Improving Authentication for Users via Better Understanding Password Use and Abuse.

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Hazel Murray

Under the supervision of:

Prof. David Malone

Department of Mathematics and Statistics
National University of Ireland Maynooth
Ollscoil na hÉireann, Má Nuad

January 2021

Contents

List of Tables

Declaration

I hereby declare that I have produced this manuscript without the prohibited assistance of any third parties and without making use of aids other than those specified.

The thesis work was conducted from October 2016 to October 2020 under the supervision of David Malone in the Hamilton Institute and the Department of Mathematics and Statistics, National University of Ireland Maynooth. This work was supported by an Irish Research Council 2017 Government of Ireland Postgraduate Scholarship and a John and Pat Hume doctoral studentship. The research was supported in part by a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

Maynooth, Ireland,

31st October 2020

Acknowledgement

Thank you to my supervisor, Prof. David Malone, for guiding me through this process for the last 4 years and being the most supportive advisor anyone could ask for. Your depth of knowledge, enthusiasm and constant willingness to answer my endless questions made my research experience enjoyable and rewarding. Thank you for your constant support of all my endeavours. This made for a truly fulfilling four years.

A special thank you to everyone in the Hamilton Institute for your friendships. To Prof. Ken Duffy for letting me cover the Institute in Christmas decorations and Halloween pumpkins, and to everyone who joined in.

Thank you to Prof. Blase Ur (and Dr. Maximilian Golla) for inviting me as a visiting researcher to the University of Chicago and adjusting plans to allow me to be able to work with SUPER Group remotely after COVID-19 struck.

Thank you to Cormac Herley, Principal Researcher at Microsoft Research, and Dr. Oliver Mason, Associate Professor at Maynooth University, for examining this thesis.

To my friends, thank you for being there for me. Aoife, Peter, Calum and Kyra playing tennis kept me going through lockdowns; Emily your postcards are always a joy to receive; Sinéad, where would I be without you catching my stats mistakes; Ciara, I loved our refreshing Thursday night sea swims; Kyra, thanks for humouring me with my art project; Anna for dropping baked goods at my door as I wrote my thesis during lockdown. To all my scouting friends here and abroad, scouts has been a joy and has provided so many rewarding experiences and skills.

Thank you to my Mam, Eileen, for being a role model and inspiration, and to Pat for supporting and encouraging me in everything I do. Also to my two sisters Shona and Ciara.

Finally, thank you to Andy for your continuous support, willingness to proofread my papers, constant encouragement and generally always being there for me.

Abstract

Passwords are our primary form of authentication. Yet passwords are a major vulnerability for computer systems due to their predictable nature, in fact Florêncio et al., conclude that human limitations makes what is often considered to be "proper password use" impossible [52]. It is vital we improve authentication with respect to both security and usability. The aim of this research is to investigate password use and abuse in order to improve authentication for users.

We investigate circulated password advice that claims to help in this security fight. We find that it is contradictory, often at odds with best practice and research findings, and can be ambiguous and taxing on users. We complete a user study investigating user and administrator perceptions of the password advice collected. We leverage knowledge of security benefits, usability and organisation costs to investigate the trade-offs that exist when security advice is enforced.

To improve password systems, effective and accurate information is needed regarding the prevalence of security vulnerabilities. We develop a guessability metric which produces guessing success results that are independent of the underlying distribution of the data. We use this to prove that small password breaches can lead to major vulnerabilities to entire cohorts of other users. We also demonstrate that a tailored learning algorithm can actively learn characteristics of the passwords it is guessing, and that it can leverage this information to improve its guessing. We demonstrate that characteristics such as nationality can be derived from data and used to improve guessing, this reduces security in an online environment and potentially leaks private information about cohorts of users.

Finally, we design models to quantify the effectiveness of security policies. We demonstrate the value of the NIST 2017 guidelines. We find that if an organisation is willing to bear costs on themselves, they can significantly improve usability for their end-users, and simultaneously increase their security.

CHAPTER 1

Introduction

In this chapter, we discuss the motivations behind the work of this thesis and provide an overview of the material presented in the following chapters. We list the publications emanating from this work and also the ethical considerations that were examined. We begin with an overview of the topics completed in this research, the motivation for their study and the key outcomes of the research.

1.1 Overview

When it comes to passwords, conflicting advice can be found everywhere. Different sources give different types of advice related to authentication. In Chapter 2 such advice is studied. First, using a sample collection of authentication advice, we observe that different organizations' advice is often contradictory and at odds with current research. We highlight the difficulties organizations and users have when determining which advice is worth following. Consequently, we develop a structure for identifying usability costs of advice. Using this model we collected information from users about the inconveniences they experience as a result of each of the security recommendations we collected from organizations' websites. Similarly, we model the security benefits brought by such advice. We then apply these models to our taxonomy of advice to indicate the potential effectiveness of the security recommendations.

Password guessing is one of the most common methods an attacker will use for

compromising end users. We often hear that passwords belonging to website users have been leaked and revealed to the public. These leaks compromise the users involved but also feed the wealth of knowledge attackers have about users' passwords. In Chapter 3, we demonstrate using proofs of convergence and real-world password data that the vulnerability of all users of a website increases as a result of a leak of any of the users' passwords from that website. We show that a leak that reveals the passwords of just 1% of the users provides an attacker with enough information to potentially have a success rate of over 84% when trying to compromise other users of the same website. For researchers, it is often difficult to quantify the effectiveness of guessing strategies, particularly when guessing different datasets. We construct a model of password guessing that can be used to offer visual comparisons and formulate theorems corresponding to guessing success.

We have lists of leaked password sets, dictionaries of words, and demographic information about the users, but we don't know which dictionary will reap the best rewards when guessing a password set. The multi-armed bandit is a mathematical interpretation of the problem a gambler faces when confronted with a number of different options which each output a reward with a certain, previously undisclosed, probability. The gambler wants to explore different machines to discover which machine offers the best rewards, but simultaneously wants to exploit the most profitable machine. A password guesser is faced with a similar dilemma. In Chapter 4, we provide a framework for using the multi-armed bandit problem in the context of the password guesser and use some examples to show that it can be effective. This demonstrates that an adaptive learning strategy is possible in real-time in password guessing systems. We also show that this method can identify the nationality of users in a password set as it guesses, and that leveraging this information will improve guessing over a generic ranked passwordset. Even when language is not a factor.

Authentication is integral to our online world. Millions are spent worldwide by organizations to achieve secure authentication. Yet we regularly see attacks which expose weak security practices. In Chapter 2, we showed that security advice relating to authentication given by one organization, can directly contradict advice given by another. There is currently no general framework an organization can use to determine what authentication practices are good for

them or which are bad. In Chapter 5, we introduce methods for quantifying the trade-off that exists between the security benefits and usability costs for authentication practices. We show how these models can be compared to provide an organization with information on the best authentication policies to implement, taking into account their own budget and priorities. We leverage this model to compare the NIST 2003 Electronic Authentication Guidelines with the NIST 2017 Digital Identity Guidelines. We demonstrate that a one-size fits all approach is unreasonable for usable security and that business size and priorities play a crucial role in determining effective security practices.

The appendices to this thesis include supplementary information related to aspects of Chapter 2 and Chapter 5. Appendix A, lists each piece of advice that was collected for Chapter 2. The advice is sorted into categories which are listed alphabetically. Appendix B, contains impressions, characteristics and notable respondent comments related to the costs that were identified by users and administrators as part of our user study. Appendix C, provides the rationale for which security benefits apply to each advice statement we collected. Finally, in Appendix D we describe in detail the methods and statistics used for assigning benefits and costs to the five NIST policies evaluated in Chapter 5.

1.2 Dissemination

In this section, we will describe the outputs and dissemination of the work described in this thesis.

1.2.1 Publications

The following publications have resulted from this work:

1. Murray, H. and Malone, D., 2017, June. Evaluating password advice. In *2017 28th Irish Signals and Systems Conference (ISSC)* (pp. 1-6). IEEE.

2. Murray, H. and Malone, D., 2018, August. Exploring the impact of password dataset distribution on guessing. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (pp. 1-8). IEEE.

3. Murray, H. and Malone, D., 2020. Convergence of Password Guessing to Optimal Success Rates. *Entropy*, 22(4), p.378.

4. Murray, H., Horgan, J., Santos, J.F., Malone, D. and Siljak, H., 2020. Implementing a Quantum Coin Scheme. In *2020 31st Irish Signals and Systems Conference (ISSC)* (pp. 1-7). IEEE. (Completed during PhD, work not included in this thesis).

5. Murray, H. and Malone, D., 2020. Multi-armed bandit approach to password guessing. In *Who Are You?! Adventures in Authentication Workshop*

6. Murray, H. and Malone, D., 2020. Characteristics and compromises: password guessing using a multi-armed bandit. *ACM Transactions on Information Forensics and Security*, **In Review**.

7. Murray, H. and Malone, D., 2020. Costs and benefits of authentication advice. *ACM Transactions on Privacy and Security (TOPS)*, **In Review**.

1.2.2 Conferences and talks

The research in this thesis has been presented to the following audiences:

Thesis in 3 National Final, Dublin. 7th November 2016. Three minute presentation of my research. Prize for best online submission.

SIAM: Society for Industrial and Applied Mathematics, National University of Ireland, Galway. 26th May 2017. Poster presentation.

Data Summit, Department of the Taoiseach, The Convention Centre, Dublin. 15th June 2017. Poster Presentation.

ISSC: Irish Signals and Systems Conference, Killarney, Co. Kerry, Ireland. 20th–21st June 2017. Presentation of research paper "Evaluating Password Advice". Winner of Best Student Paper.

USENIX Security Symposium, Vancouver, Canada. 16th–18th August 2017. Poster presentation.

Connect 2017, Maynooth University Commercialisation Event, Carton House Hotel. Three minute thesis. 23rd October 2017. Overall winner.

16th Annual Conference on Privacy, Security and Trust, Belfast, Northern Ireland. 28th–30th August 2018. Presentation of research paper "Exploring the Impact of Password Dataset Distribution on Guessing".

SecHuman Summer School, Ruhr University, Bochum, Germany. 2nd–6th September 2018. Presented a lightning talk "Password Advice: Cost Benefit Modeling".

HEAnet Conference, Galway, Ireland. 7th–9th November 2018. Speaker. "Password policies: Recent developments and possible appraise."

PasswordsCon, Stockholm, Sweden. 19th–20th November 2018. Speaker. "Advice is like Mushrooms, the wrong kind can prove fatal".

Irish Mathematical Society meeting, NUI Galway. 5th–6th September 2019. Invited Speaker. "Convergence of password guessing to optimal success rates".

Hamilton student seminar series, Hamilton Institute, Maynooth University. 15th October 2019. Speaker. "Penny-wise and pound-foolish: quantifying the effectiveness of password advice policies".

ISSC: Irish Signals and Systems Conference, Virtual, Letterkenny Institute of Technology, Co. Donegal, Ireland. 11th–12th June 2020. Presentation of research paper "Implementing a Quantum Coin Scheme". Runner up Best Student Paper.

Who Are You?! Adventures in Authentication Workshop (WAY), Virtual. 7th August 2020. Presentation of research paper "Multi-armed bandit approach to password guessing".

PasswordsCon, Virtual, Stockholm, Sweden. 23rd–24th November 2020. Speaker. "Password guessing using the Multi-armed bandit".

1.2.3 Additional outputs

The following are the list of additional outputs that have resulted from this PhD research:

- Taxonomy of 272 pieces of password advice grouped into categories such as: composition, storage, encryption, and expiry. Available on

Github [111] for use by future researchers.

- Collection of anonymous responses from users and administrators regarding the costs of following advice and their approval of this advice. Archived and publicly available on Gihub for future analysis [112, 113].
- Creation of 5 Masters level MOOC courses covering Public Key Cryptography and Security Protocols for Dublin City University.
- Member of Poster Jury for the 16th Symposium on Usable Privacy and Security (SOUPS) 2020. 10th–11th August 2020. Boston, MA, USA. Co-located with the annual USENIX Security conference.
- Program Committee for 2020 WAY Who Are You!? Adventures in Authentication, SOUPS workshop. 7th August 2020.

1.2.4 Funding

This research received the following funding:

John and Pat Hume Doctoral Studentship. Accepted 2016-2020, superseded by IRC GOI Scholarship in 2017.

Irish Research Council Government of Ireland Postgraduate Scholarship. Accepted 2017-2020.

CONNECT Funded Researcher. Duration 2016-2017, supplement to John and Pat Hume Scholarship.

Travel bursary to attend SIAM conference. Date: 26th May 2017.

Scholarship to SecHuman Summer school, Germany. Date: 2nd–6th September 2018.

1.3 Ethics

There are a number of ethical questions we considered before embarking on our research. Firstly, the use of illegally leaked password datasets and secondly the ethics of a users survey of opinions on costs of password advice. For both we applied and successful received research approval from our university Ethics Review Board. We will discuss the risks and mitigation for each of these separately as each is worthy of discussion.

1.3.1 Ethics of collecting and using leaked password datasets

As part of our research we collected password datasets that had been compromised and were subsequently leaked to the public. There arises an issue of privacy and security as a result of collecting and analyzing these password databases. We used the current best practice to minimize any harm associated with using this data. This is an account of our ethical considerations in line with our Research Ethics Board and [160].

Stakeholders The stakeholders in this scenario are the users whose password has been included in the leaked password dataset. Also the organization from which the passwords were leaked.

Informed consent It is not practical to gather consent from the stakeholders affected. The password datasets we use are already accessible to the public using common search engines. Our work is not the first publication to reference these specific password datasets [172], so we know the existence of the leaks is already known.

Harms The passwords leaked could still be in use by individuals. The passwords themselves could contain personal information. In some cases the leaked database includes other personal details such as email addresses or names.

Safeguards We recorded the frequency with which each password occurred in the database and then ranked these frequencies. This was the only information we needed to retain. Therefore, we removed the personally identifying information, usernames and emails, from our datasets before analysis. In addition, the actual passwords leaked do not appear in our publications. (We mention that the password "123456" appeared 290729 times in the rockyou.com password dataset since this has been published previously by other researchers [98, 184].)

Public interest Attackers have access to these password datasets and likely structure their attacks using the knowledge gathered from them. Therefore it is in the public interest for our analysis and defenses to be derived using the

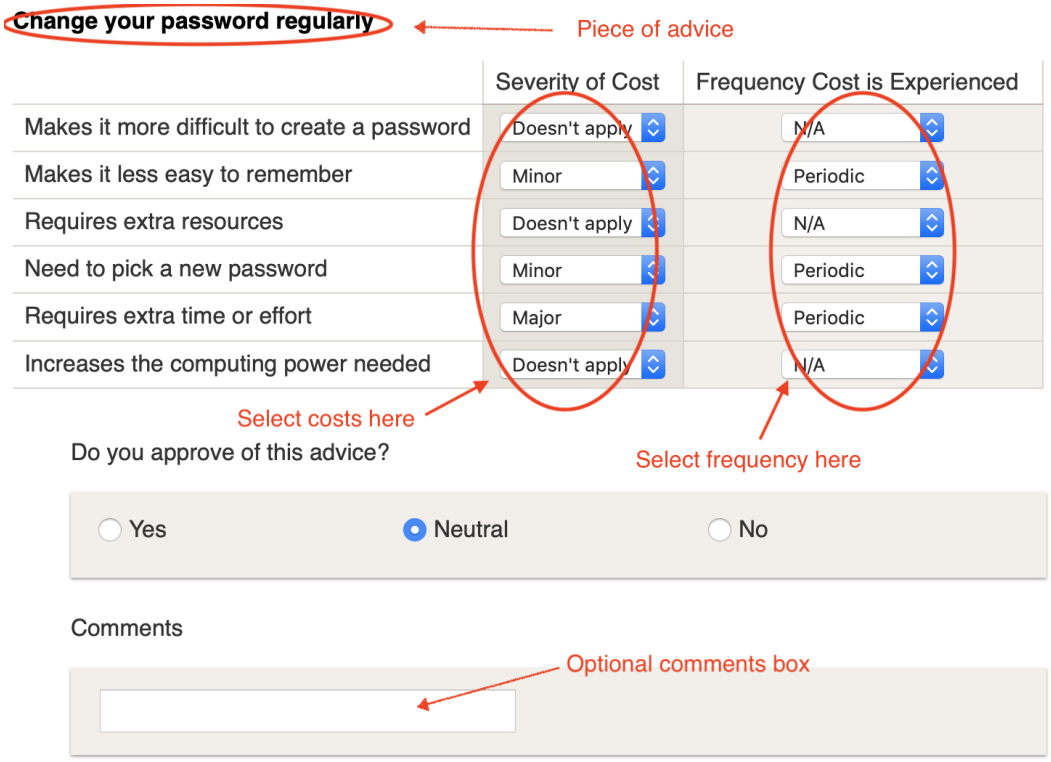


Figure 1.1: Infographic with instructions for users for the survey

knowledge we can glean from these available password datasets. The use of these datasets by multiple researchers is positive for reproducibility and offers advantages over "generated" passwords created by participants in controlled studies [46].

1.3.2 Ethics of authentication costs survey

In my research we are interested in the costs that users face as result of following password advice and requirements. We created a series of surveys for administrators and users to complete. These surveyed the severity and frequency with which they experience costs and inconveniences as a result of authentication advice. We also asked for comments and whether they approved of the given piece of advice. It was an anonymous survey.

Figure 1.1 shows an infographic providing users with an explanation of how to complete each survey question.

This research was approved by the Maynooth University Ethics Review Board. The following are the key considerations that were made in relation to this

survey:

How will participants be recruited? We used a snowball sampling technique. Initially circulating the survey to contacts in industry and in the general population via social media and other means and encouraging them to recruit future subjects from among their acquaintances.

Who will the participants be? Participants will be a mixture of end users, computer administrators/security personnel.

Justification for numbers if applicable: We have no set numbers on the survey. However, we hope to recruit 50 participants including both administrators and end users.

Will participants be remunerated and if so in what form? No

Conflict of interest? It is possible that we will recruit a disproportionate number of participants who have a vested interest in security. However, we are interested in whether we have missed any categories of costs. Therefore, this should not be a problem. If they agree with our assessment that is useful, but we are more interested in knowing whether we have missed categories of costs so their deeper security knowledge would be beneficial.

Potential Risks: It is possible that participants who have had a frustrating experience with an authentication mechanism may use the survey to express their frustration.

We have provided the advice as collected from websites and have not expressed opinions in relation to it. We are asking for opinions and frustration is one of these opinions. We are not enforcing the authentication advice on the participants in the survey. Given that we are trying to understand the costs of authentication, it is possible such participants may find the opportunity to express frustration cathartic.

Potential Benefits for this Research: Benefits to participants: In our previous research we identified that large burdens are placed on users as part of authentication procedures. Participants in this study will be given the

opportunity to voice the costs they experience as a result of authentication advice.

Benefits to usable security research: Our research project offers the first encompassing insight into the costs associated with authentication advice. Using literature reviews and the user insights gained from this survey, this is the first piece of work categorising the costs of authentication advice. It will provide a source of information for security researchers who are trying to develop authentication mechanisms that offer both security and usability for end users. This survey will indicate the shortcomings of current usability goals and potentially provide inspiration for how to rectify these.

Benefits to wider society: The wider goal of this research is to investigate the trade-offs between the costs and benefits of authentication security advice. In the past, research has shown that advice such as "regularly change your password" (i.e. change it every 90 days), offers little security advantage but results in significant inconveniences to users. Thus the usability does not trade-off well in terms of the security gained. We envision our research providing direction for security administrators so that they can enforce advice which will have a tangible benefit to the security of their organization, for minimum impact on their employees and users.

Risk/Benefit Analysis: There are few risks associated with our research and the information we learn will be valuable for improving authentication for end users, for making authentication procedures cost/time effective for organizations and providing security specialists and researchers an insight into the shortcomings of current advice in providing usability and security.

The data we collect is anonymous and it should not be possible to use it to identify an individual or their security practices.

1.4 Conclusion

In this chapter we have provided an overview of the work that is included in this thesis. We also listed key outputs of the research including publications and datasets. Finally, we provided a discussion of the ethical considerations associated with our work. We provided information regarding the steps taken to mitigate harm and the potential benefits of the work.

The rest of this thesis continues as follows. Chapter 2 investigates circulated password advice. It includes input from user studies and describes models for categorizing the costs and security benefits of this circulated authentication advice. In Chapter 3, we construct a metric for measuring success in password guessing. We leverage this metric to provide proofs of convergence which demonstrate the increased vulnerability of all users of a website as a result of a leak of any users' passwords from that website. In Chapter 4, we provide a framework for using the multi-armed bandit problem in the context of the password guesser and use real world examples to demonstrate its effectiveness. Finally, in Chapter 5 we introduce methods for quantifying the trade-off that exists between the security benefits and usability costs for authentication practices. We show how these models can be compared to provide an organization with information on the best authentication policies to implement, taking into account their own budget and priorities. We leverage this model to demonstrate the effectiveness of the NIST 2017 Digital Identity Guidelines in comparison to the NIST 2003 Electronic Authentication Guidelines.

In each chapter we include a literature review of related work and a discussion of the results.

CHAPTER 2

Evaluating Password Advice

When it comes to passwords, conflicting advice can be found everywhere. Different sources give different types of advice related to authentication. In this chapter such advice is studied. First, using a sample collection of authentication advice, we observe that different organisations' advice is often contradictory and at odds with current research. We highlight the difficulties organisations and users have when determining which advice is worth following. Consequently, we develop a structure for identifying costs of advice. Our model incorporates factors that affect organisations and users, including, for example, usability aspects. Similarly, we model the security benefits brought by such advice. We then apply these models to our taxonomy of advice to indicate the potential effectiveness of the security recommendations. We conduct a user study to inform the costs involved and to develop our model. We find that most of the advice places large burdens on humans, both system administrators, and end-users. Over 85% of the costs we identified related to the need for additional human labour or effort. This chapter is based on research published in [114] and presented at the 2017 USENIX Security Symposium [166].

2.1 Introduction

Password security is an essential part of our online security. However, the advice and restrictions placed on passwords have made them a source of considerable inconvenience for users [1]. Rules introduced around passwords and

other authentication procedures sometimes seem unsupported by research and their security objectives can be unclear [181].

In his 2003 book "Beyond Fear", Schneier explains that "almost every security measure requires trade-offs. These trade-offs might be worse usability of a system, additional financial costs or a decrease of security in another place". In this chapter we delve deeper into the trade-offs and effects of security policies and discover whether decreases to usability are justified by security increases.

Understanding the extent of the costs to the user or organisation and trading these off for the benefits is important if we envision a user and organisation as having a fixed amount of effort they are willing to exert for their security. Wasting a large amount of this *compliance budget* [9] with advice that is high cost and low impact means other advice with more effective benefits might get pushed aside. This research highlights the often-hidden usability costs of authentication security policies.

In this chapter, we describe our collection of security advice and discuss a method of categorizing this advice. We also introduce a provisional structure for identifying categories of costs. We then create cost and benefit models which can be used to evaluate authentication advice. We use these models to evaluate the 270 pieces of authentication advice we collected. We discuss the trade-off which takes place between the costs and the benefits for each piece of advice and offer insight into the general trends in the costs and benefits of authentication policies. The work in this chapter is published in part in our 2017 conference paper [114].

In Section 2.3 we describe our method for collecting authentication advice, in Section 2.4 we describe the categorization of the advice and our method for representing the advice in tables. In Section 2.5, we discuss the characteristics of the advice we collected, with particular emphasis on its relation to current literature. In Section 2.6 we define our method for identifying categories of costs that exist for authentication advice. In Section 2.7, we describe the set up of our user study which investigates how users and administrators are affected by the need to follow or implement security advice. The results of this user study are represented in Table 2.5 and described in Appendix B. Section 2.8 outlines our model of the benefits of authentication advice. Further details on the assigned benefits of advice are available in Appendix C. Section 2.9

discusses these identified costs and benefits. In this section, we also identify patterns and traits in the costs and benefits identified.

2.2 Related Work

Research into areas of authentication security is regularly conducted. Researchers are interested in; guessing password [83], password reuse [170, 57] and alternatives to passwords [13, 154], among many other areas.

Researchers have been interested in modelling and improving password guessing for a long time. The first strategic methods used dictionary attacks and were proposed by Morris and Thompson in 1979 [106]. These are still widely used today in the form of John the ripper [80] and Hashcat [63]. Users' passwords are inherently easy to guess as they often take predictable forms. In 2012, Malone and Maher investigated the prevalence of high frequency passwords in datasets [98]. They found that these passwords were often simple patterns such as "123456" and "000000" and that they often follow a theme related to the service they were chosen from. For example, a common LinkedIn password is "LinkedIn". It is possible for controls to be put in place to mitigate this, such as blocklisting common passwords or installing a password strength meter to guide users away from weak passwords. Kelley et al. looked at a method of computing the expected number of guesses before a certain password is compromised [83]. They used this *guess-number calculator* to model the security that is offered by different composition and blocklisting restrictions.[1] Ur et al. in 2012 measured users' reactions to password strength meters [163]. They found that passwords created when a stringent password meter was in place had an increased resistance to password cracking than those created with no meter in place. However, they also reported that users found these stringent meters annoying and that it led to them taking more time to create a password. Here we see a direct trade off between usability and password security.

In 2013, Nithyanand and Johnson studied the password habits of 20 participants [123]. They reported that 19 of 20 participants said they reused passwords for multiple accounts. In 2014, Das et al. estimated that 43–51%

[1] For example, your password must contain upper case and lower case letters, numbers and symbols, or, your password must be at least 16 characters long.

of users re-use passwords across sites [34]. They created a cross-site guessing algorithm which can use a users' leaked password from one site to guess their potentially similar or modified passwords on another site. In 2018, Wang et al. conducted a large scale empirical analysis of password reuse [170]. They studied a dataset of 28.8 million users across 107 services over 8 years. They observed that 38% of the users directly reused the same password at another site and a further 21% of users altered and then reused the same password at a new site. Advice regarding password reuse is evidently regularly disregarded by users. To protect their users, some providers monitor black-market stolen password databases and notify their users if they notice a password which is currently in use for that user on their service. In 2018, Golla et al. studied these notifications and investigated user perceptions showing that 84.5% of users self-reported that they would intend to take action within 24 hours of seeing the notification [57].

Passwords are not the only form of authentication, though they remain the most prevalent. Bonneau et al. completed a comprehensive study of password alternatives including: password managers, federated single sign-on, graphical passwords, one-time passwords, hardware tokens, phone-aided schemes and biometrics [13]. They consider each authentication method under the headings of DUS: deployability, usability and security. In this 2012 analysis, they found that none of these alternative methods retained the full set of benefits that can be offered by standard text-based passwords. In 2011, Stajano introduced a hardware token called Pico [154]. It was designed to replace passwords and transform the current "something you know" authentication question into a "something you have" question. Rather than providing a usability security trade-off, Pico claimed to offer greater usability and greater security. However, in Bonneau et al.'s analysis it fell short on deployability. In 2017, inspired by Pico, Peeters and Grenman introduced n-Auth [130]. N-Auth is an authentication solution designed to work with any number of accounts and using a users' mobile device. It claims to solve many of the deployability issues which surfaced for Pico, potentially offering deployability, security and usability in a single protocol. Also in 2017, Halunen et al. extended Bonneau et al.'s work evaluating password alternatives [61]. They expanded the deployability, usability and security metrics introduced by Bonneau et al. and used the five categories: flexibility, security, non-intrusive, usability and privacy.

Authentication policies place a large burden on users and organisations [75]. In the aptly titled paper "The true cost of unusable password policies: password use in the wild", Inglesant and Sasse [75] find that users are generally concerned with maintaining security, but that while an organisation or user may want to enforce strong security, if the time and monetary constraints are too high then it might not be feasible. Redmiles et al. show that the quantity of advice that users are given is too broad and that even experts struggle to prioritise it [135]. Adams and Sasse [1] find that low motivation and poor understanding of the threats lead users to circumvent password security policies. Beautement et al. find that bypassing security policies is a widely employed practice [9]. They introduce the idea of a compliance budget, which formalizes the understanding that users and organisations do not have unlimited capacity to follow new instructions and advice.

Herley [66] argues that a users' rejection of security advice is rational from an economic perspective. Herley quantifies the costs versus benefits for three specific authentication guidelines: password rules, phishing site identification advice and SSL certificate warnings. Redmiles et al. show that when participants are explicitly aware of security risks and costs they will likely make a rational (utility optimal) security choice [134]. This provides evidence that the user will follow a security policy that they perceive to be beneficial. This encourages the use of clear policies which emphasise a benefit for users that can be seen as a counteraction for the costs they will be expected to bear.

Braz et al. [17] propose a usability inspection method which can be used to compare the security problem to the usability criteria for online tasks such as; authenticate yourself, transfer funds or buy a concert ticket. Shay and Bertino [148] and Arnell et al. [6] have built simulations to model the costs versus the benefits of complexity rules, throttling and regular expiry. Shay and Bertino explain that security is an economic as well as a computer problem.

In this chapter, we demonstrate that security advice can be ambiguous, contradictory and at times may not even have any clear benefits. We expand on current work by defining a formal approach to identifying costs of security advice and instigate a user study to identify the costs that apply to a large range of authentication advice. We also apply a simple framework for analysing the authentication related security benefits of advice. This allows us to identify

costs and benefits for all classes of authentication security advice.

2.3 Collection of password advice

To begin studying password advice, we first needed to collect a selection of the advice that is distributed to users. We primarily used Internet searches to collect password advice, but also looked at advice given by standards agencies and multinational companies. We attempted to recreate the actions an individual or organisation might take when seeking to inform themselves about proper password practices. As such, while the advice given in academic papers might be more considered, if it was not easily accessible, we did not include it in our study. In total, we collected 270 pieces of password advice from 20 different sources. The 270 pieces of advice are shown in Appendix A and the types of sources from which the advice was gathered are shown in Table 2.1.

An interesting application of our work is the simplification of visualizing the costs and benefits of password advice. This can be applied to other advice we know is implemented that was not collected in our survey. Out of curiosity we added "Do not allow users to paste passwords". This has been discussed online as a piece of bad advice which unfortunately became common practice, but has no discernible security ramifications [73].

Table 2.1: Break down of advice sources.

Source	Number
General articles	6
Multinational companies	5
Security specialists	5
Universities	4

2.4 Categorization of password advice

To extract meaning from the pieces of advice that we collected, we systematically subdivided the advice into categories. Within each category we created statements, which generalized the recommendations pertaining to each category.

2.4.1 Method of categorization

Our first step after collecting the advice was to group it into categories. For this we considered each piece of advice individually. The first pieces of advice we examined suggested our starting categories. From there, each piece of advice was either included in one of our existing categories, expanded the scope of an existing category or created a new category to suit it. For example, when approaching a piece of advice which said "Use a unique password for each of your important accounts" we created the category *Reuse across Accounts*. However when a second piece of advice stated "Don't recycle passwords" we altered the name of the category to be the more general *Password reuse*. As an example, the first two columns of Figure 2.1 shows the seventeen pieces of advice which became grouped under the category *Password reuse*. Appendix A shows the grouping for all pieces of advice gathered.

In total, we identified 27 categories shown in Table 2.2. The categories are listed in two columns; one showing categories containing advice aimed at users and the second showing advice aimed primarily towards organisations. Also included are the number of pieces of advice under each category.

We collected 179 pieces of advice aimed towards users and 91 pieces of advice aimed towards organisations. Despite its greater quantity, user advice has been subdivided into fewer categories. We speculate this could be related to the wider variety of roles an organisation plays in the security of passwords. But it could also reflect our greater familiarity with user advice. While everyone is a user, not everyone has held all roles in an organisation, and therefore it was easier to interpret and categorize user advice.

2.4.2 Classification into statements

Once we divided the advice into categories, we noticed the pieces of advice within each category did not necessarily promulgate similar opinions. It was therefore necessary to subdivide the advice into statements which offer a similar message. In Figure 2.1 we can see how the seventeen pieces of advice under *Password Reuse* were grouped into three distinct statements:

- Never reuse a password.
- Alter and reuse passwords.

Table 2.2: Categories and the quantity of advice they contain.

Users	#	**Organisations**	#
Phrases	39	Expiry	27
Composition	28	Storage	16
Personal Information	21	Generated passwords	7
Reuse	17	Individual accounts	7
Personal password storage	17	Throttling guesses	6
Length	17	Keeping system safe	6
Sharing	13	Default passwords	4
Keep your account safe	10	Administrator accounts	4
Backup password options	8	Input	3
Password managers	4	Shoulder surfing	3
Two factor authentication	3	Policies	2
Username requirements	2	Transmitting passwords	2
		SNMP strings	2
		Password auditing	1
		Back up work	1
Total	**179**	**Total**	**91**

- Don't reuse certain passwords.

In this figure we make note of pieces of advice that contradict the main statement with a star (*). It is important to note that while it appears that there is no contradictory advice within the category statement "Never reuse a password", the third statement "Reuse certain passwords" is itself a contradiction. We make note of this by placing a star (*) in the text box of the third category. It is also represented by a star in Table 2.3. Already we are beginning to see inconsistencies with the advice that is circulated.

Thus, within each category we created generalized statements of the advice that was given. It is worth noting that the labels for advice are given from the perspective of the majority. For example, if two pieces of advice state that passwords should not include published phrases and one piece of advice states that it would be a good idea to use published phrases, then the advice will be labeled as "Don't include published phrases".

The statements relating to the categories of advice are shown in Table 2.3. The full breakdown of advice into statements and categories is shown in Appendix A. Table 2.3 shows how many pieces of the advice agree with the sentiment of the statement (number of sources advising), and how many dis-

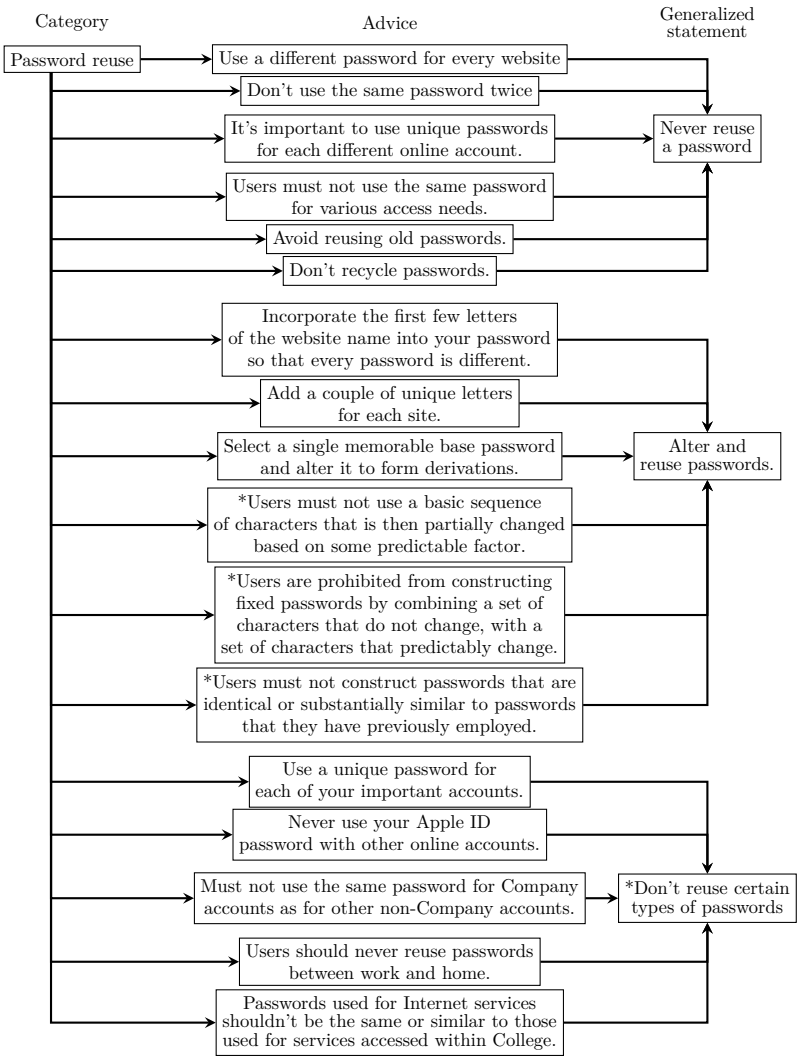


Figure 2.1: Method for categorizing advice. Example: Reuse.

agree (number of sources contradicting). This gives a clear indication of the inconsistencies in circulated password advice.

Splitting the advice statements by user or organisation made it easier for us to imply the intent of the advice that the statements are describing. For simplicity we did not separate the advice within a single category between the two tables. In the cases where the advice is not perfectly suited to the partition, we make note of it in our discussions. For the most part the split gives us a good overview.

Table 2.3: Breakdown of advice into statements

Users	#x	#✓
Backup password options		
Email up-to-date and secure.	0	3
Security answers difficult to guess.	0	3
Do not store hints.	0	2
Composition		
Must include special characters	5	7
Don't repeat characters.	0	3
Enforce restrictions on characters.	1	12
Keep your account safe		
Check web pages for SSL	0	1
Manually type URLs.	0	1
Don't open emails from strangers.	0	1
Keep software updated.	0	2
Keep anti virus updated.	0	2
Log out of public computers.	0	2
Password protect your phone.	0	1
Length		
Minimum password length.	0	13
Enforce maximum length (<40).	1	3
Password Managers		
Use a password manager.	1	2
Create long random password.	0	1
Personal Information		
Don't include personal information.	1	5
Must not match account details.	0	8
Do not include names.	1	6
Personal password storage		
Don't leave in plain sight.	0	4
Don't store in a computer file.	1	2
Write down safely.	1	6
Don't choose "remember me".	0	3
Phrases		
Don't use patterns.	0	6
Take initials of a phrase.	0	4
Don't use published phrases.	1	2
Substitute symbols for letters.	1	2
Blocklist common passwords	0	2
Don't use words.	0	16
Insert random numbers and symbols	1	4
Reuse		
Never reuse a password.	*5	6
Alter and reuse passwords	3	3
Don't reuse certain passwords.	0	5
Sharing		
Never share your password.	0	9
Don't send passwords by email.	0	3
Don't give passwords over phone.	0	1
Two factor authentication		
Use for remote accounts.	0	1
Use multi-factor authentication.	0	1
2 factor authentication using phone.	0	1
Username		
Enforce composition restrictions.	0	1
Don't reuse username.	0	1

(a) User advice statements

Organisation	#x	#✓
Administrator accounts		
Not for everyday use.	0	1
Must have it's own password.	0	2
Should have extra protection.	0	1
Backup work		
Make digital & physical back-ups.	0	1
Default passwords		
Change all default passwords.	0	4
Expiry		
Store history to eliminate reuse.	0	5
Change your password regularly.	4	8
Change if suspect compromise.	0	10
Generated passwords		
Use random bit generator.	*2	2
Must aid memory retention.	0	2
Must be issued immediately.	0	1
Only valid for first login.	0	1
Distribute in a sealed envelope.	0	1
Individual accounts		
One account per user.	0	4
Each account password protected.	0	3
Input		
Don't performed truncation.	0	1
Accept all characters.	1	1
Keeping system safe		
Implement Defense in Depth.	0	2
Implement Technical Defenses.	0	1
Apply boot protection.	0	1
Monitor and analyse intrusions.	0	1
Regularly apply security patches.	0	1
Network: SNMP community strings		
Don't define as standard defaults.	0	1
Different to login password.	0	1
Password auditing		
Attempt to crack passwords.	0	1
Policies		
Establish clear policies.	0	2
Shoulder surfing		
Offer to display password.	0	1
Enter your password discretely.	0	2
Storage		
Encrypt passwords.	*4	7
Restrict access to password files.	0	2
Encrypt password files.	0	1
Store password hashes.	0	4
Don't hardcode passwords.	0	1
Contracts state how pwds protected.	0	1
Throttling		
Throttle password guesses.	0	6
Transmitting passwords		
Don't transmit in cleartext.	0	1
Request over a protected channel.	0	1

(b) Verifier advice statements

#x = number sources contradicting.
#✓= number sources advising.

2.5 Discussion of advice collected

In this section we will discuss the advice we collected. For each advice category we will provide more information about the intention of the advice statements within a category. We will refer to interesting characteristics within the advice we collected, context in relation to how advice can be implemented and make connections to relevant academic literature. We will also refer to some interesting comments we received from users and administrators who were part of our user study (described later in Section 2.7).

The advice categories are discussed in alphabetical order. We will alternate between discussion of categories as a whole, discussion of specific advice statements within a category, and at times, dividing our discussion between the advice in a category that was contradicted by different sources and the advice that was unanimously given by sources. This section is best read with Table 2.3 at hand.

2.5.1 Administrator Accounts

This category relates to access controls. The four pieces of advice in this category recommend creating a clear distinction between administrator accounts with privileged access and normal user accounts. This means an administrator must switch between their privileged and general user account depending on their current task. This could be time consuming for an administrator. This burden is lessened by the implementation of programs such as su and sudo which allow users to easily run programs which require extra security privileges. All advice in this category is unanimous.

2.5.2 Backup password options

The advice relating to backups for forgotten passwords refers to email, security questions and hints. All the advice is given unanimously. Three organisations recommend the use of security questions. Schechter et al. show that security questions can be very easily guessed and are also easily forgotten by users [144]. The *do not store hints* advice and the requirement for security questions to be difficult to guess increases the forgettability for users.

2.5.3 Backup work

One organisation recommended making digital and physical back ups. This can save an organisation a large amount of hardship if a breach or ransom attack occurs. It takes time, computing memory and resources in advance though with the organisation needing to save regular digital and physical back ups.

2.5.4 Composition

Composition restrictions are regularly enforced by websites but the advice given is not consistent from site to site. It is interesting to note that Herley [66] hypothesizes that different websites may deliberately have policies which are restrictive to different degrees as this can help ensure that users do not share passwords between sites. Below we will discuss each of the three statements associated with composition.

Must include special characters Seven sites instructed users to *include special characters* in their passwords, but five sites recommended placing restrictions on the special characters that could be used. The main restriction on special characters was "do not use spaces". However, one piece of advice stated the more direct "do not use special characters". By not allowing users to include any special characters an attackers' search space is decreased.

Don't repeat characters Not allowing the repetition of characters deters users from choosing passwords such as "aaaaaaa" or "wwddcc". Depending on the strictness of the restriction it could eliminate words such as "bookkeeper" or "goddessship". It could also cause some inconvenience for random password generators where the word "Sdt2htTtd65c8h" could be rejected. It is likely that any passwords that consist of common sequential strings could be removed using a blocklist rather than a blanket-ban on consecutive repeating characters.

Enforce restrictions on characters We collected twelve pieces of advice encouraging composition restrictions on passwords and only one piece of advice against it. The source rejecting composition rules was the NIST 2017 authentication guidelines [59]. These guidelines have received praise from

the authentication research community [32]. This raises the question about whether organisations will begin to disseminate these new security practices or continue to enforce their stringent password restrictions?

Kelley et al. show that strict composition rules do increase the security of passwords [83]. However, similar security can be offered by password blocklists or mandating minimum 16-character passwords. In a related study, Komanduri et al. [87] showed that the 16-character password restriction was less annoying and less difficult for users. In addition, when composition rules are enforced, the probability of a user including a "1" as their number and an "!" as there symbol is high [164]. So, an attacker who knows the composition restriction in place could potentially tailor their attacks to suit it.

2.5.5 Default Passwords

Four organisations recommended changing all default passwords. Default passwords are used in Internet of Things devices, wifi routers and other equipment among other things. These default passwords are in their essence not secure. They have been shared publicly and often the same logins are used for many or all of the devices of that type manufactured by a given company. Manufacturers set a default password, which an administrator is supposed to change during setup. However, Farik et al. analysed 1096 network routers and found only 359 distinct passwords and only 259 unique passwords [47]. In fact, 18.8% of the routers studied required no login password. Other common default passwords in the routers were: 'admin' (15.8%), 'password' (6%), '1234' (2.9%), 'epicrouter' (1.6%), '0' (1.4%), 'root' (1%), 'changeme' (1%), 'tech' (1%), 'access' (0.9%) and 'router' (0.8%). In our user study (which we will describe in Section 2.7), all administrators surveyed agreed with changing default passwords. Changing the default password takes time during the set up of the device but once done should not need to be updated unless it is compromised.

2.5.6 Expiry

Unanimous We found five pieces of advice telling organisations to *Store password history to eliminate reuse*, one encouraged organisations to *Enforce a minimum password age* and ten were in favor of *Changing passwords if compromise is suspected.* If organisations do *store their users' password history*

this creates an additional security hole as the company needs to allocate resources to protect this file. Also, even though users can no longer reuse prior passwords, alterations are still possible [149]. In fact, Zhang, Monrose and Reiter [185] identify that we can easily predict new passwords from old when password aging policies force updates.

The reason given for introducing a *minimum password age* is to prevent users from bypassing the password expiry system by entering a new password and then changing it right back to the old one [103]. However, if an attacker gains access to a users' account and changes their password the user will be unable to change it again until the required number of days have elapsed, or with an administrators' help.

Ten pieces of advice recommended *changing passwords if a compromise is suspected*. This can be inconvenient for users not affected by the compromise, and also those who are. If there is a breach at the server the users were not at fault yet still they must choose a new password.

Contradicting From anecdotal evidence we know the advice *change your password regularly* is widely hated by users [60]. This is summarized by one user in our user study saying "I hate this! The only solution I've come up with is to increment a number in the password each time. So inconvenient and frustrating, especially when combined with other bad password advice". Seven pieces of the advice we collected encouraged the use of password expiry while only four pieces of advice discouraged it. This is despite research suggesting that the security benefits are minimal [24][185]. This implies the inconvenience to users is worth less to organisations than the minimal security benefits. Or, do organisations want to be seen to be enforcing strong security practices, and forcing expiry is just one way of doing this?

2.5.7 Generated Passwords

Contradicting We have two contradictory advice statements in this category. One advises creating random passwords using a random bit generator. The second states that generated passwords should aid memory retention. Random passwords are very difficult for users to remember [187].

Unanimous One piece of advice said that generated passwords must not be stored and one piece of advice said that generated passwords should only be valid for the users' first login. The former takes administrative time to distribute immediately and the user cannot clarify the password is correct if it has not been stored anywhere. The second requires the user to generate their own password as well as administrators to generate the initial generated password.

2.5.8 Individual Accounts

Having one account per user with a password is important for maintaining access controls and also tracking errors or attacks back to a source account. It is standard to enforce this in organisations. One of our user study respondents commented to say that "the alternative is unthinkable now". Though, two respondents did say that their approval of the advice would depend on the system.

2.5.9 Input

One organisation advised that truncation of the password should never be enforced. That is, if a user creates a 12 character password, the verifier should not decide to just store and compare the first 6 characters for authentication. This advice has a clear security impact as the user will not realise that their password is much weaker than they intended. The magnitude of the costs of not allowing truncation seem small, though one administrator in our user survey did flag that truncation allows "compatibility with legacy systems".

There was disagreement between two organisations about accepting all characters. One organisation argued for acceptance of all ASCII and UNICODE characters. This is expensive because it requires string normalization. The other organisation encouraged the removal of certain characters, specifically they recommended removing consecutive or all space characters.

2.5.10 Keeping system safe

All the advice in this category was unanimous. It recommends a number of background defense strategies. Implement defense in depth and apply technical defences are both vague pieces of advice but good in theory. They require a system layered with security, and technical defenses can reduce the human

factor. Applying access controls is one form of a defense in depth strategy. It makes sure that users have access only to their isolated parts of the larger system. Monitor and analyze intrusions is a continuous job and often intrusion points are only discovered retrospectively.

2.5.11 Keep your account safe

Check webpages for TLS This advice tells the user to check that the connection from the web server to the browser is encrypted, usually by identifying the closed padlock in the URL bar. However, Schechter et al. found that 53% of their study participants attempted to log in to a site after they were interrupted by a strong security warning. Likely, even if users see an unlocked padlock symbol they will still continue to use a webpage [146]. In addition, [38] show that checking a valid certificate exists is no longer a good protection against phishing. In 2018, 49.4% of phishing sites were using SSL/TLS.

Manually type URLs Manually typing URLs can be very time consuming for users and the assortment of symbols included can make transcription difficult. Manually typing URLs means a user should not be fooled by a fake link. For example, a link www.safe.com which actually directs to www.you-are-in-trouble.com. Manually typing the URLs can lead to a typographical error and a malicious website could take advantage of this. For example a website called www.faceboook.com which masquerades as facebook and asks for login credentials. Only one piece of advice asked for URLs to be manually typed.

Don't open emails from strangers The goal of this advice is to protect users against fraudulent emails and viruses in email attachments. However, it is unclear how effective a user proactively checking the "From" email address would be against these attacks. By means of Email Spoofing, an attacker can very easily create an email that appears to come from someone it does not [71]. In certain jobs it can be impossible to not open email from strangers and even in everyday life it can be very inconvenient. However, a 2012 study by Böhme and Moore found that as a result of concerns over cybercrime 42% of participants say they do not open email from strangers [11]. We only collected one piece of advice which said *Don't open emails from strangers.*

Keep software updated & keep antivirus updated Two places recommended these pieces of advice. In their paper "Tales of Software Updates: The process of updating software" Vaniea and Rashidi found that 49.3% of respondents relayed negative experiences with software updates [168]. This is likely similar for antivirus updates with the added disadvantage that antivirus may also need to be repurchased regularly.

Log out of public computers Finding the person before you on the public computer has not logged off can lead to different reactions. Some might take advantage and snoop around, maybe leave a message for the user to see to know that they were vulnerable, or just log them off. But with this privileged access an opportunistic attacker can do a lot of damage. Spending money with linked credit cards, masquerading as the user and asking for money to be transferred by the user's peers, and setting up backdoors into the accounts for future use, among other things. Despite the fact that we know these are threats we could find little evidence of reported breaches as a result of this. Is it possible that it does not occur? Or maybe do victims just not reveal this as the reason for their breach?

Password protect your phone Nowadays users can conduct most online transactions via mobile phones. Their portability makes them susceptible to theft. Therefore, password protecting phones could be more important even than password protecting our computers. Yet because phones are carried on our person, it is possible that users are more likely to leave themselves continuously logged on to applications.

2.5.12 Length

Enforce a minimum password length Enforcing a minimum length inconveniences memorability and may force users to alter or change their password. If our aim is to minimize password reuse, then this might not necessarily be a draw back [66]. A minimum length could also be seen as reducing the total search space of an attacker. However, it definitely has the advantage of restricting the use of zero or one character passwords and protecting against simple brute force attacks. Most of the advice encouraged the minimum length to be set at eight characters.

Enforce maximum password length Three pieces of advice recommended enforcing a maximum password length: 15 characters [76], 20 characters [129] and 40 characters [77]. Interestingly Paypal do not list maximum password length as one of their restrictions. Only when a user attempts to enter their password is the restriction revealed. NIST 2017 guidelines [59] state that "verifiers should permit user-chosen memorized secrets to be at least 64 characters in length". Restricting the length of a password inconveniences a users' personal system for password generation, in particular it affects users who usually choose long passwords. It also restricts the output of a random password generator and introduces an upper bound on the attacker's search space.

2.5.13 Network: SNMP community strings

A SNMP[2] community string can be compared to a username or password; knowing the string provides access to a device or router's statistics. Community strings are only used in devices supporting the SNMPv1 and SNMPv2c protocols. The newer SNMPv3 uses an encryption key along with a username and password for authentication.

Don't define SNMP network community strings as standard defaults The standard defaults for community strings are set by the vendor. Vendors can choose the same password for all their devices and can set default passwords as simple as 'public'. These defaults are generally easily guessed or accessible to attackers who can use them to find out about an organisation's network and potential access points.

SNMP community strings should be different to login passwords Generally community strings are not encrypted. Because they are transmitted in cleartext they can be read by anyone. This is an issue if an administrator has chosen a value for a community string which is the same as a password they use elsewhere.

2.5.14 Password auditing

Attempt to crack passwords Administrators attempting to crack users' passwords is a method for removing the "low hanging fruit" from attackers

[2]SNMP: Simple Network Management Protocol [152].

reach. By requiring users to change their password if it was guessed by the administrator, the hope is that a stronger password will then be chosen by the user. This is a classic application of the original "crack" program. The shellscript "Scripts/nastygram" is invoked by the password cracker to send mail to users who have guessable passwords [110]. This is a pretty standard policy for organisations [84][182] yet only one piece of advice we collected recommended it. We wonder whether organisations are unwilling to openly admit to and recommend this practice. Maybe this would change if there were standardised ethical procedures in place.

2.5.15 Password managers

Use a password manager Password managers are a convenient method for storing and auto-filling users' passwords. The single sign on means that users recall a single master password which is used to protect all other passwords. A password manager does mean that the user is relying on an external agent to store their passwords and therefore if this agent is compromised then the passwords of all accounts are compromised.

Password managers which automatically fill in the users' credentials with no user interaction do have some corner case vulnerabilities [150]. Though this same paper shows that a password manager can provide more security than the normal manual typing of the password. But it does depend on the set up of the specific password manager.

Throughout the user studies (described in Section 2.7) we saw very positive responses in favour of password managers. Respondents in our survey said "Would almost consider it essential for modern Internet usage. Some additional setup and potentially cost, but absolutely worth it" and "I think using a password manager is pretty much essential at this stage". Though one other respondent said: "Password managers can be good. Muscle memory is better".

Create long random passwords One piece of advice recommended that when a user is using a password manager they should "create long random passwords". Long random passwords are resilient to brute force guessing but generally are too much of a memory strain for users. Where a password manager is saving the password for each use, the usability issues are no longer a concern.

2.5.16 Personal information

Unanimous Eight pieces of advice instruct against choosing passwords that match account information. The advice is enforceable and protects against elementary targeted attacks [78]. In 1989, Bishop and Klein cracked 40% of 14,000 UNIX accounts using guesses derived from associated usernames or account numbers and dictionaries [10].

Contradicting The advice, *don't include personal information in your password* is issued by five sources. It is partially contradicted by the advice which says: "Personal details such as spouse's name, vehicle license plate, PPS or social security number and birthday must not be used unless accompanied by additional unrelated characters" [39]. It does not seem feasible to enforce this without cross referencing against some body of user information. However, if the advice is followed it would protect against a targeted attack. Castelluccia et al., find they can crack 5% more passwords by leveraging personal information about users [21].

Eight different sources advise against the inclusion of names in passwords, e.g. "do not choose any common name" [14]. However, one source contradicts this saying: Choose "someone else's mother's maiden name (not your own mother's maiden name)" [55]. By not including your own mother's maiden name a targeted attack will not be affected. However, it offers no protection from a bulk guessing attack. We consider a ban on names to be capable of eliminating a significant number of guesses for an attacker. In addition, words which double as names could be eliminated, "Bob", "Amber", "Jack", as a result of this restriction.

2.5.17 Personal password storage

Unanimous Four sources advised users to *not leave passwords in plain sight.* This advice is difficult for an organisation to enforce.

Of the three sources warning against the use of the "remember me" option, two advised never using it, and one said never to use it on a public computer. By not using the "remember me" option, there is an increased need to remember passwords. Also, more user time is consumed, since the user now needs to enter

their password every time. It does decrease the chance of an opportunistic or targeted attack.

Contradicting Two sources told users to *not store passwords in a plain text computer file* and one source recommended that if you were doing it, a unique name should be chosen for the file so people don't know what's inside. This advice still shows an awareness of the risks of a targeted attack.

If a user *writes their passwords down safely* then if they forget their password they can retrieve it from the safe location. Adams and Sasse [1] conducted a survey of corporate password users and found them flustered by password requirements and coping by writing passwords down on post-it notes. Komanduri et al. [87] found that most participants in their study write down or otherwise store their passwords. Interestingly they also find that, storage is correlated with the use of higher-entropy passwords. In fact, some experts recommend writing passwords down as a mechanism to cope with numerous passwords [93]. However, Shay et al. found that users are more likely to share and reuse their passwords than to write them down [149]. We found six pieces of advice recommending that users write their passwords down safely, and one piece of advice discouraging it. In our survey of administrators (Section 2.7), four administrators approved of users writing passwords down safely, two disapproved and one was neutral.

2.5.18 Phrases

Advice regarding password phrases was the most commonly given advice we encountered. This implies that advice is mostly concerned with making passwords "strong". While this is important for some attacks, for attacks such as phishing and keylogging the strength of the password is irrelevant [186, 53].

Unanimous Within the category *Phrases* there were no contradictions for the statements: *Don't use patterns, Take initials of a phrase* and *Don't use words.* The last is particularly interesting since from leaked password database we know users primarily choose word based passwords [176]. Shay et al. find that the "use of dictionary words and names are still the most common strategies for creating passwords" [149]. This depicts how ineffective some

password advice can be and is possibly a reflection on the costs appearing to not outweigh the benefits from a users' point of view.

Contradicting The statements: *Don't use published phrases* and *Substitute symbols for letters* had contradictions. For *don't use published phrases* the advice given was:

1. "Don't use song lyrics, quotes or anything else that has been published."
2. "Do not choose names from popular culture."
3. "Choose a line of a song that other people would not associate with you."

The last piece of advice directly contradicts the first. This inconsistency makes it no surprise that users seem disinclined to follow security advice [75, 1].

The advice statement *Substitute symbols for letters* is proposed by two sources but is advised against by a third. We know from Warner [174] that passwords with simple character substitutions are weak. Yet, 2 of 3 pieces of advice recommend it. This could stem from the attitude that it is "better than nothing".

2.5.19 Policies

Establish clear policies Two places encouraged organisations to *establish clear policies*. This is interesting since a lot of the advice we have collected is contradictory. Including the advice from those two organisations who gave this advice!

2.5.20 Reuse

We collected six pieces of advice telling users to *never reuse passwords* and three pieces telling users to *not reuse passwords for certain sites*. In addition, we found three pieces of advice encouraging users to *alter and reuse their passwords* and three pieces telling users to not alter and reuse their passwords. There seems to be little agreement among the distributed advice in terms of password reuse.

Never reuse a password vs. reuse for certain accounts Das et al. estimate that 43–51% of users re-use passwords across sites [34]. They also provide algorithms that improve an attacker's ability to exploit this fact. Florêncio, Herley and Van Oorschot [52] declare that, "far from being unallowable, password re-use is a necessary and sensible tool in managing a portfolio" of credentials. They recommend grouping passwords according to their importance and reusing passwords only within those groups. Interestingly, the advice we collected in the category *Don't reuse certain passwords* gave a slightly different take on this advice. The advice mostly asked users to not use the password used for *their site* anywhere else, e.g. "Never use your Apple ID password for other online accounts". Most organisations gave advice prioritizing their own accounts. Only one piece of advice suggested using a unique password for any important accounts [58].

In our user study, one respondent said "I experience regular frustration and lose valuable time at work trying to login to numerous platforms with different passwords".

Alter and reuse passwords An alternative to grouping accounts for reuse is to alter and then reuse a password. This advice was given by three sources and rejected by three sources. Alterations to password are often very predictable. Using a cross-site password guessing algorithm, Das et al. [34] were able to guess approximately 10% of non-identical password pairs in less than 10 attempts and approximately 30% in less than 100 attempts.

2.5.21 Sharing

Nine pieces of advice said "Never share your password". Three specified not sharing by email and one specified not sharing over the phone.

Weirich and Sasse find that sharing your password is regarded as a gesture of trust [179]. Refusing to share your password with someone is an indication that you do not trust them. A 2011 study of 122 people found that one third of respondents reported sharing their personal email password, a quarter shared their Facebook password and approximately 20 % of people who had work email passwords reported sharing them with colleagues [81]. However, the study did find that thought and consideration is given before the password

is shared. This tells us that users are aware of the security risks and accept them on the ground of trust.

2.5.22 Shoulder surfing

Offer to display password One piece of advice recommended offering to display the users' password when they type it at login. Jacob Nielsen in 2009 challenges the masking of passwords (i.e. password: ********) [119]. Bruce Schneier, after originally agreeing with the removal of masking, here [147] sums up why password masking needs to continue in certain contexts.

Enter your password discretely This advice is unenforceable and relies on user education. Research has analyzed technical mitigations for the threat of a shoulder surfing attack [90, 140]. However, Florêncio, Herley and Coskun suspect that shoulder surfing attacks are not very common as humans are very good at detecting people in their personal space [50]. Eiband et al. find that shoulder surfing mainly occurs in an opportunistic, non-malicious way and it is usually personal data that is observed [43].

2.5.23 Storage

Store password hashes Four pieces of advice recommended storing passwords as hashes. A cryptographic hash is a bit string of a fixed size which should uniquely represent a password. It is irreversible but is deterministic i.e. the same password will always map to the same hash. An attacker can discover the passwords by creating a large look up table (rainbow tables) which matches each password to a corresponding hash value. This is very effective for passwords up to a certain length and after this point a brute force search is still possible [51].

Two of the pieces of advice recommending hashes also recommended using a salt. A salt is a large random, non-secret value which can be stored with the password hash. The salt randomizes the output of the password hash, making the use of rainbow tables infeasible. An guessing attack is necessary for revealing each password.

One piece of advice relating to salting said to use "a unique salt for each account". The second salt related piece of advice did not specify whether the same salt could be used for all accounts or whether a unique salt was to be

used. If the same salt is used then a look up table is still an advantageous attack.

Florêncio et al. provide a decision tree in their paper "An Administrator's Guide to Internet Password Research" showing the attacks different methods of password storage are susceptible to [51].

For our evaluation of the costs and benefits of this advice we assumed that passwords would be both hashed and salted.

Encrypt passwords Encryption of passwords was the first common method recommended as a means of protecting passwords [106]. However it is often reversible and a key needs to be protected for security to be maintained. It is recommended that the hashing and salting method is used for password storage. Despite this, seven pieces of advice recommended password encryption, in comparison to only four pieces of advice recommending hashing and, of those, only two recommending hashing and salting. In the case of a password leak, either the key is revealed along with the password database or it is not. If the key is revealed all passwords are immediately decipherable. If the key is not revealed then brute force guessing can be attempted against the decryption key but this is a difficult task.

Don't hardcode passwords Hardcoding passwords will make it very difficult to automate changes making it a less usable method. Hardcoding will also make it less secure since it is stored in plaintext to a file that an administrator can directly edit.

Access to password file and encrypting password file Advice recommended encrypting password files and restricting access to those files. Not allowing all employees access to the passwords of all users seems like intuitive protections but only 1 source mentioned encrypting files and only two mentioned implementing access controls for the password file.

2.5.24 Throttling

Throttle password guesses To fight against online guessing we can have a fixed or exponentially increasing delay after each failed authentication attempt. For example after 10 incorrect guesses the account is locked for 24

hours or until it is unlocked by an administrator. Throttling (or rate limiting) password guesses drastically reduces the number of guesses an attacker can make. The attacker can no longer continuously make guesses until the correct password is accepted. However, because of the right skewed nature of password distribution, the attacker does still have a high probability of success with a small number of guesses [98, 115]. The cost of throttling could be the unintended lock out of a legitimate user. For example, Brostoff and Sasse [18] find that with a three strike system 31% of users are unfairly locked out and with a ten strike system it is 7%.

2.5.25 Transmitting passwords

Four pieces of advice recommend not transmitting passwords in plaintext. Two pieces of advice say to request passwords over a protected channel. These are very similar pieces of advice since both groupings place the onus on the organisation to oversee that passwords are sent by protected channels. If passwords are transmitted in cleartext they are susceptible to any eavesdropper on the network. Let's Encrypt has in recent years helped to make security certificates accessible to more websites [92]. However, providing a secure channel for passwords is not an easy task. In a 2017, two hour long lab study, 18.5% of their knowledgeable participants failed to set up a secure HTTPS connection [89].

2.5.26 Two factor authentication

Use-multi factor authentication Multi-factor authentication traditionally involves: *something you are*, *something you know*, and *something you have*. Examples for each of these respectively are: fingerprint, password, and a USB key token. One piece of advice recommended using multi-factor authentication. Using *something you have* as one of the requirements for authentication means that the user may need to carry an additional item around with them. In addition, this item (unlike *something you know* or *something you are*) is easily susceptible to theft. If theft does occur though, the user is still nearly as secure as if the second factor had never been used.

Notably, respondents in our user study expressed an approval for two factor authentication but only for use with certain accounts. One respondent said "I like this feature for anything related to monetary transactions".

Use two factor authentication on phone Using a phone would count as the *something you have* form of authentication. One piece of advice recommended this. In 2017, Peeters and Grenman introduced n-Auth [130]. N-Auth is an authentication solution designed to work with any number of accounts and using a users' mobile device.

Use for remote accounts One piece of advice said to use multi-factor authentication for remote accounts. Remote accounts are often more vulnerable as the user might need to connect over an insecure channel. Therefore, the multi-factor authentication process is seen as adding extra security to the account. It is also more likely that the second factor will be compromised if used remotely; either by theft or by eavesdropping.

2.5.27 Username

Enforce composition restrictions on usernames Florêncio, Herley and Coskun argue that it is better to increase the strength of the userID rather than the password [50]. They propose that this will protect against online guessing attacks but will not majorly increase the cost to users since the username can be recorded visibly.

In our user study, six of eight users asked about this advice said they did not approve of it. One user commented that it was "Stupid and pointless" and another said "Can't see a good reason for this, but with a password manager it wouldn't be too painful.". This emphasises the importance of explicitly explaining rationale when advice is passed onto users.

Don't reuse username If the same username is used for multiple accounts then once the password for one account is compromised, this password can be tried against the same person's other accounts. Das et al. find that 3% of users directly re-use passwords between sites and many others introduce small modifications to their passwords across sites [34]. Not reusing a username could be one way to protect against an attacker leveraging this vulnerability and could be less burdensome on the user than a restriction on altering and reusing passwords.

2.5.28 Summary

We have now finished our discussion of the 27 advice categories which contain the 270 pieces of authentication advice we collected. We acknowledged in our discussion that some of the advice collected contradicts what security experts and researchers consider to be best practice. Similarly, we made note of the large cohort of advice that was contradicted by different sources. This shows a lack of consensus between organisations about what advice is effective. We believe that if users are receiving contradicting advice from different sources then this is likely decreasing users' confidence in the advice they are given. This, as well as an imbalance between usability costs and benefits, is likely a reason for users' unwillingness to follow many pieces of security advice.

In the next section, Section 2.6, we will be delving deeper into the usability and implementation costs that accompany this advice. We will also be reporting the verdict of users and administrators regarding whether they approve of these pieces of security advice. In section 2.8, we will be discussing the security benefits of this advice. This will give us a clearer idea of the cost and benefit trade-offs that exist for each piece of advice. This trade-off will be discussed in Section 2.9.

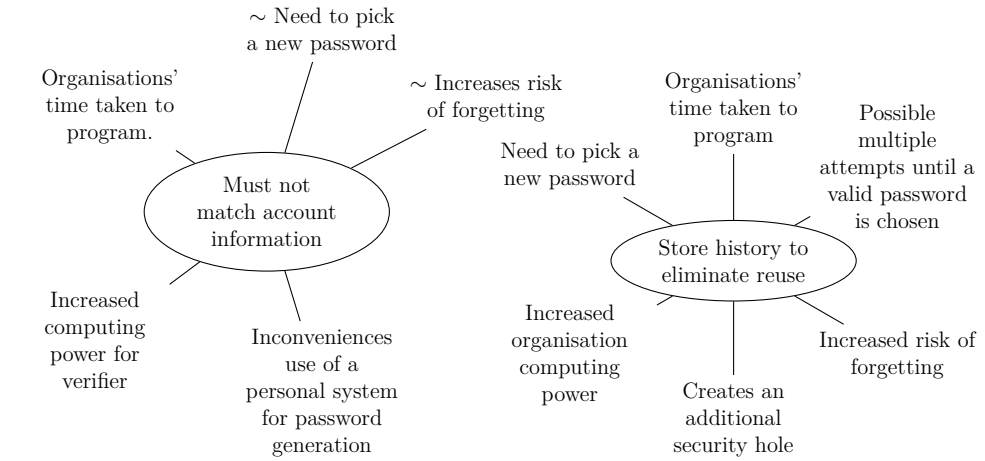
2.6 Costs Model

In this section we describe our methodology for the creation of our costs model. We began with a brainstorming exercise. At the heart of it was a conscious consideration for the usability costs; costs which are often-overlooked when security policies are implemented [33]. Table 2.4 shows our finalized cost categories.

We begin by presenting our initial identification of categories of costs and then discuss their refinement. This involved analysis of severity and frequency of costs by surveyed systems administrators and end-users.

2.6.1 Identification of Cost Categories

To begin, we considered any costs that we could think of that related to the implementation of each piece of advice we collected. We viewed costs as any burdens on the user who must follow the advice, or any burdens on the organisation who must either follow or enforce the advice. Figure 2.2 shows



(a) Personal Information: Must not match account information (b) Expiry: Store history to eliminate reuse

Figure 2.2: Identifying costs for advice statements.

two examples of this brainstorming exercise for the advice statements "Must not match account information" and "Store [password] history to eliminate reuse". After analyzing the 79 advice statements in this way we had identified 10 categories of costs. These initial cost categories were:

1. Increased risk of forgetting
2. Need to pick a new password
3. Possible multiple attempts until a valid password is chosen
4. Reduced search space for attacker guessing
5. Company/organisation's time taken to enforce
6. Hard or impossible to enforce
7. Inconveniences a user's personal system for generating passwords
8. Created an additional security hole
9. Increased Computing power needed
10. User time or inconvenience.

2.6.2 Refining cost categories

These gave us our initial categorization of what costs come into play when an organisation or user wishes to obey or enforce password advice [114]. We then went through a process of refining these categories. In the following

subsections, we explain the process of finalizing these cost categories. Our final set of costs categories is shown in Table 2.4.

2.6.2.1 Sub-costs

We notice that some categories are outcomes of others. For example, *increased risk of forgetting* is an obvious outcome of *needing to pick a new password*. Similarly, many categories can be seen to have "sub-costs" which we have not explicitly listed in our categories. For example, *increased risk of users abandoning site* and *increased number of password resets* are sub-costs of *increased risk of forgetting*. To minimize overlap we chose the highest-level categories that could occur without being a sub-cost. For example, *user time* remained as a distinct category as at times we recognize that there is no cost other than the users' time. Whereas, we removed the category *possible multiple attempts to choose a valid password* as it never occurred other than as a sub-cost of *need to pick a new password*.

2.6.2.2 Removal of cost categories

Three of our cost categories did not relate to our definition of costs as burdens on the user or organisation. These were: *creates an additional security hole*, *reduced search space for attacker guessing*, and *impossible to enforce*.

Creates an additional security hole refers to advice that while it may protect against one thing, opens up another avenue of attack. For example, offering to display the password at login makes it easier for the user to enter the password correctly but increases the chance of a shoulder surfing attack.

Placing a restriction on the maximum number of characters allowed in a password is an example of advice which could *reduce the search space for an attacker guessing*. For example, if only eight-character passwords are allowed then an attacker has a smaller search space.

Both of these categories relate more to the chance of attacks occurring than to particular burdens on the user or organisation. Therefore, we consider both of them in light of the benefits rather than costs. Benefits of password advice are discussed in Section 2.8.

Impossible to enforce refers to any advice that an organisation distributes but

cannot enforce. This can lead to an illusion of higher security than actually exists. The term 'security theatre' was coined in 2003 by Bruce Schneier [19] and could apply here. The organisation may feel they are secure with multiple authentication procedures in place. But if these are in no way enforceable then they have little guarantee of offering any improvement to security.

A piece of advice that is *Impossible to enforce* only has costs and benefits if it is voluntarily followed. An interesting question is; what is the probability optional advice such as this is followed, and what is the impact of user education on its uptake? We remove it as a cost category as it does not necessarily introduce a 'burden' on either the user of the organisation.

2.6.2.3 Additional cost categories

There was one cost which when we looked back we noticed we had overlooked. It was the cost of additional physical resources that might be needed by the user or the organisation in order to follow the advice. The prime example of this is two factor authentication which might require a specialized USB device or token. We included this need for additional resources as an additional item in our list of costs.

2.6.2.4 User versus organisation costs

Analyzing the cost categories, we wanted some way to distinguish between costs borne by the user and costs borne by the organisation. We believe it will be interesting to know who bears most of the costs; the user or the organisation. In addition, what might be a small cost for an organisation could be a large cost for a user. We therefore separated cost categories to be user and organisation specific e.g. *user computing power* separate from *organisation computing power*.

2.6.2.5 Minor costs

In our analysis we acknowledged the need to distinguish the extent to which a cost occurs. For example, both "Enforce restrictions on characters in passwords" and "Make digital and physical back ups" require organisation time. But very little organisation time is needed to enforce a composition restriction and a lot of effort is needed to digitally and physically back up work. It is

Table 2.4: Finalized cost categories.

Organisation Costs
Increased help desk/user support time
User education required
Organisation needs extra resources
Takes organisation time to implement
Increases the organisation's computing power needed
User Costs
Makes it more difficult to create a password
Makes it less easy to remember
Requires extra resources
Requires the creation of a new password
Increases the computing power needed
Requires other extra time or effort

not within the scope of this model to have a full grading of the costs, but we do acknowledge a difference between small or partially felt costs and more substantial costs.

2.6.2.6 Periodic costs

We also notice that advice such as "Make digital and physical back ups" and "Keep email up-to-date and secure" require continued action. This significantly increases the costs. We acknowledge three types of costs: once off costs usually relating to setup or account creation, costs which occur at every login and periodic costs which occur repeatedly over different time frames.

2.6.2.7 Positive costs

Finally, some pieces of advice reduced the burden on the organisation or user. These pieces of advice we acknowledge as 'positive costs'.

The costs listed in Table 2.4 are the finalised 11 cost categories that were used in our user study.

2.7 User study of costs

Once satisfied that we had a provisional set of cost categories for users and for organisations we began our user study. There were 10 surveys in total, five

aimed at administrators and five aimed towards end users. These are detailed below.

2.7.1 User and administrator surveys

We created a series of surveys which asked users and administrators to identify which categories of costs apply to each piece of advice. Participants were randomly redirected to one of the 5 surveys relevant to them. This allowed us to ask about the large number of advice statements we had collected without overburdening participants. We did include some overlap in the end-user surveys as every user was asked about:

- Include specific character types in your password. E.g. your password must include uppercase, lowercase, digit, symbol
- Change your password regularly
- Never reuse a password
- Never share your password
- You cannot paste in your passwords
- Use 2-factor authentication
- Use a password manager

There was no overlap in the administrators' surveys as they were asked a larger variety of questions. For example, administrators were quizzed on costs associated with hashing and salting passwords and monitoring and analysing intrusions which users were not asked about.

Participants were chosen using a snowball sampling technique. The study was approved by our university Ethics Review Board. For a full description of the ethical considerations, please see Section 1.3.2.

In the survey, participants were asked to specify the severity and frequency with which they experience costs and inconveniences as a result of authentication advice. We also asked participants to indicate whether they approved of the given piece of advice. Figure 2.3 shows an infographic providing users with an explanation of how to complete a survey question. Beside it, users were given the following example text:

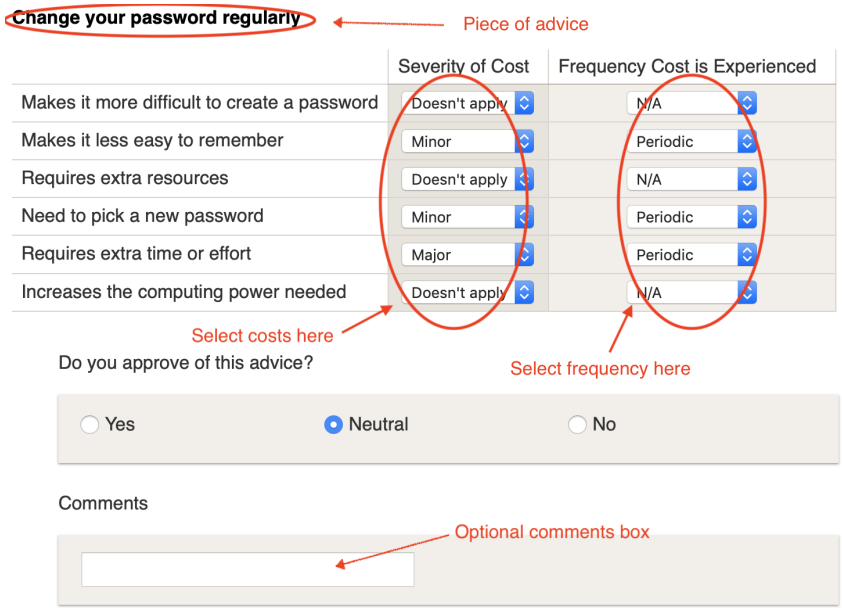


Figure 2.3: Infographic with instructions for users for the survey

For example: for the advice "Change your password regularly" you might consider the following costs:

- *Makes it less easy to remember (periodically - every time I need to change it)*
- *Need to pick a new password (periodically - every time I need to change it)*
- *Takes extra time (periodically - sometimes I can't start work until I change the password)*

You could complete the question for this advice as shown in Figure 2.3.
Note these costs are individual and may differ for you and you are not required to offer explanations.

Administrators were given a similar description. A full version of the questions and answers for the user survey [113] and administrator survey [112] is available on Github.

2.7.2 Feedback on user cost categories

After users had attempted to assign cost categories to the advice statements, they were asked at the end of the survey whether they agree with the cost categories that were used in the survey.

For the user survey the prevailing answer was between 'Somewhat' and 'Yes'. In summary, 21 end-users said Somewhat, 18 said Yes and 1 said No.

The user who chose 'No' said: "In most cases the categories seemed not applicable to the points so most were of no cost to me". This is a valid point, for many advice statements there was only one cost category that users deemed to apply, and for some advice such as "Every user in an organisation must have their own account", the majority of participants decided that there was no associated user costs.

After asking participants whether they agreed with the cost categories, we asked whether there were any cost categories that they think should be added or removed. We received the following suggestions. One person said we could remove computing power. One person said a cost could be that the advice "decreases sense of security in password". Another participant suggested maybe including the cost category "makes it harder to follow other advice". They gave the example that needing to have long passwords would make it more difficult to not write them down somewhere. Finally, one participant suggested that there is a "cost to personal stress related to constantly interacting with devices that require various different logins and passwords and eat up time and energy". All 34 other participants suggested no changes.

We did consider removing the user computing power cost, but in the end decided that in some areas it might be relevant. For example, for the pieces of advice "Keep anti-virus updated" and "Keep software updated" the majority of users said that computing power was a minor cost. This can relate to a slow-down of computing processes during an installation or waiting for restarts during updates.

The ability of a participant to follow multiple pieces of advice simultaneously is very important to consider. For example, Florêncio et al. show that never reusing a password and also random password choice is an impossible task outside the bounds of human memory [52]. In fact, many respondents in our survey made the point that the advice "Never reuse a password" is impossible to uphold unless it is coupled with the use of a password manager. For this reason, to really get a sense of the value or effect of a piece of advice it is important to consider it in light of a complete security policy. This is what we do when we quantify the value of advice in Chapter 5.

A yes/no answer system would have made the survey easier for participants to understand and simplify the completion. We were eager to get information about severity and frequency, but in retrospect a simplified version of the survey should have been considered. We did also notice that their was misunderstanding among participants about the meaning of the 'Positive costs'. We think a different term should have been used to describe these that was more self-descriptive. Based on the answers, we believe some participants indicated 'positive' when they were 'positive' there was a cost there. These participants were always the minority so largely did not affect the results.

Finally, the stress associated with current password systems is exemplified by the respondents' encouragement for "costs to personal health" to be included as a category. When we consider a security system and speak about usability, it is mitigating this stress and pressure that users are burdened by that we want to achieve. We believe this stress is a result of the human effort and mental strain that is encompassed within the existing categories.

2.7.3 Feedback on organisation cost categories

As with users, after administrators had attempted to assign organisation cost categories to the advice statements, they were asked at the end of the survey whether they agree with the cost categories that were used in the survey.

The majority of administrators agreed with the five cost categories that were used to denote organisation/administrator costs in this survey. However, nearly as many said they 'Somewhat' agreed. 13 administrators said Yes, 12 said Somewhat and 4 said No.

The most common comment we received was that user burdens were not included as a cost category. It is likely that administrators were not aware that a second survey existed aimed at users and the burdens they experience. However, the fact that many administrators insisted that user burdens must be taken into account is reassuring. It shows a changing in the ethos within security development and indicates that there could exist a changing perspective which is no longer viewing users as the enemy [1].

One respondent commented that the 'resources' cost was not specific enough. From feedback we received during the survey, we acknowledged that a misunderstanding existed with the term 'additional resources needed'. Some re-

spondents viewed resources as additional personnel needed whereas we were referring to resources as physical purchases required. We categorised a need for additional personnel under increased help desk/user support time and/or time take to implement. We did include a clarification for this partway through the study stating that "Resources refers to physical resources that may need to be acquired or purchased". After this clarification the responses seemed more aligned with this definition. For this reason though, a resource cost is sometimes identified where it does not seem clear to us why it should be. We do leave it in as significant increases in necessary personnel and other forms of resources are forms of valid burdens for an organisation.

One participant mentioned that "Everything has a cost sometimes it is small, but in may areas questioned in this survey, the benefit outweighs the cost". This a nice indication that this comparison of costs versus benefits of security advice is something that security administrators are required to do mentally for each policy consideration.

Two participant mentioned other costs. One said that "Some things need more than just help desk resources — development time, administration, auditing". The second participant said "There is more than just training and help support in terms of cost. Their is also engineering cost, audit cost, risk assessment cost, employee quality costs (not all employees can implement the policies discussed here in an enterprise environment). Technology cost. Enforcement cost. Incident response cost (for policy violations)". These are all interesting areas for consideration. We would have viewed all of these under the organisation time taken to implement the policy, though employee quality costs is not something we considered (e.g. the cost of recruiting and retaining highly skilled employees needed to implement advice).

Finally, one respondent mentioned that the extent of costs can depend strongly on the ethos of a company. They state that "Having a business leadership team that fully supports IT Security and is prepared to champion it will for example make the initial and ongoing "cost" in terms of resources much easier. Many of the things here will depend not just on technology but the culture and maturity of the organisation." This alludes to feedback we received throughout the survey and directly from some respondents: the costs experienced differ for each organisation. While the general costs felt might be similar, the extent

to which they are felt and the difficulty involved in implementing advice will strongly depend on the ethos, size and sector or services of the company. This model only provides a broad indication of the categories of costs felt by the majority. In Chapter 5, when we quantify these costs, there is a strong relationship between the costs felt and the type of organisation we consider.

2.7.4 Visualisation of survey responses

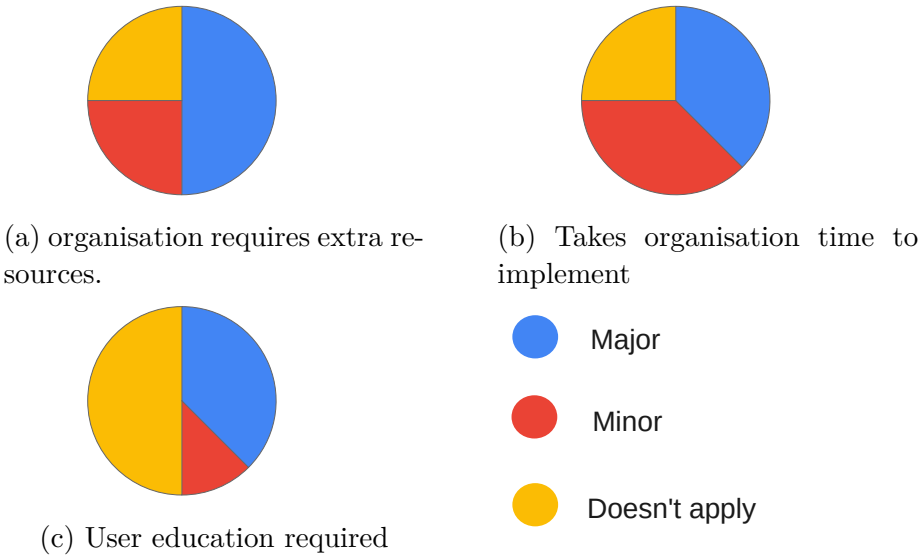
We received 41 participants for our end-user survey and 32 participants for our administrator survey. We received insightful comments about the costs from both a user and administrator perspective. A minimum of eight end-users indicated the costs they associated with each piece of advice and a minimum of five administrators indicated the organisation costs they perceived.

In Table 2.5, we detail the costs that users and administrators identified for each advice statement. A minor cost is identified as ○, and a more substantial cost is identified as ●.

A super-scripted symbol is used to denote the frequency of the costs. Costs at login are denoted by an at symbol: $\bullet^{@}$ and periodic costs are denoted by a sun: $\bullet^{\odot}$. 'Positive costs' are represented with the positive symbol: +.

The tables summarize the costs that respondents identified for each piece of advice. We use the following rules for identifying what costs the participants agreed exist for each piece of advice (i.e. did participants believe a cost major, minor, positive or doesn't apply):

- If one answer option accounts for more votes than any other then it is the majority answer. For example, in Figure 2.4a most respondents in the administrator survey indicated that the *organisation requires major extra resources* to enforce the advice "Digital and physical backups of work should be maintained".
- If two answers are selected with equal frequency we choose the one that represents the largest cost but note with an underline that there was variability in the responses, e.g. $\underline{\bullet}$. This means we are choosing to overestimate rather than underestimate costs. See Figure 2.4b which shows that for the piece of advice "Digital and physical backups of work should be maintained", 37.5% of respondents believed it has a minor



(a) organisation requires extra resources.

(b) Takes organisation time to implement

(c) User education required

Figure 2.4: Pie charts showing responses for three different cost categories for the advice "Digital and physical backups of work should be maintained"

affect on *takes organisation time to implement* and 37.5% believed it was a major cost. In this case we represent it as a major cost.

- If 'Doesn't apply' is outweighed or equaled by minor and major combined then we assign it as minor.[3] We again indicate that there is variability. In Figure 2.4c, we see that for the cost category *user education required* half the users said non-applicable. But half said that it was either a major or minor cost. Therefore, it is marked as a minor cost.

In Table 2.5 you can see these costs represented in the table under the category "Backup work" for the piece of advice "Digital and physical backups of work should be maintained".

Users and administrators were also asked about their approval of the advice statements. The results of this are represented in the table for both users and administrators. A ✔ represents the majority of respondents' approval of the advice. ✘ indicates that the majority disapproved and ■ represents the majority of respondents indicating that they were neutral about the advice. Occasionally, respondents were split between different approval ratings. This is

[3]Note that "Doesn't apply" was the default response for the survey questions.

also indicated in the table. For example, if users were split between approving of the advice and feeling neutral about the advice, we represent this as ✓/■.

In Table 2.5 we represent the costs that users and/or administrators associated with each piece of advice. In some cases, users or administrators were not always consulted for every piece of advice. This was an attempt to reduce the size of the survey. In these cases, we highlight the corresponding row in the table (see the organisation side of the row "Enforce maximum length"). At times we indicate obvious costs if they exist. For example for the "Enforce maximum length" advice, the administrative costs could be extrapolated from the "Minimum password length" costs.

Further impressions, characteristics and notable respondent comments are included in Appendix B. A discussion of the costs identified is included in our discussion section, Section 2.9.

Table 2.5: Costs of implementing password advice

	Organisation						User costs						
Advice to Users	Increased help desk/user support time needed	User education required	Requires extra resources	Takes time to implement	Increased computing power needed	*Administrator approval*	Makes it more difficult to create a password	Makes it less easy to remember	Requires extra resources	Requires the creation of a new password	Increases the computing power needed	Requires other extra time or effort	*User approval*
Backup password options													
Email up-to-date and secure						✔						○☼	✔
Security answers difficult to guess	○☼	●☼	○	○		✔	○	○				○	✓/■
Do not store hints	○☼	○☼		○☼		✔		●@				○	✔
Composition													
Include specific character types	○	○☼		○		✓/■	○	○@				○	✔
Keep your accounts safe													
Check web pages for TLS	○☼	●☼				✔						○@	✔
Manually type URLs	○☼	○☼				✘			○			●☼	■
Don't open emails from strangers	●☼	●☼	●☼	○☼		✘							✔
Keep software updated	○☼	○☼	○☼	●☼		✔					○	○☼	✔
Keep anti virus updated	○☼	○☼				✔			●☼		○☼	○☼	✔
Log out of public computers		●☼				✔						○@	✔
Password protect your phone	○☼	○☼		○		✔		○	○	○		○@	✔
Length													
Minimum password length	○	○		○		✔	●	●@		○			✓/✗
Enforce maximum length (<40)	○	○		○			○						✘
Password managers													
Use a password manager	○	○☼	○	○		✔		+	○				✔
Create long random password	○	○		○		✔		●					■
Personal information													
Don't include personal information		○☼						○@					✔
Must not match account details	○	○		○		✔		○@					✔
Personal password storage													
Don't leave in plain sight		○☼				✔		○					✔
Don't store in a computer file	○☼	○☼	○☼			✔		○@				○	✔
Write down safely		○☼						+	○			○	✔
Don't choose "remember me"		○				✓/■	○	●@				○@	■
Phrases													
Don't use patterns		○☼		○			○	○@					✔
Blocklist common password		○☼		○		✔	○	○		○		○	✔
Don't use published phrases		○☼		○			○	●@		○		○	✔
Substitute symbols for the letters		○☼					○	○@					■
Don't use dictionary words	○☼	○☼		○		✔	●	●@	○			○	■

●, filled circle: major cost. ○, empty circle: minor cost. +, plus: positive cost.
@, superscript @: cost occurs at each login. ☼, superscript ☼: cost occurs periodically.
underline: implies that variance existed in the costs that respondents indicated.
✓/■: approval split between Yes and Neutral.

	Increased help desk/user support time needed	User education required	Requires extra resources	Takes time to implement	Increased computing power needed	*Administrator approval*	Makes it more difficult to create a password	Makes it less easy to remember	Requires extra resources	Requires the creation of a new password	Increases the computing power needed	Requires other extra time or effort	*User approval*
Reuse													
Never reuse a password		●☼				✔	●☼	●@		●☼			✔
Alter and reuse passwords	○☼	○☼	○	○		✔	○☼	○☼					✘
Don't reuse certain passwords		○☼				✔	○☼	●		○☼			✔
Sharing													
Never share a password	○	○☼		○☼		✔							✔
Don't send passwords by email		○☼				✔							✔
Don't give passwords over phone	○	○☼				✔			○				✔
Two-factor authentication (2FA)													
Use 2FA using app or special device	●☼	●☼		●		✔			●@			○@	✔
Use 2FA on phone	●	●☼	●	●	○	✓/■			●@		○		✔
Use 2FA for remote accounts	○☼	○☼	○☼	●	○	✔			○@			○@	✓/✗
Username													
Enforce restrictions on characters	○☼	○☼		○		✔							✘
Don't reuse username		○				✘		●	○			●☼	■/✗

Advice to organisations

Administrator Accounts													
Not for everyday use	○	○☼	○	○		✔							
Must have it's own password	○☼	○		○		✔							
Should have extra protection				○		✔							
Backup work													
Make digital & physical back-ups	○	○	●	○	○	✔			●☼		○☼	○☼	✔
Default passwords													
Change all default passwords	○☼	●	○	○		✔				●			✔
Expiry													
Store history to eliminate reuse		○	○	○		✔	○☼	●☼		●☼			✔
Change your password regularly	●☼	●☼		○		✘	●☼	●☼		●☼		●☼	✘
Change if suspect compromise	○☼	○☼	○	○		✔		○		○			✔
Generate passwords													
Use random bit generator	○		○	○	○	✔		●@					✔
Must aid memory retention						✓/■	●	○@	○	○			■
Must be issued immediately	○					✓/■		●	○		○		✔
Distribute in a sealed envelope				○		✔		○@		○		○	✔
Only valid for first login	○	○		○☼		✔				○			✔

●, filled circle: major cost. ○, empty circle: minor cost. +, plus: positive cost.
@, superscript @: cost occurs at each login. ☼, superscript ☼: cost occurs periodically.
underline: implies that variance existed in the costs that respondents indicated.
✓/■: approval split between Yes and Neutral.

	Increased help desk/user support time needed	User education required	Requires extra resources	Takes time to implement	Increased computing power needed	*Administrator approval*	Makes it more difficult to create a password	Makes it less easy to remember	Requires extra resources	Requires the creation of a new password	Increases the computing power needed	Requires other extra time or effort	*User approval*
Individual accounts													
One account per user	○	○		○	○	✔							✔
Each account password protected	○					✔		○		○			✔
Input													
Don't perform truncation			○	○		✓/■							
Accept all ASCII characters	○	○		○		✔	+						
Keep system safe													
Implement Defense in Depth	●☼	●☼	●☼	●☼	○	✔							
Implement Technical Depth		○	○	○	○	✔							
Apply access control systems	●☼	○☼	○☼	●☼		✓/■			○			○	✔
Monitor and analyze intrusions	○☼	○☼	●☼	●	○☼	✔							
Regularly apply security patches	○	○☼	●☼	●☼	○☼	✔							
Network: SNMP community strings													
Don't define as standard defaults	○	○		○		✓/■							
Different to login password				○		✔							
Password auditing													
Attempt to crack passwords	○☼	○☼	○☼	○☼	○☼	✔	○	○		●		○	✔
Policies													
Establish clear policies	○	○☼		○		✔							
Shoulder surfing													
Offer to display password		○		○		■							✔
Enter your password discretely		○					○					○	✔
Storage													
Encrypt password files		○		○	○	✔							
Restrict access to password files	○☼	○☼		○	○	✔							
Store password hashes			○	○	○	✔							
Encrypt passwords	○		○	○	○	✔							
Don't hardcode passwords	○	○☼	○	○		✔							
Throttling													
Throttle password guesses	○☼	○☼	○☼	○		✔		○				○	✔
Transmitting passwords													
Don't transmit in cleartext	○☼	●☼	○	○		✔							
Request over a protected channel	○☼	○☼	○	○		✓/■							
Don't allow users to paste passwords	●☼	○☼		○		✘		○@				○@	✘

●, filled circle: major cost. ○, empty circle: minor cost. +, plus: positive cost.
@, superscript @: cost occurs at each login. ☼, superscript ☼: cost occurs periodically.
underline: implies that variance existed in the costs that respondents indicated.
✓/■: approval split between Yes and Neutral.

2.8 Benefits Model

Now we ask the questions: What are the benefits of this advice? What goal is the advice trying to achieve? From our analysis of costs, it is evident that advice is generally not attempting to make authentication more usable. It is our understanding that it is trying to improve security. In this section we give a method to identify such improvements, which we call benefits.

2.8.1 Defining Benefit Categories

It is broadly believed that advice is attempting to guide the creation of robust authentication systems with the goal of preventing unauthorized access to an account or to data [6, 49, 66, 148]. This is what we define as the goal of the advice. Therefore, the benefit of a piece of advice is the effect it has towards achieving this goal.

Unauthorized access has many avenues and one piece of advice is unlikely to protect against them all. For example, requiring passwords of length 12 might help against a password guessing attack, but will do nothing to protect against phishing. In order to represent how 'beneficial' a piece of advice is, we would like to know in which ways the piece of advice protects against unauthorized access, if at all.

To this end, we require a concise list of the different methods an adversary could use to gain access to a protected account. The NIST 2017 Digital Identity Guidelines [59] document includes a table of "Authenticator Threats". Given that this NIST 2017 authentication document has been well-regarded [32] and highly researched document, we accept these as a concise list of the attack vectors against authentication. Below, in our Table 2.8, we list the attacks from Table 8-1 in the Authentication and Lifecycle Management volume of the NIST 2017 Digital Identity Guidelines. Included is the name, details and example for each different attack type. In places we diverge slightly from the official description and/or example provided in the NIST document and instead indicate our interpretation of the threat.

We then considered the threats addressed by our advice statements. All the threats mitigated or created by each piece of advice fit under these 11 categories, except the threat mitigated by "Log out of public computers". If a user does not log out of a public computer then the only protection a user has is

the moral compass of the next person who uses that computer. In this way a user's account can already be thought of as in a state of compromise. It was not obvious which of the 11 NIST categories this opportunistic attack comes under. As a stopgap we placed it as Eavesdropping.

Table 2.8: Attack types on authentication

Attack type	Description	Examples
Assertion manufacture or modification	The assertion is used to communicate the result of the authentication process from the verifier to the Relying Party (RP). The verifier and the RP may be the same entity, or separate entities.	Use of assertion replay to impersonate a valid user or leakage of assertion information through the browser [153].
Physical theft	A physical authenticator or physical device used in the authentication process is stolen by an attacker.	A hardware cryptographic key (e.g. a USB authenticator), a phone, computer or one-time-password device is stolen.
Duplication	The subscriber's authenticator has been copied with or without their knowledge.	Passwords or private key written on paper or stored in an electronic file are copied. A counterfeit biometric authenticator is manufactured.
Eavesdropping	The authenticator secret or authenticator output is revealed to the attacker as the subscriber is authenticating.	Passwords are physically observed during keyboard entry or intercepted by keystroke logging software or recorded during transmission or via network packet sniffing.
Offline Cracking	An offline guessing attack is an analytical guessing attack by an attacker, it requires little to no communication with the system under attack.	A dataset of passwords or keys which are hashed and salted or encrypted are made available to an attacker (leaked). Using a dictionary or brute force guessing method the attacker attempts to guess the plaintext values of these protected secrets.
Side Channel Attack	This attack leverages an aspect of the implementation of the computer system or security device.	An attacker exploits information about a cryptographic key gathered from power, timing or audio data.
Phishing or Pharming	Fooling the subscriber into thinking the attacker is a verifier. Phishing: electronic communication masquerading as the verifier. Pharming: directing a website's traffic to a masquerading fake site.	A password or key is revealed by a bank subscriber in response to an email inquiry from a phisher pretending to represent the bank. A password is revealed by the subscriber at a fake verifier website reached through DNS spoofing. [22, 5].
Social Engineering	The attacker establishes a level of trust with a subscriber in order to convince the subscriber to reveal their authenticator secret or authenticator output.	An attacker masquerading as a system administrator makes a telephone inquiry requesting the victim's password. Attacker convinces mobile operator to redirect the victim's mobile phone messages to the attacker.
Online Guessing	The attacker connects to the online server of the verifier and attempts to guess the valid authenticator output for one or multiple users.	An attacker who knows the usernames for all the accounts guesses the top ten most popular passwords in order to try to access them. An attacker who knows neither the username nor password guesses combinations of both to try to unlock the account.
Endpoint Compromise	Malicious code on the endpoint authenticates without the victim's consent, compromises the authenticator or causes authentication to other than the intended verifier.	Malicious code can steal data from the users' device. Malicious code can be used to conduct man-in-the-middle attacks on all connections. New trusted certificates can be installed on the users' device.
Unauthorized binding	An attacker is able to cause an authenticator under their control to be bound to a subscriber's account.	Forcing a password reset to change password to one the attacker knows. Accessing the password file and changing the password (or other authenticator) bound to the user's account. Creation of a second USB or linking another authentication method to a user's account.

2.8.2 Identifying Benefits of password advice

As with costs, which could be major or minor, the benefits provided by different advice also vary. The benefits of advice therefore should reflect the probability of each attack being successful. Given some baseline chance of each attack, we reflect on whether a piece of advice increases or decreases the chance that that attack is successful. We show the result of this analysis for our 79 advice statements in Table 2.9.

Throughout our table we mark each piece of advice as either increasing or decreasing the chance of compromise. However, there is an interesting question about what baseline the advice increases or decreases the chance of compromise relative to. We describe two mechanisms for determining this baseline.

We first thought of the baseline as being a passive version of the advice. For example, if the advice is: "Email up-to-date and secure" or "Encrypt passwords" then the baseline we measure the improvements from is doing nothing. This made sense for a lot of the advice statements, but not all. For example, the advice "Do not store hints" is already a passive action, but we do not want to measure its affects against itself.

Our second consideration for the baseline was the opposite of what the advice stated. For example, for the advice "do not store hints" we consider the difference in attack threat between this and "do store hints". Or maybe less strictly, "you are allowed to store hints".

We have chosen to use this 'opposite' method as it appears the logical comparison for most pieces of advice. The downside is that the opposite can, in a few cases, be too 'strict'. For example, the opposite of "do not include names" and "don't repeat characters" are "include names" and "repeat characters" respectively, both unnatural pieces of advice. Two pieces of advice had ambiguous opposites. The opposite of "write down safely" could either be "don't write down" or "write down, but not safely". In this case, we chose to consider the opposite as "don't write down". Similarly, we choose the opposite of "alter and reuse passwords" to be that reuse is allowed even without altering.

As said, for our purposes in this chapter we identify the increase or decrease in the chance of attack by comparing to the opposite of the specific piece of advice. However, in a real-world situation a proposed policy could simply be

compared in light of the former policy in place.

No matter what baseline method is chosen the discussions in Appendix C should be relevant.

2.8.3 Representation of password advice benefits in tables

We use the symbols ⇑ and ⇓ to indicate an increase or decrease respectively in the probability of success for the attack type. ↑ and ↓ indicate less significant increases or decreases in the probabilities of success for the attacks. An underline, $\underline{\text{ }}$, indicates that the improvement is not directly enforceable. This could be because it is impossible for an organisation to bind their users to following the advice. Or, simply, that it is too vague and therefore there is ambiguity on how an organisation or user might follow it.

In Appendix C, we include explanations for why we believe the chances of certain attacks are impacted by the advice.

Table 2.9: Benefits of implementing password advice

Advice to users	Assertion manufacture or Modification	Physical Theft	Duplication	Eavesdropping	Offline Guessing Attacks	Side Channel Attack	Phishing or Pharming	Social Engineering	Online Guessing	Endpoint Compromise	Unauthorized binding
	Attack Types										
Backup password options											
Email up-to-date and secure				↓			↓		↓	↓	
Security answers difficult to guess									↓		
Do not store hints					⇓			↓	⇓		
Composition											
Must include special characters					⇓				⇓		
Don't repeat characters					⇓				⇓		
Enforce restrictions on characters					↓				↓		
Keep your account safe											
Check web pages for TLS				⇓			↓				
Manually type URLs			⇑				⇓				
Don't open emails from strangers							⇓	⇓			
Keep software updated				⇓		⇓				⇓	
Keep anti virus updated										⇓	
Log out of public computers				⇓							
Password protect your phone.										⇓	
Length											
Minimum password length					⇓				⇓		
Enforce maximum length (<40)				⇑	⇑				⇑		
Password managers											
Use a password manager			⇑		⇓				⇓		
Create long random passwords					⇓				⇓		
Personal Information											
Don't include personal information					⇓				⇓		
Must not match account details					⇓				⇓		
Do not include names					⇓				⇓		
Personal password storage											
Don't leave in plain sight			⇓								
Don't store in a computer file			⇓								
Write down safely		↑	↑								
Don't choose "remember me"			⇓								

⇓ Decreases the probability of attack. ⇑ Increases the probability of attack.
↑ Minorly increases the probability of attack. ↓ Minorly decreases the probability of attack.
, underline: advice is impossible to enforce; it must be followed voluntarily or in a certain way.

	Assertion manufacture or Modification	Physical Theft	Duplication	Eavesdropping	Offline Guessing Attacks	Side Channel Attack	Phishing or Pharming	Social Engineering	Online Guessing	Endpoint Compromise	Unauthorized binding
Phrases											
Don't use patterns					⇓				⇓		
Blocklist common passwords					⇓				⇓		
Take initials of a phrase					⇓				⇓		
Don't use published phrases					⇓				⇓		
Substitute symbols for letters					⇓				⇓		
Don't use words					⇓				⇓		
Insert random numbers and symbols					⇓				⇓		
Reuse											
Never reuse a password					⇓				⇓		
Alter and reuse passwords					↓		↑	↑	↓		
Don't reuse certain passwords					↓		↑	↑	↓		
Sharing											
Never share your password				⇓				⇓			
Don't send passwords by email				⇓				⇓			
Don't give passwords over phone				⇓				⇓			
Two factor authentication (2FA)											
Use 2FA using app or special device		↑					⇓		⇓		
Use 2FA on phone		↑		↑		↑	⇓		⇓	↑	
Use 2FA for remote accounts		↑		⇑			⇓		⇓		
Username											
Enforce composition restrictions									⇓		
Don't reuse username					↓				⇓		

Advice to organisations

Administrator accounts											
Not for everyday use				⇓		⇓					
Must have it's own password				⇓		⇓					
Should have extra protection	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Backup work											
Make digital & physical back-ups		↑									
Default passwords											
Change all default passwords									⇓		
Expiry											
Store history to eliminate reuse		⇑			⇓				⇓		
Change your password regularly					↓				↓		
Change if suspect compromise					⇓				⇓		

⇓ Decreases the probability of attack. ⇑ Increases the probability of attack.
↑ Minorly increases the probability of attack. ↓ Minorly decreases the probability of attack.
, underline: advice is impossible to enforce; it must be followed voluntarily or in a certain way.

	Assertion manufacture or Modification	Physical Theft	Duplication	Eavesdropping	Offline Guessing Attacks	Side Channel Attack	Phishing or Pharming	Social Engineering	Online Guessing	Endpoint Compromise	Unauthorized binding
Generated passwords											
Use random bit generator					⇓				⇓		
Must aid memory retention					⇑				⇑		
Must be issued immediately			⇓								
Distribute in a sealed envelope		⇑	↑		⇓						
Only valid for first login			⇓								
Individual accounts											
One account per user			⇓				⇓	⇓		⇓	⇓
Each account password protected					⇓	↓			⇓		
Input											
Don't perform truncation					⇓			⇓	⇓		
Accept all characters					⇓				⇓	↓	
Keep system safe											
Implement Defense in Depth	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
Implement Technical Defenses	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
Apply access control systems	⇓		⇓	⇓		⇓	⇓	⇓	⇓	⇓	⇓
Monitor and analyse intrusions	⇓		⇓			⇓			⇓		⇓
Regularly apply security patches				↓		↓				⇓	↓
Network: Community strings											
Don't define as standard default									⇓		
Different to login password				⇓		⇓					
Password Cracking											
Attempt to crack passwords									⇓		
Policies											
Establish clear policies											
Shoulder surfing											
Offer to display password				⇑							
Enter your password discretely				⇓							
Storage											
Encrypt password files			⇓		⇓						⇓
Restrict access to password files		⇓			⇓						⇓
Hash and salt passwords					⇓						
Encrypt passwords					⇓						
Don't hardcode passwords					⇓						
Throttling											
Throttle password guesses									⇓		
Transmitting passwords											
Don't transmit in cleartext	⇓			⇓					⇓		
Request over a protected channel	⇓			⇓					⇓		
Don't allow users to paste passwords											

⇓ Decreases the probability of attack. ⇑ Increases the probability of attack.
↑ Minorly increases the probability of attack. ↓ Minorly decreases the probability of attack.
, underline: advice is impossible to enforce; it must be followed voluntarily or in a certain way.

2.9 Discussion

In this section we will discuss the characteristics we discovered within the costs and benefits of security advice. We begin with some general observations on the cost analysis in Table 2.5.

2.9.1 Costs discussion

The costs associated with implementing and following advice are shown in Table 2.5. In this table the mode user cost category was *increased risk of forgetting* and the mode organisation cost category was *user education required*. It is important to note that while organisation time is required to develop and run user education, user time is also necessary. Therefore, each time we see a *user education* cost, we recall that this involves another burden for users.

There were 212 costs identified across all the pieces of advice for the organisation and 114 costs identified for users. Of these the users identified that 26% of them were major costs. Administrators identified that 16% of the costs to the organisation were major. 26% of these major organisation costs were attributed to user education.

Of the 212 organisation costs, 45% were repeating costs, i.e. they were felt at each login or periodically. Of the users 114 costs 44% were repeating. However, nearly half (48%) of the repeating organisation costs were attributed to user education; a cost that will be felt by users as well. Similarly, of the user repeating costs 38% were related to the cost of forgetting as many attributed this as occurring at login. The cost of forgetting is a cost that is felt by the organisation as well; since the users' only solution is often to contact the organisation help desk.

Of our eleven cost categories, help desk time, time to implement, user education and all the user costs excluding computing power involve a human directly. In these categories, advice is adding to the emotional burden or workload of the individuals involved. These categories account for 85% of all the costs identified. In the user survey we witness strong emotional responses towards the advice we asked users to analyse. We received responses such as "I am sick of passwords and logins and they are making me less productive as I have to look up the passwords so often!!" and "The cost to personal health of

the stress of constantly interacting with devices that require various different logins and passwords-and eat up time and energy!!".

Users only agreed on two pieces of advice that have a positive impact on a cost category. These are: "Use a password manager" and "Write your password down safely". Both of these had a positive impact on *memorability*.

2.9.2 Approval of advice

For each piece of advice, we asked users and administrators to indicate whether they approved of the advice, felt neutral about it or disagreed with it. In Table 2.5 we indicate the opinion agreed by the majority of respondents.

In this section, we will analyse two areas of interest. First, we will investigate advice that we received unanimous opinions of. Second, we will investigate disagreements between users and administrators about what advice was good.

Unanimous approval of advice Below we show the pieces of advice that administrators unanimously agreed with:

- Minimum password length
- Keep anti virus updated
- Use a password manager
- When using a password manager create long random password
- Don't leave your passwords in plain sight
- Don't send passwords by email
- Administrator Accounts should have extra protection
- Change all default passwords
- Change password if suspect compromise
- Regularly apply security patches
- Encrypt password files
- Don't hardcode passwords

- Throttle password guessing
- Don't transmit passwords in cleartext

There was no piece of advice that administrators or users unanimously disagreed with. For each piece of advice, an average of 7 administrators gave their opinions.

Administrators only agreed unanimously about 14 of the 68 pieces of advice they were asked about; 21%. There were 20 pieces of advice that administrators were divided between yes and no on. The remaining pieces of advice had responses which included neutral responses.

More end-users responded to advice than administrators. Therefore, it is less likely that they all agree. In particular, for the seven pieces of advice given in Section 2.7.1, all 41 end-users gave their opinions. The following pieces of advice are those that end-users unanimously agreed with:

- Log out of public computers
- Don't send passwords by email
- Change password if suspect compromise

The end-user respondents gave vastly different approval ratings for the advice. Unanimous approval was only recorded for three of the 55 pieces of advice they evaluated (5%). Users were divided between yes and no (not just neutral) on 67% of the pieces of advice. We considered a misinterpretation of advice affecting this result but looking to the users' comments this contrast in approval was reinforced by vastly contrasting comments. For example, for the advice you should not be able to paste your password when logging in, one user said:

> "If someone is trying to copy/paste a password, they're probably beyond help."

While another user said:

> "This is horrendous advice that leads to problems using password managers. It encourages using crappy passwords."

These responses represent the disagreements we noticed between users for some pieces of advice. For this advice which discouraged allowing users to paste in their password when logging in, 14 end-users said they approve of the advice, 10 were neutral and 17 users disagreed with the advice.

We also considered whether the contrast in opinions could be a result of the high level of security knowledge some users had in contrast to others. Because we used a snowball sampling technique we could have attracted a large number of users who are security advocates. Take this response to the advice "Change passwords regularly" for example:

> "This [advice] is in conflict with evidence-based approaches to password security from NIST. Frequent password change requirements lead to people using worse passwords because they keep having to remember new ones. They will generally make minimal changes, and may be driven to record them somewhere insecure. This reduces security. It can cause upheaval with access issues across heterogeneous systems."

The user who gave this response indicated that their internet security knowledge ranked as 4 out of 5.

In Figure 2.5 we plot the number of users who assigned themselves to each Internet security rank. Users were told to "Select a rank on the below scale from 1 (very poor) to 5 (very knowledgeable) to describe your internet security knowledge". No users described their knowledge as very poor but after that we seem to have a reasonable spread across the remaining four ranks with an expected lower number of people indicating that they are 5 (very knowledgeable).

User versus administrator opinions At times we noticed that users and administrators disagree in their approval of advice. In Table 2.12 we list the pieces of advice where user approval contrasted with administrator approval. For each piece of advice we indicate the proportion of each group that agrees,

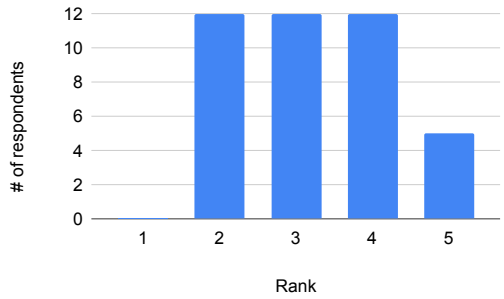


Figure 2.5: Internet security knowledge ranks from 1 (very poor) to 5 (very knowledgeable) self-assigned by user-survey participants.

disagreed and was neutral about the advice. Notice that for these pieces of advice, there was also differing opinions within the groups about whether the advice was creditable.

Administrators unanimously agreed with enforcing a minimum password length whereas users were more varied in their reaction to it. One user said that it "depends what it is protecting". Similarly for 2-factor authentication for remote logins, users were varied in their approval, whereas administrators were generally in favour.

Three of five administrators disagreed with not opening emails from strangers. But users were strongly in favour of it. A 2012 study by Böhme and Moore found that as a result of concerns over cybercrime 42% of participants said they do not open email from strangers [11].

Administrators were generally in favour with altering a password before reusing it at another site. But most users surveys did not approve of the advice. One user who disagreed with the advice said "Impossible to remember what password goes with what site". A different user who disagreed said "Any method to your password creation makes it less secure". Administrators gave no comments about why they agreed or disagreed with the advice.

The advice "Include specific characters in your username" somewhat divided administrator opinions and users strongly disagreed with it. In a 2007 research paper [50] Florêncio et al. determined that rather than making the secret password more complex, complexity could instead be added to the username. Users can choose and record a long username and this will likely offer effective

Table 2.12: Advice where users and administrators gave differing approval rating

Administrators					Users			
Decision	#Yes	#neutral	#No	**Advice**	#Yes	#neutral	#No	Decision
✔	6	0	0	Minimum password length	3	2	3	✓/✗
✘	2	0	3	Don't open emails from strangers	6	2	0	✔
✔	6	1	1	Use 2FA remote accounts	3	2	3	✓/✗
✔	4	1	1	Alter and reuse passwords	2	1	4	✘
✔	3	1	2	Include specific characters in your username	0	2	6	✘

The majority rating for each group is shown in the Decision column and the numbers indicating the Yes, Neutral and No option in both groups are also shown.

protection from a bulk guessing attack. The very attack that a complex or long password is hoping to achieve.

One administrator disapproved of this advice saying "good for passwords, over complicates usernames as password should be secure, usernames are often generated based on firstname.lastname etc". Two users commented on the advice. One said "Stupid and pointless" and another said "Can't see a good reason for this, but with a password manager it wouldn't be too painful". If this policy was being implemented for an organisation then these responses emphasise the need for it to be coupled with effective user-education and validation.

Summary Throughout the survey, user responses showed a strong sense of willingness to follow advice provided it seemed to have a benefit. For example, for the advice regularly change passwords one user commented: "Undoubtedly sensible, undoubtedly annoying". However, much of the advice that users viewed as effective has been shown by security researchers to have limited security benefits [24, 185]. While not in the majority, this was also true for a some of the administrator's responses.

There is substantial evidence in the responses (Appendix B) that user education has been effective, as users can echo the advice that was given in historic security documents [20]. However, very few users and administra-

tors showed an awareness of new recommendation which have superseded the legacy advice [59]. For example, the 2017 NIST advice recommends not introducing restrictions on the characters allowed in passwords. Yet, most users and administrators indicated that they agreed with forcing the use of certain characters in passwords. Note that this new NIST advice has been available and circulating since 2017. Our user and administrator survey took place in October 2020.

2.9.3 Costly advice

We want to consider what the most costly advice seems to be. Other than a ranked severity and frequency score, we do not at this time have the means to quantify the exact costs involved. To gather an impression of the costly advice we introduce an arbitrary scoring mechanism for each cost.

We assign 2 points to a major cost which reoccurs at each login, 1.5 to a major periodic cost, and 1 to a major one-time cost. A minor login cost is assigned 1, minor periodic cost 0.75, and a minor one time cost is 0.5. A positive cost can account for -2.

Using this elementary analysis, the six most costly pieces of advice are:

- Change your password regularly (score 9.5)
- Use 2-factor authentication on phone (score 7.5)
- Use 2-factor authentication using app or special device (score 7)
- Never reuse a password (score 6.5)
- Implement Defense in Depth (score 6.5)
- Attempt to crack passwords (score 6.25)

We can also see which pieces of advice are the most costly to end-users. To do this we can sum the scores from all the user cost categories and the user education category. This gives the six most user-costly advice as:

- Change your password regularly (score 7.5)
- Never reuse a password (score 6.5)

- Don't use dictionary words in your password (score 4.75)
- Use 2-factor authentication using app or special device (score 4.5)
- Store history to eliminate reuse (score 4.25)
- Don't use published phrases as your password (score 4.25)

Notice that the score for the never reuse password advice remains the same, as the organisation has no ability other than user education to enforce this cost.

These items may not be a surprise to those working in security operations. But it will be interesting to compare this to the benefits of these advice statements. Particularly since the majority of administrators and end-users deemed "Use 2-factor authentication using app or special device" and "never reuse a pasword" to be acceptable advice, and the only advice they disagreed with in these lists is "Change your password regularly".

2.9.4 Benefits discussion

Now we look at the benefits identified in Tables 2.9. Observe that overall, the advice does seem to decrease the chance of compromise. This should be unsurprising as this is, after all, the aim of authentication policies. However, there are some areas where there are increases in the chance of attacks. Eight advice statements have major negative benefits and six pieces of advice have minor negative benefits. That is, in some areas, these pieces of advice can increase the probability of compromise. The remaining 65 advice statements all show improvements for security.

2.9.4.1 Beneficial advice

In Section 2.9.3, we created an elementary scoring system to investigate which costs seemed to have the highest burdens on users and organisations. We can invoke a similar system here so get a sense of which pieces of advice are able to offer the most security.

For benefits we use the following rules: a large decrease in attack risk ($\Downarrow$) counts for 2 points, a minor decrease to attack risk ($\downarrow$) is 1 point. A small increase to risk ($\uparrow$) is -1 point and a large increase of risk ($\Uparrow$) is -2 points.

Using this methodology, the most security beneficial pieces of advice appear to be:

- Apply access control systems (score 12)
- Administrator accounts should have extra protection (score 11)
- Implement Defense in Depth (score 11)
- Implement Technical Defenses (score 11)
- One account per user (score 11)
- Monitor and analyze intrusions (score 10)

The advice that resulted in the greatest increase in risk was the advice Enforce maximum length (<40) which had a score of -6.

2.9.4.2 Attack protection

In the current model framework, it might not always be meaningful to compare the security impact of one piece of advice against another. Take, for example, one piece of advice which decreases the probability of compromise against one attack type, and a piece of advice that decreases the probability of compromise against three attack types. It is likely that the latter piece of advice is "better" but in reality, different attacks occur with higher frequency than others, and therefore protecting against one attack which occurs regularly might be more effective than protecting against three rare attack types. This leads us to an interesting question on the frequency with which the different types of attacks are successful, and whether there is more advice against the more frequent attack types.

Phishing, for example, occurs continuously, from targeted spear phishing attacks to mass phishing emails [5]. Whereas side channel attacks, while they attract interest from researchers, appear to have small real-world chance of occurring. We found that the attacks most protected against in our advice were online and offline guessing attacks. A similar amount of advice tried to protect against phishing attacks as side channel attacks.

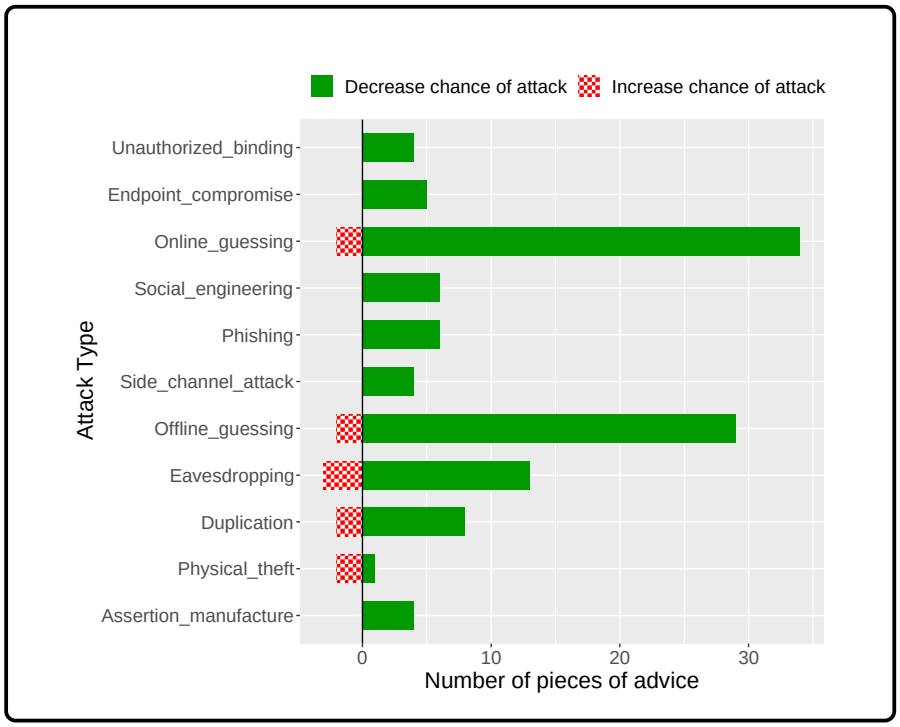


Figure 2.6: Number of times advice effects each attack type (excluding minor increases and decreases)

Figure 2.6 shows the number of times an attack was affected by a piece of advice. Green bars show the number of times a piece of advice decreased the chance of this attack occurring. The red-boxed bars show the number of times a piece of advice increased the chance of the given attack occurring.

Physical theft had the least amount of advice that helped protect against it and had a lot of a advice that leads to a minor increase in the chance of a physical theft. This is because of advice such as "use 2-factor authentication using a phone" which introduces an additional physical object into the authentication procedure.

2.9.5 Costs versus benefits trade-off

We are interested in the trade-offs between the costs and the benefits. Is high-cost advice balanced by high benefits? Or are users paying high usability costs for small increases to security?

We found in our benefits discussion that "one account per user" offered many security benefits. It has four one time minor costs and three of them are

borne by the organisation. The fourth is user education which is borne by both user and organisation. This, therefore, offers a significant increase in security without placing major inconveniences on the user.

In our benefits analysis, we found that "Enforce maximum [password] length" had no positive security value. We find that it places three minor costs on the organisation and one minor cost on the user. This is an example of advice which compromises usability for no increase in security. Unfortunately, this practice is still enforced by organisations. In 2007, Furnell assessed the policies of 10 different websites and 3 enforced maximum lengths on passwords: two limited the length to 10 characters and one limited it at 16 [54]. In addition, we found that some websites only reveal their limit on password length after the user has attempted to use a longer password [129].

"Change your password regularly" gives minor protection against offline and online guessing attacks. But it has 6 major periodic costs and 1 minor one-time cost and therefore the benefits are unlikely to offset the costs. This is in line with the results of Chiasson et al. [24].

An expensive piece of advice was "use multi-factor authentication". But it does offer increase in security against online guessing and phishing attacks, two of the most common attack types. Whether the benefits outweigh the costs will likely depend on its implementation and the needs of the specific organisation and users.

One very beneficial piece of advice appears to be: "encrypt password files". This protects against three attack types and most of the costs are borne by the organisation rather than the user. "Don't transmit in cleartext" and "Request over a protected channel" both protect against a number of attacks and the only cost to users is likely an insignificant increase in computing power.

The user piece of advice "Keep software updated" protects against three attack types but it incurs two periodic costs to the user and one minor cost. In these cases, it is more difficult to determine at a glance whether the advice is cost beneficial for the user.

The user piece of advice "Use a Password manager" was well regarded by users in our study. It can protect against three attack types and most of the costs it incurs are to the organisation. A password manager greatly reduces the users'

memory load and by extension a user can use as long, random and complex of a password as they wish. However, as with many of the pieces of advice, the value of a password manager will lie in how users utilise it. If a user uses a password manager and continues to reuse a common password choice across multiple sites then many of the potential benefits won't materialize.

Finally, we note that our "canary" worked. We included the advice "Don't allow users to paste passwords" knowing that it had been determined by the security community to have no security value [73] and both users and administrators identified it as a piece of advice that they disapprove of (users with slightly more uncertainty). We also could not identify any security benefits for the advice under our attack categories.

For the 79 advice statements we collected, and the one piece of advice we added in, we identified the costs and benefits of each one. We also notice that, in general, the costs are human effort costs; both on the system administrators who must implement the authentication policies, and on the user who must abide by them.

Most advice was concerned with protecting against online and offline guessing attacks. Keylogging and phishing or pharming attacks are not mentioned as often, despite their prevalence. Most of the protection against offline guessing is focused on improving password strength rather than back end-processes. This is one of the examples where the burden is placed on the user rather than the organisation.

2.10 Conclusion

In this chapter, we categorized and discussed currently circulated password recommendations given by security specialists, multinational companies and public bodies. We find that the advice given by different organisations is contradictory and often at odds with security and usability research findings. 41% of the 270 recommendations we collected were contradicted by recommendations given by another organisation.

Using a taxonomy of 270 pieces of collected password advice we develop a model of costs and benefits of advice. We identify the costs of authentication advice as the resources, human or otherwise, which are required for the advice

to be implemented. We define the benefits as the change in security risk. Both costs and benefits can be positive or negative; advice can reduce the number of resources needed, and advice can increase or decrease the risk.

Using input from 73 end-users and administrators we applied the classification of costs to the collected advice we gain an insight into the usability of current authentication recommendations. We find most of the advice places large burdens on humans, both system administrators, and end-users. Over 85% of the costs we identified related to the need for additional human labour or effort. We also find that a lot of the security advice focused on small improvements to security which resulted in heavy costs.

But reassuringly we did collect some advice which can offer large security benefits for small usability constraints. We also conclude that if an organisation is willing to bear costs on themselves, they can significantly improve usability for their end-users, and simultaneously increase their security.

In Chapter 5 we will build on the work introduced in this chapter. We will be leveraging attack frequency and the quantifiable human costs of authentication, in order to comprehensively evaluate the costs versus benefits of authentication security policies. But first, in Chapter 3 and Chapter 4 we will delve deeper into one of the more studied and prevalent authentication attack vectors: password guessing.

CHAPTER 3

Convergence of Password Guessing to Optimal Success Rates

Password guessing is one of the most common methods an attacker will use for compromising end users. We often hear that passwords belonging to website users have been leaked and revealed to the public. These leaks compromise the users involved but also feed the wealth of knowledge attackers have about users' passwords. The more informed attackers are about password creation, the better their password guessing becomes. In this chapter, we demonstrate using proofs of convergence and real-world password data that the vulnerability of all users of a website increases as a result of a leak of any of the users' passwords from that website. We show that a leak that reveals the passwords of just 1% of the users provides an attacker with enough information to potentially have a success rate of over 84% when trying to compromise other users of the same website. For researchers, it is often difficult to quantify the effectiveness of guessing strategies, particularly when guessing different datasets. We construct a model of password guessing that can be used to offer visual comparisons and formulate theorems corresponding to guessing success. This chapter is based on research published in [115] and [116].

3.1 Introduction

The resistance of passwords to guessing is essential to our online security. Passwords are used to protect our online banking, shopping, insurance accounts, social media, and email, among many other things. An attacker's goal is to gain access to these accounts, often by directly guessing passwords. Once an attacker has access, they can make a financial gain from that user's account directly or use the user as a base for spam, botnet, ransomware, and phishing attacks. Our work is important for gaining a better understanding of our vulnerabilities to password attackers and to raise the awareness of organizations to the damage leaked password datasets can cause.

Guessing passwords in the right order is important for an attacker as they wish to compromise as many users as possible with a small number of guesses. Password guessing can be divided into two types: online guessing and offline guessing. In an online guessing attack, the attacker attempts to guess combinations of the username and password directly on the live system, for example, on a website login page. The number of guesses is often limited by throttling techniques, such as locking a user out when a certain number of wrong guesses has been reached. In this case, an attacker will often need to make a correct guess in less than, say, 100 attempts [59].

An offline guessing attack can only occur after a dataset of passwords has been leaked. These leaks occur relatively frequently and can result in reputational damage to multinational organizations and governments [65]. Once a password dataset has been leaked, then the method that was used to store the passwords is important. If passwords are stored in plaintext or they are encrypted and the key is leaked, then they are already available to attackers. If they are hashed and unsalted, then most passwords can be found using a hash lookup table (rainbow tables) [124]. Finally, if they are hashed and salted, then offline guessing is necessary [51]. In an offline guessing attack, the attacker guesses a password, combines it with the random salt associated with each user, and then hashes the combined value. The goal of the attacker is to compromise as many users as possible with as few guesses as possible and, therefore, get the best returns for their time and resources used.

In this chapter, we formalize an understanding of password guessing success. This allows a comparison of guessing success, and is particularly useful for

comparing guessing when a variety of password datasets are used. This formulation of guessing success also allows for the development of a model of guessing that can be analytically studied. We use it to prove that convergence to optimum guessing success rates occurs when a sample of passwords from a password set is used to guess the whole password set. We continue by showing empirically that this convergence occurs when real leaked password datasets are used. In fact, we can show that, when a sample of users are compromised, information is also revealed about the characteristics of the remaining users' passwords, thus allowing us to see an effect on the guessing success when informing our guessing using the leaked sample.

In Section 3.3, we introduce a model for measuring the effectiveness of password guessing. In Section 3.4, we prove that, using a sample of passwords, we can effectively guess the password in a dataset with a loss that converges to zero as the sample becomes large. Section 3.5 provides a graphical example of the guessing function and introduces real-world leaked datasets. In Section 3.6, we use our guessing function to demonstrate the effectiveness of samples at compromising the remaining users in the dataset from which they were drawn. Section 3.7 introduces potential variants of our convergence theorems that allow for a small amount of guessing loss and therefore better describe the empirical data and the motivations of an attacker. Finally, in Section 3.8, we demonstrate the threat to an organization of a subset of their users being compromised. We show that organizations are vulnerable if a subset of their passwords is leaked.

3.2 Related work

Attackers use large dictionaries of words, as well as lists of most popular password choices to inform their password guesses. It is important for researchers to understand the methods used by attackers in order to know how to best protect online accounts. There has been considerable research developing password guessing strategies. Narayanan et al. employed Markov models to enable faster dictionary attacks [118]. Weir et al. used probabilistic context-free grammar (PCFG) which were trained using password breaches and used to assign probabilities to passwords for guessing [178]. Dürmuth et al. proposed an updated password guessing model based on Markov models, called OMEN [21]. Houshmand and Aggarwal created a method for merging

multiple grammars for dictionary-based PCFG models [70]. These methods will be discussed in more detail in Section 4.2.

User chosen passwords are inherently easy for automations to guess [98]. Often websites have offered guidance for users on how to choose stronger passwords [163]. In order to offer improvements, it is important that conclusions can be made regarding the "guessability" of a given password. In 2003, Burr [20] suggested that password strength could be measured using Claude Shannon's entropy measure: $H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$. Massey [100] showed that Shannon's entropy provides a lower bound on the expected number of guesses, but one that it not tight in general. In 2010, Weir et al. used real password datasets to show that entropy is not an accurate measure of password guessability. [177]. In 2012, Christiansen and Duffy showed that, when appropriately scaled, as the password length grows, the logarithm of the guesswork can provide direct estimates of the guesswork distribution when passwords are long [28]. Further analysis into the distribution of password has sparked an interest for many researchers. Many believed that user password choices would follow a Zipf distribution [145]. Zipf's law was originally formulated in linguistics. It states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. In 2012, Malone and Maher showed that user password choice does not follow a Zipf distribution [98].

These entropy and distribution measures are now rarely used and entropy itself is considered a poor measure of password guessability [83]. Instead, real world passwords are typically used to measure password security. The first to explore real world passwords and report their guessability were Morris and Thompson in 1979 [106]. They found that using a simple dictionary they were able to guess one third of all the users' passwords in 5 minutes. This flags some key issues with reporting the guessability of a password set using guessing as analysis. Firstly, we do not know what words were in the dictionary and what order they were tried in. In addition, we know that the guessing method they used compromised one third of the passwords in five minutes, but this does not tell us that this approach was effective or ineffective since they have not told us what the optimum number of passwords compromised in that time could have been. Therefore, while this real world guessing does indicate that the users' passwords in the dataset were possibly very easy to guess, no conclusions can

be made about the effectiveness of the guessing method.

Other researchers have developed more sophisticated methods for measuring guessing success. Melicher et al. modeled password guessability using neural networks [102], Durmuth et al. used an ordered Markov enumerator to simulate attackers' password guessing [42], Hitaj et al. used deep learning [68], and Weir et al. used probabilistic context-free grammars [178]. In 2012, Bonneau looked at a number of different options for password guessing metrics [12]. Kelley et al, developed Weir's work to create a *guess number calculator* which can determine at what point a given password guessing algorithm, trained with a given data set, would guess a specific password. However, Ur et al. show that the choice of guessing approach and the dictionaries used can bias research conclusions [165].

In this research we investigate guessing success in context. We acknowledge that depending on the password set we are guessing, a return rate for a guessing algorithm of $m\%$ of users' passwords in n guesses is either efficient or ineffective. We introduce a measure of password guessing which allows comparisons to be drawn when guessing different sized password sets with differing distributions using different dictionaries of guesses. This measure can then be used to bound guessing success. We leverage it in order to uncover the effect a small number of leaked passwords can have on the vulnerability of a website's entire user cohort.

3.3 Model

When we report the number of users that a guessing technique compromises, it is important to consider this in light of the maximum number of users it is possible to compromise for that number of guesses on that dataset. This can depend on the total number of users in the dataset and also on the distribution of password choices by those users. Therefore, reporting the number of guesses that a technique makes does not give the full picture unless context is provided. We, therefore, begin by introducing a model that can report effective guessing measured with respect to the guessing potential.

Suppose we have a set of passwords X chosen by N users. We rank and order these passwords, so the most popular password is rank 1, the second most popular is rank 2, and so on, until rank $|X|$, where $|X|$ is the number of dif-

ferent password choices; we can break ties between equally popular passwords arbitrarily. Let $p(x)$ be the probability that the password x is used by a randomly selected user from the group N; note that q will also be a probability distribution, and we can extend it to a probability measure on X. Let $\sigma(k)$ be the password of rank k in distribution p.

Question: If we take n samples from a dataset X of users' passwords, how effectively do these n samples guess the passwords of the other N users from the same dataset?
First, we define a method for measuring the effectiveness of guessing.

3.3.1 Optimal Guessing

The optimal guessing strategy [100] involves using the rank and order of p. If we guess the first g passwords, then the fraction of passwords guessed is

$$\mathcal{F}(g) = \sum_{k=1}^{g} p(\sigma(k)). \tag{3.1}$$

The function is cumulative because we want to know after g guesses how many users we have compromised, rather than caring about how many users were compromised on the g-th guess.

3.3.2 Guessing With a Sample

Suppose we take a sample of n passwords and rank and order them to form a second distribution q^n. Let $\sigma_{q^n}(k)$ be the password of rank k in the distribution q^n. Now, we use the distribution q^n to guess passwords that are actually distributed with frequency p. We can define a function similar to the one above that will tell us how many users in p will be compromised when using the rank and order of the passwords in q^n.

$$\mathcal{G}_{q^n}(g) = \sum_{k=1}^{g} p(\sigma_{q^n}(k)). \tag{3.2}$$

3.3.3 Guessing Loss

Combining the optimal guessing strategy with guessing using a sample, we define a method for describing how well a set of ordered password guesses can

guess a dataset of passwords as:

$$\mathcal{H}_{q^n}(g) = \mathcal{F}(g) - \mathcal{G}_{q^n}(g) = \sum_{k=1}^{g} p(\sigma(k)) - p(\sigma_{q^n}(k)). \tag{3.3}$$

This function measures the gap between the optimal strategy and the ability of a probability distribution q^n to guess a second probability distribution p.

3.4 Proof of Convergence of Password Guessing

Using this model, we might hope to use a sample to drive password guessing with zero loss. When $\mathcal{H}_{q^n}(g) \equiv 0$, we have guessed every user's password as fast as possible. We will now prove that, as the number of samples increases, we converge towards a zero-loss system.

Definition 3.4.1 (δ)**.** Consider the minimum size of the gap between two distinct adjacent probabilities of p:

$$\delta = \min_g \left\{ p(\sigma(g)) - p(\sigma(g+1)) \,|\, p(\sigma(g)) \neq p(\sigma(g+1)) \right\},$$

where we take $p(\sigma(g)) = 0$ if $g > |X|$.

Lemma 3.4.1. *Suppose q^n is a sample of size n drawn from the distribution p. A lower bound on the probability of zero loss can be given by*

$$\mathbb{P}[\mathcal{H}_{q^n} = 0] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta/2].$$

Proof: Consider probability distributions q, where $\|p - q\|_\infty < \delta/2$. Note that, for the two passwords w and w', if we had $p(w) > p(w')$, then $q(w) > q(w')$, as the difference between $p(w)$ and $p(w')$ must have been at least δ, but $|p(w) - q(w)| < \delta/2$, and likewise for w'. Thus, σ_q orders passwords of differing p probability in the same order as p.

On the other hand, if $p(w) = p(w')$, and w is replaced by w' in the ordering σ_q, then the contribution to $\mathcal{H}_{q^n}$ is $p(w) - p(w') = 0$. We conclude that if $\|p - q\|_\infty < \delta/2$, then $\mathcal{H}_{q^n} = 0$. $\square$

We will now show that the second probability in Lemma 3.4.1 goes to 1 as $n \to \infty$ using Sanov's Theorem.

Theorem 3.4.2 (Convergence to optimal guessing using Sanov's theorem)**.** *Suppose q^n is a sample of size n drawn from the distribution p.*

$$\mathbb{P}[\mathcal{H}_{q^n} = 0] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta/2] \to 1 \quad \text{as } n \to \infty$$

and

$$\mathbb{P}[\|p - q^n\|_\infty < \delta/2] \geq 1 - (n+1)^{|X|} 2^{-n\frac{\delta^2}{2}}.$$

Proof: Let A be the set of q with $\|p - q\|_\infty \geq \delta/2$. Pinsker's inequality [162] tells us that, for any $q \in A$,

$$\max_{E \subset X} \{|p(E) - q(E)|\} \leq \sqrt{\frac{D_{KL}(q||p)}{2}},$$

where E is any event. In particular, if $q \in A$, then the gap between the probabilities is always at least $\delta/2$, and the max on the left must be at least $\delta/2$, so

$$\frac{\delta}{2} \leq \sqrt{\frac{D_{KL}(q||p)}{2}},$$

and so $\delta^2/2 \leq D_{KL}(q||p)$. Sanov's Theorem [31] says

$$\mathbb{P}[q^n \in A] \leq (n+1)^{|X|} 2^{-nd^*},$$

where

$$d^* = \min_{q \in A} D_{KL}(q||p).$$

$$\implies \mathbb{P}[q^n \in A] \leq (n+1)^{|X|} 2^{-n\frac{\delta^2}{2}}.$$

We conclude that $\mathbb{P}[q^n \in A] \to 0$, so $\mathbb{P}[\|p - q^n\|_\infty < \delta/2] \to 1$ as $n \to \infty$.
□

Theorem 3.4.2 shows that a sample of size n will converge to the distribution of the whole dataset p with a probability approaching 1 as $n \to \infty$.

However, this convergence is slow. The turning point of the probability bound

is

$$\frac{\delta^2 \ln 2}{2} - \frac{|X|}{n+1} = 0.$$

This tells us the point at which the function must start increasing towards 1. It is determined by the ratio of the sample size n to the total size of the dataset $|X|$. Rearranging, we can find the value of n that the bound must have turned by

$$n = \frac{2|X|}{\delta^2 \ln 2} + 1.$$

Since δ is a probability value between 0 and 1, we can see that the number of samples n in q will need to be at least two times greater than the number of different passwords in the dataset.

In fact, typical values of δ will result in considerably larger n. δ relates to the difference in frequency between two adjacently ranked passwords. This difference in frequency is often 1. Given a dataset of N users, and a probability difference between a password chosen by two users and a password chosen by one user of $\delta = 1/N$, this will give

$$n = \frac{2|X|N^2}{\ln 2} + 1.$$

While Theorem 3.4.2 does demonstrate that our guesses success will converge to zero, the rate of convergence is too slow to be useful. We, therefore, present a second method for identifying an alternative lower bound for the convergence.

Theorem 3.4.3 (Convergence to optimal guessing using the central limit theorem)**.** *Suppose q^n is a sample of size n drawn from the distribution p. Then, as $n \to \infty$*

$$\mathbb{P}[\mathcal{H}_{q^n} = 0] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta/2] \geq \left(1 - Q\left(\frac{\delta\sqrt{n}}{2\|A\|_\infty}\right)\right)^{|X|} \to 1,$$

where $Q(x)$ is the probability that a standard normal random variable takes a value larger than x.

Proof: Given that q^n is sampled from distribution p in an essentially multinomial way, we know that $\mathbb{E}[q^n] = p$ and that the covariance for the entries of

q^n will be Σ/n [188]. Where

$$\Sigma_{i,j} = \begin{cases} p_i(1-p_i) & \text{if } i = j \\ -p_i p_j & \text{otherwise.} \end{cases}$$

The multivariate central limit theorem tells us that q^n is approximately normal $N(p, \Sigma/n)$.

Note that Σ is real valued, symmetric, and diagonally dominant, and so positive semidefinite, thus having non-negative eigenvalues. Consequently, by spectral decomposition [69], we can write $\Sigma = U\Lambda U^T$, where $UU^T = I$ and Λ is a matrix with Σ's eigenvalues on the diagonal. Let $A = U\Lambda^{1/2}$, and then $AA^T = \Sigma$ and samples from $N(p, \Sigma)$ can be generated by forming a vector z with coordinates that are $N(0, 1)$ and then taking $p + Az/\sqrt{n}$.

Now,

$$\mathbb{P}\left[\|p - q^n\|_\infty < \frac{\delta}{2}\right] = \mathbb{P}\left[\frac{\|Az\|_\infty}{\sqrt{n}} < \frac{\delta}{2}\right] \geq \mathbb{P}\left[\frac{\|A\|_\infty \|z\|_\infty}{\sqrt{n}} < \frac{\delta}{2}\right] = \mathbb{P}\left[\|z\|_\infty < \frac{\delta\sqrt{n}}{2\|A\|_\infty}\right].$$

Or, in terms of the Q function, noting that z has $|X|$ independent components,

$$\mathbb{P}\left[\|z\|_\infty < \frac{\delta\sqrt{n}}{2\|A\|_\infty}\right] = \left(1 - Q\left(\frac{\delta\sqrt{n}}{2\|A\|_\infty}\right)\right)^{|X|}.$$

So, as $n \to \infty$, $Q(.) \to 0$, so $\mathbb{P}[.] \to 1$. $\square$

In fact, to simplify this expression, we can give an estimate for $\|A\|_\infty$. The entries of A, $a_{ij} = u_{ij}\lambda_j^{1/2}$, and so by the Cauchy–Schwartz inequality,

$$\|A\|_\infty = \max_i \Big(\sum_j |a_{ij}|\Big) = \max_i \Big(\sum_j |u_{ij}\lambda_j^{1/2}|\Big) = \max_i |\langle u_i, \lambda^{1/2}\rangle| \leq \max_i \|u_i\|_2 \, \|\lambda^{1/2}\|_2.$$

However, $\|u_i\|_2 = 1$ because the rows of U are orthogonal vectors and

$$\|\lambda^{1/2}\|_2 = \sqrt{\sum_j (\lambda_j^{1/2})^2} = \sqrt{\text{trace}(\Sigma)}.$$

So,

$$\|A\|_\infty \leq \sqrt{\text{trace}(\Sigma)} = \sqrt{p_1(1-p_1) + p_2(1-p_2) + \ldots}.$$

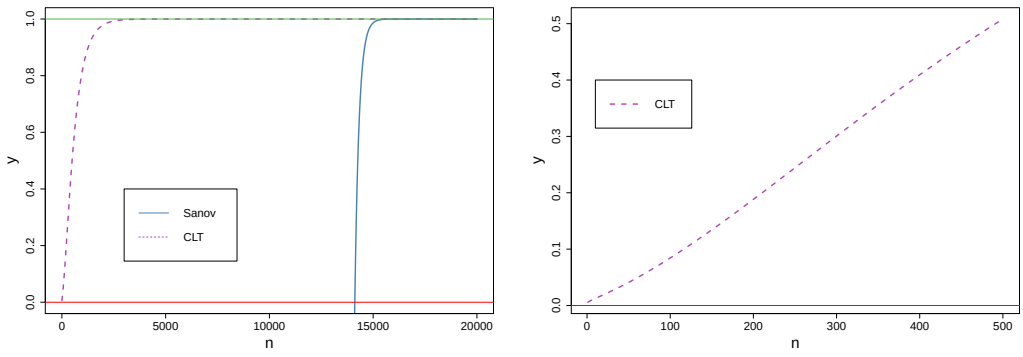
Note that, as $\sum p_i = 1$, we can find the largest possible value of $p_1(1-p_1) + p_2(1-p_2) + \ldots$ to be $1 - 1/|X| \leq 1$.

We conclude that

$$\mathbb{P}\left[\|p - q^n\|_\infty < \frac{\delta}{2}\right] \geq \left(1 - Q\left(\frac{\delta\sqrt{n}}{2\|A\|_\infty}\right)\right)^{|X|}$$
$$\geq \left(1 - Q\left(\frac{\delta\sqrt{n}}{2\sqrt{\operatorname{trace}\Sigma}}\right)\right)^{|X|} \geq \left(1 - Q\left(\frac{\delta\sqrt{n}}{2}\right)\right)^{|X|}.$$

Unlike our previous bound, this bound derived by the central limit theorem is monotonic in n. The convergence function is monotonic and between 0 and 1. The parameters $|X|$ and n determine the rate of convergence but there is no turning point of the function. This function offers a better bound on the convergence.

For example, take a toy dataset with just 8 users and a minimum distance between the probability of password choices as $\delta = 1/N = 0.125$. Figure 3.1a shows the convergence to 1 for both bounds. Even for this small dataset size, the number of samples needed for a probability of close to 1 is large. We can see the bound provided by the central limit theorem is much better than the Sanov bound. This effect is further emphasized the larger the size of the dataset X.



(a) Comparison of Convergence bounds. (b) CLT bound: $n = 0 \to 500$.

Figure 3.1: **(a)** Convergence bound with Sanov's theorem versus with **(b)** the central limit theorem.

3.5 Test on Real-World Leaked Password Datasets

Theorems 3.4.2 and 3.4.3 tell us that if we continue to choose samples from a dataset, then these will have zero guessing loss with probability approaching 1 as $n \to \infty$. We are interested in the application of this convergence to a real-world password leak. Our question is whether a leak of a subset of a dataset can, in practice, give away valuable information about the distribution of the remaining passwords in that same dataset.

In this section, we introduce five datasets of passwords that have been leaked to the public from real organizations. We comment on the distribution of these passwords. Then, sampling passwords from these datasets, we see whether the samples provide an effective guessing strategy with respect to our guessing loss function.

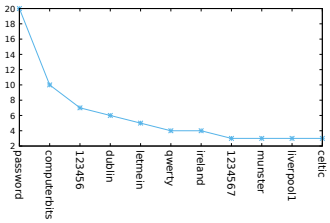
3.5.1 Datasets

3.5.1.1 Computerbits.ie Dataset: $N = 1795$

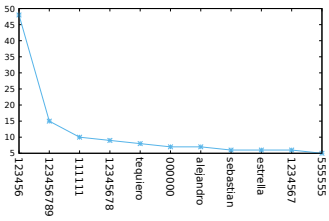
In 2009, 1795 users' passwords were leaked from the Irish website Computerbits.ie. Figure 3.2a shows the top 11 most popular passwords in this dataset. We can see many Irish-orientated words: dublin, ireland, munster, celtic. The second most popular password for the website Computerbits.ie was "computerbits", reinforcing the idea that the service provider has an impact on the user's choice of password [175].

3.5.1.2 Hotmail.com Dataset: $N = 7300$

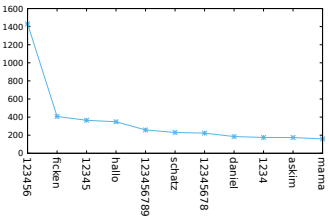
Ten thousand users' passwords from the website Hotmail.com were made public in 2009 when they were uploaded to pastebin.com [36] by an anonymous user. Though it is still unknown, it is suspected that the users had been compromised by means of phishing scams [8]. Figure 3.2b shows the frequency of the first 11 most popular passwords chosen by users in the Hotmail dataset. The most popular password is "123456" which occurs with frequency 48. The password of rank 11 occurs with frequency 5. So, all passwords of rank greater than ten have a frequency less than or equal to 5. In total, there are 6670 ranks in the Hotmail distribution, i.e., there are 6670 distinct passwords in the



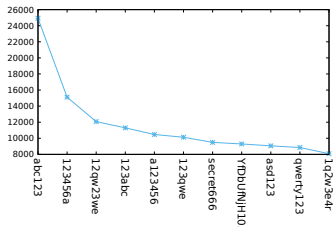
(a) Computerbits.ie distribution



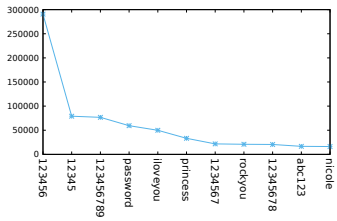
(b) Hotmail.com distribution.



(c) Flirtlife.de distribution.



(d) 000webhost.com distribution.



(e) Rockyou.com distribution.

Figure 3.2: Distribution of password choices: frequency of passwords in ranks 1 to 11.

dataset. The top password represents 0.66% of the Hotmail users' passwords, and the rank 11 password represents 0.068% of the Hotmail users' passwords.

3.5.1.3 Flirtlife.de Dataset: $N = 98{,}912$

In 2006, over 100,000 passwords were leaked from a German dating site Flirtlife.de. A write-up by Heise online [141], a German security information website, states that the leaked file contained many usernames and passwords with typographic errors. It seems that attackers were harvesting the data during log-in attempts.

This means that, in the dataset, for a small number of users, we had multiple passwords. We followed the clean up method specified in [98]: we took the user's password as the last entry seen for that user. In this way, we ignore passwords where the users first attempts were the wrong password or previous passwords when a password change has occurred. There were also 18 blank password fields, and we removed these from the dataset, too. After this cleaning process, we were left with 98,912 users and 43,838 different passwords.

The distribution of the first 11 passwords from the Flirtlife dataset are shown in Figure 3.2c. As noticed by [98], we see that passwords in these top ranks

have meanings in German or Turkish. For example, 'schatz' and 'askin' are German and Turkish terms of endearment, respectively. The rank 1 password represents 1.45% of the Flirtlife users' passwords, and the rank 11 password represents 0.16% of the Flirtlife users' passwords.

3.5.1.4 000webhost.com Dataset: $N = 15{,}252{,}206$

In 2015, 15 million users' passwords were leaked from 000webhost.com [56]. The attacker exploited a bug in an outdated version of PHP. The passwords were plaintext and created with a composition policy that forced them to be at least 6 characters long and must include both letters and numbers. Today, a lot of websites will enforce a composition policy on passwords. It will be interesting to see the impact this has on the effectiveness of guessing the password dataset distribution. The leaked dataset was cleaned in the same way as in [56]. All passwords longer than 256 characters and passwords that were not ASCII were removed.

The distribution of the top 11 passwords in the 000webhost.com dataset is shown in Figure 3.2d. All passwords are a mixture of letters and numbers. There is a clear pattern to each password, though less obviously for the password 'YfDbUfNjH10305070'. The letters, YfDbUfNjH, can be mapped to a Russian word which means "navigator" [173]. It is unclear why this password is so popular. It could be a Russian botnet that is using the same password for each of its bot accounts [183]. There are 10 million distinct passwords in the dataset. The rank 1 password represents a surprisingly low 0.16% of the users' passwords, and the rank 11 password represents 0.05%.

3.5.1.5 Rockyou.com Dataset: $N = 32{,}602{,}877$

In December 2009, 32 million user credentials were leaked by the company Rockyou. The passwords were stored in plaintext and the hackers used a 10-year-old SQL vulnerability to gain access to the database. The platform did not allow the inclusion of special characters in passwords. Figure 3.2e shows the distribution of the first 11 Rockyou passwords. The rank 1 password in the Rockyou dataset represents 0.89% of the total dataset, and the rank 11 password represents 0.05% of the total users' passwords in the dataset.

3.5.2 Demonstration of Guessing Function

In Section 3.3.3, we defined a method for measuring the effectiveness of password guessing. We will now graph the functions $\mathcal{F}_p(g)$, $\mathcal{G}_{q^n}(g)$ and $\mathcal{H}_{q^n}(g)$. Note that the y-axis for functions $\mathcal{F}_p(g)$, $\mathcal{G}_{q^n}(g)$ is a measurement of number of successes and therefore we want large values. But, $\mathcal{H}_{q^n}(g)$ is a measurement of cumulative loss, so we want $\mathcal{H}_{q^n}(g)$ to be small. The goal is to have the sample guessing, $\mathcal{G}_{q^n}(g)$, as close as possible to the optimal guessing, $\mathcal{F}_p(g)$, thus keeping the loss, $\mathcal{H}_{q^n}(g)$, as low as possible.

In Figure 3.3, we chose n =100 users' passwords, with replacement, from the Hotmail dataset. The 100 users' passwords were ranked and ordered and used to guess the passwords of all the 7300 Hotmail users whose passwords are in our dataset. We were able to compromise a total of 234 users, 134 more than those in the original sample. In the Figure, the blue dashed line represents the optimal number of successes, $\mathcal{F}_p(g)$. The green dot and dash line shows the number of success for each guess using the sample of 100 users' passwords, $\mathcal{G}_{q^n}(g)$. The red line is reporting loss rather than success. It is the difference between the optimal guessing and guessing using the password sample, $\mathcal{H}_{q^n}(g) = \mathcal{F}_p(g) - \mathcal{G}_{q^n}(g)$.

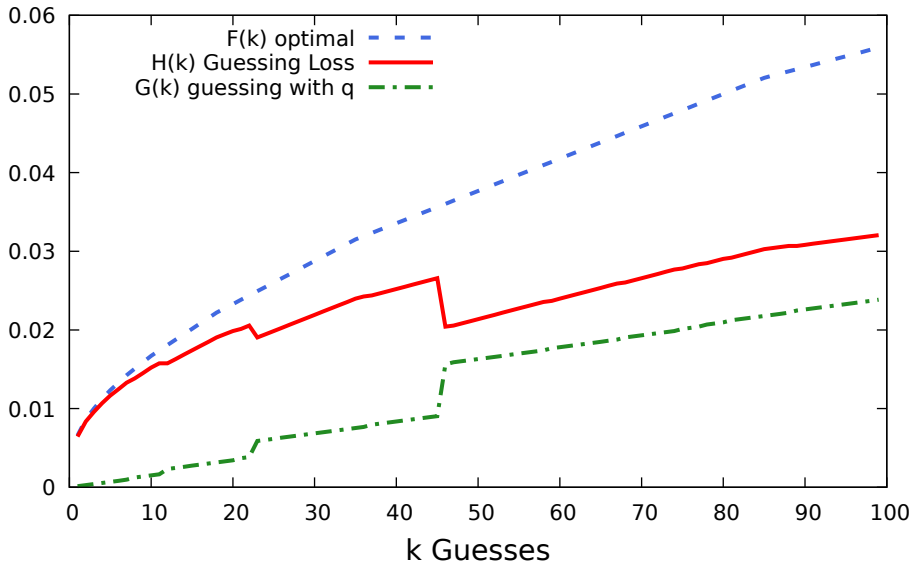


Figure 3.3: Hotmail dataset guessed using a sample of size $n = 100$ users.

The sample of 100 users was chosen randomly from the 7300 users in the Hotmail dataset. But, because we sample with replacement, we managed to

choose the same user twice. We discover this because there were 99 guesses made instead of 100, but the password of frequency 1 in q resulted in only 1 success in p. Note that, because all passwords after rank 1 in q occur with frequency 1, the ordering is arbitrary. Therefore, in this example, the jumps in loss could occur at any point after guess 1.

We can see a small reduction in loss at guess number 23, where 15 users were compromised. This was when the second most popular password "123456789" was guessed. On guess 46, the most popular password in the Hotmail dataset was guessed. This guess of password "123456" compromised 48 users.

Even for this small sample size (only 1% of the total dataset), we have a low loss. We only failed to compromise 0.03 of the users we could have potentially compromised with the optimum first 100 guesses. $\mathcal{G}_{q^n}(g)$ achieved almost half the efficiency of the optimal strategy with 100 guesses.

3.6 Empirical Evidence of Convergence

Our convergence proof, Theorem 3.4.2, leveraging Sanov's Theorem tells us that the convergence of the loss function goes towards zero with respect to $\frac{|X|}{n+1}$. That is, an important value for convergence is the ratio of the number of different passwords in the full dataset over the number of users' passwords in the sample.

In this way, the larger the sample size, the smaller the loss we expect. In Figure 3.4, we take samples of size 1%, 10%, 25%, 50%, 75%, 100%, and 200% of the original dataset size. For example, a sample of 200% of the Hotmail dataset involves drawing from the 7300 users' passwords 14,600 times with replacement.

As expected, we found that the increase in the number of samples does reduce loss. We initially see an increase in loss as the number of guesses increases. This increase occurs because we are dealing with a cumulative function, and the more guesses we make, the greater the potential we have to diverge from the optimal guessing. Notice that a peak occurs at approximately 420 guesses. In the Hotmail dataset, there are only 420 non-unique passwords (i.e., 420 password choices were chosen by more than one user). So, passwords in the Hotmail dataset after rank 420 have frequency 1; therefore each guess can only

be equal to or more successful than the optimal at that rank.

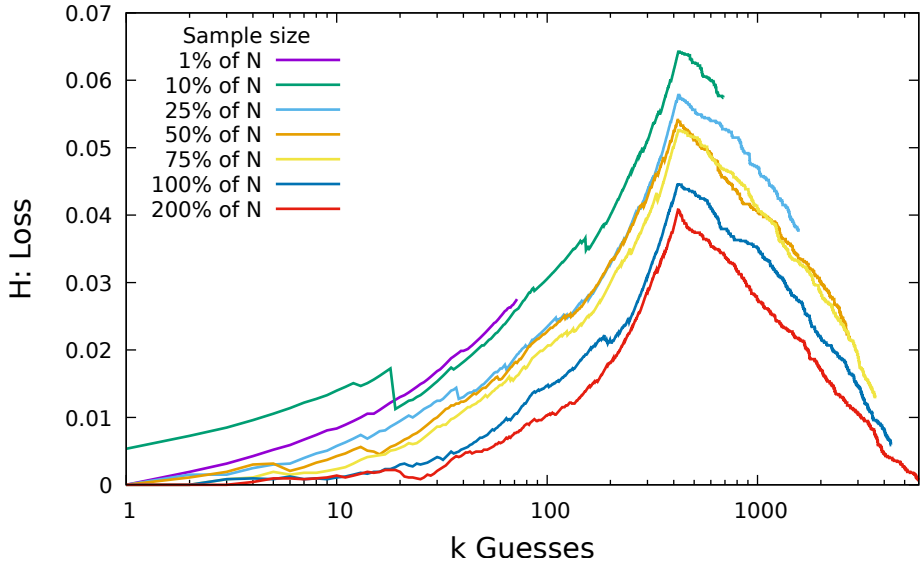
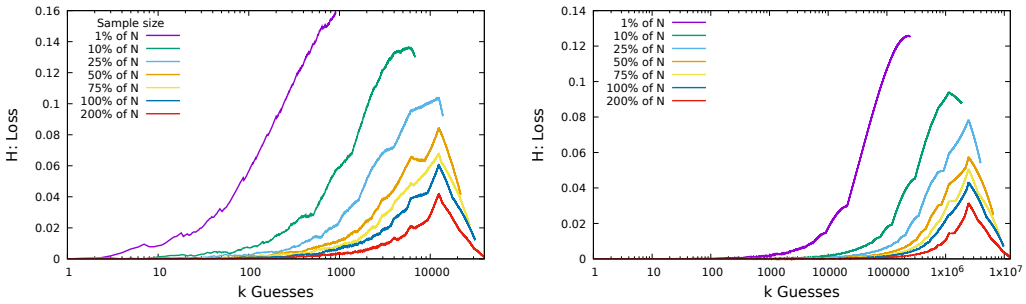


Figure 3.4: Probability loss: using samples of users from the Hotmail dataset to guess the passwords of all N users in the Hotmail dataset.

The Hotmail dataset has a relatively large number of unique passwords. The ratio of number of users N to number of passwords $|X|$ for the Hotmail dataset is 0.91, and 0.86 of the users chose a unique password. The ratio of users to passwords in the Flirtlife dataset is 0.44, and 0.32 of the users chose unique passwords. The ratio for the Rockyou dataset is also 0.44, and the fraction of users who chose unique passwords is 0.3645. In Figure 3.5, we plot the same loss function, for the same proportions of the dataset, using the Flirtlife dataset and the Rockyou dataset.



(a) Flirtlife (b) Rockyou

Figure 3.5: Probability loss for sample sizes n determined as a proportion of N.

Again, we see the larger sample sizes result in smaller loss. The largest value of loss for both was from the sample of size 1%. This resulted in a loss of 0.16 for Flirtlife and 0.125 for Rockyou, much higher than the largest Hotmail loss. Though it is not shown in the graphs, we note that the maximum loss for a sample from the Computerbits dataset was 0.055 and the maximum loss for a sample from the 000webhost dataset was 0.12. The turning point in the loss function occurs at approximately 12,442 guesses for Flirtlife. 12,442 is the number of non-unique passwords in the Flirtlife dataset. Similarly, we see the turning point for the Rockyou plots at ~2,450,000, matching the number of unique passwords in the Rockyou dataset.

In all three graphs, we can see that guessing using the sample is effective. When guessing using a sample, we lose, at most, 16% of the total users we could have compromised with the same number of optimal guesses. The loss of 16% for Flirtlife is the maximum over the *entire* range and with a sample of just 1% of the total number of users. If we look at the first 100 guesses when we have a sample of 10% of the users, we can see our loss is down at 0.007; that is, we are unable to compromise just 0.7% of the potentially compromisable users.

Figures 3.4 and 3.5 show that we were effectively able to guess users' passwords using a sample of passwords from other users of the same website. Even with a sample representing just 1% of the total number of users, we have a loss less than 0.16.

3.6.1 Spread of Results

In each of the previous graphs, we have only plotted results for one sample set of each size. But, results could vary depending on passwords chosen in the sample. Because we are anticipating convergence as $n \to \infty$, we expect the spread to decrease for increases in sample sizes.

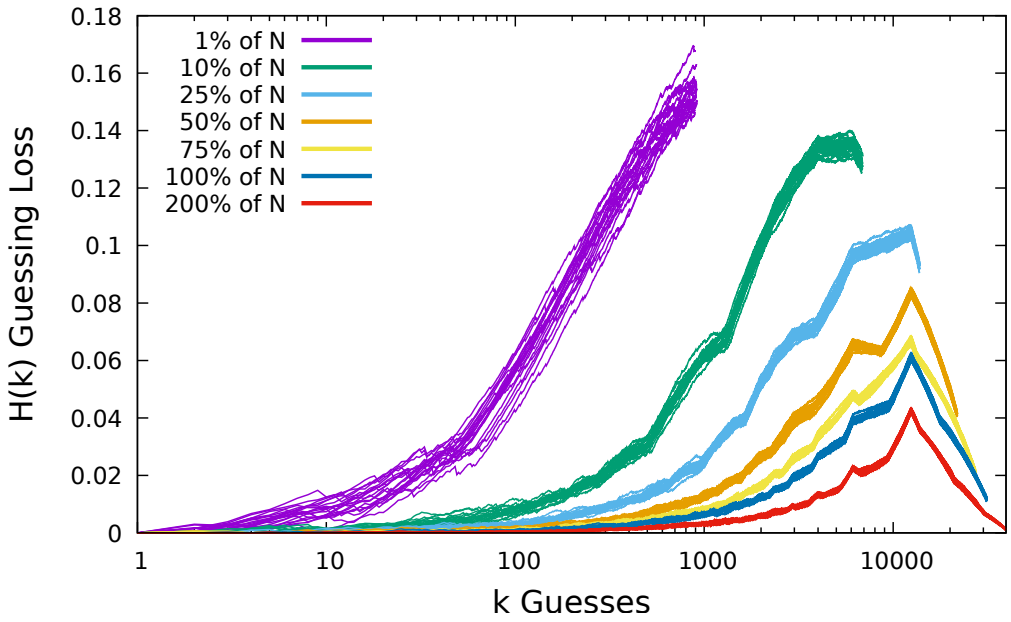


Figure 3.6: Guessing Flirtlife.de dataset, size N = 98,912, using multiple samples of each size.

In Figure 3.6, we plot the results of guessing the Flirtlife dataset using multiple samples of each size. We plot 20 different samples for each sample size. We can see that for a sample of size 1% of the dataset we have more spread in the data than for a sample 50% the size of the dataset and much more than for the sample of size 200%. At the end of the guessing, the difference between the max and min value for the samples of size 1% was 0.021, representing a difference in users compromised of 2121. In Table 3.1, we summarize the spread of the results for guessing the Flirtlife dataset. We report these results as frequency loss, i.e., the number of users from N = 98,912 that we were unable to compromise.

Table 3.1: Spread of loss results for 20 samples of each size n which are guessing the Flirtlife dataset.

Sample size		**Spread of Loss**			
% of N	**n**	**mean**	**max**	**min**	**range**
1%	989	15151.0	16608	14487	2121
10%	9891	12651.3	12980	12390	590
25%	24728	9194.9	9394	8946	448
50%	49456	4109.6	4219	3987	232
75%	74184	2100.1	2181	2018	163
100%	98912	1139.1	1210	1073	137
200%	197824	122.7	136	106	30

3.6.2 When Does Loss Reach Zero?

Despite the fact that we have low loss, in none of these graphs do we reach a loss of zero. In Rockyou, for the sample of size 200%, we still had 26,795 users to compromise. Flirtlife we had 130 users, and Hotmail we had 5 users left to compromise.

Focusing on Hotmail, we investigated how many guesses it takes for our loss function to reach zero. Using a sample 300% the size of the number of users, we converge to zero with guess number 6172. In Figure 3.1b, we had a dataset with just 8 users, and for both Sanov and Central Limit Theorem (CLT), at 300% of 8, $n = 24$, we can see we are still only just above 0 probability of convergence.

For the Flirtlife dataset, incrementing our sample size by 100% of N each time, we did not get a loss of zero until we had a sample 600% the size of N. In this case, we had a loss of zero repeatedly within the first 100 guesses, but as the number of guesses increased, we had approximately 120 users whose passwords we could not guess. We returned to a loss of zero at guess number 43,098. For a sample 600% of 8 in Figure 3.1a, $n = 48$, we still have a probability near to 0. This implies that our construction of the bounds in the convergence proofs are more strict than desirable for representing real-world guessing effectiveness.

3.7 Improvements to Models

For the Flirtlife dataset, we need a sample 600% the size of the number of users in the set in order to get a loss of 0. But, with a sample size that is 50% the number of users in the dataset, we have a loss of just 4%. An attacker is unlikely to care about getting every single user and will likely be interested in compromising the most users with a reasonable number of guesses. In Theorems 3.4.2 and 3.4.3, we show that the loss will converge to zero as $n \to \infty$. But, instead, we could look at a situation where we just want our loss to be small. We now provide additional theorems that show that we can limit the amount of loss we are willing to accept.

Specifically, we want to show that, if we take n samples drawn from a distribution p of passwords and form the resulting distribution q^n, then as $n \to \infty$ we have $\mathbb{P}[\mathcal{H}_{q^n} < \epsilon] \to 1$, i.e., in our guessing effectiveness we can choose to accept a small amount of loss, ϵ. We consider two methods for accepting an ϵ amount of loss. The first is a cut-off method, and the second is a blocking method.

3.7.1 Cut-Off Point to Allow ϵ Loss

The premise of this method is to divide the guessing into two parts, as shown in Figure 3.7. We consider the first part to be the high frequency passwords. For these, we want the guessing to be exact, i.e., the order of q matches the order of p for these ranks. After some cutoff point G_ϵ, we are not particularly concerned with the order of the remaining passwords. We wish to define the cut-off point such that the loss does not exceed a value ϵ.

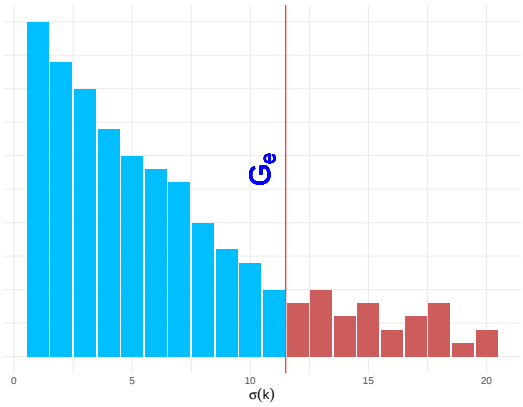


Figure 3.7: Demonstration of the cut-off point G_ϵ.

Lemma 3.7.1. *The lower bound on the probability of ϵ loss is defined by*

$$\mathbb{P}[||\mathcal{H}_{q^n}||_\infty < \epsilon] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta_\epsilon/2],$$

where G_ϵ is chosen such that

$$\sum_{g=0}^{\lfloor n/2 \rfloor - 1} p(\sigma(G_\epsilon + g)) - p(\sigma(|X| - g)) < \epsilon.$$

Proof: When $g < G_\epsilon$, we proceed in the same way as in Lemma 3.4.1 to achieve zero loss, i.e., we choose

$$\delta_\epsilon = \min_{g < G_\epsilon} \{p(\sigma(g)) - p(\sigma(g+1)) \,|\, p(\sigma(g)) \neq p(\sigma(g+1))\}.$$

Note that this δ_ϵ will typically be larger than δ.

$$\mathbb{P}[\mathcal{H}_{q^n}|_{\{g<G_\epsilon\}} = 0] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta_\epsilon/2].$$

When $g \geq G_\epsilon$, we know that the loss function $\mathcal{H}$ will be bounded by the case where q is ordered in the worst possible way. That is, the order of $q = p(\sigma(|X|)), \ldots, p(\sigma(G_\epsilon))$, where $\sigma(|X|)$ is the last ranked password in p.

If there are n ranks between $\sigma(G_\epsilon)$ and $\sigma(|X|)$, then the loss will increase for at most the first $\lfloor n/2 \rfloor$ ranks; by the symmetry of $p(\sigma(G_\epsilon + g))$ and $p(\sigma(|X| - g))$. Therefore, the maximum value the loss can be is

$$\sum_{g=0}^{\lfloor n/2 \rfloor - 1} p(\sigma(G_\epsilon + g)) - p(\sigma(|X| - g)) < \epsilon.$$

□

Using this method, we decide what value we are willing to accept as our ϵ loss and then work from there to compute the cut-off G_ϵ. The assumption in this method is that an attacker is concerned with guessing the high frequency passwords in the exact order and cares less about the exact ordering of the low frequency passwords. This method can be employed to effectively model the actions of an attacker performing an online guessing attack.

3.7.2 Blocking Method to Allow ϵ Loss

In this method, we employ some leniency to allow small changes between the ranks of passwords. The assumption is that, if an attacker is making a block of 10 password guesses against all accounts, then it matters little to an attacker whether the rank 4 guess is more successful than the rank 2 guess because the loss will have balanced out after the 10 guesses.

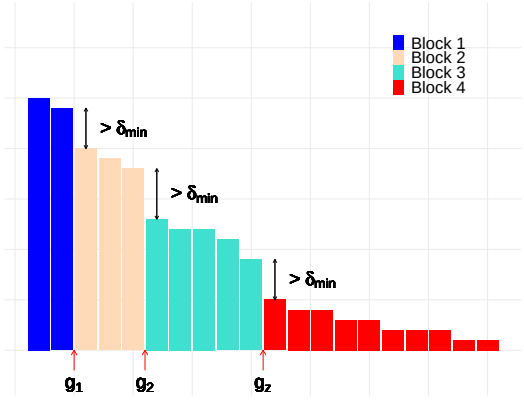


Figure 3.8: Demonstration of the δ block split.

Our motivation for this method is another attempt to increase the value δ. This should provide us with a stricter bound since an increase in δ will in turn increase $\mathbb{P}[\|p-q^n\|_\infty < \delta/2]$, which bounds our convergence in Theorems 3.4.2 and 3.4.3.

The premise is that we can divide the passwords into blocks with the jump between each block of a defined size δ, as in Figure 3.8. We concede that all passwords can change order within the blocks, but no password can move outside the block, and the blocks cannot change order. The loss, ϵ, is a result of the passwords that we are unable to group into a block.

It is possible to maintain a model with a notional zero loss by mandating that passwords remaining outside the blocks are guessed with accuracy. However, by the logic that these will be the high ranks and, therefore are likely to have low frequency, we decide not to assign any effort to correctly ordering these passwords and, instead, accept the loss. In the same way as for Lemma 3.7.1, we assume that such remaining passwords take the worst possible ordering; therefore, this provides us with a bound on our loss ϵ.

Lemma 3.7.2. *Given a block of ranks separated by* $\{g_1, \ldots, g_z\}$ *and a* $\delta_{min} > 0$ *so that*

$$p(\sigma(g_i)) - p(\sigma(g_i + 1)) > \delta_{min},$$

$\forall\, i = \{1, \ldots, z\}$*, we have*

$$\mathbb{P}\left[\mathcal{H}_{q^n}(g_i) = 0\ \forall g_i \text{ and } \mathcal{H}_{q^n}(g) \leq \epsilon\ \forall g > g_z\right] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta/2].$$

Proof: Take δ to be

$$\delta = \min_{1 \leq i \leq z} \{p(\sigma(g_i)) - p(\sigma(g_i + 1))\} \geq \delta_{min}.$$

Then,

$$\mathbb{P}[\mathcal{H}_{q^n}(g_i) = 0] \geq \mathbb{P}[\|p - q^n\|_\infty < \delta/2].$$

The condition on the right ensures the blocks do not change order; so, at g_i, the end of a block, all passwords in previous blocks have been guessed, meaning $\mathcal{H}_{q^n}(g_i) = 0$.

If the last ranked password, $\sigma(|X|)$, is contained inside the last block, then we have zero error. However, if we cannot create an end by using a jump of size δ_{min} for our last block, then those ranks outside of a block will denote our loss ϵ. This loss is bound by

$$\sum_{k=0}^{\lfloor n/2 \rfloor - 1} p(\sigma(g_z + k)) - p(\sigma(|X| - k)) < \epsilon.$$

□

Note that this blocking method may have applications for certain types of password guessing. For most of our attacks, we assume that passwords are guessed one at a time against all users, which is a reasonable match for some online guessing or offline guessing where an attacker has assigned a GPU core to each user/salt. Here, passwords could be guessed one at a time against each user by hashing them with respect to the user's specific salt. However, another method of guessing could be to feed blocks of passwords to a GPU, with each core working on a different password. This might apply when password hashes were unsalted or all hashed with a common salt. In the latter case, a successful guess would happen when any password in a block matches; so, a model that

disregards reordering within a block could be useful.

3.8 The Threat of Compromise from a Leaked Sample of Passwords

One of our initial claims was that revealing a sample of passwords from a dataset could help an attacker more than generic information about password choices. In this section, we provide evidence for this claim by using samples from one dataset to guess passwords in a different dataset. If there is a relationship between the passwords chosen by users at the same site, then we expect to get the lowest loss when guessing using passwords sampled from the same website.

3.8.1 Methodology

We took samples of size $n = 1000$ users' passwords from each of our four datasets and used these to guess the passwords of all the users in the full datasets. Below, we will discuss specific details regarding our methodology, and we then discuss results in Section 3.8.2.

3.8.1.1 Sample Size

Our samples were size $n = 1000$. It might have been desirable to use a larger sample size, for example, 10,000; however, we were limited by the number of passwords in the Computerbits password set and wanted the examples below to be comparable. In Section 3.6.1, we found that, for small sample sizes, there was a spread in the result of the guessing.

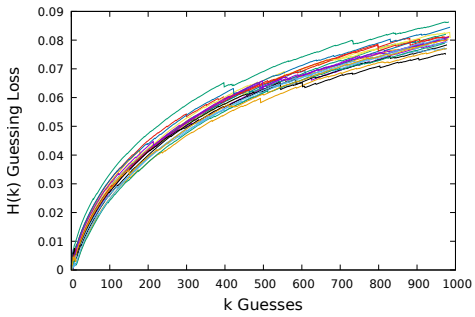


Figure 3.9: Rockyou 20 samples, size $n = 1000$.

For example, in Figure 3.9, we plot the loss for 20 Rockyou samples of size $n = 1000$. $n = 1000$ represents just 0.003% of the Rockyou dataset; as a result, we see a diversity in the success of the sample at guessing. As a result of the potential for differing results, we will run multiple trials for each sample size.

3.8.1.2 John the Ripper

We also included in the results the loss if the passwords are guessed using the top 1000 passwords in the John the Ripper (JtR) basic wordlist [80]. John the Ripper is one of many password cracking software tools. Its strength lies in its password mangling tools. It takes a base password and *mangles* it to create other related passwords. For example, it might start with the password 'password' and mangle it by exchanging vowels with symbols to create a password guess "p@ssw0rd". For our comparison, we take the basic wordlist that JtR can work on, which has just 3546 words. In the classic configuration of JtR, these 3546 would be guessed and then word mangling rules would be applied to the words to produce subsequent guesses. By including the guessing success for JtR, we provide a reference point for the effectiveness of the guessing relative to this common tool. The first 10 passwords in the John the Ripper wordlist we used are listed in Table 3.2. These have similarities to the top 10 passwords in the leaked datasets, shown in Figure 3.2, but they cannot include website, topic, nor demographic-specific word choices.

Table 3.2: First 10 passwords in the John the Ripper wordlist.

Rank	Password
1	123456
2	12345
3	password
4	password1
5	123456789
6	12345678
7	1234567890
8	abc123
9	computer
10	tigger

3.8.1.3 Sampling With and Without Replacement

We present the results of guessing when we sample both with and without replacement. When we sample with replacement, we might consider that we have a false sense of success because we are counting successes when we compromise users whose passwords we already knew and were part of our sample. However, an attacker may not be able to remove the users they already know from a dataset and, instead, risks wasting guesses retrieving data they already know.

First, in Section 3.8.2, we discuss the graphs that show sampling with replacement. When guessing with replacement, we simply rank and order our samples and use the ordering to guess the passwords in the dataset. We mark how many users we managed to compromise and subtract this from the optimum number of users we could have compromised had we used the best choice of password for that guess number, as per the function $\mathcal{H}_{q^n}(g)$. This loss value is plotted as a function of the number of guesses.

Second, in Section 3.8.3, guessing without replacement removes the bias from marking a success when the user's password in question was already part of our sample. Our method for guessing without replacement involves sampling 1000 passwords from each dataset without replacement. These samples are then used to guess the passwords in each dataset. However, when guessing using a sample, x, chosen from the dataset we are now guessing, X, we remove all the users who were used to form our sample from the dataset; $X - x$. Thus, we are now using x to try to guess $X - x$. All samples that were chosen from a different dataset can be used to guess dataset X as normal.

3.8.2 Guessing Results: With Replacement

Figure 3.10 shows the comparisons of guessing effectiveness when sampling with replacement for each sample source. Notice that, depending on the sample source, there are a different number of guesses made. For example, in Figure 3.10a, the sample from Computerbits made just 726 guesses, whereas the Rockyou sample made 984 guesses. This is because, when we sampled 1000 users from the Computerbits dataset, we returned only 726 distinct passwords. We can use the expectation of the number of distinct passwords to estimate

the number of distinct passwords we actually see in a sample of size n:

$$\sum_i (1 - (1 - p_i)^n)$$

Using this, we expect to see 732 distinct passwords in a sample of 1000 users' passwords from the Computerbits dataset, 915 in a Hotmail sample, 909 for Flirtlife, 995 in a 000webhost sample, and 979 in a Rockyou sample. This is a reflection of the proportion of the probability that lies in the high frequency passwords in each dataset.

In Figure 3.10a we see that the Computerbits sample is by far the most effective at guessing the Computerbits dataset. It offers improvements over guessing with the basic JtR wordlist or with samples from the other four datasets. A turning point occurs at $g = 88$, when we guess with the sample from Computerbits. This is because all passwords of rank greater than $g = 88$ have frequency 1.

Figure 3.10b shows that the Hotmail sample guesses the Hotmail dataset most effectively. We can also see that the distance from the optimum begins decreasing from $g = 420$ guesses, where we encounter the passwords that occur with frequency 1. For both the Hotmail dataset and the Computerbits dataset, the JtR basic wordlist performs only slightly more effectively than guessing with the samples of passwords collected from the other datasets.

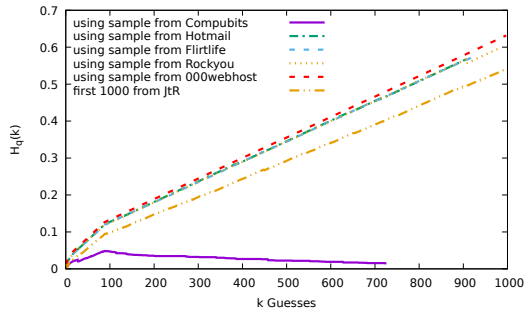
Figure 3.10c depicts the Flirtlife sample guessing the Flirtlife dataset the most effectively. The JtR list performs more effectively than all but the Flirtlife sample.

In Figure 3.10d, we can see that the sample chosen from the 000webhost dataset guesses the 000webhost dataset most effectively. In every other plot, the 000webhost sample is significantly worse than all other samples when used for guessing. This is likely a consequence of the composition rules that the 000webhost website enforced; while the other sites enforced no rules, 000webhost passwords must include both letters and numbers.

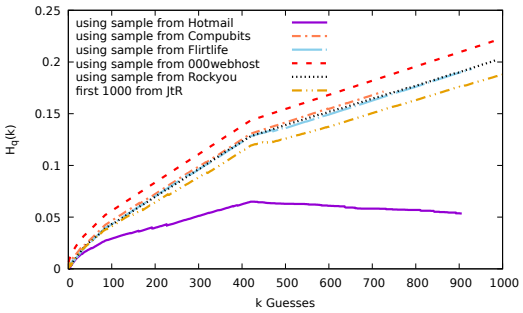
The John the Ripper set does a good job of guessing the 000webhost dataset, and the Rockyou sample is the next best option.

Figure 3.10e shows the Rockyou password set guessed using each of our five

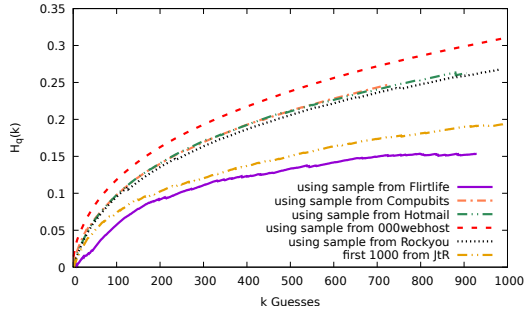
samples. The 000webhost sample is least effective at guessing the Rockyou dataset. This is followed by the Computerbits and Hotmail samples. There is very little difference between the returns from the Flirtlife sample and the Rockyou sample. However, when repeated for 100 trials, we find that the mean loss is higher for the Flirtlife sample; therefore, the Rockyou samples do overall perform better. Surprisingly, the John the Ripper wordlist outperforms the Rockyou sample. The Rockyou dataset is a wordlist option for the John the Ripper tool. It is likely that the Rockyou password distribution was used to form the contents and ordering of the passwords in the John the Ripper basic wordlist we are using. We suspect this explains why it can so effectively guess the Rockyou data and outperforms a sample of 1000 Rockyou users.



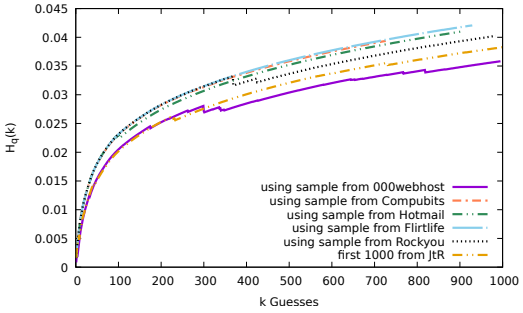
(a) Loss guessing Computerbits.ie dataset: $N = 1795$.



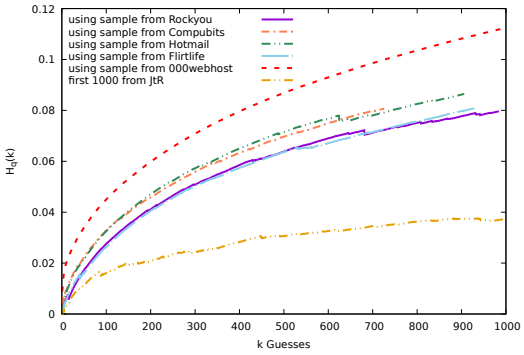
(b) Loss guessing Hotmail.com dataset: $N = 7300$.



(c) Loss guessing Flirtlife.de dataset: $N = 98,912$.



(d) Loss guessing 000webhost.com: $N = 15,252,206$.



(e) Loss guessing Rockyou.com dataset: $N = 32,602,877$.

Figure 3.10: Datasets guessed using $n = 1000$ samples from Computerbits, Hotmail, Flirtlife, 000webhost, Rockyou, and John the Ripper (JtR) with replacement.

3.8.3 Guessing Results: Without Replacement

Now, we report the results of our guessing without replacement. Again, we will find that when guessing using the sample chosen from the dataset, we are

guessing most effectively. In this case, there is less of a spread between the results from using the samples from different datasets. Therefore, in Figure 3.11, we show the results of 5 guessing trials for each sample. The results are generally much closer to the results of guessing using the established John the Ripper basic wordlist. We will discuss each graph in Figure 3.11 individually.

Figure 3.11a shows that the sample from Computerbits guesses the remaining users' passwords in Computerbits with less loss than guessing with samples from Rockyou, 000webhost, Flirtlife, or Hotmail. However, though Computerbits offers the best guessing, the loss as a result of guessing the Computerbits dataset is very high ($<\sim 0.6$) in comparison to the loss when guessing the other four datasets ($<\sim 0.3$).

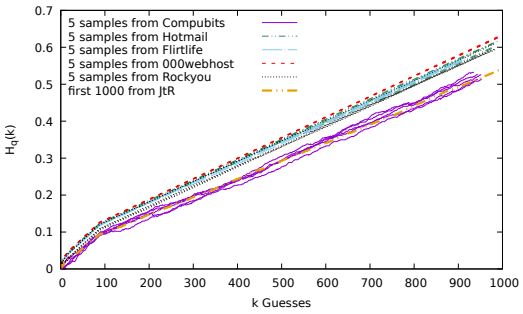
In Figure 3.11b, again, we see that guessing with the sample from the Hotmail dataset offers more successful guessing of Hotmail than guessing with the samples from the other datasets. Similar to the situation for the other datasets, the 000webhost sample gives the worst returns when used to guess the Hotmail dataset.

In Figure 3.11c, the sample from Flirtlife performs best at guessing Flirtlife, showing significant improvements over guessing with the other samples and over guessing with the JtR basic wordlist.

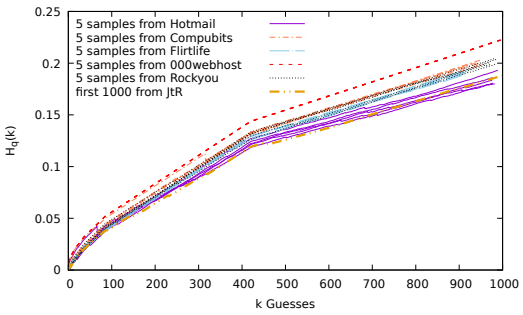
Figure 3.11d shows the sample from 000webhost guessing the remaining users' passwords in 000webhost better than the other samples and JtR. In fact, we can see that the loss when guessing 000webhost passwords is very low. For the 000webhost sample, we get a final loss of less than 0.035. That is, we failed to compromise just 3.5% of the users we could have optimally compromised in 1000 guesses. Even for guessing using the other datasets, we had a loss of less that 0.045 when guessing 000webhost.

Similar to when we guessed the Rockyou dataset using samples chosen with replacement, in Figure 3.10e, we again find that guessing using a Rockyou sample and a Flirtlife sample offer similar results. Running 100 trials, we get a 90% confidence interval for the end loss as a result of guessing Rockyou using samples from Rockyou of [0.0770, 0.083], and a 90% confidence interval for samples drawn from Flirtlife gives a loss interval of [0.0774, 0.083]. These are close, but one thing to note is that the loss is consistently increasing

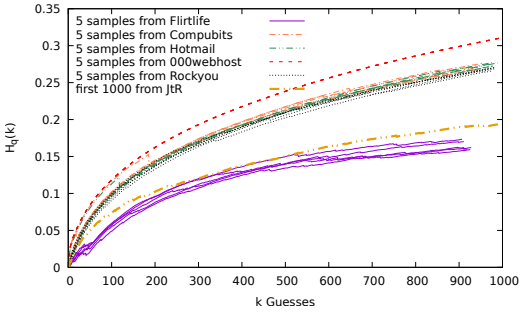
with each guess made. The Rockyou samples can make an average of 979 guesses, whereas Flirtlife samples make an average of only 912 guesses. In Figure 3.11e, we see that at the end of the guessing the Flirtlife loss stops increasing, whereas the Rockyou loss continues to increase for the additional guesses it makes, making the end result of the loss from guessing higher in comparison for Rockyou.



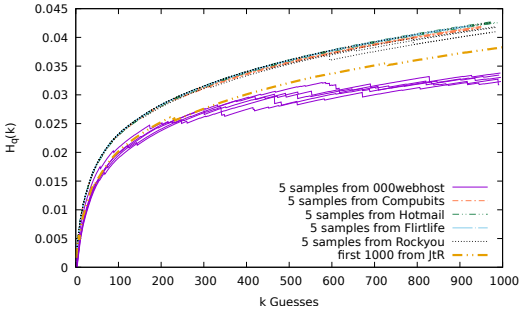
(a) Loss guessing Computerbits.ie dataset.



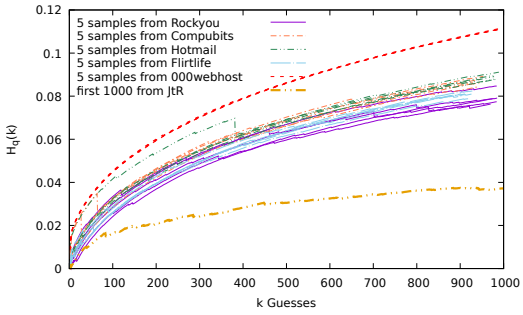
(b) Loss guessing Hotmail.com dataset.



(c) Loss guessing Flirtlife.de dataset.



(d) Loss guessing 000webhost.com dataset.



(e) Loss guessing Rockyou.com dataset.

Figure 3.11: Datasets guessed using samples, of size $n = 1000$, chosen without replacement from Computerbits, Hotmail, Flirtlife, 000webhost, Rockyou, and JtR.

3.9 Discussion

In Figure 3.11a, we find that a sample from Computerbits guesses the remaining users' passwords in Computerbits better than any of the samples from Rockyou, 000webhost, Flirtlife, or Hotmail. This is interesting because if we assume all users choose passwords with a certain distribution, then we would expect Rockyou with 32 million users' passwords to give the best approximation of that distribution. Instead, Computerbits, with just 1795 users, offers the best guessing success. This tells us that there are certain characteristics within a dataset, such as demographic of users and inclusion of site-specific terminology, that impact choice of passwords. This supports the work of other researchers [175, 98] and confirms that access to a small number of users' passwords from a particular website can be leveraged to compromise remaining users of the same site.

We have shown both theoretically and in practice that a sample of passwords taken from a dataset will help compromise the remaining users in that dataset. This research provides evidence supporting the advice: "Users should be prohibited from creating passwords obtained from previous breaches" [59, 30]. Our findings also highlight the importance of protecting every user of a website or online service. Sometimes, a small number of users can be compromised by phishing, guessing attacks, or other means, but, by proof of convergence of this sample towards the distribution of the whole dataset, we know that the information gained by the attackers about the passwords of these users can be used to compromise the remaining users in the dataset.

We observed that samples from the 000webhost dataset were less effective for guessing the other datasets we considered. It seems likely that this means that one should consider composition policy when doing cross-website password guessing. This suggests that, while passwords from the same source are most likely to compromise users, sources of passwords with matching composition policies are more likely to produce matches.

When guessing the 000webhost dataset, we found that we were able to guess with very little loss. Figure 3.11d shows that, using 1000 users to guess the 000webhost dataset results in loss less than 0.045, whereas guessing the Rockyou password set (Fig. 3.11e) results in a loss of up to 0.12. Looking at the frequency with which passwords are chosen can give us some insight into why

the loss for 000webhost is low. The 10 most popular passwords in the Rockyou dataset account for 2.05% of all the Rockyou users' passwords, whereas the 10 most popular 000webhost passwords account for 0.8% of the users' passwords. We conjecture that the introduction of the composition policy reduces the non-uniformity of the password distribution.

In 2016, Florêncio, Herley, and Van Oorschot pointed out that, once a few accounts are compromised, control over these can be leveraged to bring down a whole system [53]. They introduced the concept of a saturation point, α_{sat}, a value between 0 and 1 which represents the fraction of the user population that needs to be compromised before the entire system can be said to be overrun. The authors suggest that, once $\alpha = 0.5$, a system can be considered to be completely overrun. At this point, there would be very few resources the attacker cannot access, and, due to malicious actions from the compromised accounts, such as spam, the system would become unusable. In our work on this topic [115], we have demonstrated that if we have a sample which includes just 10% of the users if flirtlife.de, then we can leverage the passwords in this sample to compromise approximately 45% of the remaining users in the Flirtlife set. Thus, jumping the number of compromised accounts much closer to the saturation point.

We are impressed by how well the basic John the Ripper wordlist does at guessing. When guessing without replacement, the basic JtR wordlist, at times, performs well at guessing the Computerbits and Hotmail datasets and is consistently best at guessing the Rockyou dataset. The strength of JtR lies in mangling rules. We believe this work provides evidence for the power of seeding JtR input wordlists with passwords gained from previous small leaks from the same website. Mangling rules can also then be applied to these inputs.

3.10 Conclusion

In this chapter, we introduced a model for measuring the effectiveness of password guessing. Leveraging this model, we proved that, by using a sample of passwords, we can capably guess the passwords in a dataset with a loss which converges to zero as the number of samples gets large. We also provided variations of these proofs which allow for a small amount of guessing loss and,

therefore, more accurately describe the empirical data and the motivations of an attacker. Our guessing model is used to visualize the effectiveness of samples at guessing and allows us to compare the performance of different samples at guessing datasets of different sizes. We demonstrated that organizations face the risk of subsequent compromises when a subset of their users' passwords are revealed. We have shown both theoretically and in practice that a sample of passwords taken from a dataset will help compromise the remaining users in that dataset effectively.

CHAPTER 4

Multi-armed bandit approach to password guessing

The multi-armed bandit is a mathematical interpretation of the problem a gambler faces when confronted with a number of different machines (bandits). The gambler wants to explore different machines to discover which machine offers the best rewards, but simultaneously wants to exploit the most profitable machine. A password guesser is faced with a similar dilemma. They have lists of leaked password sets, dictionaries of words, and demographic information about the users, but they don't know which dictionary will reap the best rewards. In this chapter, we provide a framework for using the multi-armed bandit problem in the context of the password guesser and use some examples to show that it can be effective. This chapter is based on research presented at WAY '20 [117] and PasswordsCon '20.

4.1 Introduction

Passwords are a widely used form of authentication online. However, one major weakness of passwords is that human chosen passwords can be easily guessed by automated attacks. In fact, with the regular occurrence of leaks of password datasets, attackers are provided with an increasing amount of data to inform password guesses. It is important for security advocates and researchers to understand the capabilities of attackers given they have access

to this data. This way, we can create countermeasures to protect the security of Internet users.

Users from similar demographics will often choose similar passwords [98, 116]. In addition, users have been observed choosing passwords that reflect the nature of the website they are choosing the password for [175]. In this chapter, we investigate whether an automated learning algorithm can identify these idiosyncrasies within a password set and whether it can leverage this knowledge in order to improve the success of password guessing rates.

A commonly used method for guessing passwords involves using dictionaries of password guesses and guessing these in an optimum order in order to compromise as many users as possible. Guessing passwords in the optimal order is important for an attacker as they wish to compromise many users with a small number of guesses. In particular, an online attacker is often using automated attacks and is limited to a certain number of guesses before a lockout is triggered. It is a challenge for an attacker to discover which guesses will result in the highest success rates.

In this chapter, we suggest and develop an *explore and exploit* protocol based on the classic multi-armed bandit problem. This protocol can be used to guess a password set effectively using guesses from a selection of dictionaries. Because of the learning nature of the model over a small number of guesses, we view this as a method which could prove effective as an online guessing method.

In Section 4.2 we describe related work. In Section 4.3 we describe the multi-armed bandit problem and Section 4.4 describes the formal set-up of the bandit problem in the context of password guessing. Section 4.5 describes the tools used for solving the problem and offers a validation of these methods. Section 4.6 introduces a selection of variable values which can be used within the model. Section 4.7 runs the full multi-armed bandit model on a selection of simulated password sets. We demonstrate the model's ability to assign appropriate weightings to dictionaries and that, given these weightings, we can effectively guess the password sets. Section 4.8 discusses these results and Section 4.9 uses the results to inform decisions regarding the optimal variable values for use in the model.

Given that we were able to differentiate characteristics within the simulated password sets, in Section 4.10 we show that the multi-armed bandit can match password sets and dictionaries created by users of the same nationality. We also demonstrate that using a subset of passwords from users known to be of the same nationality as those whose passwords exist in the password set will improve guessing successes over passwords from a general population of users.

Finally, in Section 4.11 we investigate whether using the multi-armed bandit model can be used to improve guessing for a password set. We find that the multi-armed bandit does offer improvements over direct guessing with a single dictionary or guessing from a dictionary at random. However, we acknowledge that the set-up of the bandit could be altered in order to be optimised for guessing successes. Section 4.12 discusses the results of our multi-armed bandit work.

4.2 Related work

For a long time researchers have been interested in modelling and improving password guessing. The first strategic methods involved dictionary attacks. These were proposed by Morris and Thompson in 1979 [106] and are still widely used today in the form of John the Ripper [80] and HashCat [63]. HashCat takes dictionaries of words and for the first word in the dictionary HashCat applies *word mangling* rules to it. For example, the first word is password and HashCat applies mangling rules to guess password, password1, Password, P@ssword, P@$$W0rd etc. Then HashCat moves onto the next word and applies the same word mangling rules to it. John the ripper has a similar approach but it takes the first mangling rule and runs through the whole dictionary with it. Then moves onto the next mangling rule. In both cases, a dictionary of ordered guesses is required. This is usually made up of previous leaked password datasets, a dictionary of words, a dictionary of names and maybe a list of particular guesses based on the demographics of the individuals whose passwords we hope to guess.

In 2005, Narayanan and Shmatikov employed Markov models to enable faster dictionary attacks [118]. A Markov model is a stochastic model used to model a randomly changing system. A typical Markov model predicts the next character in a sequence based on the current character. Markov models

are a standard technique used in natural language processing. Narayanan and Shmatikov applied them to passwords and were able to significantly reduce the search space and improve guessing returns for a selection of password datasets.

In 2009, Weir et al. used probabilistic context-free grammars (PCFG) to guess passwords [178]. Probabilistic context free grammars are used in computational linguistics to help model and understand the structure of languages. Instead of characterising passwords character by character, a PCFG characterises a password according to its "structures". Structures can be password guesses or word mangling templates that can take dictionary words as input. Weir et al. automated the creation of probabilistic context-free grammars for passwords by training them on password breaches and then used them to assign probabilities to passwords for guessing. Researchers have developed Weir et al.'s PCFG into a tailored password cracking system [27, 35] and into a measurement system for password guessability [83].

In 2013, Dürmuth et al. proposed an updated password guessing model based on Markov models, called OMEN [42]. As part of their initial paper they demonstrated an OMEN specific method for merging personal information with a dictionary of guesses [21]. They acknowledged the difficulty of merging guesses from two different sources. Independently, Ma et al. also showed that a correctly tailed Markov model will outperform PCFGs at password guessing [97].

In 2016, Wang et al. developed a targeted password guessing model which seeds guesses using users' personally identifiable information [173]. Wang et al. leverage existing probabilistic techniques including Markov models and PCFG as well as Bayesian theory. They create tags for specific personally identifying information (PII) associated with a user. In their most successful version, TarGuess-I, they use training with these type-based PII tags to create a semantic aware PCFG. Independently, Li et al. also created a method for seeding password guesses with personal information. Their guessing also extended the probabilistic context free grammar method [94].

Also in 2016, the use of artificial neural networks for password guessing was proposed by Melicher et al. [102]. Artificial Neural networks (or, simply, neural networks) are computation models inspired by biological neural networks. Artificial neural networks are a machine learning technique particularly useful

for fuzzy classification problems and generating novel sequences (such as a password not in the training data). Melicher et al. show that their neural network method can be more effective than both Markov and PCFG methods. In addition, because neural networks can be highly compressed, they show that they can be used to efficiently carry out client-side password strength checking. In 2017, Houshmand and Aggarwal created a method for merging multiple grammars for dictionary-based PCFG models [70].

Hitaj et al. proposed using deep generative adversarial networks (GAN) to create password guesses [68]. A generative adversarial network pits one neural network against another in a zero-sum game. A GAN is able to generate new data which exhibits the same distribution as the training set. PassGAN is able to autonomously learn the distribution of real passwords from leaked passwords and can leverage these to generate guesses. In contrast to Markov and PCFG models, PassGAN does not require a-priori knowledge of password structures.

In 2018, Liu et al. suggested a deep learning model of password guessing which could combine PCFG with the artificial recurrent neural network LSTM [180]. This method, called GENPass, was designed to overcome the limitation of neural networks that means they can not, in their raw form, be used for cross-site attacks. This was an important contribution as leveraging passwords leaked from one website to use to guess the same users' passwords on another site is a common attack.

Pal et al. in 2019, developed a password manipulation tool called PASS2PATH [126]. Leveraging the knowledge that users alter and reuse their passwords, this guessing model can transform a base user password into effective targeted password guesses.

Probably the most similar to our work is work by Pasquini et al. [127]. In 2019 they introduced the idea of "password strong locality" to describe the grouping together of passwords that share fine-grained characteristics. By grouping passwords into these localities, they can then train their learning model to generate passwords that are similar to those that it has previously seen. For example, passwords with the structure "jimmy****" would all be grouped together such that "jimmybean" and "jimmybear" are both guess options within the group. In this way, conditional password guessing becomes

possible, for example if the first five characters are known and the remaining four need to be guessed. This password locality can also be leveraged to help with password guessing. In particular, it allows the guessing of passwords that are unique to the attacked password set as they will be guessed if they share a locality with already guessed passwords.

Determining the order to make password guesses is a hard problem. Particularly when drawing from multiple sources which often do not provide obviously compatible probability scores to order them.

The most widely used guessing strategy involves combining wordlists with word mangling rules. The success of this strategy, which is demonstrated by Hashcat and John the Ripper, is best highlighted through the success of both teams in the annual KoreLogic "Crack Me If You Can" contest, and also in the wide spread use of both software [88, 80, 63]. Our research expands on the existing literature and practices by exploring a method for effectively ordering the input wordlist.

A password guesser will have access to information from multiple sources: previous password leaks, language dictionaries, site specific terms and demographic information about the users, among other things. In this chapter, we describe our work optimising the choice of guesses by combining the information from all the available sources (called dictionaries).

This is a new piece of research that, to the best of our knowledge, has not been tackled analytically. Previous research has focused on creating words to guess. We investigate the optimum ordering of the wordlist which can contain words from multiple sources including PCFG created words or simply from leaked password sets. Our contributions are as follows:

- Our learning algorithm develops its learning about the distribution of the password set it is guessing with every guess made. Importantly, it requires no a-priori training.
- Its adaptive nature allows it to react to new information and tailor the guessing strategy accordingly and in real-time. This strategy makes it a particularly dominant strategy for automated and adaptive online guessing. In simulations we show that within 1 to 10 guesses the model

can accurately determine the distribution of a password set.

- Given a password set formed by users predominantly from a single nationality our model can accurately and efficiently recognise this characteristic and tailor the guessing to use an appropriate dictionary.
- We also show that this improves guessing, revealing that using passwords generated by other users from that same nationality will improve guessing over using a general ranked wordlist (even when language is not a factor, i.e. both nationalities use the same language).
- In many previous wordlist approaches, a single ordered wordlist is created. In our method, wordlists are separated based on their source or characteristics. In this chapter, we demonstrate that this method allows for effective guessing from the promising dictionaries which are tailored to the characteristics discovered within the password set.
- We show that a method (denoted $\mathcal{Q}$-method) of guessing which combines guesses from multiple dictionaries according to the perceived weightings of those dictionaries, is generally more effective at guessing than any single wordlist of leaked passwords.

This adaptive learning model demonstrates that a password guesser can learn about a password set with each guess made. This demonstrates the effectiveness of dynamic real-time analysis of guessing returns.

4.3 The multi-armed bandit problem

The multi-armed bandit problem describes the trade-off a gambler faces when faced with a number of different gambling machines [156]. Each machine provides a random reward from a probability distribution specific to that machine. The crucial problem the gambler faces is how much time to spend *exploring* different machines and how much time to spend *exploiting* the machine that seems to offer the best rewards. The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls.

In our scenario, we regard each dictionary as a machine which will give a certain distribution of successes. We want to explore the returns from each dictionary and also exploit the most promising dictionary, in order to make

effective guesses. With each guess we learn more about the distribution of the password set we are trying to guess. Leveraging this knowledge, we can guess using the dictionary that best matches the password set distribution, thus maximising rewards.

In the following sections we describe our set up of the multi-armed bandit problem in the context of password guessing.

4.4 Password guessing problem set up

Suppose we have n dictionaries. Each dictionary $i = 1 \dots n$, has a probability distribution p_i, and $\sigma_i(k)$ denotes the position of password k in dictionary i. So, the probability assigned to password k in dictionary i is $p_{i,\sigma_i(k)}$.

Suppose we make m guesses where the words guessed are k_j for $j = 1 \dots m$. Each of these words is guessed against the N users in the password set and we find N_j, the number of users' passwords compromised with guess number j.

To model the password set that we are trying to guess, we suppose it has been generated by choosing passwords from our n dictionaries. Let q_i be the proportion of passwords from dictionary i that generated the password set. Our aim will be to estimate $q_1, \dots, q_n$ noting that

$$\sum_{i}^{n} q_i = 1 \qquad \text{and} \qquad q_i \geq 0. \tag{4.1}$$

This means that the q_i are coordinates of a point in a probability simplex. If the password set was really composed from the dictionaries with proportions q_i, the probability of seeing password k in the password set would be

$$Q_k := \sum_{i=1}^{n} q_i p_{i,\sigma_i(k)}. \tag{4.2}$$

Given the N_j, we will use this probability to build a maximum likelihood estimator.

4.5 Maximum likelihood estimation

Given this problem set up, we can construct a likelihood function which will describe the likelihood that a given set of parameters $q_1, \ldots, q_n$ accurately describe the password set. In this section, we introduce the likelihood estimator function, prove that a unique maximum value exists and demonstrate the convergence to this maximum.

4.5.1 Likelihood estimator

Maximum likelihood estimation is a method for estimating the parameters of a probability distribution using observed data [44]. It does so by selecting the parameters so that the observed data is most probable.

We construct the following likelihood for our model with m guesses:

$$\mathcal{L} = \binom{N}{N_1 \cdots N_m \, (N - N_1 \cdots - N_m)} \cdot Q_{k_1}^{N_1} Q_{k_2}^{N_2} \cdots Q_{k_m}^{N_m} \cdot (1 - Q_{k_1} \cdots - Q_{k_m})^{N - N_1 \cdots - N_m}, \tag{4.3}$$

where the first term is the multinomial coefficient representing each unique way the successes can be structured. The second term denotes how many times password k_j is expected to be seen in the passwordset, $\mathcal{Q}_k$, to the power of how many times it was actually seen. The final term represents the remaining guesses and states that they account for the remaining users' passwords in the passwordset that have not yet been compromised.

Our goal is to maximise this likelihood function by choosing good estimates for $q_1, \ldots q_n$ based on our observed rewards from each previous guess. Note, with each guess we learn more about q_i for all the dictionaries. In fact, one of the features of this model compared to a traditional multi-armed bandit model is that when we make a guess we learn something about all the dictionaries.

We can take the log of the likelihood function to create a simplified expression. In addition, we can remove the multinomial which is simply a constant for any values of $\vec{Q}$. This leaves us with:

$$\log \mathcal{L} = \text{const} + N_1 \log Q_{k_1} + N_2 \log Q_{k_2} + \cdots + N_m \log Q_{k_m} \\ + (N - N_1 - \cdots - N_m) \log (1 - Q_{k_1} - \cdots - Q_{k_m}). \tag{4.4}$$

4.5.2 Maximising the likelihood function

Except in limited cases, the likelihood equation (Equation 4.3) cannot be solved explicitly. We were able to prove that the log-likelihood function for our system is concave, and therefore the likelihood function has a unique maximum value.

Theorem 4.5.1 (Concavity of log likelihood function)**.** *The log likelihood function,* $\log \mathcal{L}$*, for* $\mathcal{L}$ *defined in 4.3 is concave.*

Proof: Required to prove that

$$\log \mathcal{L}(\alpha\vec{q} + (1-\alpha)\vec{r}) \geq \alpha \log \mathcal{L}(\vec{q}) + (1-\alpha) \log \mathcal{L}(\vec{r}).$$

We begin with a simplification of the notation used in the Likelihood function by converting to vector notation.

Let

$$g(\vec{Q}) = \sum_{j=1}^{m+1} N_j \log Q_j$$

where $N_{m+1} = N - \sum_{j=1}^{m} N_j$ and $Q_{m+1} = 1 - \sum_{j=1}^{m} Q_j$.

As before (4.2), $Q_j = \sum_{i=1}^{n} q_i p_{i,\sigma_i(k_j)}$ and therefore

$$Q_{m+1} = 1 - \sum_{j=1}^{m} \sum_{i=1}^{n} q_i p_{i,\sigma_i(k_j)} = \sum_{i=1}^{n} q_i (1 - \sum_{j=1}^{m} p_{i,\sigma_i(k_j)}).$$

Now, let

$$\log \mathcal{L}(\vec{q}) = \sum_{j=1}^{m+1} N_j \log Q_j = g(\vec{Q}), \text{ where } \vec{Q} = P\vec{q}$$

Therefore, we can now rewrite

$$\begin{aligned}\log \mathcal{L}(\alpha\vec{q} + (1-\alpha)\vec{r}) &= g\left(P(\alpha\vec{q} + (1-\alpha)\vec{r})\right) \\ &= g\left(\alpha P\vec{q} + (1-\alpha)P\vec{r}\right) \\ &= \sum_{j=1}^{m+1} N_j \log(\alpha(P\vec{q})_j + (1-\alpha)(P\vec{r})_j)\end{aligned}$$

But, log is a concave function so we know that for any $a, b \in \mathcal{R}^+$ $\log(\alpha(a) + (1-\alpha)(b)) \geq \alpha\log(a) + (1-\alpha)\log(b)$. Therefore for $a = (P\vec{q})_j$ and $b = (P\vec{r})_j$:

$$\begin{aligned}&\geq \sum_{j=1}^{m+1} N_j(\alpha \log(P\vec{q})_j + (1-\alpha)\log(P\vec{r})_j) \\ &= \alpha \sum_{j=1}^{m+1} N_j \log(P\vec{q})_j + (1-\alpha) \sum_{j=1}^{m+1} N_j \log(P\vec{r})_j \\ &= \alpha g(\vec{Q}) + (1-\alpha) g(\vec{R}) \\ &= \alpha \log \mathcal{L}(\vec{q}) + (1-\alpha)\log \mathcal{L}(\vec{r})\end{aligned}$$

□

Because the log-likelihood function is concave, any local maximum we find will be a global maximum. We use gradient descent techniques to find the q_i that maximises $\mathcal{L}$ after m guesses subject to the constraints in (4.1). We iteratively change the estimated q_i values, $\hat{q}_i$.

4.5.3 Maximising within a constrained environment

As we iterate through the gradient descent, it is important that we stay within the constraints of the system. In particular, we are required to meet the following constraints:

$$\sum_{i}^{n} q_i = 1 \qquad \text{and} \qquad q_i \geq 0.$$

To meet these requirements we project the gradient vector onto a probability simplex and then constrain our steps so that we always stay within that space. Projecting the gradient vector onto the probability simplex corresponds to adding a constant to each entry of the gradient so that the entries sum to 0.

We then move in the direction of this gradient with checks that we never go outside the boundaries of the simplex.

With each iteration of our gradient descent we move a step in the direction which maximises our likelihood function. The size of the possible step is computed within β and then the actual size of the step taken is scaled by α (α is a gradient descent constant described in Section 4.6.3):

$$\text{Move from } \vec{p} \text{ to } \vec{p} + \alpha\beta\vec{g}.$$

For each gradient descent we compute the value β such that:

- $\beta \leq 1$
- $\beta||\vec{g}|| \leq 1$
- $\vec{p}+\beta\vec{g}$ lies within the simplex. i.e. $0 \leq \vec{p}+\beta\vec{g} \leq 1$, where the inequalities hold entrywise.

We can now use this gradient descent method to estimate the parameters $q_1, \ldots, q_n$ which maximise the likelihood function. These parameters $q_1, \ldots, q_n$ denote the weightings assigned to the dictionaries and are informed based on the information collected from each password guess.

4.5.4 Gradient descent validation

The goal of the gradient descent is to converge towards the maximum of the likelihood function and thus find the proportions q_i that provide the best explanation of the distribution of the password set seen after m guesses. For initial validation of the gradient descent performance we take four different leaked password datasets as dictionaries.

The datasets we used are leaked passwords from users of hotmail.com, flirtlife.de, computerbits.ie and 000webhost.com[1]. They contain 7300, 98912, 1795 and 15,252,206 users' passwords respectively. The datasets were compromised by various methods so the lists may only contain a random, and possibly bi-

[1]These are the same datasets as those used and described in Section 3.5.1.

ased, sample of users [116]. As far as we can tell only the 000webhost dataset imposed composition[2] policies on users [56].

We took a random sample of 1000 users' passwords from the 98912 users in the Flirtlife dataset. In Chapter 3, we showed that when guessing a sample of leaked passwords from a website, the most effective guesses will come from the passwords of other users of that same site [116]. Therefore, if the gradient descent is effective we expect it to show that the sample most closely compares to the Flirtlife dictionary.

In Figure 4.1, we show the q value estimates during the convergence of the gradient descent for the likelihood function seeded with just one guess. This single guess was the password *123456* because it is widely considered to be the most commonly used password. We began by setting the $\hat{q}$-values to $1/n = 0.25$, and then used 100 steps of gradient descent with a scaling of $\alpha = 0.1$ on the step size to estimate the proportions. We recorded the q_i values that, during the gradient descent, gave the maximum Likelihood value. This way, we remember the best value if we overstep the maximum. As expected, the estimation suggests that a high proportion of the passwords in the sample were drawn from the flirtlife.de dictionary.

The password *123456* occurred in all four password sets but using the distribution of those datasets the likelihood function was able to determine that the proportion in the sample best matched the proportion in Flirtlife. If we were guessing the full Flirtlife dataset with several guesses, rather than just a sample from it with one guess, then this proportion will be closer to 100%.

The optimization assigns a non-zero proportion to two of the other dictionaries. This implies these dictionaries hold some guessing value. As we collect more information from additional guesses we expect that the proportion assigned to those dictionaries will converge to zero.

In the above example, all 1000 passwords came from a random sample of Flirtlife. We will investigate in later examples whether the maximum likelihood estimation can determine the breakdown of where passwords come from

[2]Composition of a password describes what characters are allowed or must be included in the password. For example, a password composition policy which is often used is "you must include an uppercase character, a lowercase character, a number and symbol in your password".

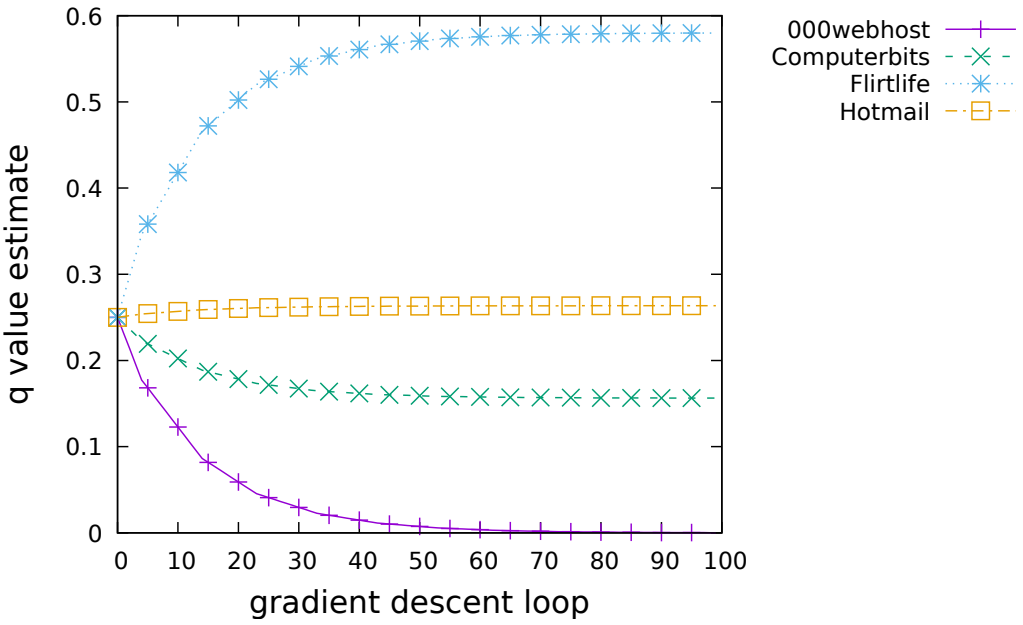


Figure 4.1: Estimating the distribution of a password set using information from 1 guess and $\alpha = 0.1$.

when composed of different dictionaries.

4.6 Variables in the multi-armed bandit model

Let us now suggest some variations of the gradient descent and the maximum likelihood estimates when forming a password guessing multi-armed bandit. We are interested in which of these variations produces the best results.

4.6.1 Gradient descent initialization variables

We expect the gradient descent to improve with each guess made since every guess provides it with more information. There are a number of ways of initialising the gradient descent after each guess provides new information. The following are three different methods for choosing the initialisation value:

Random Randomly pick starting values for $\hat{q}_i$, subject to (4.1),

Average Choose the average starting value, i.e. assume the passwords are distributed evenly between the n dictionaries, so $\hat{q}_i = 1/n$,

Best Use our previous best estimate for the $\hat{q}$-values, based on the gradient descent results for the previous guess.

4.6.2 Informing our guess choices

Once we have generated our estimate of the $\hat{q}$-values, we want to use them to inform our next guess. We suggest three options for how to choose our next guess:

Random Randomly choose a dictionary and guess the next most popular password in that dictionary.

Best dictionary Guess the next most popular password from the dictionary with the highest corresponding $\hat{q}$-value.

$\mathcal{Q}$-method Use all the information from all the $\hat{q}$-values for the dictionaries and all the frequencies of the passwords in the dictionaries to inform our next guess.

These options have different advantages. In the first option, we randomly choose a dictionary to guess from, but we are still taking the most probable guess from the dictionary we choose. This option emphasises the continued exploration of all the dictionaries. In the second option, we are choosing the dictionary we believe accounts for the largest proportion of the password set.

The last option is specifically basing password guess choices on equation (4.2). It uses our predicted $\hat{q}$-values to estimate the probability of seeing each word k. If, for example, we have a word k which has frequency $f_1(k)$ in dictionary 1 but also occurs in dictionary 2 and 3 with frequencies $f_2(k)$ and $f_3(k)$ respectively. Using equation (4.2), where $p_{i,\sigma_i(k)} = f_i(k)/\textit{size of dictionary } i$, we can compute the total probability of this word occurring in the password set. This method should determine which word k has the highest probability of being in the password set and use this word as our next guess.

4.6.3 Gradient descent step size

As part of our gradient descent convergence we iteratively take steps in the direction of the likelihood maximum. The scaling on the size of these steps taken is a variable α as part of our system. This step size variable, α, was

introduced in Section 4.5.3. A step size which is too large could fail to converge to the maximum and instead overstep it, a step size too small means we may never reach the maximum value. There are a number of options for choosing this step size, three popular options are listed below:

Constant alpha Let α be a small constant. For example 0.1. This setup means that the step size will become smaller as we converge towards the maximum.

Constant step size Keep the step size constant by choosing an α value that depends on the size of the step taken, $\alpha = \frac{\epsilon}{||\vec{g}||}$. A downside of this fixed step size is that when we get close to the maximum, we could take a fixed size step past it.

Adaptive The step size can be adapted based on the perceived gains. For example, if the Likelihood function is increasing, then we might try increasing α on the next step. Whereas, if the movement will make the function decrease then we can scale back α until we get a value which will not cause a decrease. This option means that the function will definitely increase on each step. However, it is important to verify that we remain within the probability simplex with each adaption of α.

Advantages and disadvantages exist for each choice for computing the step-size variable. The choice of the α value is important as a poor α value could mean the system never converges to the maximum. The choice of an effective step-size variable is dependant on the specific system.

4.7 Multi-armed bandit Validation

We will now look at some examples of the performance of our multi-armed bandit model against simulated password sets. Guessing against simulated password sets allows us to identify whether the multi-armed bandit model is capable of identifying the characteristics of a password set (i.e. the q_i) in a controlled environment. It also allows us to compare and contrast the effectiveness of the different model variables (introduced above in Section 4.6).

In these simulations we guess one word at a time and then compute the estimated weighting of each dictionary. We also report separately the number

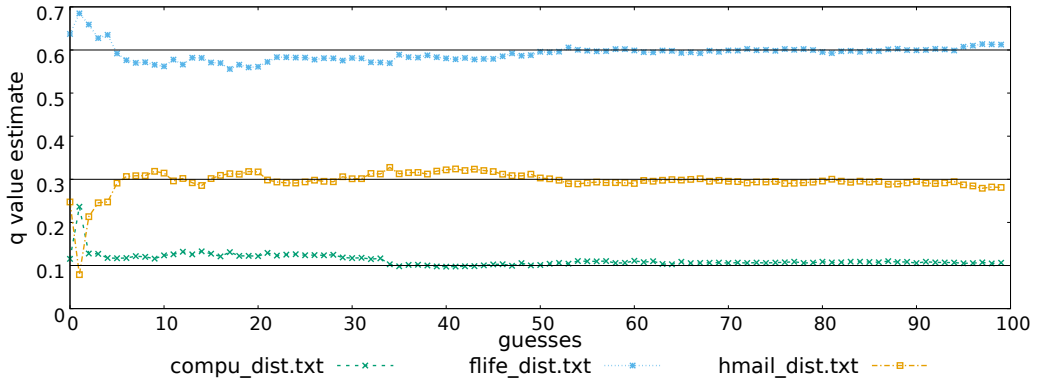


Figure 4.2: Password set 1 q-value estimates. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

of users compromised after each guess is made. We hope to show that the simulations are able to approximate the true distribution of the password set. We also expect to find that the "$\mathcal{Q}$-method" of guessing is more effective than a random dictionary choice strategy and we could also find that it is better or as good as simply guessing using the estimated "best" dictionary.

In each of the below plots we have used the **constant alpha** method for computing the gradient step size and set this value to 0.1. We show the q-value estimates plot for the best combination of initialisation and guess choice methods. We discuss the alternative methods later in Section 4.9.

4.7.1 Password set 1: 60% Flirtlife, 30% Hotmail, 10% Computerbits

We begin by creating a password set made up of 10,000 users' passwords; 60% were selected randomly from the flirtlife.de dataset, 30% from the hotmail.com dataset and 10% from the computerbits.ie dataset.

In Figure 4.2, we plot the estimated q-values after the gradient descent was completed for each guess. For this graph, the gradient descent was initialised using average $\hat{q}$-values, $\hat{q}_i = 1/3$, and the $\mathcal{Q}$-method was used for guessing. The actual proportions are shown as solid horizontal lines. Even after a small number of guesses we have good predictions for how the password set is distributed between the three dictionaries.

In Figure 4.3, we show the number of users successfully compromised as the

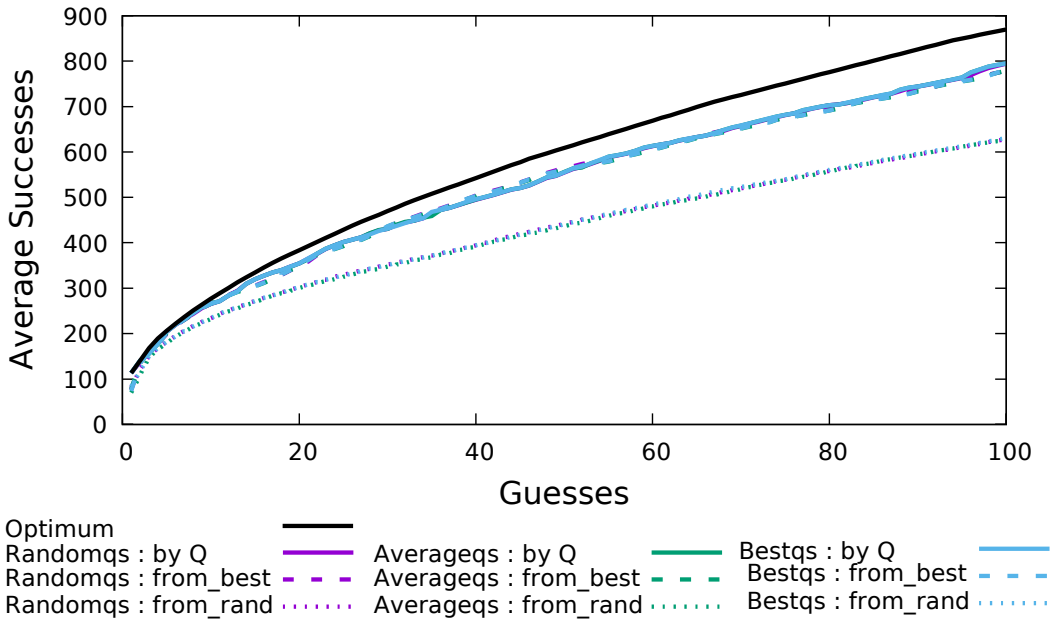


Figure 4.3: Guessing returns for password set 1.

number of guesses increases. The successes are the average over fifty runs to reduce the variance in the random guessing method. Results are shown for each combination of initialisation and guessing method. As one might expect, picking guesses from a dictionary at random resulted in the lowest success rates. Both the $\mathcal{Q}$-method and guessing from the best dictionary resulted in successes very close to the optimal line. After 100 guesses these methods had compromised 795 users, in comparison to the 870 users compromised by guessing the correct password in the correct order for every guess. In this case all the initialisation methods (random, average and best qs) perform similarly.

4.7.2 Password set 2: 60% 000webhost, 30% Hotmail, 10% Computerbits

In Figure 4.4, we show the estimated q-values for a 10,000 user password set made from 000webhost, Hotmail and Computerbits with a 6:3:1 split. Again, we get good estimates for the q-values. As the 000webhost passwords had composition rules in force but the other dictionaries did not, we may see different behaviour for the guessing successes.

In Figure 4.5, we show the guessing success rate. Interestingly, in this case, guessing from the best dictionary performed even worse than guessing from a

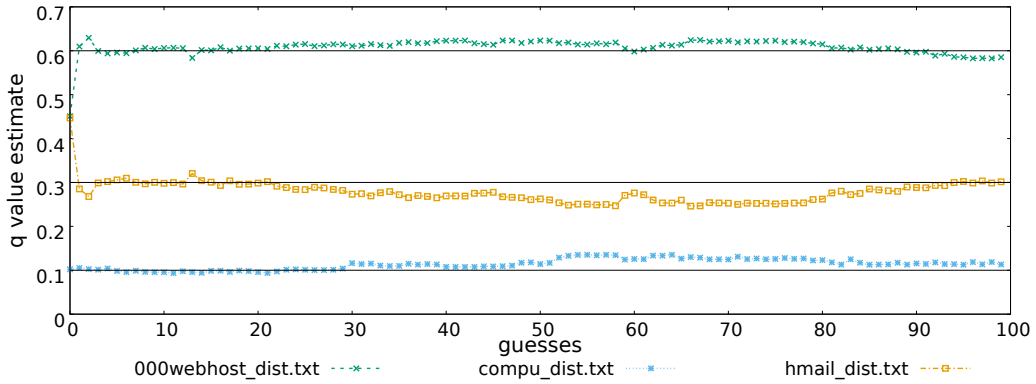


Figure 4.4: Password set 2 q-value estimates. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

random dictionary. In Chapter 3, we discovered that the 000webhost dataset is particularly poor at guessing Hotmail and Computerbits users' passwords. According to the characteristics of this password set, 000webhost is accurately weighted as accounting for the largest proportion of password set. However, when we guess solely from the best ranked dictionary (dashed lines) we get lower guessing returns than when we randomly choose a dictionary to guess from (dotted lines). This highlights an important distinction between optimising our model for effective guessing and optimising to best represent the characteristics within the password set (this is explored further in Section 4.11.1). As per Chapter 3, we hypothesise that the low guessing success of 000webhost is due to it being the only dataset we analyse which included rules on how passwords should be formatted [56].

It is worth noting that the $\mathcal{Q}$-method of guessing would also be skewed by the high ranking of the 000webhost passwords and their low guessing success. However, it still performs slightly better than the random method, and significantly better than guessing from the best dictionary (avg. results over 100 trials).

All the initialisation methods in this password set that guess from the ranked best dictionary do poorly. However, initialising with random q-values (purple dashed line) does perform better than other initialisation methods (green and blue dashed lines) when guessing using the best dictionary method. We believe this is a feature of the occasional failure of the random q-value to successfully converge. We will discuss this more in Section 4.9.2. In brief, when q-values

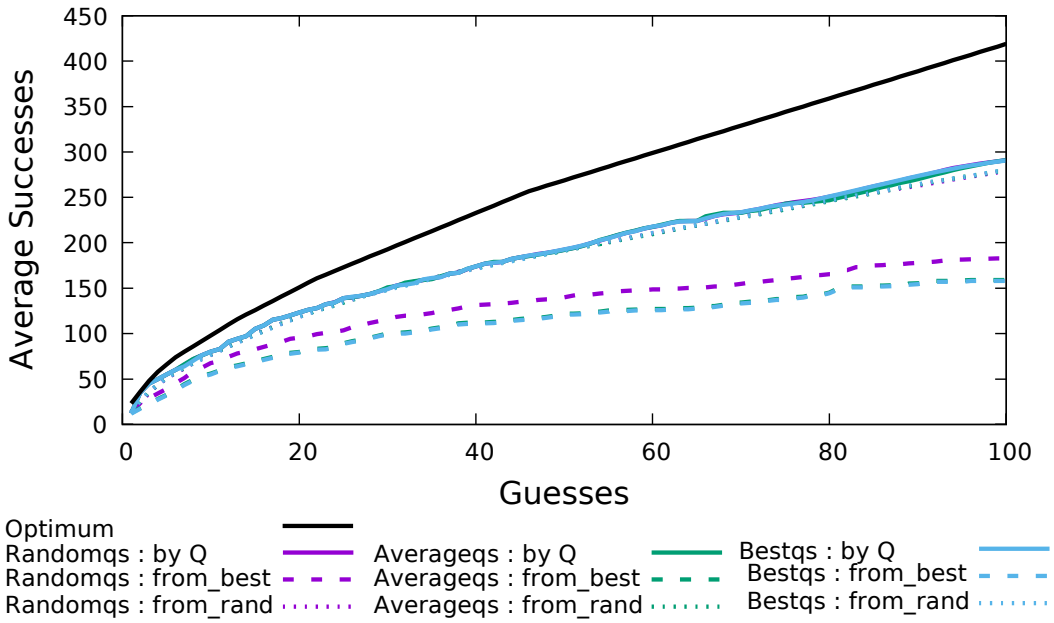


Figure 4.5: Guessing returns for password set 2.

are initialised randomly, one dictionary can begin by being ranked very high in comparison to the others. In this case, it is sometimes possible that the gradient descent does not have sufficient time to converge. In this case, a password set other than 000webhost can be ranked as best. Analysing the returns, it is this that allows the random method to perform better than the other initialisation methods when guesses are chosen from the best ranked dictionary (i.e. guessing returns are occasionally improved when 000webhost is not ranked as the best).

4.7.3 Password set 3: 60% Hotmail, 30% Flirtlife, 10% Computerbits

In Figure 4.6, we show the estimated q-values for a 10,000 user password set made from Hotmail, Flirtlife and Computerbits with a 6:3:1 split. The approximation for the Computerbits dictionary falls slightly below the correct level and the Flirtlife estimate is slightly above. The approximation for the strongest dictionary, Hotmail, is accurate. The estimates have mostly converged by guess 10 and there is little divergence after that point.

Figure 4.7 shows the guessing success rate for this password set. In this case we see that the $\mathcal{Q}$-method fares better than the random and best dictionary

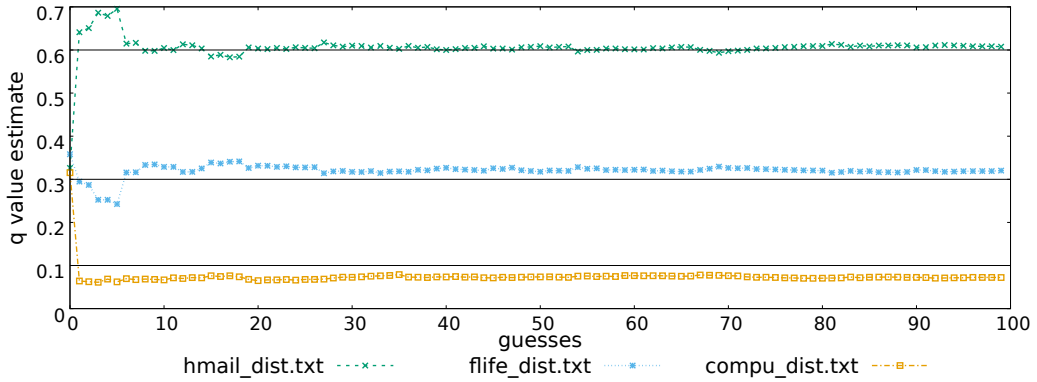


Figure 4.6: Password set 3 q-value estimates. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

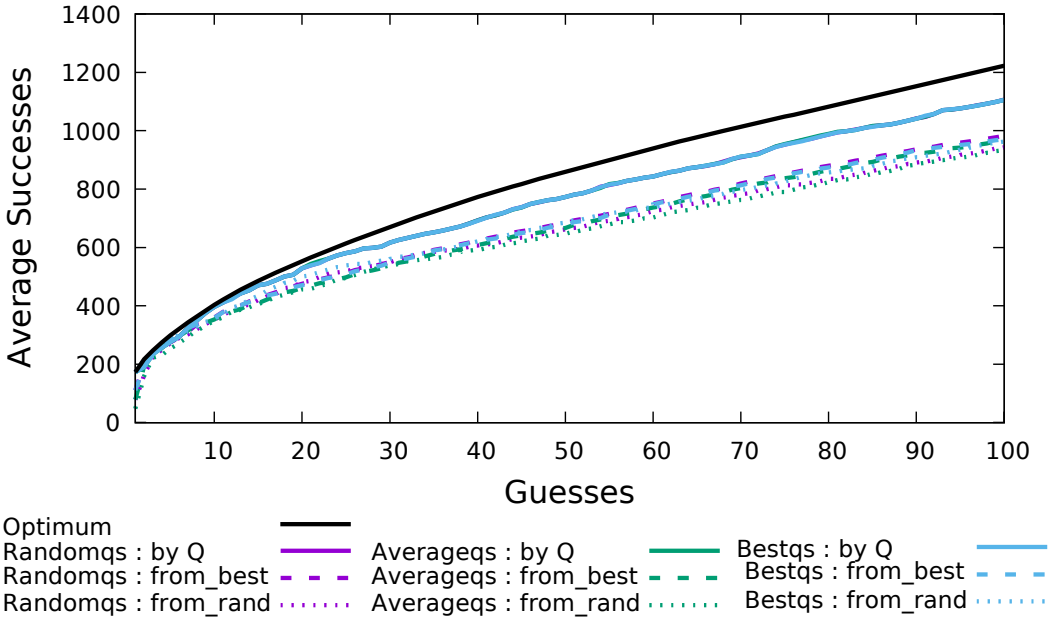


Figure 4.7: Guessing returns for password set 3.

methods. In fact, it is close to the optimal guessing method. By the end of the guessing the $\mathcal{Q}$-method has compromised an average of 1106 users. An optimal strategy at this point would have compromised 1223 users. The best dictionary method compromised an average of 980 users and random compromised an average of 962 over 5 runs. We see very little difference in success rates for the different initialisation methods within each guess choice.

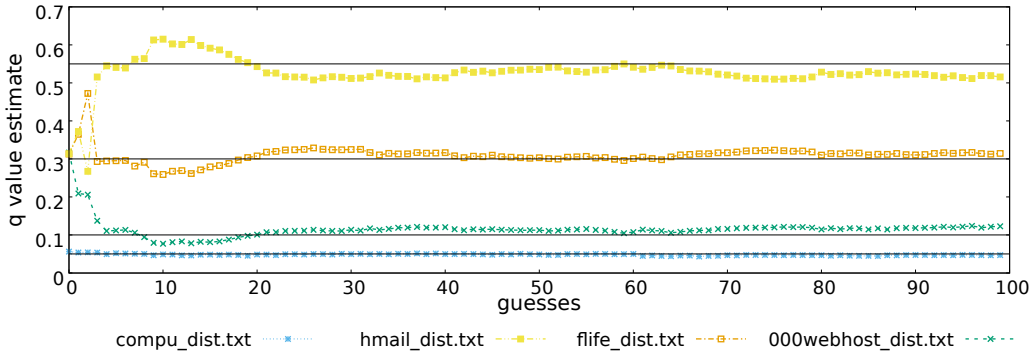


Figure 4.8: Password set 4 q-value estimates. Initialization: average $\hat{q}$-values, Guessing: best dictionary.

4.7.4 Password set 4: 55% hotmail, 30% flirtlife, 10% 000webhost, 5% Computerbits

The final password set we look at is composed of 10,000 users' passwords from 4 different dictionaries. In Figure 4.8, we display the estimated q-values. The model gives an accurate approximation of the q-values. Figure 4.9 shows the successes when guessing this password set. Again, we see that the $\mathcal{Q}$-method is effective at guessing, this time performing significantly better than the other guessing methods. We notice that the successes are close to the optimal. Particularly for the first 20 guesses, where the $\mathcal{Q}$-method compromised 303 users in comparison to 317 compromised by optimal guessing.

4.8 Discussion of results for simulated password sets

The multi-armed bandit automation is able to match characteristics in a password set to characteristics in the dictionaries used for guessing. We have seen that for a variety of synthetic examples, guessing using the multi-armed bandit technique can be effective both for compromising users and estimating how the passwords have been chosen.

In all examples we saw that guessing using the $\mathcal{Q}$-method is consistently effective in comparison to other dictionary selection methods. In general, we found that the initialization method had little bearing on the success results. This stems from the concave nature of our log-likelihood function, meaning

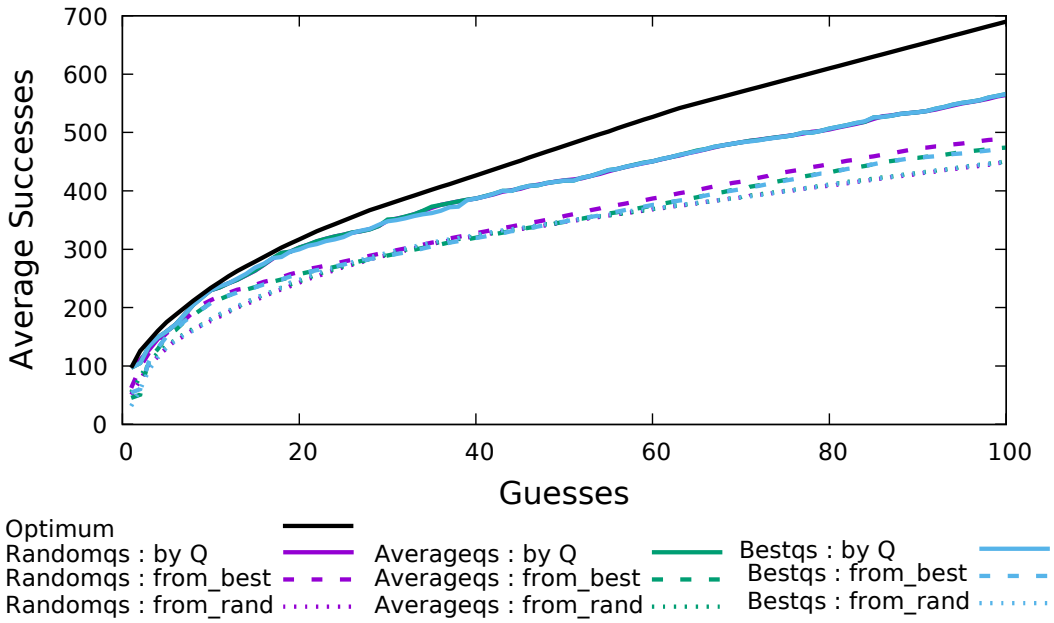


Figure 4.9: Guessing returns for password set 4.

that, for most set-ups, the function will converge to a single maximum when estimating the distributions.

These initial results demonstrate that the relationship between password choice and user cohorts is tangible and identifiable by automation. We are now motivated to investigate whether these characteristics can be identified in real password leaks. In the next section, Section 4.9, we investigate the optimum set-up for our multi-armed bandit model. Leveraging this, we then explore whether the demographic information of users in a password leak can be determined (Section 4.10.1); whether this information can successfully inform guessing (Section 4.10.2); and whether a multi-armed bandit can outperform simple dictionaries of guesses at password guessing (Section 4.10.2).

4.9 Choosing variables in the multi-armed bandit model

By creating and guessing simulated password sets we have been able to test the various variables within the set up of our multi-armed bandit model. In this section we will discuss each variable type and discuss the value of this variable that we found worked most effectively. In future experiments, this will allow us to generate results with the model set up effectively.

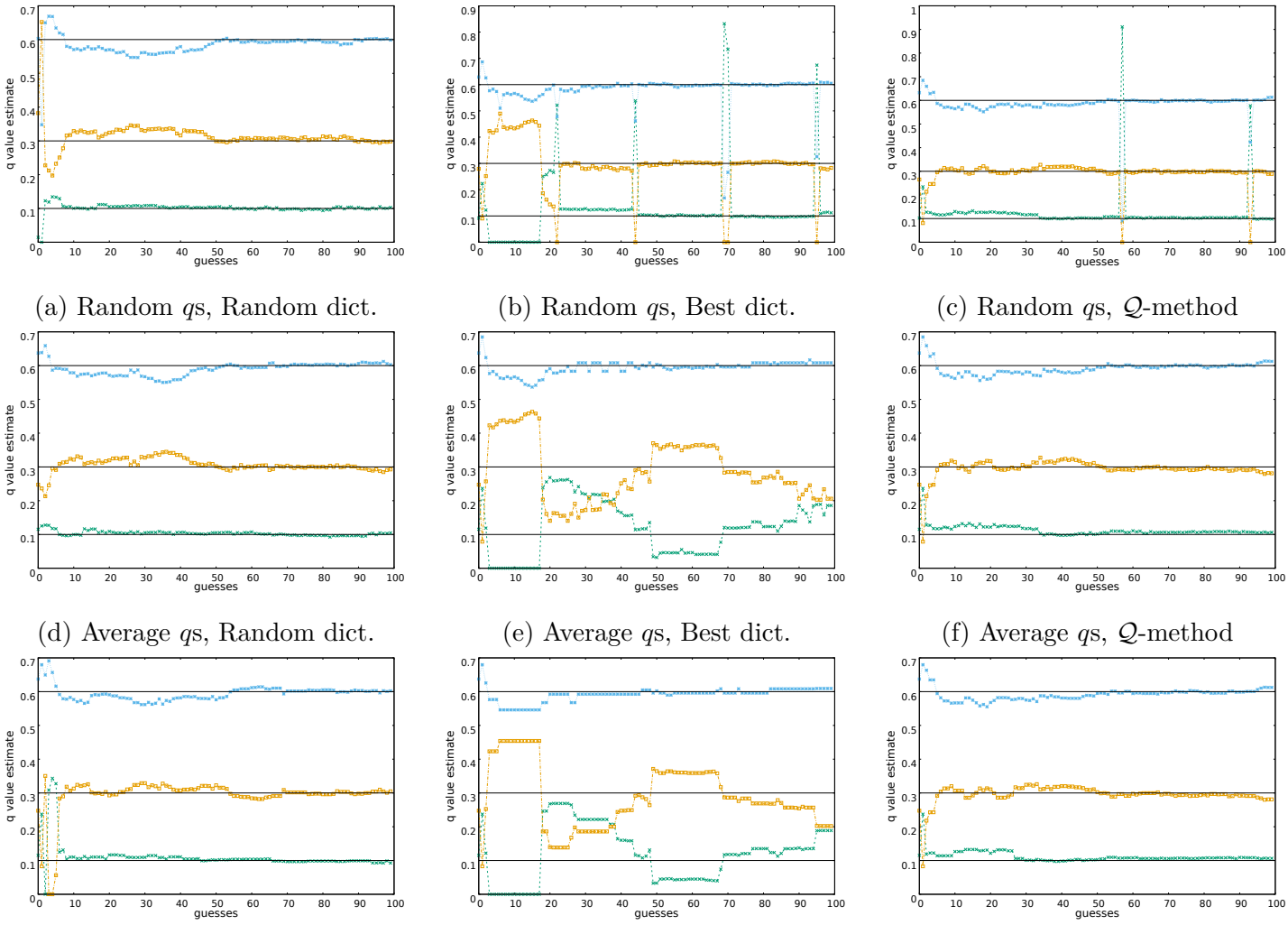
4.9.1 Variables for optimising success

In Section 4.6, we introduced 3 methods for informing our guess choice: Random dictionary, Best dictionary, and $\mathcal{Q}$-method. We also included three methods for initialising the q-values: Random, Average and Best. In Section 4.7, we plotted the successes for each combination of the variables. We found that all the options which informed the guess choice using the $\mathcal{Q}$-method produced the best results. However, there was little variation in output between the different initialisation methods. In Figure 4.5 and Figure 4.9 we found that initialising with a random q-value performed marginally better than the other two options when used in conjunction with the Best dictionary guessing choice method. Therefore, if a single method is desired then using the Random method of initialising the q-values and the $\mathcal{Q}$-method for informing the guess choice seems to offer the best returns for compromising users.

4.9.2 Variables for estimating the characteristics of the password set

When estimating the q-values assigned to each dictionary, the variables in the multi-armed bandit play an important role.

Let us first look at the nine graphs produced by combinations of the guess choice and initialisation variables for password set 1 (as defined in Section 4.7.1). The results are shown in Figure 4.10.



(a) Random qs, Random dict.

(b) Random qs, Best dict.

(c) Random qs, $\mathcal{Q}$-method

(d) Average qs, Random dict.

(e) Average qs, Best dict.

(f) Average qs, $\mathcal{Q}$-method

(g) Best qs, Random dict.

(h) Best qs, Best dict.

(i) Best qs, $\mathcal{Q}$-method

Figure 4.10: Password set 1 q-value estimates. Shown for each combination of initialisation and guess choice method.

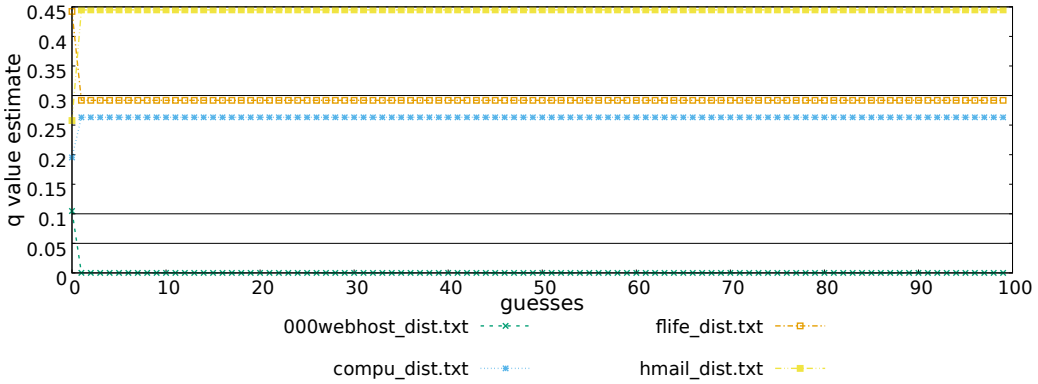


Figure 4.11: Password set 4 q-value estimates. Initialization: best $\hat{q}$-values, Guessing: best dictionary.

Guess choice: Best dictionary We can see that the best dictionary method for choosing guesses is not effective for determining the characteristics of a password set. This is because of the emphasis on exploitation of the best dictionary over exploring all the dictionaries. Where there is large overlap between the passwords in the dictionaries, the best method will provide information about the distribution of each dictionary. However, when this is not the case we do not learn about the relationship between the password set and all the dictionaries. A particularly poor set up is guessing from the best dictionary and initialising gradient descent with the previous best values. Figure 4.11 shows the outcome of this set-up for password set 4. Notice that despite incorrectly approximating the 0.55:0.3:0.1:0.05 split of the password set, it does not diverge from its initial approximation. Below we will explain why this can occur in the model.

Initialisation: Best qs When the number of guesses is less than the number of dictionaries the likelihood function can be degenerate. This means multiple combinations of q-values can maximise the likelihood function. This is not as much of an issue in the Average and Random initialisation methods but in the Best qs initialisation method we are seeding the next guess with the best q-values from our previous guess. But because we are constrained by the probability simplex defined by (4.1), movement to leave this initial approximation can be constrained. See Figure 4.12 depicting a 3-dictionary probability simplex which shows $\vec{q}$ in a corner of the simplex. q-values in a corner position are limited in the valid directions they can move in. This

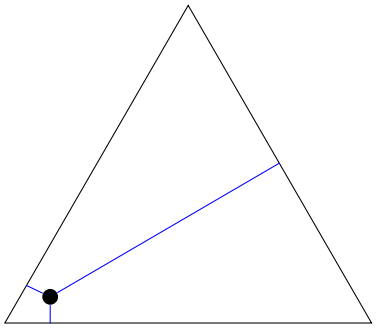


Figure 4.12: Diagram impression of a probability simplex for a 3-dictionary set. Depicts q-values starting in one corner of the simplex.

creates the potential to become "stuck". It is further emphasised when the Best dictionary method is used for guess choice as it is compounded now by limiting the information gathered about any password set other than the one estimated as the best. A simple solution is to reset the q-values before each new guess is made, as is done in the Average values method. A more complex solution could involve avoiding seeding guesses until a non-degenerate likelihood function can be derived.

Initialisation: Random qs Notice the spikes in two of the graphs which show initialisation with random q-values.[3] These spikes represent a failure to converge when the randomly chosen initial $\hat{q}$ values are far from the true values. In our other simulated password sets, these spikes were also present in the graphs which used random qs for initialisation and a random dictionary for guessing. For this reason we will avoid using the random initialisation method when determining the make-up of a password set.

4.9.2.1 Conclusion

Based on this analysis, we find that the results from the models initialised using both Random q-values and previous Best q-values are not reliable. In addition when guesses are chosen from the Best dictionary only, the model does not fare well at approximating the q-values. The Average q initialisation method paired with either the Random dictionary guess choice or the $\mathcal{Q}$-method guess choice performs consistently well at approximating charac-

[3]Because the random initialisation is non-deterministic, these spikes can occur to a greater or lesser extent in different iterations. To demonstrate the existence of the spikes, we have intentionally chosen graphs from an iteration in which the spikes are prominent.

teristics. One advantage of the $\mathcal{Q}$-method over the random dictionary choice is that it is deterministic.

4.9.3 Variables for Gradient descent step size

As well as different options for initialisation and guess choice, we also have a number of choices for the gradient descent step sizes. In Section 4.6, we introduced three popular options for the step size variable within gradient descent literature [161]. Each method has advantages and disadvantages and it is important to choose a method that works for a given set-up of the multi-armed bandit.

We implemented these three options in four ways:

- Constant alpha
- Constant step size
- Starting with constant alpha and then adapting step size
- Starting with constant step size and then adapting.

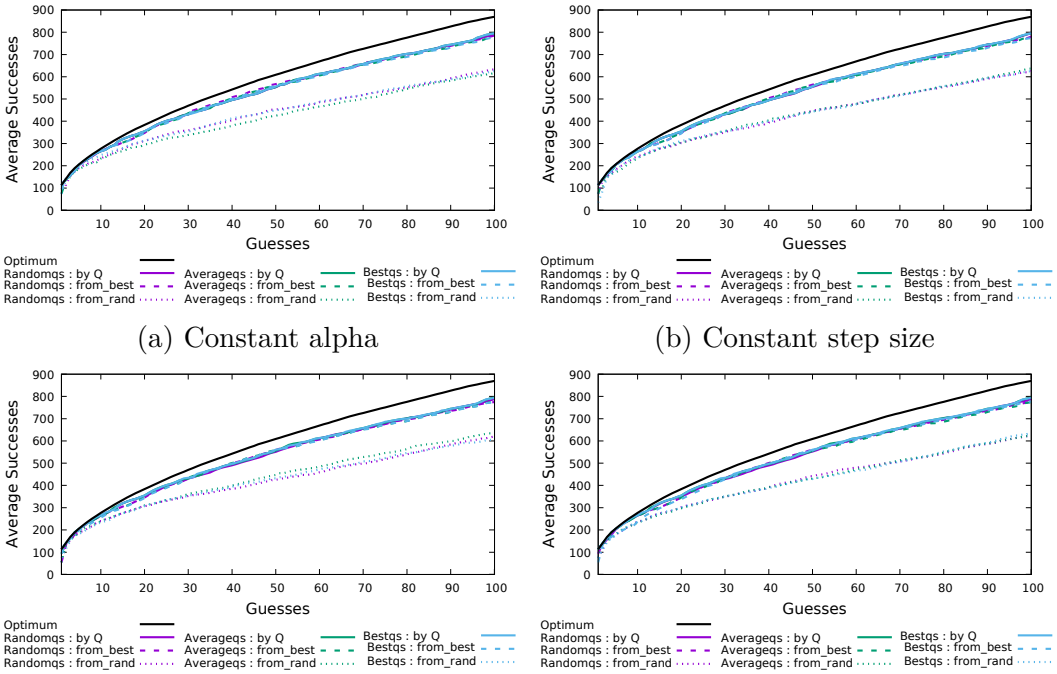
Recall the adaptation involves increasing the step size if it will result in an increased likelihood value and similarly decreasing the step size incrementally until the resulting likelihood is an improvement on the last likelihood. Thus we can guarantee that each step results in an increase to the likelihood function.

4.9.3.1 Successes

Figure 4.13 shows the success plots for password set 1 using the 4 different step size methods. There does not appear to be a significant difference in the success rates for the different step size options.

4.9.3.2 Estimating the q-values

Our goal is to achieve an accurate prediction of the q-values for a selection of dictionaries to help us determine the characteristics of the password set. We therefore compare how effectively the step size options estimate the q-values. We can do this in two ways. By looking directly at the q-values and visually comparing them against the actual values. Alternatively by plotting



(a) Constant alpha

(b) Constant step size

(c) Adapting constant alpha

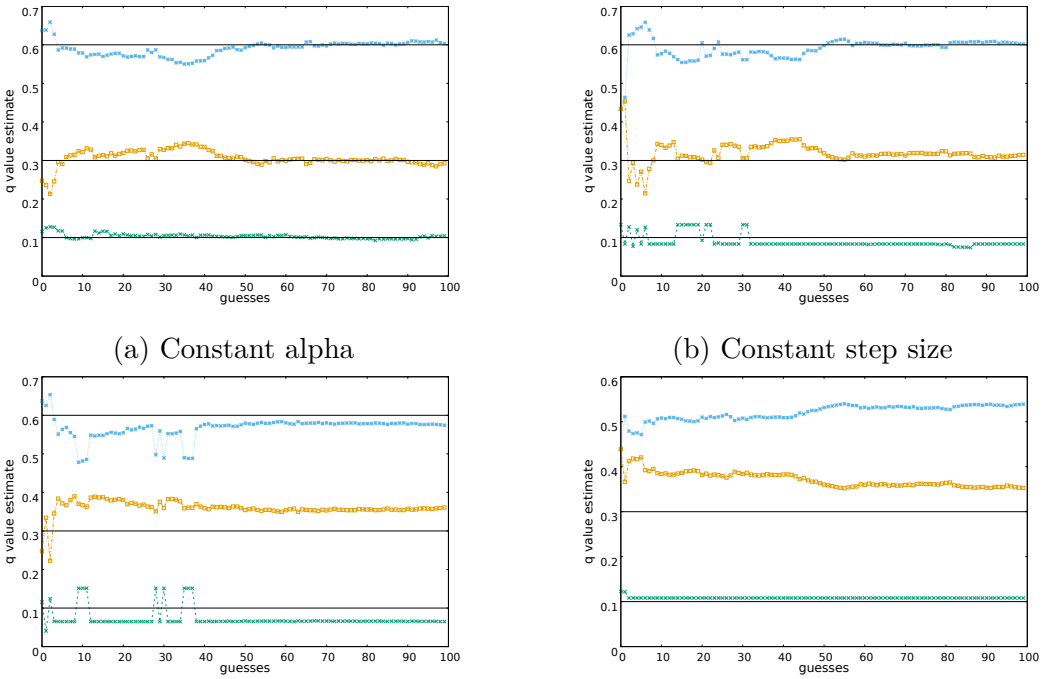
(d) Adapting constant step size

Figure 4.13: Password set 1 guessing successes. Shown for each combination of gradient descent step size methods.

the distance the q-values are from the optimal and analysing this graph. Below we do both and see that both show us that the constant alpha approach seems to work best with our model.

For a single initialisation and guess choice In the previous section we determined that the Average qs Random dictionary setup was a reliable set-up for estimating the q values. Therefore, in Figure 4.14 we show the Average qs Random dictionary set-up of the model for estimating password set 1 for the four step size options. We can see that constant alpha seems to give the best approximation of the true q-values for this set-up of the initialisation and guess choice. Both adaptive options give a poor estimation and the constant step size option performs only slightly worse than constant alpha.

L_1 norms An L_1 norm is the sum of the magnitudes of vectors in a space. It can be used to define a metric which is the sum of the absolute difference of the components of the vectors.



(a) Constant alpha

(b) Constant step size

(c) Adapting constant alpha

(d) Adapting constant step size

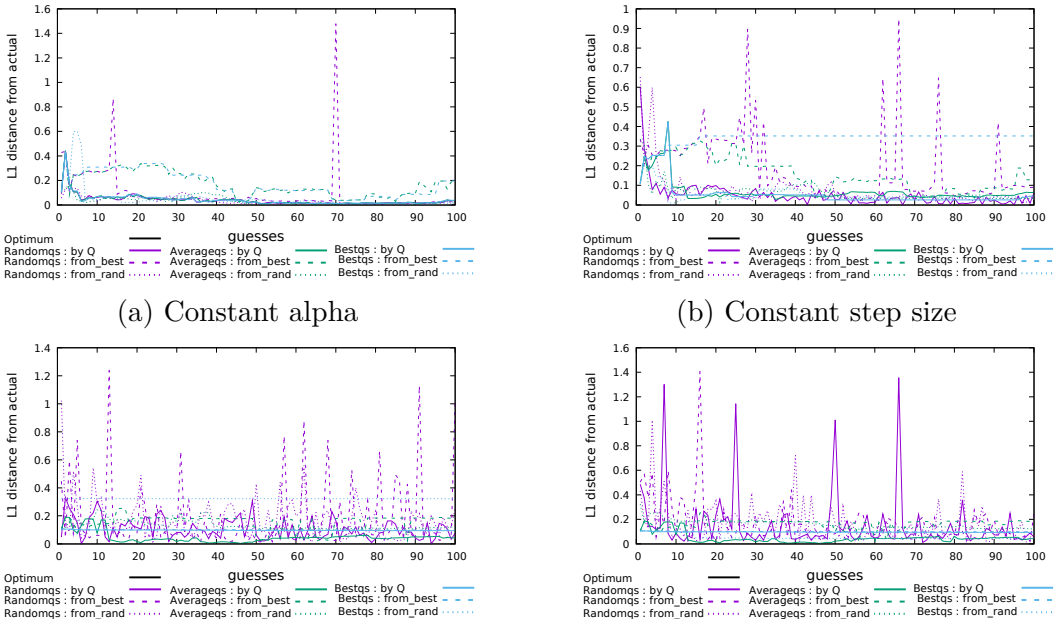
Figure 4.14: Password set 1 q-value estimates. Shown for each combination of gradient descent step size methods.

$$||x||_1 = \sum_{i=1}^{n} |x|.$$

In this norm, all the components of the vector are weighted equally. We can use the L_1 norm to measure the distance between the actual and estimated q values at each guessing point. The result is the sum of the differences between the actual and estimated q for each dictionary.

Figure 4.15 compares the L_1 norms for the four step methods. In Figure 4.15a we see the L_1 norms for the constant alpha set up. All of the $\mathcal{Q}$-method approximations have L_1 norms consistently close to zero after an initial peak within the first 3 guesses. We see spikes in the Random qs Best dictionary plot and the other two Best dictionary options perform poorly. Random dictionary choice performs well with an L_1 norm generally less than 0.1.

Figure 4.15b shows the L_1 norms for the constant step size method. We see regular spikes in the Random qs Best dictionary method. There is more variation in the $\mathcal{Q}$-method results. The L_1 norm for the Best qs Best dictionary



(a) Constant alpha

(b) Constant step size

(c) Adapting constant alpha

(d) Adapting constant step size

Figure 4.15: Password set 1 L_1 norms. Shown for each combination of gradient descent step size methods.

method never converges to zero.

The general impression from the two adaptive plots (Figures 4.15c and 4.15d) is that they are inconsistent in their estimation. We see spikes in all the Random q initialisation methods. The adapting constant step size method reports reasonably small L_1 values when initialised using the Average qs method and choosing guesses using the $\mathcal{Q}$-method, or when it is initialised using Best qs and guesses are chosen from a Random dictionary. The adapting constant alpha method only shows low L_1 values for Average qs initialisation and $\mathcal{Q}$-method guess choice.

From this analysis we can conclude that the best estimates for the q values are provided using the constant alpha method for determining step size. This method involved a set alpha included in the iteration which results in a reduced step size as we approach the maximum. For our model we used the value $\alpha = 0.1$ though other values were also tested including 0.5 and 0.01.

In this section we have covered in detail the variables that are used in our multi-armed bandit model. It is important for us to test the combination of variables on the data we know the actual composition of. This allows us to

compare against true values. Going forward we will test our bandit model on password sets for which we hold no information about. It is important that we have tested our model and tailored it in a way that verifies the results of these exotic password sets.

4.10 Demographics

Users' demographics such as nationality and language play an important role in their password choices [98, 3, 62]. We are interested in two key questions:

- Given a password set formed by users predominantly from a single nationality. Can the multi-armed bandit recognise which dictionary best matches this locality.
- Does using passwords generated by other users from that same nationality improve guessing?

4.10.1 Matching nationality characteristics

In Chapter 3.5.1, we introduced 5 password datasets. Of these, two have a distinct demographic origin. One is the 2006 Flirtlife password leak which revealed over 100,000 passwords from the German dating site Flirtlife.de. Its users are believed to be predominantly German and Turkish. In 2009, 1795 users' passwords were leaked from the Irish website Computerbits.ie.

Computerbits will represent an Irish password set. We will use Flirtlife as an example of a German password set.

We formed dictionaries to guess these password sets. We required a dictionary which included passwords created by Irish users and one with passwords by German users.

In 2019, over 773 million passwords and email address pairs were leaked in a collection called Collection #1 [74]. This collection contains 31 sub collections, believed to have been collected over 2000 site breaches. I could verify the authenticity of two of these sub collections by identifying my own old credentials in the breach.

We chose to work with one of these collections titled ‘‘`Collection__#1_NEW_combo_semi_private_EU_combo.tar.gz`’’. This contains 366,471,471 unique

email address password pairs. From this we extracted all the passwords whose corresponding email address contained the country code top-level domain ".ie" and separately ".de". These formed our nationality specific user dictionaries from Ireland and Germany with 90,583 and 6,541,691 users respectively.

Irish passwords We are interested in whether the multi-armed bandit will match the distribution of the Irish password set computerbits.ie to the extrapolated Irish dictionary taken from the subset of Collection #1 (denoted "*Irish users*" from now on). To test this we ran the multi-armed bandit set up as per the optimal parameters found in Section 4.9.2.

In Figure 4.16, we included three dictionaries, the hotmail.com leaked passwords, the flirtlife.de password set and the *Irish users*. Hotmail.com is an international website. However, it is suspected that the Hotmail users in the dataset we have were compromised by means of phishing scams aimed at the Latino community. Flirtlife is a dating site with predominantly German and Turkish users. Figure 4.16 plots the breakdown estimated by the multi-armed bandit. From the first guess it estimates that the passwords in the computerbits.ie set match closely to the passwords chosen by the Irish users from the Collection #1 subset. This plot was initialised using average $\hat{q}$-values and guesses were chosen using the $\mathcal{Q}$-method. However, every combination of initialisation and guess choice provided an estimate which assigned the *Irish users* dictionary a rating substantially higher than the other dictionaries. Notice that some weighting is assigned to the Hotmail dictionary but none to the flirtlife.de password set.

In Figure 4.17 we included four dictionaries. The additional dictionary we include is the Rockyou.com password set leaked in 2009. It includes 32 million users' passwords and had an international audience. The language used on Rockyou applications was English. Given the common spoken tongue in Ireland is English and that the Rockyou password set is often used as an effective base to seed guessing, we expect the Rockyou users to be somewhat representative of the Irish Computerbits users.

Figure 4.17 shows the estimated breakdown for the computerbits.ie passwords. In the beginning Rockyou is assigned a weighting nearly as high as the *Irish users*. However, the value of Rockyou declines as the number of guesses in-

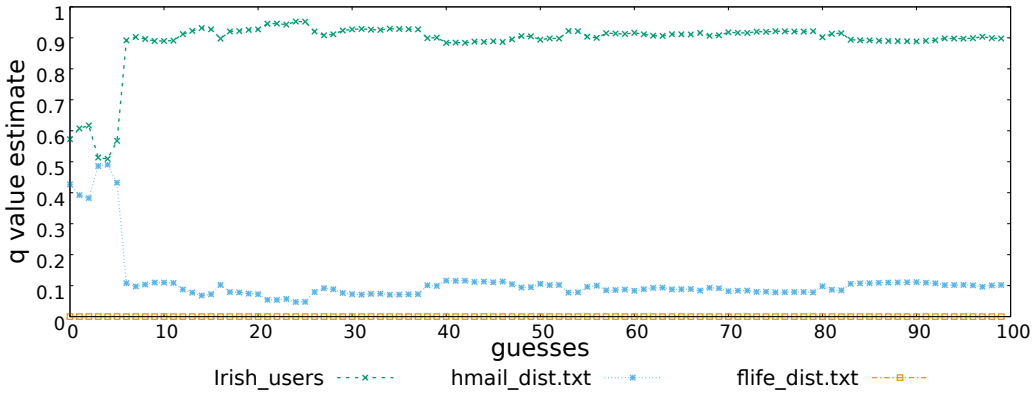


Figure 4.16: q-value estimates for the Irish password set from Computerbits.ie estimated using three dictionaries. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

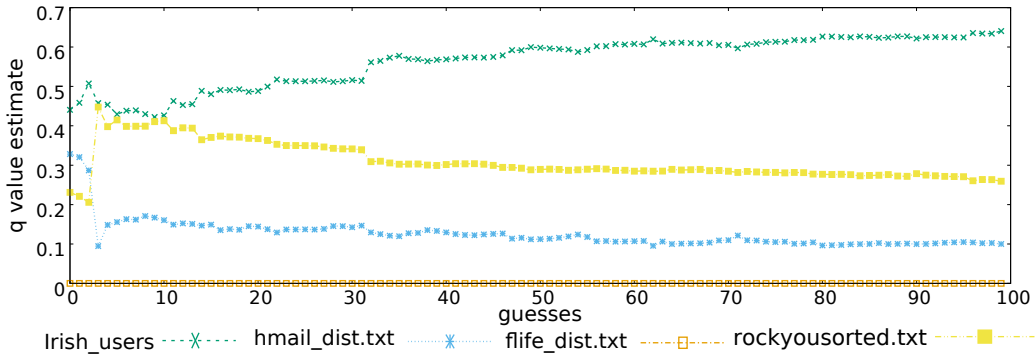


Figure 4.17: q-value estimates for the Irish password set from Computerbits.ie estimated using four dictionaries. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

creases. In all combinations of initialisation and guess methods, the multi-armed bandit was able to identify that the Computerbits passwords most closely matched the Irish subset of users.

German password We now try guess the flirtlife.de password set using the dictionary of *German users*. While flirtlife.de is a German dating site, its main users were both German and Turkish. Therefore, we do expect the multi-armed bandit to find it more difficult to match it to the solely German user dictionary.

Figure 4.18 shows the approximation of how flirtlife is characterized by three dictionaries: *German users*, rockyou.com and hotmail.com. Up until 50 guesses,

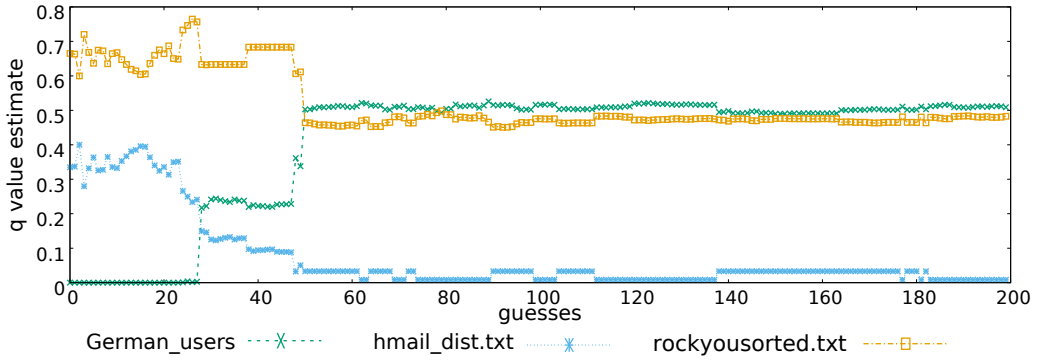


Figure 4.18: q-value estimates for the German password set from flirtlife.de estimated using three dictionaries. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

most weighting is assigned to the Rockyou dictionary. However after 50 guesses, the German users' passwords overtake Rockyou and remain slightly ahead up to at least guess 200.

The multi-armed bandit was still able to identify that the characteristics in Flirtlife best matched those of the German users. However, the effect does not take place until the high frequency passwords, up to 50, have been guessed. This is likely a result of Flirtlife not being a solely German password set and Rockyou being a better representation of a general population.

4.10.2 Password nationality to inform guessing

We saw that the multi-armed bandit can link a password set to a dictionary based on characteristics within the passwords that divulge the demographics of the users. Does using passwords generated by these other users of the same nationality improve guessing?

Irish users In Figure 4.19 we guess the passwords in the Irish computerbits.ie password set. We guess them using the order and passwords from the full ‘‘`Collection__#1_NEW_ combo_semi_private_EU_combo.tar.gz`’’. We label this full dataset *“all users”*. We made 100 guesses against the 1795 users in the Computerbits password set. The top 100 most popular words were chosen in order from each dictionary. The dictionary composed of only Irish users performed better at guessing than the dictionary with all users' passwords in it. We also include the guessing success for two versions of our multi-armed

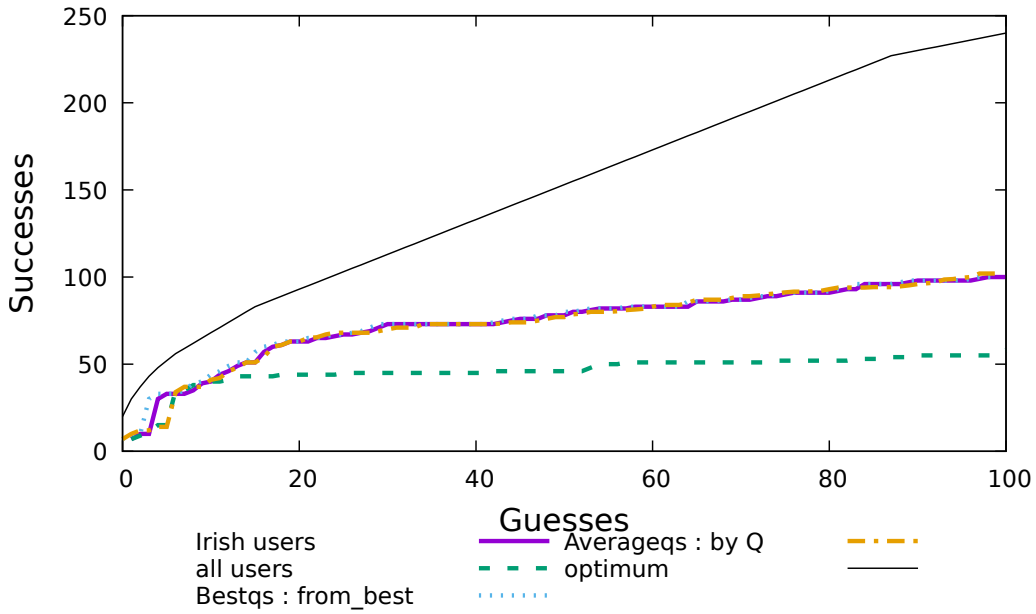


Figure 4.19: Guessing the Irish password set Computerbits.ie. Compares successes between guessing using a full dictionary of passwords and just those passwords belonging to Irish users.

bandit model. These both perform as well as guessing using the *Irish users* set. The black line shows the returns for an optimum first 100 guesses.

German users In Figure 4.20, we guess the Flirtlife password set using two dictionaries similar to above. We can see that using just the German users' passwords ranked in order, is more effective that using *all users* passwords. Both multi-armed bandit models perform better than simply using the distribution of *all users*' passwords to rank and order guesses.

4.11 Improving password guessing

Given a set of leaked passwords, that we have no a-priori knowledge about, we are interested in whether the multi-armed bandit can learn which dictionaries to choose guesses from. In this section we investigate whether the $\mathcal{Q}$-method of choosing passwords between dictionaries can offer a guessing improvement over a random choice of dictionary and choosing from the predicted "best" dictionary.

Recall that the $\mathcal{Q}$-method used the weighting of dictionaries and the proportion of each password in those dictionaries to decide on the next guess.

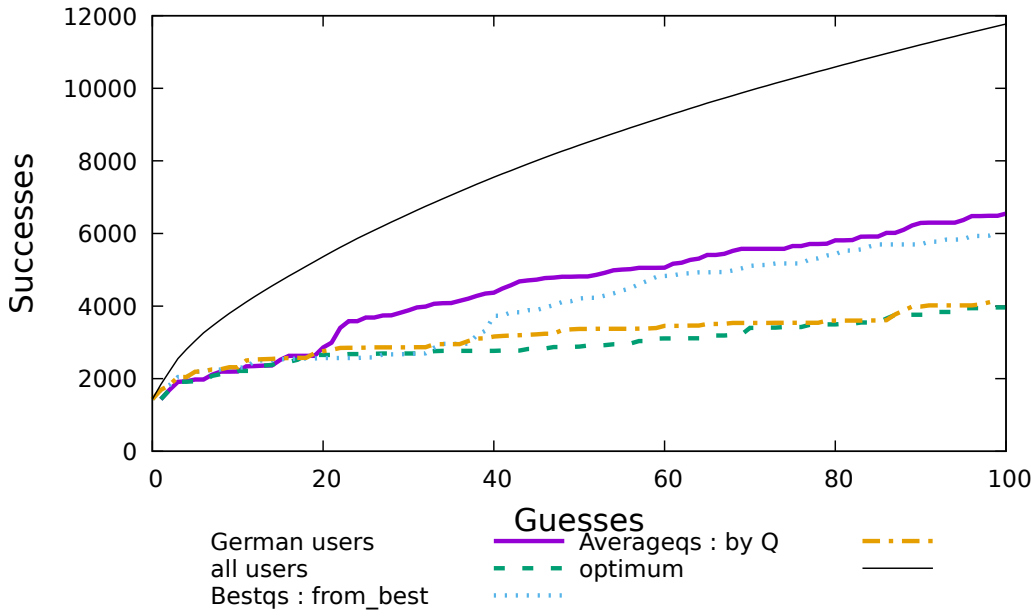


Figure 4.20: Guessing the German password set from Flirtlife.de. Compares successes between guessing using a full dictionary of passwords and just those passwords belonging to German users.

For this investigation we use two leaked password sets. The 2009, rockyou.com password leak which included 32 million user credentials. The passwords were stored in plaintext and the hackers used a 10-year-old SQL vulnerability to gain access to the database. The 2012 yahoo.com Yahoo Voices password leak which included 453,492 users' passwords [15], all in plaintext and also compromised using an SQL injection. We chose these password sets as we can be reasonably confident that they give a good representation of users. The plaintext passwords means we are not excluding low frequency passwords which might not be easily guessable. In addition, a compromise due to phishing or social engineering might mean we would only have a small subset of users from a similar demographic.

4.11.1 Rockyou.com password set

In this section, we describe the guessing of the Rockyou password set. Four dictionaries were used for the guessing: Computerbits, Hotmail, Flirtlife and 000webhost.

Figure 4.21 shows the estimated breakdown of Rockyou between the four dictionaries. Hotmail is assigned the highest rating with 000webhost, flirtlife

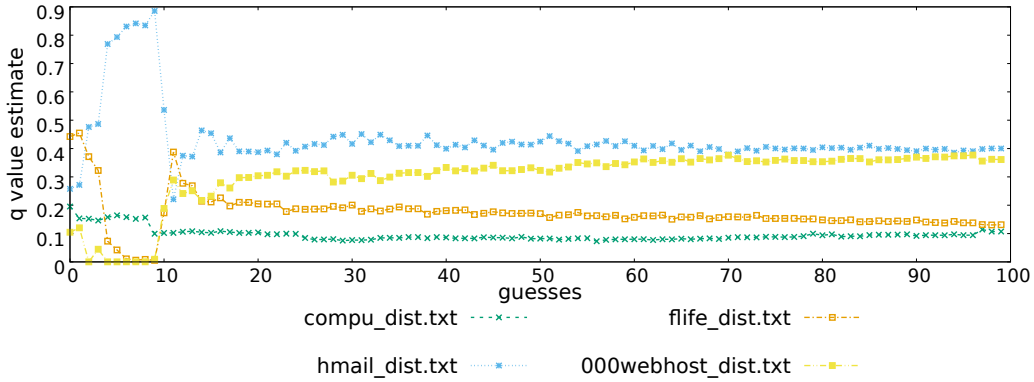


Figure 4.21: q-value estimates for Rockyou. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

and computerbits falling below it respectively. In terms of the breadth of the audience demographic in each of the dictionaries, this assessment of the breakdown seems logical. The nationality specific websites such as Computerbits and Flirtlife fall lowest and 000webhost, which enforces composition restrictions, fares slightly worse that Hotmail.

Figure 4.22 shows the guessing successes for the Rockyou password set guessed using the four dictionaries. There is little differentiation between the initialisation methods. However, the guess choice significantly impacted the number of successes. The optimum number of successes for 100 guesses against the Rockyou password set is 1,483,668 users compromised. The $\mathcal{Q}$-method compromised an average of 945,371 users. Choosing from the estimated best dictionary compromised 846,772 users on average, and choosing a dictionary at random to guess from resulted in the lowest number of average successes at 781,164.

We can see the $\mathcal{Q}$-method performs better than the next best method by compromising just under 100,000 more users.

Comparison to single dictionary guessing Our multi-armed bandit was designed to be able to distinguish the weightings assigned to a selection of password guessing dictionaries in order to characterise a given password set. To converge to the optimum characterisation, we maximise a likelihood function which is designed to match the distributions in the dictionaries to those in the files.

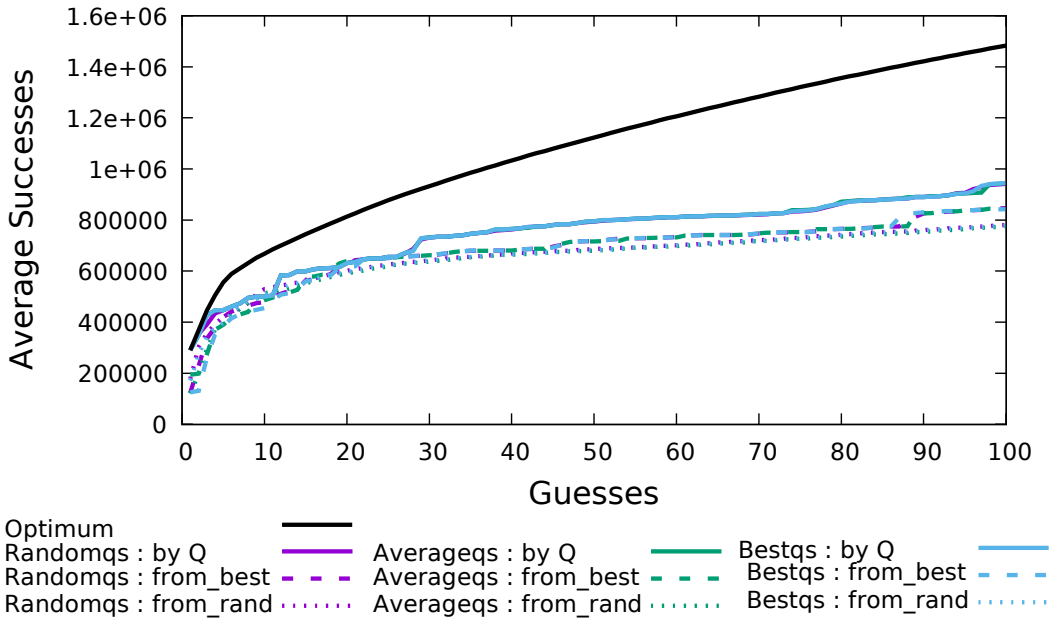


Figure 4.22: Guessing returns for rockyou.com

Here we are investigating whether leveraging this information about the distribution of the password set can directly aid guessing. We therefore compare the guessing returns for the $\mathcal{Q}$-method to guessing returns from simply guessing using each dictionary separately. Figure 4.23 shows the Rockyou password set guessed using 4 dictionaries separately. The purple solid line also shows the guessing returns when the dictionaries are combined using our multi-armed bandit $\mathcal{Q}$-method approach. The $\mathcal{Q}$-method performs well, compromising 945,371 users in comparison to 804,731, 703,041, 603,783 and 64,024 from Flirtlife, Hotmail, Computerbits and 000webhost respectively.

Compare the ordering in Figure 4.21 to that in Figure 4.23. Notice that, the weightings assigned to the dictionaries do not necessarily correspond to a better guessing result when the dictionaries are used individually. This is because the multi-armed bandit was designed with the goal of matching characteristics not optimising guessing. That is, the maximum likelihood will match a password set to dictionaries that follow a similar distribution. Let's take for example a password set which does not include the word "hellothere". If only one of the 3 dictionaries also does not contain this word then it will be assigned a good match for that password set for that guess. However, not containing this word has little baring on its ability to guess the password set.

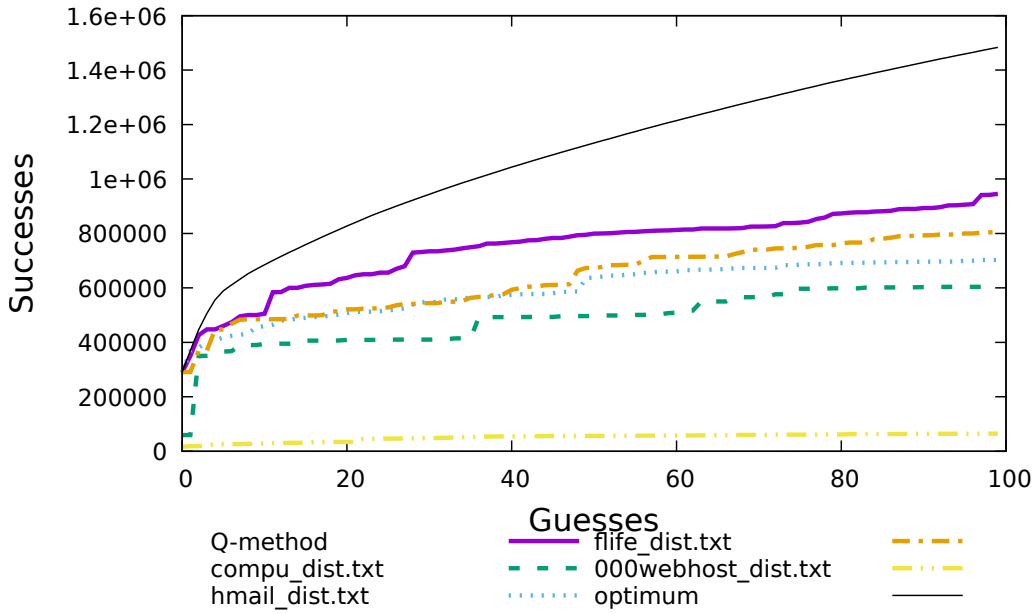


Figure 4.23: Single dictionary guessing returns for rockyou.com versus multi-armed bandit guessing

It simple means it matches the distribution.

If we guess a large number of passwords which do not exist in the password set, then we will see dictionaries matched against it which also do not contain these passwords. This provides a motivation for using a function other than maximum likelihood. If optimising guessing returns is the goal, then we suggest one which can weight successes more than weighing failures.

4.11.2 Yahoo.com password set

The yahoo.com password set was leaked in 2012 and contains 453,492 users' passwords. Figure 4.24 shows the estimated weighting of each dictionary for the Yahoo password set. Again we see what the flirtlife and computerbits password sets are weighted as low impact. The hotmail password set starts off strong but as the guessing continues its impact decreases towards zero. The 000webhost dictionary is consistently weighted at 0.5 but is overtaken as the best dictionary by Rockyou after 60 guesses.

In Figure 4.25 we plot the guessing returns for the yahoo password set. Again we have multiple set-ups for guessing. We are interested in comparing the $\mathcal{Q}$-method of guessing (solid lines) against guessing using the "best" dictionary

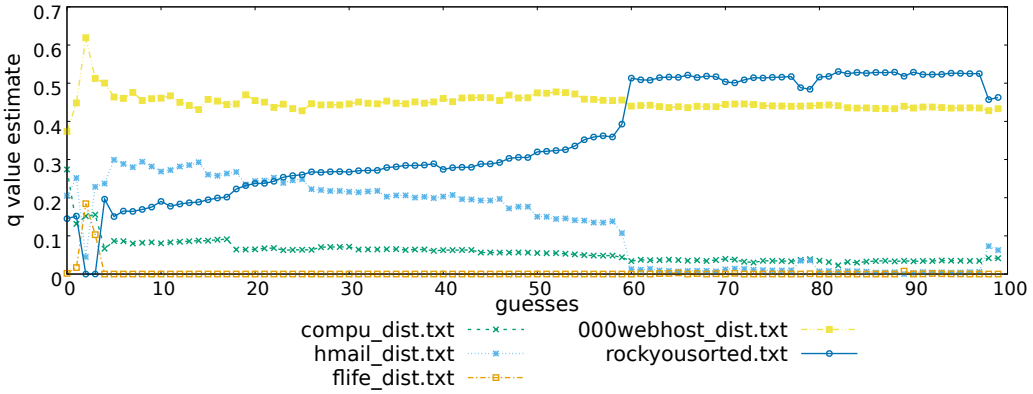


Figure 4.24: q-value estimates for Yahoo. Initialization: average $\hat{q}$-values, Guessing: $\mathcal{Q}$-method.

(dashed lines) and choosing a dictionary at random to guess from (dotted lines).

The $\mathcal{Q}$-method performs better than the other options with a return after 100 guesses of 8812 users when initialised with average q-values. The worst method initialised using average q-values and guessed from the best dictionary. Notice, that it does begin to improve at guess number 90. It is only at this point that the Initialization: average $\hat{q}$-values, Guessing: Best dictionary, consistently assigns the Rockyou password set as being the highest ranked (best) dictionary.

Comparison to single dictionary guessing Figure 4.26 shows the Yahoo password set guessed using 5 dictionaries separately. The blue solid line represents the guessing returns when guessing using the Rockyou dictionary of passwords. It performs the best of all the dictionaries and better than our multi-armed bandit $\mathcal{Q}$-method approach. The purple solid line represents the $\mathcal{Q}$-method which aims to combine the words from each of the dictionaries to guess in an effective order. It performs better than four of the dictionaries, but, as mentioned, performs worse than simple guessing using the Rockyou dictionary. However, as we would expect, it does begin to improve once it has ranked Rockyou as the best dictionary at guess 60. The $\mathcal{Q}$-method depicted has been initialised using average q-values.

Importantly, the bandit method of guessing can still be viewed as preferable to single dictionary guessing for this password set. Because the bandit model

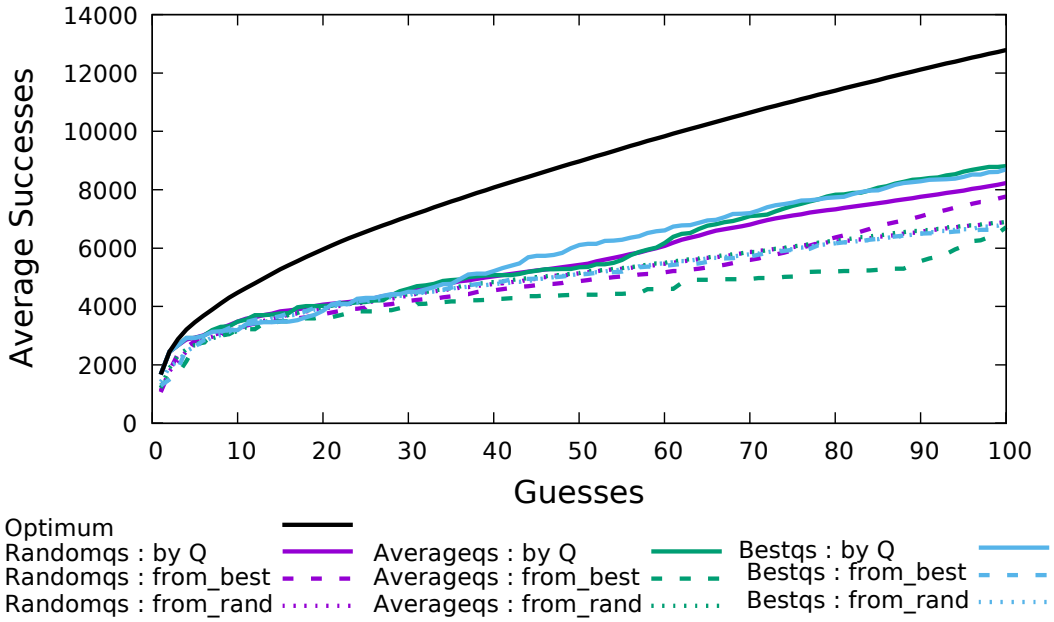


Figure 4.25: Guessing returns for yahoo.com

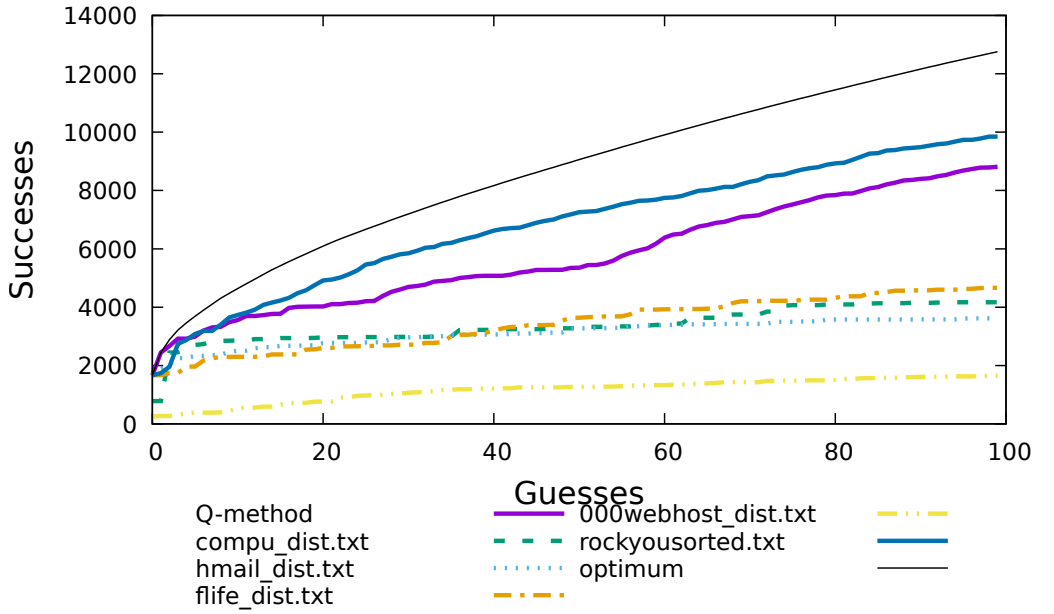


Figure 4.26: Single dictionary guessing returns for yahoo.com versus multi-armed bandit guessing

has the ability to learn, it is able to adjust its guesses to favour a dictionary that it rates highly. In comparison, guessing using a single password set does not offer learning and all 5 possible dictionaries would need to be tried in order to determine which is performing best.

4.12 Discussion of Results

The multi-armed bandit has the ability to distinguish a demographic category such as nationality from a password set. This can offer insights into password datasets and their sources. Leveraging this demographic information gathered, we showed that guessing tailored to the demographic in question will improve guessing successes.

For guessing a password set that we have no demographic insights about, the multi-armed bandit offers an automated method for optimising the choice of guesses when a selection of dictionaries is available. Though occasionally a single dictionary might offer better guessing returns, the value of the multi-armed bandit lies in its ability to learn. The adaptive model feeds on the information gathered from each guess and updates its recommendations with each new piece of information received.

There is a broad potential for expanding this work. We suggest three ways we believe this work can be developed. Given access to password datasets which contain passwords created under different composition policies, investigate whether the multi-armed bandit can identify the policies used to link it to a dictionary with an equivalent policy in place. The multi-armed bandit is not dependent on a strictly structured ranking from dictionaries. Therefore we suggest an investigation into using the bandit model to seed guesses with a users' personal information. As mentioned, when the goal is optimising guesses rather than identifying characteristics, an underlying optimization function other than maximum likelihood estimation could be used in the bandit model.

4.13 Conclusion

We used a multi-armed bandit approach to uncover the distribution of a password set and to optimise the order we choose passwords for guessing from dictionaries, thus improving the success of our guessing.

We showed that the multi-armed bandit learning algorithm could identify the dictionary which best matches a password set. We also showed that this can be used to identify the nationality of the users in a password set, and that guessing can be improved by guessing using passwords chosen by other users of the same nationality.

We identified that the multi-armed bandit can be used to improve how passwords are chosen when a selection of dictionaries are available. The ability to learn from each guess makes it a viable threat in online password guessing. A multi-armed bandit learning model should recognise composition restrictions, language, content topics and demographic information from each successful password guess and tailor the *mangling rules* and dictionaries used in order to target this.

Knowing the potential of this guessing model is useful for both users and organizations. It provides further evidence for the importance of guiding users away from passwords which reflect characteristics associated with demographic or website specific terms. It also demonstrates that password choices differ measurably depending on their source use. This could indicate that websites could consider tailored blocklisting techniques. In particular, websites who have experienced previous password leaks could work at restricting future users from using passwords which occurred with a high frequency in that leak.

CHAPTER 5

Quantifying the costs and benefits of authentication policies

Authentication is integral to our online world. Millions are spent worldwide by organizations to achieve authentication security. Yet we regularly see attacks which expose weak security practices. We showed in Chapter 2 that security advice relating to authentication given by one organization can directly contradict advice given by another. There is currently no general framework an organization can use to determine what authentication practices are good for them or bad. In Chapter 2 we modelled the costs users and administrators perceive with authentication advice and also which security benefits exist for the same pieces of advice. In this chapter we introduce our methods for quantifying the trade-off between these costs and benefits. We show how these models can be compared to provide an organization with information on the best authentication policies to implement, taking into account their own security budget and priorities. We leverage this model to compare the NIST 2003 Electronic Authentication Guidelines with the NIST 2017 Digital Identity Guidelines. In doing so, we demonstrate the value of the NIST 2017 advice in terms of both security and usability. This chapter is based on research presented at USENIX '17, HEAnet '18 and PasswordsCon '18.

5.1 Introduction

Authentication is an essential part of our online world. Yet many organizations implement weak security procedures. We consider two reasons for this: first, that organizations do not receive clear consistent advice on best practice security procedures; second, that the benefits of a strong security policy do not always outweigh the costs of implementing and maintaining such a policy.

In Chapter 2, we described our collection of authentication advice and found that it is often inconsistent. In addition we found that, at times, advice circulated contradicts what researchers might consider to be the best practice.

In this chapter, we introduce quantitative models which can allow an organization to trade-off the costs and benefits of authentication advice in order to make informed security and usability decisions.

We use these models to quantify the costs and benefits of the NIST 2003 Electronic Authentication Guidelines and the NIST 2017 Digital identity guidelines. The NIST 2017 policy superseded the 2003 policy and at the time of publication there was discussion about the failings of the 2003 policy [41]. We were interested in quantifying the increase in usable security that the NIST 2017 policy claimed to offer over the obsolete 2003 policy.

This research will allow organizations to make informed decisions about their authentication security policies. The adjustable model allows an organization to tailor the output to identify the best policy with respect the their personal security and usability needs. An organization with 50 internal employees may want a different policy to an organization with 1000 remote users. This model allows this to be represented. We hope that considering a security policy with respect to its usability will allow organizations to make more informed choices which will in turn lead to stronger security [1, 25].

In Section 5.3, we recall the cost categories that were introduced and evaluated in Chapter 2. In Section 5.4, we formally define our model. Section 5.4.1 details the benefits model and how it can be quantified and Section 5.4.2 describes the same for the costs model. In Section 5.5, we provide a brief description of the five varieties of NIST authentication policies that we will evaluate. In Section 5.6 we use our model to identify the value of each NIST policy for a simulated company. Then, in Section 5.7 we compare the impact

of organisation size and security and usability priorities on the value of each policy. We begin, in Section 5.2, with related work.

5.2 Related work

Lampson [91] in 2009 said

> "The root cause of the problem is economics: we don't know the costs either of getting security or of not having it" ... "To fix this we need to measure the cost of security, and especially the time users spend on it."

Usability with security has become a subject of interest in the last few years. It gained attention in 1999, with the publication of Adams and Sasse's paper "Users are not the enemy" [1]. In this work, Adams and Sasse gathered survey responses from 139 users asked about their password practices and perceptions. This was then followed by in-depth interviews with 30 of these users. The authors demonstrated that users were often forced to comply with security mechanisms that were incompatible with their work procedures. In these cases, they found that users employed workarounds for the security policies, which would circumvent the whole procedure. Adams and Sasse emphasise that security systems must be designed with usability in mind, as otherwise mechanisms that might look secure on paper will fail in practice.

In 2005, Cranor and Garfinkel published their book "Security and Usability: Designing Secure Systems that People Can Use" [33]. This book covered fundamental topics in the usable security discussion. It included chapters on *Usability design and evaluation for privacy and security solutions*, *The memorability and security of passwords*, and *Usable biometrics*. Of particular relevance to our work is the chapter "Evaluating Authentication Mechanisms" by Karen Renaud. In this chapter [138] and in the related paper [137], Renaud quantified the quality of web authentication mechanisms. Uniquely at this time, Renaud took into account usability metrics. Renaud evaluated a selection of different authentication mechanisms including: biometrics, passwords (syntactic[1], semantic[2] and one-time), graphical codes, recognition based sys-

[1]Syntactic passwords were defined as a remembered sequence of letters and digits.
[2]Semantic passwords rely on a cognitive process to produce the required password.

tems, location based systems, and public key authentication schemes (using a cryptographic hardware device or certificate stored on the user's machine). Each of these authentication mechanisms were evaluated with respect to accessibility, memorability, security and vulnerability. The paper concludes that graphical mechanisms show potential and that semantic passwords could also be a good alternative to the classical password. However, Renaud does note that most developers use semantic passwords only to request the mother's maiden name, which is a misuse and worthless. Renaud dismissed public key authentication schemes as being poorly implemented and difficult to use at the time the paper was written. Renaud's work highlights the importance of quantifying both security and usability in order for information security decisions to be made.

We are interested in usability and it's relationship to authentication security. In particular, when we think of security, we refer to the reduction in risk that a security mechanism or procedure can provide.

In their 2007 paper, Jøsang et al. showed that many security risks are caused by poor usability [79]. They emphasise the importance of considering security usability as part of vulnerability analysis and risk assessment. They describe a set of security usability principles. These include: users must understand which security actions are required of them, users must have sufficient knowledge and ability to take correct security actions, the mental and physical load of a security action must be tolerable, and the system must provide the user with sufficient information for deriving a security conclusion. They use these principles to assess the usability of security actions expected of users. They argue that, using these it is now possible to quantify the trade-offs between theoretical security and practical security.

In 2007, Renaud continued her work evaluating authentication mechanisms with a thorough analysis of security risks associated with them [139]. Renaud defines risk in the web authentication context as: "the possibility of an intrusion, and consequent harm, to a web site, by means of exploitation of authentication mechanism vulnerabilities". Renaud highlights that we do not have an idea of any website's actual exposure to vulnerabilities. However, we do know what opportunities an attacker has for an attack. Renaud therefore

equates security risk to opportunity and summarizes opportunity as:

$$Opportunity = \frac{Guessability + Observability + Recordability + Analysability}{Resistability}.$$

Renaud uses a point system to assign scores for each of *Guessability, Observability, Recordability, Analysability* and *Resistability* for a given authentication mechanism. In this way, Renaud created a quantification framework to support choice between different authentication mechanisms.

Building on Renaud's work, Mihajlov et al. released two papers in 2011. The first looked at quantifying the usability and security in authentication [104]. The second, leveraged this methodology to build a conceptual framework for evaluating usable security in authentication mechanisms [105]. Their work involved a scoring system similar to the one developed by Renaud [137]. They began by identifying the quality criteria of import in both usability and security. They could then assess a given authentication mechanism with respect to this quality criteria. The mechanism is assigned a value between 0 and 1 for each quality criteria to indicate the level of quality it provides. A total quality score can then be derived for each mechanism and this allows authentication mechanisms to be compared to each other in terms of both usability and security.

Braz et al. [17] proposed a usability inspection method which could be used to compare the security problem to the usability criteria for online tasks such as; authenticate yourself, transfer funds or buy a concert ticket.

Shay and Bertino [148] built a simulation tool that can model the costs versus the benefits of complexity rules, throttling, and regular password expiry. Their model simulates users and accounts so that an organisation can test out a security policy before rolling it out to actual users. The model can take into account: increased help desk time as a result of mandated password changes and the increased ability of a user to memorize their password after they have had it for a long time. They also include a probability of a compromise as a variable read into the simulation. The password strength as a result of a particular policy is measured in entropy. The value of the password rules is measured based on the balance of $Income - Cost$, where income is the income generated by users utilizing the service, and cost includes both

fixed daily operating costs and variable costs such as help desk calls and costs resulting from an account compromise. Their model can allow analysis of the impact of enforcing variations of these three rules on passwords and can allow organizations to determine the net value of implementing them. However, their model is solely concerned with protecting against compromise due to online guessing attacks and compromise resulting from a user writing their password down. Shay and Bertino explain that security is an economic as well as a computer problem.

In 2009, Herley [66] argued that a users' rejections of security advice is rational from an economic perspective. Herley quantifies the costs versus benefits for three specific authentication guidelines: password rules, phishing site identification advice and SSL certificate warnings. Herley concludes that most security advice offers a poor cost-benefit trade-off for users.

In 2011, Altinkemer et al. [4] quantified costs and benefits associated with implementing a new authentication system. In particular, they were interested in the effects, for an online service or product provider, of implementing a one-factor versus a two-factor system. They investigate the impact of both authentication options with reference to the additional implementation costs, the probability that a customer will switch to a competitor's product or service, and the expected losses should the new system fail.

In 2012, Bonneau et al. developed a qualitative mechanism for assessing the value of different authentication mechanisms [13]. They created a qualitative framework which assessed whether each mechanism met quality goals under three key categories: usability, deployability and security. In 2016, Realpe et al. created a set of 153 heuristics to grade applications on their user authentication processes [133]. They created these qualitative heuristics under the headings of: usability, security, operability, accessibility, reliability, and performance.

Similar to Shay et al [148], Arnell et al. use simulations to compare the effectiveness of two different password policies [6]. Arnell et al. created utility equations for an organisation to allow them to determine the best policy for them. The utility was based on three factors: breaches, productivity loss and necessary investment in support services (such as help desk staff). For a given policy, they computed the Cost of Breaches (the total annual breach costs to

the organization) and the Cost of Performance Loss (the annual performance loss due to password resets for the organization). The performance loss statistics were gathered using studies which recorded users' time taken to generate a password, number of failed password entry attempts and the frequency that users forgot their passwords. They collected these statistics for three different password composition rules. As a demonstration of their model they compare a policy which requires a six character password to one which required a 12 character password. Their work demonstrated that it is possible to quantifiably measure the impact of security decisions in a utility-driven context.

Authentication mechanisms cannot be considered outside of the policy they are implemented in. For example, there is little point mandating complex password composition if the passwords will be stored in plaintext, with no rate limiting against online guessing, and they are communicated in the clear. Our work in this chapter, builds on the current research in the following ways:

1. Previous work has quantified the value of password composition and regular password expiry rules or specific pieces of advice such as SSL certificate warnings. We provide a plenary analysis of the costs and benefits of an entire authentication policy.

2. Instead of arbitrary scoring as an evaluation of usability costs, we can provide exact costs based on time. This allows for differentiation between costs of different magnitude that are reoccurring at different rates.

3. This finer analysis allows us to analyse characteristics within costs and benefits for any authentication policy. This provides us with the data needed to investigate the impact of changing parameters such as organisation size, security priority and user value on the effectiveness of a policy.

4. These models are applied to password policies which researchers have intuition about the advantages of. The NIST 2017 policy was a fully considered policy, praised by many security experts. Our analysis highlights the value of this policy and the failings of those it was compared to.

Table 5.1: Cost categories as defined in Chapter 2.

Organisation Costs
Increased help desk/user support time
User education required
Organisation needs extra resources
Takes organisation time to implement
Increases the organisation's computing power needed
User Costs
Makes it more difficult to create a password
Makes it less easy to remember
Requires extra resources
Requires the creation of a new password
Increases the computing power needed
Requires other extra time or effort

5. Authentication policies are enforced by organisations and therefore it is an organisation who must analyse the cost benefit trade-offs based on their own priorities and needs. Our model is adaptable to tailor to a specific organisation and offers a financial evaluation of the trade-offs.

This chapter will provide our methodology for quantifying the security benefits and usability costs of enforcing a password policy.

5.3 Cost and benefit categories

The value of an authentication policy can be defined as the security benefits the policy brings minus the costs it invokes.

In Chapter 2, we defined 11 cost categories, 5 costs to the company enforcing the advice and 6 categories of potential costs for the users following the advice.

These costs are listed in Table 5.1. Recall that the benefits of following authentication advice can be categorized according to the NIST 2017 Authenticator Threats. These are listed here in Table 5.2 and further descriptions are available in Table 2.8 in Chapter 2.

5.4 Model

Modelling the value of an authentication policy will trade off these costs versus benefits. In this section, we formally describe this model and derive equations

Table 5.2: Categories of security benefits.

1.	Assertion manufacture or modification
2.	Physical theft
3.	Duplication
4.	Eavesdropping
5.	Offline Cracking
6.	Side Channel Attack
7.	Phishing or Pharming
8.	Social Engineering
9.	Online Guessing
10.	Endpoint Compromise
11.	Unauthorized binding

which permit its quantification. As mentioned, we define the value of a password policy as the benefits minus the costs.

$$\mathbb{E}[\text{Value}] = \mathbb{E}[\text{Benefits} - \text{Costs}] \tag{5.1}$$

By design, our benefits, B, are determined by the possible attacks, ω_a, and the costs, C, are dependent on the possible user or organization actions, ω_u. Both ω_a and ω_u are results of the password policy.

$$\sum_{\omega_\text{a},\omega_\text{u}} \mathbb{E}[\text{Benefits} - \text{Costs}] = \sum_{\omega_\text{a},\omega_\text{u}} \mathrm{P}_{\omega_\text{a},\omega_\text{u}}\left(\mathrm{B}_{\omega_\text{a}} - \mathrm{C}_{\omega_\text{a}}\right) \tag{5.2}$$

$$= \sum_{\omega_\text{a},\omega_\text{u}} \left(\mathrm{P}_{\omega_\text{a},\omega_\text{u}}\mathrm{B}_{\omega_\text{a}}\right) - \left(\mathrm{P}_{\omega_\text{a},\omega_\text{u}}\mathrm{C}_{\omega_\text{a}}\right) \tag{5.3}$$

We assume ω_a and ω_u are independent

$$= \sum_{\omega_\text{a},\omega_\text{u}} \left(\mathrm{P}_{\omega_\text{a}}\mathrm{P}_{\omega_\text{u}}\mathrm{B}_{\omega_\text{a}} - \mathrm{P}_{\omega_\text{a}}\mathrm{P}_{\omega_\text{u}}\mathrm{C}_{\omega_\text{a}}\right) \tag{5.4}$$

Because $\sum_{\omega_\text{a}} \left(\mathrm{P}_{\omega_\text{a}}\right) = 1$ and $\sum_{\omega_\text{u}} \left(\mathrm{P}_{\omega_\text{u}}\right) = 1$

$$= \sum_{\omega_\text{a}} (\mathrm{P}_{\omega_\text{a}}\mathrm{B}_{\omega_\text{a}}) - \sum_{\omega_\text{u}} (\mathrm{P}_{\omega_\text{u}}\mathrm{C}_{\omega_\text{u}}) \tag{5.5}$$

So we have that the benefit is the sum of all the benefits the authentication

policy offers against each attack type a. The cost is the sum of the cost of each user/organization action resulting from the authentication policy.

We now define the individual models for the benefit and the cost. Section 5.4.1 will describe our methods for modeling the benefits and Section 5.4.2 describes our model of the costs.

5.4.1 Benefits

We define the expected benefit of an authentication policy as: the difference in the expected losses with and without the policy.

$$\mathbb{E}[\text{Benefits}] = \mathbb{E}[\text{Loss without policy}] - \mathbb{E}[\text{Loss with policy}] \tag{5.6}$$

The $\mathbb{E}$[Loss without policy] and $\mathbb{E}$[Loss with policy] are calculated using the same method. Therefore we will describe the method for calculating losses and it can be applied to the scenario with and without an authentication policy.

We define two classes of Losses: L_1, the loss as a result of each individual user compromise and F_{system}, the loss as a result of system overrun or system failure.

$$L_{\omega_a} = F_{system_{\omega_a}} + \sum_n L_{1_{n,\omega_a}} \tag{5.7}$$

To define system failure, F_{system}, we use the concept of a saturation point α which Flôrencio et al. develop [53]. This α represents the fraction of accounts that need to be compromised before a system can be consider to be completely overrun. This point $0 < \alpha \leq 1$ can be relatively low when each currently compromised account can be leveraged to compromise others. When a fraction α of the total N number of users are compromised we can consider a system to be completely overrun.

We define X_n to be our indicator of compromise and F_{system} to be our indicator

of system failure.

$$X_n = \begin{cases} 1 & \textit{if user } n \textit{ compromised by any attack} \\ 0 & \textit{otherwise} \end{cases} \tag{5.8}$$

$$F_{system_{\omega_a}} = \begin{cases} L_{system} & \textit{if } \sum_n X_n > \alpha N \\ 0 & \textit{otherwise} \end{cases} \tag{5.9}$$

L_{system} is the losses as a result of system failure or system overrun. This loss could include reputation cost to the organization, loss of ability to function, and could potentially equal or exceed the total value of the organization.

$$\mathbb{E}[\text{Loss}] = \sum P_{\omega_a} L_{\omega_a} = \sum P_{\omega_a} \left(F_{system_{\omega_a}} + \sum_n L_{1_{n,\omega_a}} \right) \tag{5.10}$$

$$= \sum_{\omega_a} P_{\omega_a} F_{system_{\omega_a}} - \sum_n L_{1_{n,\omega_a}} \sum_{\omega_a} P_{\omega_a} \tag{5.11}$$

Unless $X_n = 1$ or $F_{system} = L_{system}$ the term is 0 for all ω_a possibilities. Also assuming X_n are independent we get:

$$= L_{system}.\mathbb{P}\left[\textstyle\sum_n X_n > \alpha N\right] + \textstyle\sum_n \mathbb{P}[X_n = 1] L_{1_n} \tag{5.12}$$

We make the assumption that up until αN accounts are compromised X_n are identically distributed. Also that L_1 is constant for each user n. Then given large N we can use a normal approximation [37] $\mathcal{N}(\mu, \sigma^2)$ to estimate the probability that a fraction α of the accounts are compromised. Finally, let $\mathbb{P}[X_n = 1] := p$.

$$= \mathbb{P}\left[\mathcal{N}\left(p, \frac{p(1-p)}{N}\right) > \alpha\right] L_{system} + NpL_1 \tag{5.13}$$

Now we can calculate the $\mathbb{E}$[Loss without policy] and $\mathbb{E}$[Loss with policy] to find the $\mathbb{E}$[Benefits]. The probability p for $\mathbb{E}$[Loss without policy] is the application of Equation 5.13 where p is the probability of a compromise when no mitigation is in place.

5.4.1.1 Quantifying benefit

In this section we show the additional steps necessary for quantifying Equation 5.13. We define the probability that X_n is compromised as the probability that at least one of the a attacks is successful.

$$\mathbb{P}[X_n = 1] := p := 1 - \prod_a (1 - P_a) \tag{5.14}$$

This method of quantification will work when all the attack types are independent of each other. In most cases this is true, however, we encounter an exception with offline guessing.

Offline guessing Offline guessing is a trial and error guessing attack against an offline dataset of passwords. In order for the guessing against all accounts to take place, the password dataset must first have been compromised. We call this a dataset leak.

To model this, we look at the expectation, first when no password dataset leak occurs, and secondly when a leak has occurred.

$$\mathbb{E}[\text{Loss}] = \mathbb{E}[\text{Loss}|\text{no leak}]\mathbb{P}[\text{no leak}] + \mathbb{E}[\text{Loss}|\text{leak}]\mathbb{P}[\text{leak}] \tag{5.15}$$

This simplifies our calculations. In the situation when there is no leak the $\mathbb{P}[\text{offline guessing}] = 0$. So all attacks can be calculated using $p = 1 - \prod_a (1 - P_a)$ normally. We denote the answer $p_{l'}$. When a leak does occur the $\mathbb{P}[\text{offline guessing}] = \mathbb{P}[\text{password cracked}]$. So again this can be included in the $p = 1 - \prod_a (1 - P_a)$ normally. We denote it p_l.

5.4.2 Costs

To develop the cost part of equation 5.1 we look at costs in terms of a set of ω_u outcomes that occur as a result of the advice,

$$\mathbb{E}[Costs] = \sum_{\omega_u} P_{\omega_u} C_{\omega_u}. \tag{5.16}$$

However, we are only interested in the outcomes ω_u which will result in a cost to the user or organization. To identify the costs associated with enforcing advice we leverage the cost categories we created in Chapter 2. These 12 types of costs were listed again in Table 5.1.

Because any w_u which does not result in a cost will make C_{ω_u} zero, it is reasonable to re-write our equation just in terms of categories we have identified costs for. We will refer to these as C_i for $i = 1 \dots m$. The cost output from these categories is dependent on the piece of advice j and also a piece of advice could result in a number of repetitions of each cost. We denote the number of repetitions of each cost as R. The probability P of a cost occurring is dependent on the advice and category.

$$\mathbb{E}[Costs] = \sum_{i.j} P_{i,j} C_{i,j} R_{i,j} \tag{5.17}$$

5.4.2.1 Quantifying costs

In Chapter 2, we identified categories of costs associated with authentication advice. These related to both user and organisation costs. We ran a user and administrator survey to gather feedback about these cost categories.

We need to be able to quantify the monetary cost from each of the cost categories identified in Table 5.1.

In Section 2.6.2.1 we recognised that each cost category can contain sub-costs. At that time, for clarity in the model, we chose the high level costs only. However, now for each cost category we continue to identify sub-costs until we have costs that can be defined financially. Figure 5.1 shows this visually for the category *Increased risk of forgetting*.

Increased risk of forgetting inconveniences the organization as a result of necessitating password resets and also takes up more of the users' time. The inconvenience can also result in a user simply abandoning the website. In this case, the profit from that user is lost. In some situations the password policy enforcer or organization will not care about the inconvenience to users and in other situations users will be high priority. We create an optional variable $0 \leq U \leq 1$ which indicates the weight an organization places on their users'

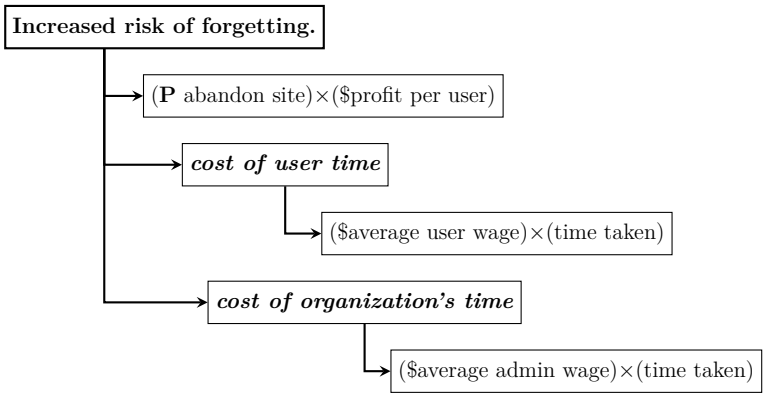


Figure 5.1: Costs associated with *Increased risk of forgetting*.

satisfaction.

Table 5.3a shows the quantification of each cost category. All-capitalized words refer to constants; these can be inputted based on the organization's data. These are defined in Table 5.3b. Lower case variables are dependent on the pieces of advice in the policy. Each equation is named $C_i()$ and the input variables are indicated in the brackets. t refers to time, t_c to computing time, and u to costs borne by the user and o to costs borne by the organization. Table 5.3c describes the estimates for the probabilities which are required for the equations in Table 5.3a.

There are two things to note in Table 5.3a. First, the user education category has been split into two parts. This is because there are costs to both users and organisation as a result of user education, and the costs felt will differ in duration and repetition depending on the group. Secondly, we note that in the organisation cost categories, we have three categories which are defined by the same cost equations. These are the help desk, implementation, and providing user education costs. This is because each of these costs relate to an employee time cost. Similarly the user education incurs the same costs as the simple user time cost. We leave these as separate categories for clarity with regard to the type of cost being assigned and also to match with the costs identified by users and administrators in our user study (Section 2.7).

Table 5.3: Quantifying functions and variables for policy costs.

Define function		**Equation**
$C_1(t_o)$: Increased help desk/user support time	=	supportTimeTaken ∗ $ADMIN WAGES
$C_2(t_o)$: User education provided	=	EducatorTimeTaken ∗ $ADMIN WAGES
$C_3(r_o)$: Cost of additional organization resources needed	=	$orgResources
$C_4(t_o)$: Cost of the Organization's time taken to implement	=	timeToImplement ∗ $ADMIN WAGES
$C_5(t_{c_o})$: Cost of the organization's computing power	=	orgCompTime ∗ $COMP POWER COST
C_6(timeChoosePwd): Cost of increasing difficulty of password creation	=	P[abandon|created authenticator rejected] ∗ ($ORG PROFIT PER USER) + C_{11}(timeChoosePwd) + P[forget] ∗ $C_7()$
$C_7()$: Cost of forgetting	=	P[abandon|forget] ∗ ($ORG PROFIT PER USER) + C_1(TIME ADMIN RESET) + C_{11}(TIME ADMIN RESET)
$C_8(r_u)$: Cost of additional user resources needed	=	U ∗ $userResources
C_9(timeChoosePwd): Cost of needing to create a new password	=	P[abandon during creation] ∗ ($ORG PROFIT PER USER) + C_{11}(timeChoosePwd) + C_5(TIME HASH SALT ENCRYPT) + P[forget|new password] ∗ $C_7()$
$C_{10}(t_{c_u})$: Cost of the users' computing power	=	U ∗ userCompTime ∗ $COMP POWER COST
$C_{11}(t_u) =$: Cost of the users' time and inconvenience	=	U ∗ userTime ∗ $USER WAGES
$C_{12}(t_u)$: User education	=	U ∗ EducationTimeTaken ∗ $USER WAGES

(a) Quantification for each cost category.

Constants	
$ADMIN WAGES: the cost of the administrator's time.	$USER WAGES: the cost to the users' time.
$ORG PROFIT PER USER: organization's profit per user.	$COMP POWER COST: cost of computing.
TIME ADMIN RESET: time taken for the administrator to reset the authenticator (e.g. password).	U: weight on user importance.
TIME CHOOSE PWD: time taken for a user to create a new password.	TIME HASH SALT ENCRYPT: time taken to hash salt or otherwise encrypt a users' passwords.
L_1 = $ORG PROFIT PER USER + (U)(TIME ADMIN RESET)($USER WAGES) + (TIME ADMIN RESET)($ADMIN WAGES) + $DAMAGES.	

(b) Table of constants.

Estimating Probabilities		
P[abandon|forget]:	=	Fraction of users who have abandoned the site after they fail to recall their password.
P[abandon|created authenticator rejected]:	=	Fraction of users who have abandoned the site after their password was rejected at creation.
P[abandon during creation]:	=	Fraction of users who have abandoned the site any time during password creation.
P[forget]:	=	Fraction representing the average number of resets per user.
P[forget|new password]:	=	Fraction of users whose authenticator needed to be reset directly after it has been created.

(c) Table of probabilities.

5.5 NIST authentication policies

We will use this model to quantify the value of the NIST (National Institute of Standards and Technology) authentication policies. But first let us introduce these policies. The exact advice we quantify from these policies is included in Appendix D. Here, we provide an overview of each policy and simply highlight the key components.

5.5.1 NIST 2003

We begin with the NIST 2003 "Electronic Authentication Guideline". These 2003 guidelines have received widespread criticism in recent years [101, 155]. However, in 2003, very few understood the impact the advice would have [41]. The 2003 advice was created with federal agencies in mind and was designed to have varying levels which could each offer a different level of security. However, the advice was widely taken on board by universities and large companies and was enforced without consideration for user time. Furthermore, it was the highest level of password rules that were taken on board, rather than the weakest. This was exacerbated by future NIST policies augmenting the original NIST document by introducing additional restrictions.

In this section, we describe the NIST 2003 Level 1 security advice and the NIST 2003 Level 4 security advice (highest level). We also look at the evolved NIST 2003 advice which represents advice that was widely used in practice. It includes a 90 day expiration of passwords and strict composition restrictions. We refer to this advice as the 2007 NIST advice, since the first written example we found for it was in the 2007 NIST documents, [122] and [121].

NIST 2003 Level 1 The NIST 2003 Level 1 guidelines describe a challenge response protocol. There is no requirement at this level to use approved cryptographic techniques and this level does not require cryptographic methods that block offline analysis by eavesdroppers. There is no composition requirements[3] on the choice of passwords. A lockout is enforced for 1 minute after 3 incorrect guesses. The password files contain an "inversion"[4] of the password rather than the password itself.

[3]Composition of a password describes what characters are allowed or must be included in the password. For example, a typical composition policy is "you must include an uppercase character, a lowercase character, a number and symbol in your password".

[4]We consider an "inversion" to be an invertible encryption method using a key.

NIST 2003 Level 4 Level 4 is the highest authentication policy in the NIST 2003 standards. It requires authentication using a hardware cryptographic token and a password. Client authenticated TLS is recommended for the authentication process. This involves the client holding a certificate and corresponding private key for that website. Users need to re-authenticate themselves every 24 hours. Passwords are stored in encrypted form. An account is locked for 24 hours after 6 successive failed authentication attempts. Passwords must be: at least 8 characters long and must contain at least one upper case letter, one lower case letter, one number and one special character, and a dictionary is used to remove common words and prevent permutations of the username as a password. Passwords must be changed every two years.

NIST 2007 We use the term "NIST 2007" to describe a debasement of the NIST 2003 policy. It represents a policy most similar to those enforced within universities, federal bodies and companies [101, 155]. In this policy, cryptography is used to secure passwords when they are transmitted. Passwords stored on a server are encrypted. An account is locked for 24 hours after 6 successive failed authentication attempts. Passwords must be: at least 8 characters long and must contain at least one upper case letter, one lower case letter, one number and one special character, and a dictionary is used to remove common words and prevent permutations of the username as a password. Passwords must be changed every 90 days. Notice that, unlike the NIST 2003 Level 4 policy, this 2007 policy does not require Client authenticated TLS or a second factor cryptographic device, but it does involve a 90 day password expiration policy.

5.5.2 NIST 2017

In 2017, NIST released their "Digital Identity Guidelines" to supersede the 2003 document. The new guidelines received input from notable security researchers and involved a public comment period. The four volume series includes extensive and detailed guidelines related to authentication, identity proofing, and assertions [59]. We focus on the authentication guidelines. Similar to the 2003 document, the guidelines are structured according to the level of security required. There are three levels of security detailed. We will provide an analysis of the Level 1 and Level 3 (highest) recommendations.

NIST 2017 Level 1 The NIST 2017 Level 1 policy allows a number of different authentication methods. We focus on 'subscriber chosen memorized secrets' (we refer to them as passwords) which are the most common form of authentication [67]. In the Level 1 policy, users must re-authenticate every 30 days. No more than 100 consecutive failed authentication attempts on a single account can be made. The password should be communicated via an authenticated protected channel. Two physical authenticators should be used to bind the user to their online identity.

Passwords must be more than 8 characters long. There should be no composition requirements in place and all printable ASCII and UNICODE characters should be accepted. Blocklisting of password choices should be in place. That is, when a user creates a password, it should be compared against lists made up of passwords obtained from previous breach corpuses, dictionary words, repetitive or sequential characters (e.g. 'aaaaaa', '1234abcd') and context-specific words, such as the name of the service, the username, and derivatives thereof. Users should have the option to view their password - rather than a series of dots or asterisks - until it is entered. Passwords should be stored in a hashed and salted form and a pepper[5] should be used on the entire password file.

NIST 2017 Level 3 Level 3 is the highest authentication assurance level in the NIST 2017 policy. A cryptographic USB key and password used together are a valid means of authentication at this level.

The rules for using a memorized secret for authentication are the same as for the NIST 2017 Level 1 policy. Re-authentication must be performed after 15 minutes of inactivity or after 12 hours. Two physical authenticators should be used to bind the user to their online identity. Client authenticated TLS is recommended for securing communications between the user and the verifier. The cryptographic device authenticators should require a physical input (e.g., the pressing of a button) in order to operate. It should have tamper detection and response and the private key should be non-exportable.

[5]A pepper is typically an additional salt value that is used on the entire password file. Unlike a traditional salt, a pepper is kept secret and should be stored separately from the hashed passwords [99].

5.6 Value of the NIST 2017 policy

We have introduced 5 different authentication policies. We now use our model to quantify the value of the NIST 2017 Level 1 policy. We do note that the version of the NIST 2017 policy that we quantify is a high-level overview of the authentication policy. This is simply our interpretation and many of the nuances in the advice may be lost in this brief account.

5.6.1 Single result for a fictional company

In Appendix D we detail a full working example for applying our model to the NIST 2017 Level 1 policy for a fictional company.

We say that the company is worth $\$10,000,000$. It has 500 remote users and the satisfaction of their users is very important to them: $U = 1$. Because the users are external to the organization a compromised user would not immediately be seen as a threat to the overall system; the saturation point $\alpha = 0.5$. That is, 50% of users would need to be compromised before the reputation and actual damage combine to bring down the organization. The damages as a result of a single breach we set at \$148 (taking this value from [132]). This means our calculated L_1, loss from a single users' compromise, is \$166. We set both the user and administrator wages at \$18 per hour. Each continued user is worth a \$15 profit to the organization.

5.6.1.1 Estimating the benefits

Using these values, we implement our model to calculate the value of the NIST 2017 Level 1 policy when the authenticator used is passwords. We encountered the difficulty of not having accurate attack breach data. We could find little to no statistics on the probabilities of different attacks occurring. Even when looking at the statistics that we did find on the frequency of successful attacks, we found that the statistics from different sources were often not comparable. It is vital for this model that the statistics used for each attack are compatible with each other. Therefore we chose to take all our statistics from the one source. We chose the 2017 Verizon's Data Breach Investigations Report (DBIR) [169] because the statistics were the most granular we could find from a single source. The values we take from this source can be conceptually thought of as weightings. While the specific figure may not reflect a

true probability of attack. The weighting of the attack type in comparison to other attack types is the true importance of the statistics. Therefore, provided all the values come from the same data source, they should be appropriate relative to the other statistics.

Figure D.1 in Appendix D, shows the table from the Verizon DBIR that we used. It depicts the number of breaches they recorded, separated by the attack vector and attack variety. From this figure, we can gather statistics such as: given there is a breach the probability the attack variety was phishing = 653/4788.

Given there is a breach these values tell us the probability it is of the variety we are interested in. We still need the probability that a breach occurs at all. The UK 2018 Cyber Security Breaches Survey [167] found that 72% of UK large businesses (64% for medium, 47% for small and 40% for micro businesses) experienced cyber security breaches or attacks in the last 12 months.[6] We take the large business value and include the probability of a breach 0.72 in all our calculations. Naturally, one could adjust this for various factors, such as undetected breaches or multiple breaches being counted as a single breach. However, we will use 0.72 as an indicative value.

There are 12 attack vectors specified in the NIST 2017 Digital Authentication Guidelines. For each of these we estimate the probability of an attack occurring.

For example for the NIST 2017 level 1 policy the passwords are encrypted before transmission. Therefore the probability of an eavesdropping attack is:

$$\begin{aligned} \text{P[eavesdropping]} &= \text{P[keylogger breach]} + \text{P[physical surveillance]} \\ &= \left(\frac{(595)(0.72)}{4788}\right) + \left(\frac{(21)(0.72)}{4788}\right) = 0.0926 \end{aligned} \tag{5.18}$$

Whereas when there is no policy in place the probability an eavesdropping

[6]Micro businesses (1 to 9 employees), small businesses (10 to 49 employees), medium businesses (50 to 249 employees) and large businesses (250 employees or more)

attack is successful is:

$$\begin{aligned} \text{P[eavesdropping]} =& \text{P[network eavesdropping breach]} \\ &+ \text{P[keylogger breach]} + \text{P[physical surveillance breach]} \\ =& \left(\frac{(118)(0.72)}{3231}\right) + \left(\frac{(595)(0.72)}{4788}\right) + \left(\frac{(21)(0.72)}{4788}\right) = 0.1189 \end{aligned} \tag{5.19}$$

If there is no authentication procedure in place then any attempt to access an account will result in a success. To quantify the probability that each type of attack is attempted we use the same weightings from the DBIR table (Figure D.1). However, the survey measured the breach type, when there was a successful breach and each of the companies surveyed would have likely had some security policy in place. This does mean that we are underestimating the true benefit of an authentication policy since the Verizon DBIR reports statistics for companies who had authentication policies in place.

This method of underestimating the benefits is prudent. We decided it was best to underestimate the benefits and overestimate the costs and therefore gain a better understanding of whether a policy is actually valuable. Furthermore, as the underestimation exists for each policy, a comparison is still informative.

Using this method we will now calculate the expected losses from authentication related attacks when no policy is in place. For this organization this is estimated as: \$105,571.91. The expected losses when the NIST 2017 Level 1 policy is in place is \$24,529.71. This calculation is shown in detail in Appendix D.

$$\begin{aligned} \mathbb{E}[\text{Benefits}] &= \mathbb{E}[\text{Loss without policy}] - \mathbb{E}[\text{Loss with policy}] \\ &= \$105571.91 - \$24529.71 \\ &= \$81042.19 \end{aligned} \tag{5.20}$$

This tells us that the benefit for this organization of implementing the NIST 2017 Level 1 policy is \$81,042.

Table 5.4: Security benefits of the five NIST policies. These do not take into account the costs of implementing the policies.

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Benefits	$4,119	$105,548	$79,250	$81,042	$105,551

For each of the five policies (for this set up of the fictional company), Table 5.7 shows the security benefit of implementing the policy. These benefits will be more significant when put in terms of the costs of achieving them. The full computations for each policy are in Appendix D.

Notice that the security benefits from the NIST 2007 policy and the 2017 Level 1 policy are similar. Also, the 2003 Level 4 policy and the 2017 Level 3 policy have similar security benefits. However, as we will see later, the costs of the 2007 policy and 2003 Level 4 policy are significantly higher than their counterparts.

5.6.2 Do the benefits outweigh the costs?

We use our model to calculate the costs of implementing the NIST 2017 policy. For each piece of advice we identify the costs relating to it and the probability and number of times they will occur.

For example, the advice *Hash and salt passwords* will require: organization time to set up (C_4) and at each login a small amount of computing power (C_5) will be required to compare the supplied password with the stored hashed and salted password. In addition, a user will now need to create a new password (C_9) if they forget theirs, as it will not be possible for the organization to remind the user of their old one. The following equation shows the full computation for the additional cost as a result of hashing and salting passwords, notice that for each relevant cost category, C_i, we substitute a time variable into the equations defined in Table 5.3a.

$$\begin{aligned}\mathbb{E}[\text{Cost: hash \& salt}] &= (1)(C_4(1\text{ day}))(1) + (1)(C_5(2\text{ secs}))(\#\text{logins}) \\ &\quad + (0.0204)(C_9(85.4\text{ secs}))(\#\text{users}) \\ &= (1)(28800 \times 0.005)(1) + (1)(2 * 0.01)(130500) + (.0204)\big((20/100)(15) \\ &+ (1)(85.4)(0.005) + (2)(0.01) + (20/100)(10.5)\big)(500) \\ &= \$2810.57\end{aligned} \tag{5.21}$$

In Appendix D.4, the computations for all the key pieces of advice in the NIST 2017 Level 1 policy are given. Summing the costs for all the advice in the policy gives the total cost as:

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=21}} P_{i,j}.C_{i,j}.R_{i,j} = \$99778.94 \tag{5.22}$$

Table 5.5 shows the sum of all costs for this fictional company who wants to follow one of the five NIST policy options. We notice incredibly high costs for the NIST 2003 Level 4 policy and the 2007 policy. Recall that these values represent everyone inconvenienced and burdened by the policy. It accounts for the time taken out of the working day of every user who is forced to abide by these rules.

Notice that both the NIST 2003 L4 policy and the NIST 2017 L3 policy employ client authenticated TLS and Two-factor authentication using a specialised device, but the cost of the 2003 L4 policy is significantly higher than the cost of the 2017 L3 policy. Despite on the surface having similar policies, having strict lock-out and password composition rules as well as requiring a change of password every 2 years makes the NIST 2003 L4 policy significantly less user-friendly.

Notice that the 2007 policy which involved password expiry every 90 days is nearly as expensive as the 2003 L4 policy which involves client authenticated TLS and two factor authentication. This is more significant when we consider that the 2007 policy offered the second lowest contribution to security of the 5 policies, and now we also see that it comes with the second highest costs. Its

Table 5.5: Costs of implementation for the five NIST policies for this fictional company.

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Costs	$26,469	$12,155,278	$11,931,637	$99,779	$438,787

Table 5.6: Costs to the organisation of implementing the five NIST policies for this fictional company.

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Organisation's Costs	$4,527	$35,430	$32,037	$41,020	$59,197
Percentage of overall costs	17%	0.29%	0.27%	41%	13.5%

costs were 11858% those of the NIST 2017 L1 policy which offered a similar security benefit.

If the 2007 policy comes with such high costs, then why has it been one of the most implemented forms of an authentication policy over the last 20 years? Partially it is because this was the advice circulated at the time [20], and also Florêncio and Herley find that websites that are insulated from the effects of poor usability have no issue implementing complex password policies [49]. This is echoed here by looking at Table 5.6. Nearly all the cost for the NIST 2007 policy are borne by the user. 99.73% of the costs are felt by users rather than the organisation (the organisation only bears 0.27% of the total costs). Let us compare that to the 2017 L1 policy where the organisation carries 41% of the costs and the 2017 L3 policy where 13.5% of the costs are borne by the organisation. The figures for the 2017 policies demonstrate that if the organisation is willing to bear costs on themselves, the usability and security of the authentication policy can both be significantly improved.

Given that we now have the costs and benefits can we determine the Value of the policy to the organization? In Table 5.8, we can compare the costs versus the benefits for each of the five security policies.

Table 5.7: Security benefits of the five NIST policies. These do not take into account the costs of implementing the policies.

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Benefits	\$4,119	\$105,548	\$79,250	\$81,042	\$105,551
Costs	\$26,469	\$12,155,278	\$11,931,637	\$99,779	\$438,787

Organisation's Costs	\$4,527	\$35,430	\$32,037	\$41,020	\$59,197
Percentage of overall costs	17%	0.29%	0.27%	41%	13.5%

However, we do note that we encountered difficulty identifying the frequency and severity of security attacks in the wild. Therefore, while the benefits can certainly be compared to one another, there is some uncertainty when extrapolating benefit as a real world value. We believe an interesting future piece of work would be identifying the margins of error for the statistics we have provided and the impact this variation would have on the results.

In Table 5.8 row 1, all the the Values are negative. This implies that, when user costs are taken into account and prioritised with $U = 1$, the benefits of none of the policies outweighs the costs. However, note that if we look just at the organisation costs in Table 5.6 and the second row in Table 5.8, then all policies except for NIST 2003 Level 1 are cost-beneficial for the organisation to implement. In fact, to an organisation, the 2003 Level 4 policy appears to be the policy of most Value. This is despite offering essentially the same security as the 2017 Level 3 policy, but bearing 2670% times the cost of the 2017 Level 3 policy. This emphasises the importance of considering both user and organisation costs. In particular, if all users are employees within the organisation, then user time *is* organisation time and the costs cannot be avoided, though much of the cost may be hidden not in a security or IT budget, but in the salary cost of employees.

In Figure 5.2, we plot the Cost (x-axis) versus Benefit (y-axis) for each of the 5 NIST policies. For a policy to hold Value, we wish it to have a large benefit

Table 5.8: Value of the five NIST policies. This compares the security benefits and the implementation costs to determine Value.

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Value: Benefit - Cost	-$22,349	-$12,049,730	-$11,852,386	-$18,737	-$333,237
Benefit - only org. Costs	-$408	$70,118	$47,213	$40,023	$46,354

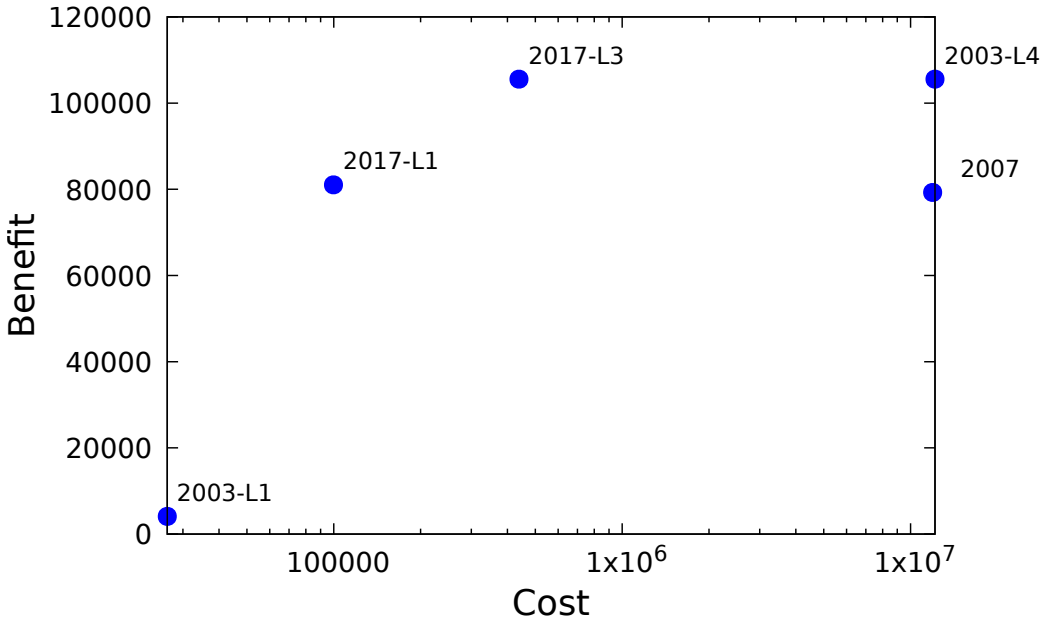


Figure 5.2: Scatterplot of Cost versus Benefit for the 5 NIST policies. Note log scale x-axis.

and a small cost. The costs for the 2003 L4 and 2007 policy are very high. Notice the log scale used on the x-axis. The costs and benefits for the 2003 L1 policy are both low. Looking at the advice, this is unsurprising. It offers little to no benefits, for little to no cost. Both the 2017 L1 and L3 policies look more promising. They both fall within the top left quadrant (more obvious had we not used a log scale on the x-axis) showing that for relatively low costs they offer relatively high benefits.

We noted that we might not have good comparability of the costs and benefits, due to the uncertainty in some of the parameters. However, we believe that there is reasonable comparability within the costs and within the benefits. We are going to use a z-score-based technique [16] which facilitates comparisons in this situation. To get a clearer comparison of the relative Value of each

Table 5.9: Value of the five NIST policies. Compares the security benefits and the implementation costs using composite z-scores

NIST policy	L1 2003	L4 2003	2007	L1 2017	L3 2017
Benefits z-score	-1.92	0.72	0.17	0.17	0.86
Costs z-score	0.84	-1.24	-1.21	0.83	0.77
Value of policy: Composite z-score	-0.52	-0.21	-0.53	0.48	0.78

NIST policy we convert the benefits and the costs for the five security policies into z-scores.

A z-score is simply a value that represents the number of standard deviations away from the mean that a value is. With these values we can create a composite z-score. This is done by summing the Benefit and the Cost z-scores and then computing the standard deviation of the resulting values [45]. The composite z-score is the sum of the Cost and Benefit z-scores for that policy divided by the standard deviation of the composite values. It is given as:

$$Z_i = \frac{(Bz_i + Cz_i) - \mu_{(Bz+Cz)}}{\sigma_{(Bz+Cz)}}$$

where Bz_i is the Benefits z-score for policy i and Cz_i is the Costs z-score for policy i. σ_{Bz+Cz} and μ_{Bz+Cz} are the standard deviation and mean respectively of all the $(Bz_i + Cz_i)$ scores for $i = 1 \ldots 5$.

Using this adjusted scoring method we can effectively compare the Value of each policy. Table 5.9 shows the composite z-scores which represent the comparative Value of each security policy. Notice that the two 2017 security policies have a z-score which is above the mean. The L1 2017 policy is just under half a standard deviation above the mean and the 2017 Level 3 policy is 0.78 of a standard deviation above the mean. The 2003 Level 1 policy and the 2007 policy both have cost benefit composite z-scores which are just over half a standard deviation below the mean. The Level 4 2003 policy lies at 0.21 of a standard deviation below the mean. These z-values give a single value indication representing the value of each authentication policy. It provides an effective means of comparing the policies and for this fictional company, they give an indication of which policy may be the most effective to implement.

5.7 General analysis of security policies

The above results tell us that the NIST 2017 policies are effective in comparison to the 2003 and 2007 NIST policies. We are now interested in evaluating the effect aspects of an organisation's characteristics and priorities have on the effectiveness of security policies.

In this section we investigate how the costs and benefits are affected by:

- The number of users?
- The saturation point α of the organization?
- The weight placed on user satisfaction, U?

5.7.1 Number of users

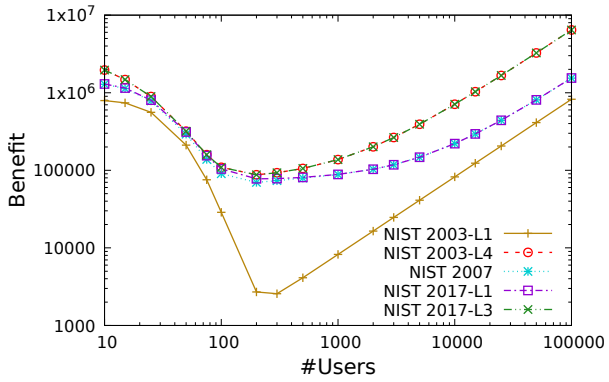
To investigate the effect that the number of users has on an authentication policy, we will plot the number of users against the Cost, Benefit, Value, and the standardized z-value for each of the NIST authentication policies. We will keep α and U constant at 0.5 and 1 respectively.

In Figure 5.3a the security benefits of each of the 5 policies is plotted. Notice that the Benefits is decreasing as we go towards 200 users. We are witnessing the affect of the L_{system} coefficient in our Losses model (Equation 5.13),

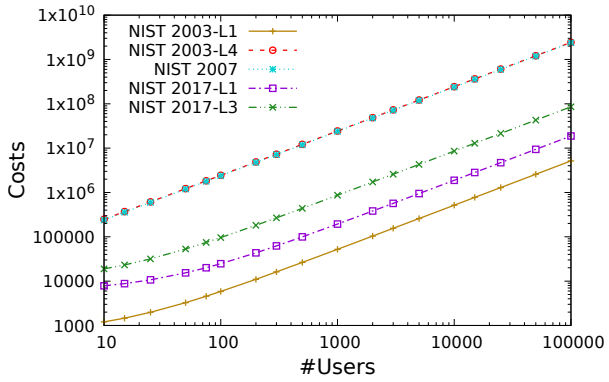
$$\mathbb{E}[\text{Loss}] = \underbrace{\mathbb{P}\left[\mathcal{N}\left(p, \frac{p(1-p)}{N}\right) > \alpha\right]} L_{system} + NpL_1 \tag{5.23}$$

This value tells us the probability that greater than $\alpha = 0.5$ users are compromised. After this point we consider the company's system to be overrun and the large L_{system} cost takes effect. When the number of users is small this L_{system} term is dominant. After the number of users exceeds ~ 200 (depending on policy), the L_{system} coefficient effectively becomes zero. The benefit then increases linearly according to NpL_1, the cost of each individual users' compromise.

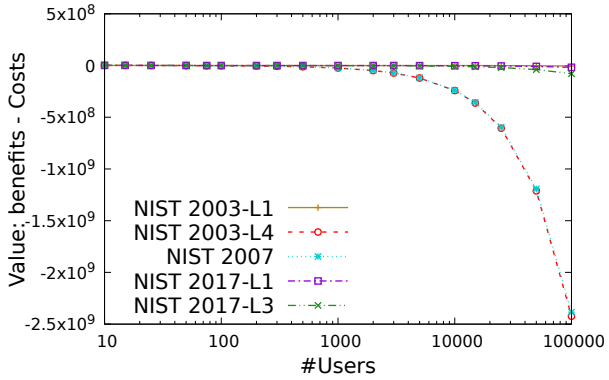
In Figure 5.3a the NIST 2003 Level 4 and NIST 2017 Level 3 policies have consistently the highest benefits. The 2017 Level 1 and 2007 policies have similar benefits throughout, and the 2003 Level 1 policy offers significantly



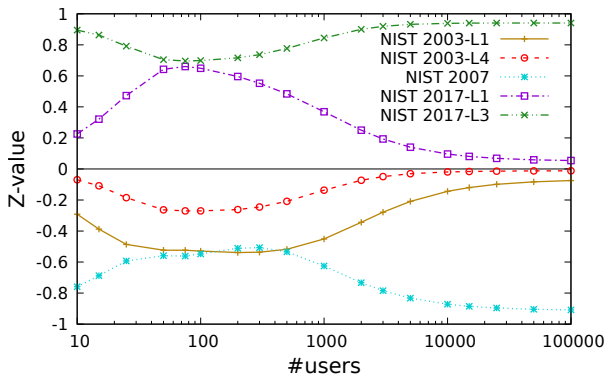
(a) Security benefit (log-log plot)



(b) Cost (log-log plot)



(c) Value



(d) z-value

Figure 5.3: Cost, Benefit, Value (benefits − costs) and Standardized value (z-value) for each NIST policy. Plotted against number of users.

lower benefits than all other policies. Notice the log scale on both the x- and y- axes in this plot.

Figure 5.3b shows the costs associated with each NIST policy. The costs are increasing roughly linearly with the number of users. Again, this graph is plotted using a logscale on both the x- and y- axes. The costs of the NIST 2007 and NIST 2003 Level 4 policies are overlapping. The NIST 2003 Level 1 policy has the lowest costs.

Figure 5.3c does not have a logscale y-axis as many values are negative. The Value of the NIST 2007 and NIST 2003 Level 4 policies decreases rapidly after $N \approx 1000$ users. Though they look similar, at the end point ($N = 100{,}000$), the NIST 2003-L1 policy has a Value $\approx -\$4$ Million, the NIST 2017-L1 policy has a Value $\approx -\$17$ Million and the NIST 2017-L3 policy has a Value $\approx -\$79$ Million. Recall that these values are for a User value of $U = 1$.

Unsurprisingly, when 100,000 users will be affected by a policy that a company implements, the costs to all of these users far exceeds the benefits that the policy brings to the organisation. In fact, for the NIST 2007 and NIST 2003 Level 4 policy, the Value falls below zero after just 30 users are affected by the policy. For the NIST 2017 Level 3 policy this happens when 115 users are affected, for the NIST 2003 Level 1 policy it is at 175 users and for the NIST 2017 Level 1 policy this happens at 390 users.

Finally, in Figure 5.3d we plot the standardized score which compares the value of the 5 security policies. This gives us a direct comparison of the policies against the number of users. Again, we can see the affect of the L_{system} coefficient and its impact on the effectiveness of the policies. However, despite this, for $U = 1$, $\alpha = 0.5$ we see that the order of value of the policies stays consistent along the number of users. The NIST 2017 Level 3 policy consistently shows the best benefits for costs trade off, while the NIST 2007 policy is the worst.

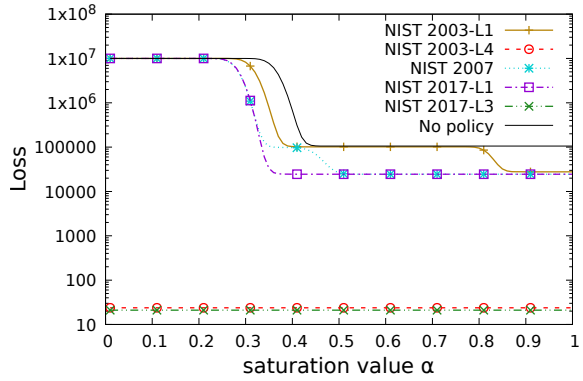
5.7.2 Changing saturation point α

The saturation point α represents the fraction of user accounts that need to be compromised before a system can be considered to be completely overrun. This point $0 < \alpha \leq 1$ can be relatively low, particularly if each currently compromised account can be leveraged to compromise others or if the organisation operates under the assurance of complete security. The α can in some ways be regarded as a crude measure of the level of risk an organisation can tolerate.

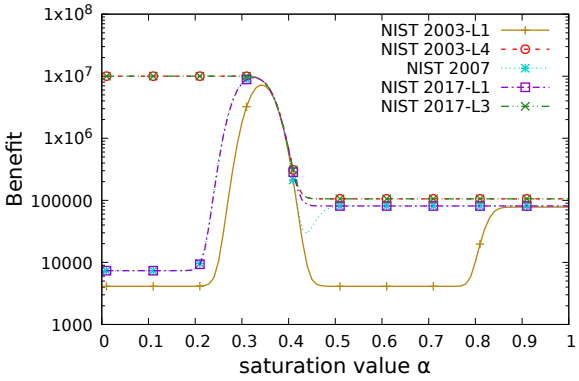
When an organisation is considered to be completely overrun it incurs a large L_{system} cost. This value can represent the cost of a total collapse of an organisations' system, the reputational damage an organisation might suffer as a result of a major breach, and also direct costs such as data protection fines [136], among other things.

Changing the value α has no impact on the costs. We plot the affect of changing alpha against Losses, Benefit, Value and z-value. In each plot we keep the number of users N and the user value U constant at 500 and 1 respectively.

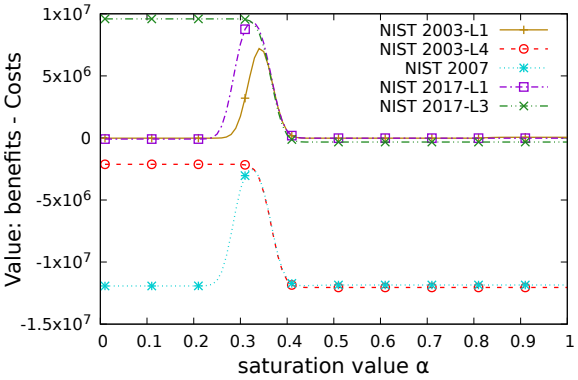
In Figure 5.4a we plot changing saturation point α against the Losses from



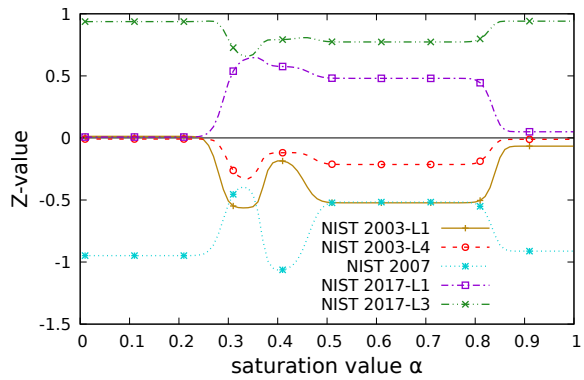
(a) Security loss (log y-axis)



(b) Security benefit (log y-axis)



(c) Value



(d) z-value

Figure 5.4: Cost, Benefit, Value (benefits − costs) and Standardized value (z-value) for each NIST policy. Plotted against saturation point α.

each policy. We also include in the plot the Losses as a result of No policy. The maximum losses when no policy is in place are generally dictated by the value of the organisation in question. For our set up of a fictional company, the value of the organisation is $L_{system} = \$10^7$. Notice that when $\alpha < 0.3$, for user cohort of size 500, none of the NIST 2003 L1, NIST 2007 or NIST 2017 L1 policies differ significantly from the No policy losses. This is because, at this low saturation level, it is highly likely that the threshold α of users will be compromised and the system will become overrun, thus causing the with-policy losses to be similar to the without-policy losses.

The NIST 2003 Level 4 and NIST 2017 Level 3 policies, which employ client authenticated TLS and two factor authentication, exhibit a constant Loss across all saturation values. This is because, with these policies in place, even at the low, $\alpha = 0.01$, saturation level, it is still unlikely that the threshold α of users will be compromised and the system will become overrun. This

security is a result of users of these policies being protected by three factors, their password, their client-authentication and their specialised cryptographic hardware device. An attacker wishing to compromise a single user, must overcome all three mechanisms.

In Figure 5.4b we plot the policy benefits against saturation value α. Recall that the benefit of an authentication policy is the difference between the losses when no policy is in place and the losses when the policy in question is in place. The smallest benefits are consistently for the NIST 2003 Level 1 policy. The NIST 2003 Level 4 and NIST 2017 Level 3 policies both have high benefits. Note the log scale on the y-axis.

Figure 5.4c demonstrates how the Value of each policy changes as a result of different saturation values α. We can see that up until a saturation point of 0.3, for this fictional company with 500 users, the NIST 2017 L3 policy has significant economic and usability value over all other policies. In fact, the NIST 2017 L3 policy has economic value for companies with a low saturation threshold, even when up to $N = 12{,}500$ users are impacted. This indicates that for organisations where security is critical, the strict 2017 L3 policy is advantageous. As the saturation point increases, the NIST 2017 L3 policy decreases to fall below the NIST 2017 Level 1 and NIST 2003 Level 1 policies.

In Figure 5.4d the z-value for each policy is plotted against the saturation value. There is some oscillation in the exact z-values for each policy as the saturation value changes. However, the 2017 Level 1 and 2017 Level 3 policy are consistently above the mean. Showing that they have good combined scores in security and usability in comparison to the other policies. At saturation point ~ 0.3 we see the 2017 Level 3 policy decrease in value and the 2017 Level 1 policy increase in value. Showing that for high risk organisations, the NIST 2017 Level 3 policy is highly valuable and for lower risk organisations both the 2017 Level 1 and 2017 Level 3 policies hold value.

Overall, we find that the saturation point has a significant impact on the size of the benefits of each policy. The security benefits can differ in orders of magnitude depending on the assigned saturation point. In addition, different policies hold more Value depending on the saturation value. This emphasises the importance of choosing a policy with respect to the risk framework in which it is expected to operate.

5.7.3 Weight on user satisfaction

In our costs model we introduced a weighting on the importance placed on user satisfaction. Where a user is part of an organisation, all user time and login time should be considered as organisation time, and therefore the only logical setting for the user weighting is $U = 1$. This is the value we have used throughout the above examples.

However, when the user is external to the organisation or, equivalently, the user's time is considered to be mutually exclusive from the organisation time, then this user satisfaction value can take any value between 0 and 1.

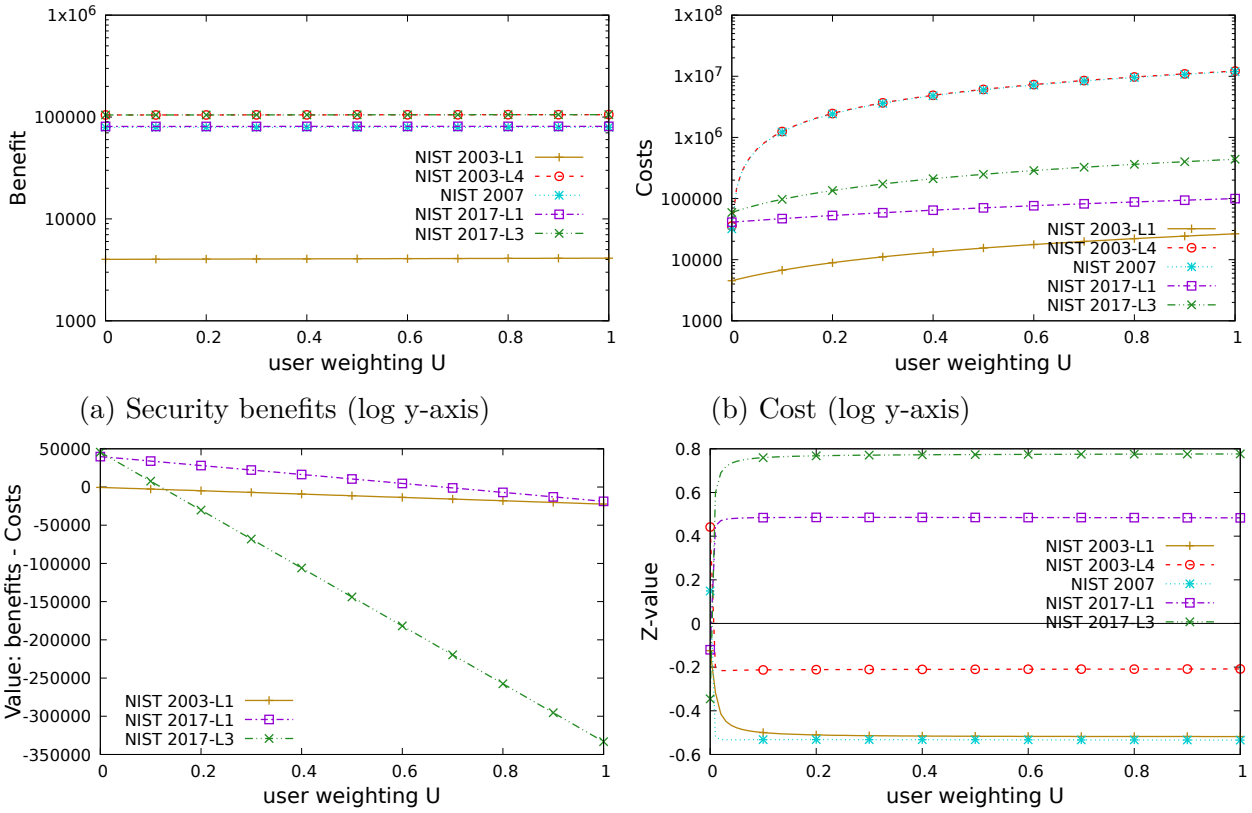
The value of U impacts the Benefits in a minor way. This is because the value L_1, which represents the loss as a result of each individual user compromise, includes the time the user must wait to regain access to their account:

$$L_1 = \$\text{ORG PROFIT PER USER} + (\text{U})(\text{TIME ADMIN RESET})(\$\text{USER WAGES}) + (\text{TIME ADMIN RESET})(\$\text{ADMIN WAGES}) + \$\text{DAMAGES}.$$

In the Figure 5.5 plots we set $\alpha = 0.5$, $N = 500$ and vary the user weighting value U.

In Figure 5.5a, we plot the security benefits against the changing user weighting value U. There is a minor increase in the benefits. For example, the NIST 2017 L1 policy has a benefit of \$80,850 at $U = 0$ and a benefit of \$81,042 at $U = 1$, a difference of \$192. However, this is insignificant when compared to the overall benefit values.

Figure 5.5b plots the costs against the user weighting. There is a roughly linear relationship between the costs and the user weighting (Note the log scale on the y-axis). This is because each increase in user weighting will bring a corresponding increase to the costs. The largest increase gradient is between the first two discrete values plotted: $U = 0$ and $U = 0.01$. When $U = 0$, the NIST 2017 Level 3 policy has the highest costs of any policy with the NIST 2017 Level 1 policy in second place. However, as soon as a $U = 0.01$ user weighting is applied, this ordering changes dramatically. The NIST 2007 and NIST 2003 Level 4 policy become the most expensive by a significant margin.



(a) Security benefits (log y-axis)

(b) Cost (log y-axis)

(c) Value

(d) z-value

Figure 5.5: Cost, Benefit, Value (benefits − costs) and Standardized value (z-value) for each NIST policy. Plotted against user weighting U.

Notice that the increasing user weighting has a relatively small impact on the costs of the NIST 2017 Level 1 policy.

In Figure 5.5c we do not include the Value of the NIST 2003 Level 4 policy and the NIST 2007 policy. The Value of both linearly decreases down to $-\$1.2 \times 10^7$ as $U \to 1$. We remove these in order to show a clearer plot of the three remaining policies: NIST 2003 Level 1, NIST 2017 Level 1 and NIST 2017 Level 3. The NIST 2017 Level 3 policy has a positive economic value when the users are weighted as having an importance less that $U = 0.13$. All three policies have Value which decreases as user importance increases. This is to be expected when trading off benefits against (user and organisation) costs. The Value of the NIST 2017 Level 1 policy is the least affected by changing U. Up until $U = 0.8$ the Value of the policy remains positive.

Figure 5.5d shows the standardized z scores for the 5 policies. This offers a

simple comparison of the policies' value is relation to each other. We can see that other than for $U = 0$, the z-values of the policies remain largely unaffected by changing U. At $U = 0$ though the order of policies according to z-value is significantly different. The NIST 2003 Level 4 policy shows the highest z-value with NIST 2007 second and then NIST 2017 Level 1, NIST 2003 Level 1 and finally NIST 2017 Level 3. This mirrors the (Benefits - costs) Value at $U = 0$. As shown in Table 5.8 in the "Benefit - only org. Costs" row, we see that the NIST 2003 Level 4 policy has the highest Value. This is a result of the 2017 Level 3 and 2017 Level 1 policies having the highest costs when $U = 0$, i.e., having the highest 'organisation only' costs (Table 5.6). This tells us that when only the organisation costs are of any importance and their is no value assigned to user time, burden, or effort then the NIST 2003 Level 4 policy could be seen as effective.

It is important to note, that we do not see any situation where the weighting assigned to the value of users is set to $U = 0$. As Cranor and Garfinkel [33] put it:

> "Computers that are theoretically secure but not usable do little to improve the security of their users, because these machines push their users away to less secure platforms."

To consider users' time and effort to be of no value is naive when considering an authentication system.

5.7.4 General analysis summary

We find that the number of users that an authentication policy will affect will have an impact on the expected benefit and costs of a policy. The saturation point α is defined as the fraction of users who need to be compromised before the system can be completely overrun. An α close to 1 represents an organisation that is resilient to security risks due to individual user compromise. A changing saturation point has an impact on which policy will be the most effective for an organisation to enforce. We find that the difference in the Value of a security policy is significant when considered with and without users in mind ($U = 0$ versus $U = 1$). However, variation in user weightings between $0.1 \leq U \leq 1$ has little impact on policy value.

Overall we find that the NIST 2017 Level 1 policy shows value for widespread deployment and the NIST 2017 Level 3 policy is very effective when security risk is high ($\alpha < 0.3$). The NIST 2003 Level 1 policy brings no significant improvement to security, but correspondingly comes with very little user or organisation cost. The NIST 2007 and NIST 2003 Level 4 policies place significant burdens on users and can only claim to offer the same amount of security as the NIST 2017 Level 1 and NIST 2017 Level 3 policies respectively. Despite coming with a 2670% increase in costs.

5.8 Conclusion

In this chapter, we provided methodology for quantifying the costs and benefits of an organisations' authentication policy.

To test these models we applied them to the NIST 2003, 2007 and 2017 authentication guidelines. We expected the 2003 and 2007 guidelines to show poor usability based on the wide scale criticism they recently received [101, 155]. In contrast, the new 2017 guidelines have been praised as user friendly and security effective [32]. In Appendix D we provide the calculations for the costs and benefits for each of these security policies.

This evaluation demonstrated the Value of the 2017 policies and emphasised the problems with the 2003 policies. We showed that the 2003 Level 1 policy provided little to no security benefit. The 2007 policy which involved expiry and complex password composition requirements offered some security but major usability costs. We found that, of the five policies evaluated, the 2017 policies were the only policies which could offer both high security and high usability.

Using these policies as a basis, we investigated the impact an organisation's characteristics have on the effectiveness of the policy they choose. We modified the number of users, the organisation's security risk, and the importance placed on users, and assessed the impacts this had on the Value of a policy. We found that the most impact is derived from the organisation's security risk. A high risk organisation will need a different security policy in place than a low risk organisation. Similarly the number of users expected to follow a policy will impact the Value of that policy in relation to the Cost and Benefit trade off. This demonstrates that a one-size-fits-all policy is not effective for security

policies. Policies need to be applied when and how they are needed and not as blanket-restrictions to all users. The NIST 2017 Level 1 and Level 3 policies showed evidence in our evaluations of being able to offer effective differential security at their respective levels. We would be interested, in future work, to also assess the effectiveness of the NIST 2017 Level 2 policy.

While modeling the organisation's risks and benefits, we found that we couldn't always find good estimates for the quantities we needed. The further development of this work, particularly with the aid of additional security breach statistics and enforcement cost information would improve the realism of our work. We believe this research provides a valuable basis for the effective comparison of security policies which will allow security administrators to quantify and support security concerns and security progress within organisations.

CHAPTER 6

Conclusions

This thesis aimed to provide information about password use and abuse in order to help improve authentication for users. In this chapter, we will review what we achieved and give recommendations for possible extensions.

6.1 Summary

The core contributions of this research are discussed in four chapters. These respectively look at: evaluating password advice, convergence of password guessing to optimal success rates, multi-armed bandit password guessing and quantifying the costs and benefits of authentication policies. Below, we will discuss each chapter in turn before discussing challenges and future work in the next section.

In Chapter 2, we began with an evaluation of currently circulated password advice. We categorized and discussed 272 password recommendations given by security specialists, multinational companies and public bodies. This highlighted stark variations between advice given by different sources. 41% of the recommendations we collected were contradicted by recommendations given by another source. We surveyed administrators and users about the costs associated with following this advice. The responses were catalogued in tables illustrating the costs and perceptions that accompany password advice. The collected advice and the survey responses have been made available on

Github for use by future researchers [111, 112, 113]. Our research exposed the disconnect that exists between the recommendations of security research, and the advice given by organisations and believed by users. Finally in this chapter, we qualified which attack vectors each piece of advice offered protection against. We found that most advice was concerned with protecting against online and offline guessing attacks, with little emphasis on the security of back-end processes.

In Chapter 3, we delved deeper into the security concerns associated with online and offline password guessing. We were specifically interested in methods to quantify the concept of guessability in order to offer guessing success results that are independent of the underlying distribution of the data. This allows guessing success to be compared between two password datasets in a meaningful way. Our method utilised the comparison of optimal guessing to actual guessing to provide a guessability statistic based on loss. The creation of such a statistic allowed for detailed proofs to evolve which could offer insight into the characteristics of password guessing. In particular, we proved that a sample of users' passwords selected from a chosen passwordset, can be leveraged to guess the remaining users' passwords in that dataset with guessing loss which converges to zero as the number of samples gets large. We showed both theoretically and in practice that a sample of passwords taken from a dataset will help compromise the remaining users in that dataset effectively. In fact, a sample accounting for just 1% of a website's users provides an attacker with enough information to potentially have a success rate of over 84% when trying to compromise other users of the same website.

In Chapter 4, we showed that it is possible for a learning algorithm to actively learn characteristics of the passwords it is guessing, and that it can leverage this information to tailor and improve its guessing. The learning model is based on the multi-armed bandit problem which describes the trade-off a gambler faces when faced with a number of different gambling machines. We showed that the techniques used to solve the multi-armed bandit problem can be used to help an automated attacker adaptively choose between a selection of password dictionaries by optimising according to feedback received after each guess is made. The multi-armed bandit model has the ability to distinguish a demographic category such as nationality from a password set. This can offer insights into password datasets and their sources. Knowing the potential of

this guessing model is useful for both users and organizations. It provides further evidence for the importance of guiding users away from passwords which reflect characteristics associated with demographic or website specific terms. It also demonstrates that password choices differ measurably depending on their source use. This could indicate that websites could consider tailored blocklisting techniques.

Chapter 5 revisits the costs and benefits that were qualified in Chapter 2, but this time our goal was to use these categories to build a model for quantifying the value of an authentication policy. We built a model which took into account users and organisation time associated with following an authentication policy, and compared this to the security benefits the policy provides. We used these to quantify the value of the NIST 2003 Level 1 and Level 4, 2007, and 2017 Level 1 and Level 3 authentication policies. Of these five policies evaluated, we found that only the 2017 policies offer both high security and high usability. We were also able to leverage our model to investigate the characteristics that have the greatest influence on the effectiveness of authentication policies. We investigated the impact of number of users, organisation's security risk, and the importance of users, on the Value of a policy. We found that a high risk organisation will need a different security policy in place than a low risk organisation. We also conclude that if an organisation is willing to bear costs on themselves, they can significantly improve usability for their end-users, and simultaneously increase their security.

6.2 Challenges and Future Work

In Chapter 2, we created a concept of how to categorise password guessing. This method involved assigning a severity (major, minor, positive) and frequency (once-off, periodic, at login) to each cost category for each piece of advice. After developing this methodology we wished to ask users for their input. While we believe this model is an accurate way of differentiating costs, it made for a convoluted user study. Some users misunderstood the ranking, in particular the concept of a 'positive cost' being one that has a positive impact on their authentication process. If repeating this survey, we would simplify the general questionnaire and potentially conduct detailed interviews with a smaller cohort of participants where we can learn about the finer details. On the whole, we still received meaningful feedback from our user and admin-

istrator surveys. After the writing of Chapter 2, our user surveys remained open and we plan to update our cost results in future work as we gather a larger number of responses.

In Chapter 3, we were able to leverage our model of guessing loss in order to derive theorems about the characteristics of passwords. However, the bounds on the guessing functions were broad, and in our empirical results we observed much faster convergence to the optimal than our theorem's convergence rate suggested. We believe structured functions describing guessing are valuable for password security research, as they offer insights into bounds and characteristics. We would be interested in future work which could develop those introduced in our research.

There is broad potential for expanding our multi-armed bandit research. We would be particularly interested in research that explored the extent to which it is possible to learn personally identifiable information (PII) about users within a password database. In Chapter 4, we showed that it is possible to infer information about the users' nationality. When a password dataset is released, often PII is released along with it, such as age and gender. If this information was combined with information that could be inferred from the passwordset and used to identify individuals, then users would face even more serious risks when their credentials are leaked.

In a similar direction, often password databases are found but it is not known whether they are genuine password leaks, and if they are, then it is not known where they come from. Combining the work from both Chapter 3 and Chapter 4, we would be interested in knowing whether characteristics in samples of passwords could be linked back to the source dataset.

Chapter 5 contains the research in this thesis that proved the most challenging. What began as an interesting, small project turned out to be a much larger undertaking. We were constantly thwarted by a lack of available data. In particular, we struggled to find data about security breach attempts and security breach successes for each of the different types of attacks that we know exist. In addition to this, we did not have access to company specific data which would have been helpful for a more accurate 'fictional company'. We would like to thank our university IT department who provided some information related to uptake for two factor authentication and provided feedback during

our modelling. We encourage the further development of this work, particularly with the aid of additional security breach statistics and enforcement cost information from within an organisation. We believe this will be possible, as organisations who wish to make decisions about their authentication policies can input their own data into our model.

APPENDIX A

Advice Statements

This appendix lists each piece of advice that was collected for Chapter 2. This illustrates which pieces of advice were contained under each category and in each advice statement how all the pieces of advice were categorized. A star (*) at the beginning of the advice means it contradicts the statement it is listed under. The categories are shown in alphabetical order:

There were 27 categories of advice in total. These are listed below. The advice under each category are listed in this appendix.

- Administrator Accounts
- Backup Password Options
- Backup Work
- Composition
- Default passwords
- Expiry
- Generated Passwords
- Individual Accounts
- Input
- Keeping system safe
- Keep your Account safe
- Length
- Network: SNMP Community strings
- Password Auditing
- Password Manager

- Personal Information
- Personal Password Storage
- Phrases
- Policies
- Reuse
- Sharing
- Shoulder surfing
- Storage
- Throttling
- Transmitting passwords
- Two factor authentication
- Username

A.1 Administrator Accounts

Administrator account not for everyday use

1. Do not use an account with administrator privileges for everyday use.

Password different for administrators account than users' other accounts

1. User accounts that have system-level privileges granted through group memberships or programs such as sudo must have a unique password from all other accounts held by that user to access system-level privileges.
2. Administrators must use different passwords for their administrative and non-administrative accounts.

Give administrator accounts extra protection

1. Give administrators, remote users and mobile devices extra protection.

A.2 Backup Password Options

Email up-to-date and secure

1. Make sure your backup password options are up-to-date and secure.
2. Make sure to regularly update your recovery email address

3. Make sure your email password is also strong.

Security answers difficult to guess

1. The answer shouldn't be something that someone can guess by scanning information you've posted online on blogs or social networking profiles.
2. If you have to choose a question from a list of options, such as the city where you were born, try to find a way to make your answer unique.
3. Verifiers also SHALL NOT prompt subscribers to use specific types of information (e.g., "What was the name of your first pet?") when choosing memorized secrets.

Do not store hints

1. Do not hint at the format of a password
2. Memorized secret verifiers SHALL NOT permit the subscriber to store a "hint" that is accessible to an unauthenticated claimant.

A.3 Backup work

Make digital & physical back-ups.

1. Make electronic and physical back-ups or copies of all your important work.

A.4 Composition

Must include special characters

1. Use a mix of letters, numbers, and symbols.
2. Include numbers, capital letters and symbols.
3. Numbers, symbols and combinations of upper and lower case can be used.
4. Use a combination of upper & lower case letters, numbers and keyboard symbols.
5. Passwords less than 12 characters long should contain mixed case letters, digits, and symbols.
6. Use lower case, upper case, a number, and a special character.

7. Use combinations of numbers, symbols, and letters (uppercase and lowercase).
8. *No special characters.
9. *Passwords must not contain non-English characters.
10. *No spaces
11. *Spaces shouldn't be used as some applications may trim them away.
12. *The password cannot contain the space character.

Don't repeat characters

1. Passwords should not contain more than two (2) consecutive repeated characters.
2. Passwords must not contain repeating strings of 3 or more identical characters.
3. The password cannot contain three or more repeated characters.

Enforce restrictions on characters

1. Include upper and lowercase letters, and at least one number.
2. Contain at least 1 letter. Contain at least 1 number.
3. Use a mix of alphabetical and numeric characters.
4. Do not choose any all-numeral passwords.
5. Passwords must use at least three of the four available character types: lowercase letters, uppercase letters, numbers, and symbols.
6. The password should contain a minimum of one (1) non-alphabetic character.
7. Passwords should contain at least one number and at least one special character.
8. Must contain at least one non-alphabetic character and at least four alphabetic characters.
9. Passwords between 12 and 15 characters long (inclusive) should contain mixed case letters and digits.
10. Use a mixture of upper- and lowercase.
11. Use a mixture of upper- and lowercase.
12. Passwords 20 characters or longer can contain just a single case of characters.

13. *Verifiers SHOULD NOT impose other composition rules (mixtures of different character types, for example) on memorized secrets.

A.5 Default Passwords

Change all default passwords

1. Change the manufacturer's default passwords on all of your software.
2. Change all default passwords.
3. All vendor-supplied default passwords must be changed before any computer or communications system is used.
4. Make sure that absolutely no default administrator passwords are used.

A.6 Expiry

Store history to eliminate reuse

1. Enforce Password History. Set how frequently old passwords can be reused. Users are not allowed to reuse any of the stored passwords.
2. A password history must be maintained for all domain level.
3. On all multi-user machines, system software or locally developed software must be used to maintain an encrypted history of previous fixed passwords.
4. Passwords must not be the same as the previous password.
5. The password cannot match any of the last eight passwords.

Change your password regularly

1. Passwords should be changed periodically.
2. Enforce a Maximum Password Age, users should change passwords regularly.
3. Change your password regularly.
4. Have a minimum Password Age
5. Password expiration should be enforced on all accounts.
6. Change your passwords regularly (every 45 to 90 days).
7. All user-level passwords must be changed at least every six months.
8. All system-level passwords must be changed on at least a quarterly basis.
9. *The routine changing of passwords is not recommended.

10. *Don't change your passwords, unless you suspect they've been compromised.
11. *Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically)
12. *Normally, there should be no reason to change your password or PIN.

Change password if compromise is suspected

1. If you think that someone else knows your password, change it immediately.
2. Unless the accounts to which they apply have been hacked, in which case they should be changed immediately.
3. If you suspect that someone else knows your password, you should change it immediately.
4. Don't change your passwords, unless you suspect they've been compromised.
5. Any user suspecting that his/her password may have been compromised must report the incident and change all passwords.
6. Unless there is evidence of compromise of the authenticator or a subscriber requests a change.
7. Only ask users to change their passwords on indication or suspicion of compromise.
8. Whenever an unauthorised party has compromised a system the relevant Autonomous network manager or application administrator must immediately change every password on the involved system.
9. Passwords must always be changed if it is known or suspected that another person has become aware of the password.
10. If you notice something suspicious on your PayPal account. You suspect that someone you don't trust has your password.

A.7 Generated Passwords

Create using a random bit generator

1. SHALL be generated using an approved random bit generator.
2. Must be generated using the low order bits of system clock time or some other frequently changing unpredictable source.

Must aid memory retention

1. Choose a scheme that produces passwords that are easier to remember.
2. Offer a choice of passwords, so users can select one they find memorable.

Generated passwords must be issued immediately

1. Generated passwords and pins must always be issued immediately after they are generated.

Only valid for first login

1. Generated password valid only for the involved user's first on-line session.

Distribute passwords in an envelope.

1. Passwords must be concealed inside an opaque envelope that will readily reveal tampering.

A.8 Individual Accounts

One account per user

1. Everybody who uses a computer should be assigned their own user account.
2. Applications must support authentication of individual users, not groups.
3. Computer and communication system access control must have passwords unique to each individual user.
4. Do not allow password sharing.

Accounts must be password protected

1. Each user account should be accessible only by entering a username and password.
2. Computer and communication system access control must be achieved via passwords.
3. Applications must provide for some sort of role management, such that one user can take over the functions of another without having to know the other's password.

A.9 Input

Truncation should not be performed

1. Truncation of the secret SHALL NOT be performed.

All ASCII and UNICODE characters should be accepted

1. All printing ASCII[RFC 20]characters as well as the space character SHOULD be acceptable in memorized secrets. Unicode[ISO/ISC 10646]characters SHOULD be accepted as well.
2. *To make allowances for likely mistyping, verifiers MAY replace multiple consecutive space characters with a single space character prior to verification, provided that the result is at least 8 characters in length.

A.10 Keeping system safe

Implement Defense-in-Depth

1. Implement Defense-in-Depth: a layered defense strategy includes technical, organizational, and operational controls.
2. Computer and communication systems must be designed, tested, and controlled so as to prevent both the retrieval of, and unauthorized use of stored passwords,

Implement Technical Defenses

1. Implement Technical Defenses: firewalls, intrusion detection systems, and Internet content filtering.

Update anti-virus

1. Update your system's anti-virus software daily.
2. All devices connected to the Internet must be equipped with the latest versions of anti-virus software

Regularly apply security patches

1. Regularly download vendor security "patches" for all of your software.

Monitor and analyze successful and attempted intrusions

1. Monitor, log, analyze, and report successful and attempted intrusions to your systems and networks.

Boot protection

1. All workstations, no matter where they are located, must use screen-savers with fixed-password-based boot protection along with a time-out-after-no-activity feature.

A.11 Keep your account safe

Check for encryption and SSL

1. Check for encryption and SSL.-when accessing account.

Manually type URLs

1. Log on manually by typing what you know to be the site's URL into your browser window.

Don't open emails from strangers

1. Do NOT open emails, links, or attachments from strangers.

Keep software updated

1. Make sure you're using up-to-date anti-malware software
2. Make sure that your operating system is up-to-date.

Log out of public computers

1. When using a public computer, always sign out when your session is complete to prevent other people from accessing your account.
2. Don't forget to log out on a cybercafe computer.

Password protect your phone

1. Use a "password" or fingerprints for your phone too.

A.12 Length

Minimum password length

1. Set a minimum password Length. At least eight characters.
2. Minimum password length should be 8 characters.
3. The minimum password length is 8 characters.
4. Eight or more characters.
5. Length of 6 to 40 characters.
6. Make the password at least 8 characters long.
7. At least 8 characters in length.
8. More than 8 characters long.
9. Minimum of 8 characters.
10. All passwords must have at least eight (8) characters.
11. Passwords must be at least 9 characters long.
12. Do not choose passwords of fewer than six characters.
13. Choose a password with at least eight characters. If you want greater security, set the minimum password length to 14 characters.

Maximum password length (<40)

1. Passwords should be no more than fifteen characters in length.
2. Maximum password length 40.
3. Less than 20 characters long.
4. *Verifiers SHOULD permit user-chosen memorized secrets to be at least 64 characters in length.

A.13 Network: SNMP Community strings

Don't define community strings as the standard defaults.

1. The community strings must be defined as something other than the standard defaults of public, private, and system

Community string must differ from login passwords.

1. The community strings must be different from the passwords used to log in interactively.

A.14 Password Auditing

Try guessing passwords

1. Password cracking or guessing may be performed on a periodic or random basis by the Infosec Team or its delegates.

A.15 Password Managers

Use a Password Manager

1. If you have a difficult time remembering multiple passwords, a trusted password manager may be a good solution.
2. An alternative to writing down passwords is to use an online password vault or safe. Use a Password Manager
3. *Password management software can help users, but carries risks.

Create long random passwords with password manager

1. Configure your password manager to create 30-50 random characters with a mixture of upper- and lower-case letters, numbers, and symbols.

A.16 Personal information

Password should be unrelated to personal information

1. Create a unique password that's unrelated to your personal information.
2. Don't use family members' or pets' names. Or family birthdays. Or your favorite football or F1 team or other words easy to work out with a little background knowledge.
3. A car license plate number, a spouse's name, or an address must not be used.
4. Not a word or date associated with you (like a pet's name, family names, or birth dates).
5. Don't use any personal information. Even when combined with letters and numbers, someone who knows you, or can research you online, can easily guess a password with this information.
6. *Personal details such as spouse's name, vehicle license plate, PPS or social security number and birthday must not be used unless accompanied by additional unrelated characters.

Passwords must not match account information

1. Do not choose any ID number or user ID in any form, even spelled backwards.
2. Passwords can't contain the username.
3. Passwords must not contain your LoginID
4. Passwords must not contain any email address on record for the user profile.
5. The password cannot contain email address.
6. Can not match username.
7. Don't use your username or business name.
8. Do not choose part of your userid.

Do not include names

1. Do not choose your name in any form - first, middle, last, maiden, spelled backwards, nickname or initials.
2. Do not choose any common name, e.g., Sue, Joe.
3. Passwords can't contain parts of the user's full name such as his first name.
4. The password cannot contain your name.
5. Don't use your actual name
6. Do not choose part of your name.
7. *You could use someone else's mother's maiden name.

A.17 Personal Password Storage

Don't leave passwords in plain sight

1. Don't leave notes with your passwords to various sites on your computer or desk.
2. Don't post it in plain sight.
3. Do not write passwords down and store them anywhere in your office.
4. Passwords must not be written down and left in a place where unauthorised persons might discover them.

Don't store in a computer file

1. Don't store passwords in a document on your computer.
2. Do not store passwords in a file on a computer system or mobile devices (phone, tablet) without Encryption.
3. *If you decide to save your passwords in a file on your computer, create a unique name for the file so people don't know what's inside.

Write down safely

1. If you must write it down, hide the note somewhere where no one can find it.
2. If you must write passwords down in order to remember them, encrypt them in a way that is familiar to you but makes them indecipherable by others.
3. If you have to keep your passwords somewhere, it is safer to write them on a piece of paper and store that paper in a secure location, like a safe.
4. The display and printing of passwords should be masked, suppressed, or otherwise obscured.
5. Store unrecoverable passwords.
6. Allow users to securely record and store their passwords.
7. *Don't write down your password

Do not allow applications to remember your password

1. Don't save your passwords in a web browser.
2. Do not use the "Remember Password" feature of applications (for example, web browsers).
3. Don't save passwords or use "Remember Me" options on a public computer.

A.18 Phrases

Don't use patterns

1. Don't use keyboard patterns such as qwerty or qazwsx,
2. Dont use sequential patterns such as abcd1234
3. Don't use numerical sequences.
4. Don't use repetitive patterns on the keyboard.
5. Common character sequences must not be employed.

6. Do not use ascending or descending numbers (for example 4321 or 12345), duplicated numbers (such as 1111) or easily recognisable keypad patterns (such as 14789 or 2580).

Blacklist common passwords

1. Prohibit the most common passwords by blacklisting.
2. Verifiers SHALL compare the prospective secrets against a list that contains values known to be commonly-used, expected, or compromised.

Don't use published phrases

1. Don't use song lyrics, quotes or anything else that's been published.
2. Do not choose names from popular culture, e.g., Harry_Potter, Sleepy.
3. *Choose a line of a song that other people would not associate with you.

Substitute symbols for letters

1. Consider using symbols and numbers in place of letters.
2. Replace a letter with another letter, symbol or combination, but don't be too obvious about it.
3. *Do not choose any word with any of the following substitutions: a $\rightarrow$ 2, a $\rightarrow$ 4, e $\rightarrow$ 3, h $\rightarrow$ 4, i $\rightarrow$ 1, l $\rightarrow$ 1, o $\rightarrow$ 0, s $\rightarrow$ \$, s $\rightarrow$ 5, z $\rightarrow$ 5

Don't use a single word

1. Don't use a single commonplace dictionary word
2. Do not choose acronyms, geographical or product names, and technical terms.
3. Do not choose a single word either preceded or followed by a digit, a punctuation mark, up arrow, or space.
4. Passwords must not be a word found in the dictionary or some other part of speech.
5. Dont use Words like password or letmein
6. Don't use the word password
7. Compare the prospective passwords against a list of known commonly-used, expected, and/or compromised values.
8. Prohibit the most common passwords by blacklisting.

9. Pick a deliberately misspelled term
10. Words in a dictionary must not be employed.
11. User-chosen passwords must also not be any part of speech.
12. Pick an odd character in an otherwise familiar term, such as phnybon instead of funnybone
13. Choose an easily phonetically pronounceable nonsense word.
14. Combine random words.
15. Combine random words to create a passphrase.
16. Choose a combination of two unrelated words.

Insert random numbers and symbols

1. Do use a random word or phrase and insert letters and numbers into the beginning, middle, and end.
2. Chose a phrase and pad with symbols, uppercase letters and numbers.
3. Two words separated by a non-alphabetic, non-numeric, or punctuation character.
4. A combination of words with unusual capitalization, numbers, and special characters interspersed.
5. *Don't use passwords with combinations of random letters, numbers and symbols. e.g. jal43#Koo%a.

Take initials of a phrase

1. Use a phrase and use the initial of each word.
2. Pick a phrase known to you and take the first character from each word.
3. Take initials of a phrase and swap letters for numbers.
4. Initials of an easy to remember quote of phrase.

A.19 Policies

Establish clear policies

1. You should set account policies that define a secure password for your systems.
2. Establish clear policies and procedures for employee use of your organization's information technologies.

A.20 Reuse

Never reuse a password

1. Use a different password for every website
2. Don't use the same password twice
3. It's important to use unique passwords for each different online account.
4. Where possible, users must not use the same password for various access needs.
5. Avoid reusing old passwords.
6. Don't recycle passwords.

Alter and reuse passwords

1. Users must not use a basic sequence of characters that is then partially changed based on some predictable factor.
2. Users are prohibited from constructing fixed passwords by combining a set of characters that do not change, with a set of characters that predictably change.
3. Users must not construct passwords that are identical or substantially similar to passwords that they had previously employed.
4. *Incorporate the first few letters of the website name into your password so that every password is different.
5. *Add a couple of unique letters for each site.
6. *Select a single memorable base password and alter it to form derivations

Don't reuse certain types of passwords

1. Use a unique password for each of your important accounts.
2. Never use your Apple ID password with other online accounts.
3. Users must not use the same password for Company accounts as for other non-Company accounts.
4. Users should never reuse passwords between work and home.
5. Passwords used for Internet services should not be the same or similar to passwords used for services accessed within College.

A.21 Sharing

Don't share your password with anyone.

1. Don't share your Apple ID with other people, even family members.
2. Never give out your password to anyone.
3. Never disclose your passwords to anyone else.
4. Don't share your passwords.
5. Passwords must not be shared with anyone.
6. Do not share passwords with anyone,
7. Do NOT give any of your usernames, passwords, or other computer/ website access codes to anyone.
8. Never share your Maynooth University password with anyone.
9. Passwords must never be shared or revealed to anyone else besides the authorized user.

Don't send password by email

1. Do not send your password by email.
2. Don't send your password to anyone in an email.
3. Passwords must not be inserted into email messages, Alliance cases or other forms of electronic communication.

Don't share passwords over the phone.

1. Passwords must not be revealed over the phone to anyone.

A.22 Shoulder surfing

Offer to display password

1. Verifier SHOULD offer an option to display the secret (rather than a series of dots or asterisks, typically) as it is typed.

Conspicuously enter password.

1. Make sure no one sees you typing your password.
2. Don't enter your password when others can see what you are typing.

A.23 Storage

Encrypt passwords

1. Store Password Using Reversible Encryption For All Users enable the option on a per-user basis and then only as required to meet the user's actual needs.
2. Passwords must always be encrypted when held in storage for any significant period of time or when transmitted over communications system.
3. Passwords must always be encrypted (non-clear text) when held in storage for any period of time
4. Never store passwords as plain text.
5. The verifier SHALL use approved encryption.
6. Applications must not store passwords in clear text or in any easily reversible form.
7. Passwords must not be stored in readable form in batch files, automatic login scripts, software macros, terminal function keys, in computers without access control, or in other locations where unauthorised persons might discover them.

Restrict access to password files.

1. Restrict access to files that contain passwords.
2. Ensure you protect files containing encrypted or hashed passwords from unauthorized system or user access.

Encrypt password files

1. Encrypt files that contain passwords.

Store password hashes

1. Store one-way cryptographic hashes for passwords instead of storing the passwords themselves.
2. Produce hashed representations of passwords using a unique salt for each account.
3. Store passwords in a hashed format
4. Secrets SHALL be hashed with a salt value using an approved hash function.

Don't hardcode passwords into software

1. To allow passwords to be changed when needed, passwords should not be hard-coded (incorporated) into software.

Contractual agreements should stipulate how credentials are protected

1. When outsourcing, contractual agreements should stipulate how user credentials are protected.

A.24 Throttling

Throttle password guesses.

1. Lock out a user account after a number of consecutive failed authentication attempts.
2. Have a fixed or exponentially increasing delay after each failed authentication attempt.
3. Verifiers SHALL implement a throttling mechanism that effectively limits the number of failed authentication attempts an attacker can make on the subscriber's account.
4. Allow users around 10 login attempts before locking out accounts.
5. The number of consecutive attempts to enter an incorrect password must be strictly limited.
6. The number of consecutive attempts to enter an incorrect password must be strictly limited.

A.25 Transmitting passwords

Do not transmit clear text password

1. Applications must not transmit passwords in clear text over the network.

Request passwords over a protected channel

1. The verifier SHALL utilize an authenticated protected channel when requesting memorized secrets in order to provide resistance to eavesdropping and man-in-the-middle attacks.

A.26 Two factor Authentication

Use for remote accounts

1. Consider implementing two factor authentication for all remote accounts.

Use multi-factor authentication

1. Consider using multi-factor authentication.

Enable two factor authentication using phone

1. Enable two step verification with your phone.

A.27 Username

Enforce Composition Restrictions

1. Lower case only, no spaces, no special characters. Contain a minimum of 3 letters and 3 numbers. Length of 8 or 9 characters

Don't reuse usernames

1. Any username used for the Internet services should not be the same or similar to a College username.

APPENDIX B

Password Advice Costs

This appendix contains a description of the costs associated with each advice category and statement. The costs correspond to those indicated by users and administrators in our user study (Section 2.7) which are displayed in Table 2.5 in Chapter 2. For each category, we provide further impressions, characteristics and notable respondent comments.

B.1 User advice

We begin by discussing the advice associated with users. These are the top 12 categories in Table 2.5.

For some pieces of advice in this section, administrators were not asked about the costs to an organisation as these costs could largely be extrapolated based on related advice statements.

B.1.1 Backup password options

Email up-to-date and secure Most organization can not practically check that each user has kept their email up-to-date and secure. For a compliant user, this is a continuous process and will cost the user their time. In the survey users were divided about whether this has no cost or a minor periodic cost. We suspect that for most users their email client will automatically update the email software. However, we do mark it as a minor periodic cost. Administrators also marked this as advice as having no cost. This reinforces

the automated nature of email updates which will often be handled by the external email client.

Security answers difficult to guess Making security answers difficult to guess is a challenging task for an organisation. Administrators noted a large number of costs associated with the advice. Most notable is major user education periodically.

In reality, it is unlikely an organization can verify that security answers are difficult to guess. One option could be to employ guessing and require users to change their answers if they have been guessed. But in reality a practice such as this would have its own additional costs and privacy concerns associated with it. Respondent to our user survey indicated that security answers which are difficult to guess will likely be difficult to remember and will also take time for the user to create.

Do not store hints This statement represents two pieces of advice. One of the pieces of advice tells organizations to not allow users to store a hint, and one tells a user to not store a hint. Administrators in our survey were asked "Users should not set password hints on websites" and end-users were asked "Do not store hints about your password". The major cost for users was that their password would be less easy to remember and that it would require extra time or effort. For the organisation the major cost is periodic user education.

One administrator survey respondent noted in the comments that they did not understand the question and the interpretation they provided spoke of password reset practices. We therefore removed their response for this question.

B.1.2 Composition

In our user survey we asked about the composition advice "Include specific character types in your password. E.g. your password must include uppercase, lowercase, digit, symbol".

Both users and administrators indicated a number of minor costs associated with this advice.

B.1.3 Keep your account safe

The advice "Check web pages for TLS"and "Manually type URLs" take user time. Users interpreted the check webpages for TLS to be only at login. They marked it as advice they approved of. As with a lot of the advice circulates, this advice would be very difficult for an organisation to enforce. One administrator said "Not sure how to implement this, users often ignore instructions".

User identified no costs with "Don't open emails from strangers". We would have thought it could interfere with daily life or work but a 2012 study by Böhme and Moore found that as a result of concerns over cybercrime 42% of participants say they do not open email from strangers [11].

Both "A user's anti-virus software should be kept up to date " and "All users should keep software updated" require user *additional computing power* and *additional resources*for the users. Respondents to the user study also said that it "required extra time". Though in the case of anti-virus software, respondents disagreed about whether it was a minor or major cost. "Log out of public computers" has a small extra effort cost for users. The focus on administrators is user education.

"Password protect your phone" introduces an increased password memory burden on users. For administrators, there were varying views on whether there was a help desk/user support cost associated with this advice. It likely depends on how much oversight the organisation has. For example, given the user forgets their password, is it the organisation they go to or their personal phone provider?

B.1.4 Length

Minimum password length This advice has a major affect on the users' *risk of forgetting*. It also *inconveniences the users' password creation* and can mean that a user needs to choose a new password.

Enforce maximum length (<40) This advice *inconveniences the users' personal system for password generation* as a user may wish to choose a longer password. In fact one of the pieces of advice we collected, told the organization to limit the password length to less than 15 characters.

Administrators were not asked directly about the costs to the organisation associated with enforcing a maximum length. The assumption is that it will bear similar costs for an organisation as enforce a minimum length.

B.1.5 Password managers

Use a password manager A password manager helps a user by remembering their passwords and saves the user the time of typing the password at each site. Users indicated that using a password manager has a positive impact on remembering passwords. Some organizations will be able to force all their users to use a password manger but most organizations will not have this capability. This requires additional resources for the user as the user may need to purchase and/or download and maintain a password manager.

Respondents in the administrator survey indicated that encouraging users to use a password manager would result in a minor increase in help/desk user support time. They marked this as either a once off or periodic cost. We suspect that this relates to support user may need setting up their password manager and they may also need ongoing support.

Organisation requires extra resources if they are supplying the password management software for their employees/users.

Throughout the user studies we saw very positive responses in favour of password managers. One respondent said "Would almost consider it essential for modern Internet usage. Some additional setup and potentially cost, but absolutely worth it". However, others had comments such as "Haven't seriously considered using them before" and "I've never been recommended one or used one".

Create long random passwords when using a password manager One of the advantages of a password manager is that, because a user no longer needs to recall their passwords, the password can be as long and complex as a user wishes. If the password created is different to the users' general structure and is random, the user may never be able to remember it. One administrator mentioned that this "assumes that the user will always have access to their password manager. SaaS on off site work may limit this".[1]

[1]SaaS: Software as a service [142].

B.1.6 Personal Information

Don't include personal information This is a difficult thing for an organization to enforce. In fact, there is no reasonable way for an organization to eliminate all personal information from passwords. We did not ask administrators about this information as user education is the only method they have for enforcing the advice.

Users identified an *Increased risk of forgetting* as personal details could have made the password more memorable.

Must not match account details Users were asked about the inclusion of both personal information and account details in their passwords in the single statement: "Don't include personal information, account details or names in your password".

Because on organisation can restrict the inclusion of account information in passwords, administrators were also asked about this advice.

B.1.7 Personal Password Storage

Don't leave in plain sight If the users are internal to the organization and work areas are monitored then it could be possible for an organization to enforce this advice. However in many situations it will be impossible. Administrators saw user education as the only organisation cost for this piece of advice.

If the user follows the advice they have two options. They can memorize the password in which case there is a chance it is forgotten. Or they can store it in a hidden location which will require extra user effort or time to retrieve. Users indicated an *increased risk of forgetting* as the cost associated with this advice.

Don't store your passwords in a computer file Unusually, for this piece of advice, administrators were split between the *help desk/user support time* being either a major cost or non-applicable.

This that marked it as a major cost indicated that it would be periodic. We suspect they are envisioning an increase in help desk call as a result of users

forgetting their passwords over time. As a compromise between those who consider it a major cost and those who saw no cost, we mark it as a minor periodic cost and indicate variability.

Administrators were generally not in favour of users storing their passwords in a computer file. However, many people noted that provided the file was encrypted by the user then it was fine.

Write down safely We did not include this in the advice administrators were asked about as they can have little impact on the practice except for user education.

Three out of the seven respondents marked this as having a positive impact on "Makes it less likely to forget". Four said it was non-applicable. Perhaps because a user could still forge their password but it will save them the inconvenience of a reset when they do as they will simply need to locate the recorded password. We do mark it as a positive cost for forgettability since writing it down means the cost of forgetting won't exist.

Two users disagreed with this advice and 4 agreed. One person was neutral.

Don't choose "remember me" The user will now need to remember their password instead of it being saved in the browser. In addition, at each login the user will need to physically type their password.

B.1.8 Phrases

There exists a PAM module called 'cracklib' which automates blocklisting [109]. User education is required to explain the risks of choosing a common password and which types of passwords are likely to be common. One administrator made the point that blocklists that are not transparent lead to a lot of user support.

Contrary to the NIST 2017 advice, one administrator responded to this advice by saying "Force the password complexity, then this is not needed".

We did not ask administrators about the advice "Don't use patterns in a password", "Don't use published phrases as your password" and "Substitute symbols for the letters in your password". The first two would come under the

remit of a blocklist. The third seems impossible for an organisation to enforce and therefore would only require user education costs.

In the user study, respondents said that "Don't use published phrases" makes password creation more difficult, makes a password less easy to remember, and requires a minor amount of extra time or effort on top of this. When asked how frequently this extra time or effort was required, the prevailing answer was N/A. The next most popular was periodic so this is how we recorded it.

B.1.9 Reuse

Never reuse a password If users never reuse a password they must create a new password when they open an account.

Password reuse could be within an organization or across organizations. It is very hard to enforce a no reuse policy across organizations. Administrators marked this advice as having a major user education cost.

Alter and reuse passwords Respondents to our user study asked about the affect this advice has on ease of password creation were split between it being a positive and negative cost (See Figure B.1). Three users said it was a positive cost, i.e. the advice made it easier to create a password. We suspect this was in relation to needing to create a completely new password at every site which was the previous piece of advice that users were asked about. Antithetically, one user said it was a major cost when creating a password and three users said it was a minor cost. We choose to mark it as a minor cost as more people said major and minor combined than said positive.

There was similar discrepancy in determining whether the advice *made the password less easy to remember* was positive or negative. In this case the major and minor votes combined equaled positive. Most users typically use the exact same password for multiple websites. Therefore, we see the baseline as keeping all passwords the same, rather than having all random. For this reason, and to keep with the same narrative as above, we mark it as a minor cost.

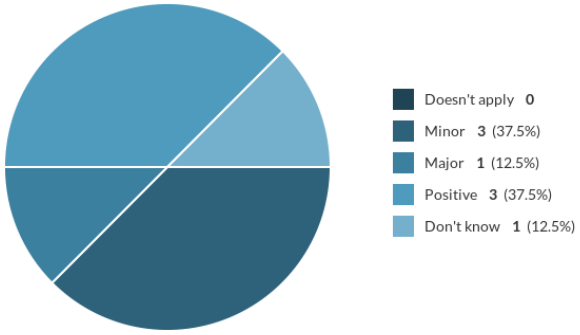


Figure B.1: Alter and reuse passwords. Results from user study for the cost category *Makes it more difficult to create a password*

B.1.10 Sharing

Sharing All user surveys include the advice "Never share your password" and 36 out of 40 users agrred with it.

User comments did vary though. One user said "This feel obvious and not something a website needs to point out". While other pointed out that exceptions apply and sometimes a system is not capable of dealing with it: "Sometimes it makes sense to share an account with close collaborators or family (if the underlying system doesn't otherwise support collaborative use). Sharing passwords may be a justifiable risk".

B.1.11 Two-factor authentication

Use multi-factor authentication All users were asked about implementing two factor authentication. The phrasing they were given was: "When you want to log in, you should have to enter an additional code from an app or a special device ".

A subset of these users were also asked specifically about two factor authentication using their phone: "When you want to log in, you should have to enter an additional code that is sent to you by text message (or phone call)".

In both cases users agreed that an extra resource was required at login. Using a specific app or device also incurred an additional time cost.

Administrators were asked about "Users should have to use some form of 2-factor authentication" and "Some form of 2-factor authentication should be available to users". The distinct difference being the option of choice for the

user. There was little difference in the costs to the organisation between the two pieces of advice. Though it is important to note that different groups of administrators answered each question with 6 answering the first version and a different seven answering the second. The major difference was that 5 of 6 (83%) agreed that some form of 2-factor authentication should be available to users, whereas only 4 of 7 (57%) agreed that users should have to use some form of 2 factor authentication. In Table 2.5 we show the costs that administrators related to users *having* to use 2-factor authentication.

One participant gave the strong opinion that 2-factor authentication "should be completely mandatory. Be it a push notification to their phone or some randomly generated OTP or both. Some form of MFA should be used".[2]

If a phone is not used then it is likely that either the user or the organization will have to provide the *additional resource needed* for authentication, for example a USB device.

Setting up 2-factor authentication using texts to phones will also require extra resources for the organisation as they will need the ability to initiate authentication texts or phone calls of app alerts.

Additional user time is needed to complete the authentication process since multiple factors are needed at each login. Some devices do offer a "remember me on this device option" which would ease the burden on the users. But this will have it's own security trade-offs.

If the user is using 'something you have' to authenticate and loses it, this will also have an impact for *user time and inconvenience*. Also, the user must remember to bring the device with them.

Colnago et al. found that [29] implementing two factor authentication (2FA) in their university increased the help desk calls by 10%. In particular, as the deadline for implementation approached, 2FA related tickets represented 24% of all support that was provided by their help desk.

Use for remote accounts The costs for this piece of advice are similar to those for "Use multi-factor authentication". The difference is that the costs only need to apply to remote accounts. In the user study, users were unsure

[2]OTP: One time password [108], MFA: Multi-factor authentication.

whether this advice *required an additional resource*. This could be because users maybe already have a phone on their person which will simply be used as the second factor. Three respondents said it did not apply, two said it was a minor cost and two said it was major. We therefore mark the severity as minor. Also, respondents were split between the extra user time being needed periodically or at login. Assuming the two factors are required at every login, we mark it as a login cost.

The exact advice statement given in the user study was phrased as: "You should use 2-factor authentication (e.g. an extra PIN) when logging into accounts remotely". Some respondents marked it as *requiring the creation of a new password*. Maybe it was unclear to users whether they had to create this additional PIN or not. Because we envision it as being a PIN sent to a device and created automatically, we do not include this cost.

B.1.12 Username

Enforce restrictions on characters The majority of users in our study assigned no cost to needing to include specific character types in their username. Some said it might require a minor amount of extra time. Interestingly, despite this, 6 respondents said they did not approve of the advice, none said they approved and two were neutral. One participant commented to say it was "Stupid and pointless" and another said "Can't see a good reason for this, but with a password manager it wouldn't be too painful".

B.2 Organization advice

In this section we discuss advice that was directed towards the organisation. It relates to back-end security such as storage of passwords and other security decisions that the organisation has control over. For some pieces of advice in this section end-users were not consulted about the costs as we deemed them not to be overly relevant to them in terms of costs.

B.2.1 Administrator Accounts

Not for everyday use The main cost here is to the administrator who must switch between accounts for different tasks. This burden is lessened by the implementation of programs such as su and sudo which allow users to

easily run programs which require extra security privileges. The organization must also have the resources to create two accounts for the one user.

Must have its own password This requires the organization to set up password protection on the administrator account. The administrator must create a new password and enter it in some way at each login. Multiple users with privileged access may all need to know the same administrator password. This is the disadvantage of using su protocols over sudo.

Should have extra protection In relation to not using admin accounts for everyday use, one respondent commented that "standard admin accounts are common vectors of attack". Another respondent mentioned using two factor authentication for admin accounts. All administrators agreed with additional security for administrator accounts. The only cost noted was a minor implementation cost. However, the advice does not specify what extra protection are recommended. If it is two factor authentication that is recommended, then this would come with its own costs for the administrator and organisation.

B.2.2 Backup work

We asked both users and administrators about the costs of needing to digitally and physically backing up work. From a user point of view it was their own work. From an organisation point of view it is the organisation's files and data. For both it requires time and resources. For the organisation there is also a need for user education and help desk support if a back-up policy is in place.

B.2.3 Default passwords

Change all default passwords This requires changing from default passwords but does not require that each password is unique and they often may be recorded, thus not requiring much additional burden on users. Users marked the only cost as a need to create a new password. Though admittedly, we would see figuring out how to change the default password as another major time cost. Often administrator accounts exist on devices and have associated default passwords that the user of the device may know nothing about.

B.2.4 Expiry

Store history to eliminate reuse The organization must store all previous passwords, requiring memory. The user will need to pick a new password as old passwords cannot be reused. There is also an *Increased risk of forgetting* as the user may forget which passwords have been expired and which is the current password in use.

Change your password regularly Regularly needing to change passwords have an impact on users' memory load as well as taking time out of their day. Users will repeatedly *need to create a new password*.

Only 50% of the 40 end-users surveyed disagreed with regular password expiry. 25% were neutral and 25% agreed with it. For administrators we have a much smaller number of responses. However, 3 administrator respondents disagreed with expiry, 1 was neutral and 1 agreed.

Change if suspect compromise An organization can internally monitor breaches or link with a breach application. For example receiving notifications about their users' credentials from `https://haveibeenpwned.com` [72]. The notifications to users will require organization time, and user education and support.

All users and administrators agreed with this advice.

B.2.5 Generated passwords

The assumption by users for most of the advice related to generated passwords is that they will create their own password afterwards. This is evidenced by the inclusion of a costs for the user of creating a new password for most of the related advice. One user said "Should also be replaced hence the added cost to create."

One issue an administrator flagged with generated passwords is that it is difficult to distribute them safely when off-site.

Must aid memory retention Interestingly, the eight respondents who answered this question were split according to Figure B.2. Notice that 3 said it was a major cost and 3 said it was a positive cost. This is interesting as it

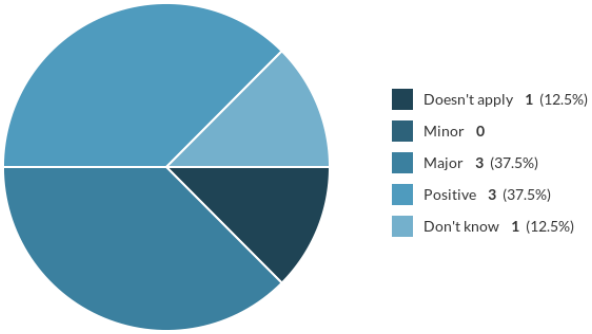


Figure B.2: Generated passwords must aid memory retention. Results for cost category *makes it less easy to remember*.

depends on the interpretation. In comparison to choosing their own password, any generated password will be more difficult to remember. But in terms of a generated password that is randomly generated, it is easier to remember. We mark it as a minor *inconvenience to remembering a password* as the status quo of creating their own password would be easier to remember.

Must be issued immediately Uses users time as the user must be available to receive the newly created password. Takes administrator time to distribute.

B.2.6 Distributed in a sealed envelope

Respondents to the user survey disagreed with whether this advice *required extra resources*. Two said does not apply, two said minor and two said major. We suspect that though the organization may need the extra resources of envelopes, the end-users shouldn't need any extra resources.

Only valid for first login Requires the user to generate their own password as well as administrators to generate the initial generated password. Most users said it had no costs but some said *requires the creation of a new password*, which we mark in.

B.2.7 Individual accounts

One account per user This will require organization time to set up. The cost can be high in an environment where there are shared computers. However, in our user study the majority of users assigned no cost to this piece of advice.

Each user account must be password protected User study respondents assigned very few costs to this advice. Three of eight answering the question said that it brought a major cost to *easy memorability* and one respondent said it was a minor cost. This cost was presumably assigned because it requires another pair of account details to be remembered. However, four of the eight respondents marked N/A for it's affect on memorability. Therefore we mark it as a minor variable cost.

Surprisingly, half respondents said that *need to create a new password* was a non-applicable cost. This could be because, even though a new password needs to be set, they don't necessarily need to create a new one as potentially an existing password could be used. However, we mark it as yes as it does require a new password to be set.

For the administrator survey, the only cost marked was for help desk/user support time. The majority of administrators said there was no cost for implementation. We expect this is because it is considered standard on most systems. In fact, one administrators commented to say "I haven't awarded severity/frequency costs because this should be mandatory and the cost is unimportant".

B.2.8 Input

Don't performed truncation Users often will not notice if their password has been truncated. One administrator gives a reason for truncation as "compatibility with legacy systems". It takes organisation time to be able to implement this advice and also requires extra resources.

Accept all ASCII characters The organization must create a system which has the ability to accept all ASCII characters in a consistent way. Accepting all characters should reduce the likelihood that the policy will inconvenience a user's password choice.

B.2.9 Keep accounts safe

Implement Defense in Depth Because we do not know what defense in depth strategies would be deployed, it is very difficult for administrators to assess what costs will incur. One administrator said "Its context specific based on the burden of costs associated with it".

Implement Technical Defenses Similar to above, this advice is not specific enough. It shows the importance of describing exactly what systems are intended to be put in place when giving security advice. Some respondents suggested technical defence that should be in place. For example, "..MFA should be mandatory, any company resources should have to be accessed over a VPN and the access policy on that VPN should have you running some sort of company approved anti-virus". Others said that it is "not specific or meaningful enough to be useful advice".

Apply access control systems This advice was worded for administrators as: access controls should be applied to access particular features or systems. One respondent commented to say that the advice statement was not clear. There was disparity in responses about the cost to organisation time for implementing these access controls. Two said that implementation costs were non-applicable, 1 said minor, 1 positive, and 2 major. We marked it as major as we believe implementing a fully access controlled system will take time for an organisation. However we mark it as having variability in the responses. All except those that marked it as non-applicable indicated that it is a periodic cost.

Access control can cause problems for end-users. One user survey respondent noted that "The problem is that someone has to decide which parts of the system are relevant for everyone else, and it's easy to err on the 'safe' side, which can create lots of friction."

Intrusions should be monitored and analysed One respondent mentions that in order to follow this advice it is "Assuming you know the attack vector". This could mean working with users of the system and also engaging in user education so that users can recognise compromises. Computing power is also important for identifying and monitoring and analysing breaches. It is a major organisation cost to implement.

Regularly apply security patches In order for security patches to be applied across an organisation, users must be compliant. Administrators assigned *user education* and *user support time* to this advice. Administrator survey respondents also mentioned the difficultly of this task. One said that

it “requires a lot of additional management to be done properly and audited.” Another respondent mentioned that some patches can break existing functionality and a third emphasised that it is a big but necessary job.

B.2.10 Network: SNMP community strings

An SNMP community strings can be compared to a username or password, knowing the string provides access to a device’s statistics.

As one respondent pointed out, community strings are only used in devices supporting the SNMPv1 and SNMPv2c protocols. The newer SNMPv3 uses an encryption key along with a username and password for authentication.

B.2.11 Password auditing

Attempt to crack passwords Both administrators and end-users were asked about this advice. There are a large number of costs for both users and administrators. One administrator commented to say “Don’t let it run forever or using spare cycles on your nearby supercomputer or you’ll start cracking relatively strong passwords too!”.

B.2.12 Establish clear policies

The exact wording of this question for administrators was: “Clear policies should be established (e.g what passwords will be accepted or security advice for use of IT systems)”. Users were not asked about this advice.

An administrator commented to say “Most questions of info sec come down to good user education, good policy and effective "policing" (and that does not necessarily mean punishment for non compliance)”.

B.2.13 Shoulder surfing

Offer to display password Administrators were asked “When logging in there should be an option to view a password after it is typed”. Users were asked “You should have an option to view your password after you have typed it”.

Users saw no costs for them regarding this advice. One user commented to say “There’s a trade-off to be made. Context matters. If you’re in a public space, it is riskier. If you’re at home or otherwise alone, it can be helpful. Suitable

for low security applications (meme generator account?). Not suitable for high security applications (bank?)."

Enter your password discretely Administrators were not asked about this advice as all they can so is provide user education. One user said "Absolutely! All password boxes should be starred or invisible to type".

When a user is logging in it may take some *extra time or inconvenience* for them to verify that they are entering their password discretely. Users also marked it as impacting the *ease of creating their password.*

B.2.14 Storage

Encrypt password files For the organization, at each system start up the password needs to be provided. This can be done manually, which would require periodic organization time. Or it could be automated in which case the password is accessible to the computer system, which would bear security risks, by reducing the effective secrecy provided by the encryption. One administrator noted that "It's common practice to ensure password files are encrypted.. To me this is a no brainer".

Access to password files should be restricted Administrators disagreed about whether *help desk support* and *user education* was necessary for this piece of advice. For both, half the administrators said non-applicable, while the other half were split between it being a minor and major cost. We mark it as a minor cost.

One administrator commented to say that "It is dependent on the type of organisation". We would be curious to know which organisations would prefer to not have access controls in place for their password files.

Hash and salt passwords In the authentication advice we collected in Section 2.3 we found four pieces of advice recommending storing passwords as hashes. However, of these, only two also recommended including a salt. Because a hashed password should always be salted, we asked administrators about hashing and salting together.

Administrator said it would require extra resources, organisation computing power and time to implement. We did not ask users about this advice as many would not be familiar with the terms. However, we do know that if the hash of the password is stored, then if the user forgets their password, the password cannot be recovered from the hash. Therefore, the user will need to create a new password.

Encrypt passwords One respondent said "Usually you would hash, but encryption can be used too". Another mentioned the "Minor cost to help desk, as passwords cannot be retrieved, only reset. Could be offset with self-service reset, at additional overhead cost". Another respondent said "Assume by encrypted you mean hashed".

Passwords should not be hardcoded One respondent did not understand the question. They commented to ask if this meant using certificates instead of passwords or MFA only. We removed this response as we believe the hard-coding is referring to the storage of the password on the organisation's system.

Another administrator gave the comment "Users should be allowed set their own password. Orgs should not keep lists of unencrypted passwords that have been hardcoded". This is more in line with our interpretation of the advice.

B.2.15 Throttling

Throttle password guesses Throttling involves limiting the number of wrong guesses that can be made against an account. The cost of this is that a legitimate user could accidentally be locked out if they mistype or forget their password a certain number of times. For example, Brostoff and Sasse [18] find that with a three strike system 31% of users are unfairly locked out is. With ten strikes it is 7%. Smart systems can help to minimize the risk for real users [95]. Users noted that this has a cost associated with time and effort and forgetting.

One user said "means you want to be extra sure of your password, either having it linked to the site/service or else stored somewhere that you can be sure of getting it correct". Another said "Depends on the value of what is being protected and there are better ways of providing extra security".

All administrators agreed with this advice (7/7) whereas users were less sure. One user said they disagreed and one was neutral. Four (4/7) users did agree with the advice.

One administrator emphasised the need to "let the user know that this can happen, to reduce confusion". Another administrator respondent mentioned that depending on the system in use, setting up throttling can range from a very simple 'tick box' task to a complete system overhaul.

B.2.16 Transmitting passwords

Different subsets of administrators were asked about "Don't transmit in cleartext" and "Request over a protected channel". Users were not asked about this advice.

The costs administrators noted for the two are very similar. Except for the transmission of passwords which involved *increased computing power* as well.

Both involve user education and help desk time. This is to ensure users or employees are not sending and asking for passwords using unencrypted means.

One administrator said "Passwords should not be transmitted in clear text - having a policy that states that is one thing (with little cost), educating the userbase to the risks of not adhering to the policy is more involved from a cost perspective and technically policing the policy can be quite expensive. So in effect you can have a range of answers here. I agree with going all the way to the technical policing (systems in place to discover clear text passwords)". Another said that mandating that passwords should be requested over protected channels is "Good in theory, in practise my experience is that users start to work around it".

B.2.17 Additional advice

Don't allow users to paste passwords Administrators and users disagreed with this advice. One administrators said: "Terrible idea. Increased help desk costs and user frustration. Makes it impossible to use password managers".

One user said "This is horrendous advice that leads to problems using password managers. It encourages using crappy passwords." But some 14/31

agreed with not allowing passwords to be pasted. One user said: “auto-fill is okay, paste is not”.

APPENDIX C

Password Advice Benefits

This appendix contains the rationale for how the security benefits were assigned to each advice statement in Table 2.5 in Chapter 2.

C.1 User advice

C.1.1 Back up password options

Email up-to-date and secure. Email is used for password reset links and often as the method that a generated password is passed to users. Therefore having a secure email account can help against eavesdropping of passwords by attackers. It can also prevent unauthorized binding of a new password to a users' account through the form of an emailed password reset. Having an up-to-date and secure email system with a working spam/malware filter can help to protect against phishing and pharming attacks and compromise of an endpoint due to malware.

Do not store hints If hints were stolen then these hints could be used to facilitate online or offline guessing or to aid a social engineering attack.

C.1.2 Composition

Enforce restrictions on characters Researchers have shown that having complex password composition rules can make the resulting passwords more difficult to guess [83]. Though simply allowing only long (greater than 16 characters) passwords has a similar effect on guessability and may not cause

as much hardship for users [83]. Having very stringent composition rules does have the effect of limiting the search space an attacker needs to look through. For example, a brute force exhaustive attacker has to search more possibilities for a password which can be made up of any type of 8 characters, than for an 8 characters password which has to have two numbers, two uppercase letters, two lower case letters and two symbols. But this only takes effect if the unconstrained user actually chooses from all 95 possible character options.

C.1.3 Keep your account safe

Check web pages for TLS This task helps users verify that communications to this webpage will be transmitted securely with encryption. However, it only has limited effectiveness as a strategy against phishing. In 2018, 49.4% of phishing sites were using SSL/TLS [38].

Manually type URLs Manually typing URLs can save a user from a Phishing attack as the user should recognize that the URL is not linking to the correct website. However, manually typing URLs makes a user vulnerable to typo-squatting/URL hijacking [157] e.g., `www.goggle.com`. A user is sent to the site with the similar URL which masquerades as the website of the user's intended destination. The site can then ask the user to enter their login details and store them to use on the real site. Thus the user's password is duplicated.

Keep software updated By keeping software updated a user gains protection against vulnerabilities as soon as the patch is released. This can save a user from eavesdropping, side channel and endpoint compromise.

Log out of public computers If a user does not log out of a public computer then the only protection a user has is the moral compass of the next person person who uses that computer. In this way a user's account can already be thought of as in a state of compromise. It is not obvious which of the 11 categories this opportunist attack comes under. We will somewhat arbitrarily place it under Eavesdropping.

Password protect your phone If a phone is password protected then the probability of endpoint compromise is lower.

C.1.4 Length

Minimum password length Inhibits brute force guessing as there are no passwords to guess with a very small number of characters, which are sometimes favoured by both users and guessers.

Enforce maximum length (<40) Requiring that passwords are less than a certain number of characters makes them easier for an eavesdropper to record them as they are less likely to cross packet boundaries. It also makes online and offline guessing easier as the attacker now need only guess passwords within the given range.

C.1.5 Password managers

Use a password manager The security benefits of a password manager will depend heavily on both how it is utilized by the user and also on the capabilities on of the specific software that users are using. For this reason all benefits are dependent on the implementation.

A password manager greatly reduce the users' memory load and by extension then a user can use as long, random and complex of a password as they wish. Thus this act will increase security. A password manager does mean that the user is relying on an external agent to store their passwords and therefore if this agent is compromised or if the users password for this single account is compromised then the passwords of all the users' accounts are compromised. Therefore we consider this to be a new way in which the users' passwords can be duplicated.

Password managers which automatically fill in the users' credentials with no user interaction do have some corner case vulnerabilities [150]. Though this same paper shows that a password manager can provide more security that the normal manual typing of the password. For example, password managers can be effective against phishing and pharming attacks.

Create long random passwords This piece of advice was given in the context of a password manager. "Configure your password manager to create 30–50 random characters with a mixture of upper- and lower-case letters, numbers, and symbols." It has the same benefits as creating a complex long password but without the user memory costs.

C.1.6 Personal password storage

Don't store in a computer file. An attacker accessing this password file can duplicate the password.

Write down safely Even if the password is stored safely, the very act of writing it down makes it's duplication and physical theft possible. There is discussion as to whether the security risks of writing passwords are in fact very low [93]. And in fact, if users write down passwords, then they may be more confident making stronger password choices [23][87][66].

Don't choose "remember me". If the "remember me" option is not used then if an attacker steals a laptop or computer they should not automatically have access to the accounts on it. It is equivalent to not logging out of an account.

C.1.7 Reuse

Never reuse a password Reusing passwords has the security disadvantage that if an attacker compromises a password on one site then the password can be used to gain access to other sites. This means if passwords are reused then online and offline guessing becomes much easier for an attacker. In fact, if the password is leaked elsewhere the chance of it being compromised for this given organization is just equal to be chance that an attacker tries.

However, even with different passwords at different sites, the attacker has a good chance of being able to leverage the information from a separate compromised site to mount effective phishing, social engineering and guessing attacks [34]. These will be at a higher cost to the attacker though.

Alter and reuse passwords Altering and reusing passwords means not directly reusing passwords between sites. It will make a guessing attack necessary for an attacker even if they have access to a password belonging to the same user from a different site. However, Das et al. [34] were able to guess approximately 10% of non-identical password pairs in less than 10 attempts and approximately 30% in less than 100 attempts. Therefore we mark it as a limited security improvement.

Don't reuse certain passwords. Asking users to not reuse certain passwords is equivalent to saying that a user can reuse some passwords.

In fact, if we look at the specific advice in this category we can see that most organizations are asking users to not reuse the password for *their* site. This does provide some security advantage as the attacker will not be able to directly access the protected account using the revealed password. But, as with "don't reuse you passwords" we know that an attacker can still leverage information from other compromised sites to attempt phishing, social engineering and guessing attacks.

C.1.8 Sharing

Never share your password In the process of sharing a password it could be eavesdropped. Not allowing users to share their passwords also helps to protect against social engineering. Though this is through the form of user education.

C.1.9 Two-factor authentication

Use Multi-factor authentication As mentioned before multi-factor authentication traditionally involves: *something you are*, *something you know*, and *something you have*. The *something you have* is susceptible to theft. However, if it is stolen the user if still protected by their other authentication factor. Using multi-factor authentication decreases the success of phishing (as second factors are often not subject to replay) and online guessing attacks (as both factors must be guessed). We underline some of the benefits as it depends on which factors the user or organization choose to use.

Use 2-step verification on phone The phone can be stolen or the code can be revealed by eavesdropping or a side channel attack. But again, if the phone is compromised, it is possible that the first step of the authentication process will keep the users' account secure. 2-step verification decreases the success of an online guessing attack and a phishing attack.

Use for remote accounts Without knowledge of a specific second factor it is hard to say what the security effects are. Therefore, depending on what the

second factor is, there is the potential for physical theft or endpoint compromise to jeopardize the authentication. The probability of the exchange being eavesdropped is much higher if used for remote accounts.

C.1.10 Username

Enforce composition restrictions on usernames Florêncio, Herley and Coskun argue that it is better to increase the strength of the userID rather than the passwords [50]. They propose that this will protect against online guessing attacks but will not majorly increase the cost to users since the username can be recorded visibly.

Don't reuse username If the same username is used for multiple accounts then once the password for one account is compromised, this password can be tried against the same person's other accounts. Das et al find that 3%, of users directly re-use passwords between sites and many others introduce small modifications to their passwords across sites [34]. Not reusing a username could be one way to protect against an attacker leveraging this vulnerability and could be less burdensome on the user than a restriction on altering and reusing passwords.

C.2 Organization advice

C.2.1 Administrator accounts

Not for everyday use It can be argued that the more times the authentication process is completed by the user, the more times it is susceptible to compromise during entry or transmission. We therefore say that not logging into the administrator account for everyday tasks decreases the chance of eavesdropping and side channel attacks.

Must have its own password If there is one administrator then ensuring that this administrator account has a distinct password means it is less vulnerable to eavesdropping and side channel attacks. However in most situations, many users will require privileged administrator access. In this case the problem depends on how you choose to implement this advice. If users access the administrator privileges by typing the administrator password via su then all privileged users must know the same password. This makes social

engineering, phishing and endpoint compromise more likely. In addition if multiple users are recording or sharing with others the same credentials then they are more likely to be duplicated and fall into the hands of an attacker.

Alternatively, there might be administrative access via the user's own password (e.g. via sudo) or a second administrative account/password corresponding to each user with administrative privileges.

All of these have associated security risks. In our table we have represented the case where there is one administrator who must create a second password which allows them access to administrator privileges.

Should have extra protection Depending on the extra protection the account is given this will have different benefits.

C.2.2 Backup work

Make digital & physical back-ups Having a back up of work means that attacks can be less harmful to the organization. Having backups does not directly decrease the chance of an attack but would be factored in relation to the costs of a breach. Having physical backups of work does mean that the potential for physical theft now exists.

C.2.3 Expiry

Store History to eliminate reuse This advice is given alongside "Change your password regularly". The password must now also not match any previous passwords. This means that knowledge of old passwords will not directly lead to an attacker knowing a current password.

However, even though users can no longer reuse prior passwords, alterations are still possible [149]. In fact, Zhang, Monrose and Reiter [185] identify that we can easily predict new passwords from old when password aging policies force updates.

In addition, if an attacker gains access to a users' account and changes their password, the user will be unable to change it again until the required number of days have elapsed, or with an administrator's help.

Finally, storing the history means there is an additional password file which needs to be protected. Because of the close relationship between old and new passwords [185], if this file is revealed then the information in it can be used to effectively guess the current password [34].

Change your password regularly In a certain situation changing your password regularly does decrease the probability of success of online guessing. Imagine an attacker cycling through a list of guesses. If a password is changed to something new during this guessing, then an attacker wishing to guess it must start their guessing process again from scratch.

However, most attackers will guess the most probable guesses first and since passwords follow a long tailed distribution [115, 98] a rational attacker will typically stop and move onto a new account if the password is not captured within the first few million guesses.

If the attacker correctly guesses the password within the time frame. Then the password will be changed at the beginning of the new period. This does bring some additional security but in reality once an attacker has access to the account they can set up a backdoor and will not need the password in future. Even if the attacker creates no backdoor the probability that they can guess the next period's password is high as users base their next password heavily on their previous password [185, 34]. Therefore, knowledge of the password from one period will strongly aid the attacker in guessing subsequent passwords.

Change if suspect compromise If the password has been leaked elsewhere then the advice is to change your password. This protects you from online or offline guessing attacks as otherwise an attacker with access to compromised password has immediate access to the account. Some of the caveats discussed above still apply. But the hope is that if a compromise is suspected, a user may be less likely to create a new password very similar to their old one. In addition the time scale to the creation of backdoor may be longer.

C.2.4 Generated passwords

Must be issued immediately This decreases the chance that generated passwords are stolen before they are told to the user. If passwords were created in advance they would likely be recorded as administrators could not

remember multiple generated passwords. Therefore these passwords could be duplicated while in storage.

Distribute in a sealed envelope This increases the chance that the password is physically stolen as the envelope could be taken. The password could also be duplicated since it has been recorded. If an adversary opens the envelope and duplicates the password then it will go undiscovered if the adversary places the password page in a new envelope and reseals it. The benefit of the sealed envelope is that an observational, audible or network eavesdropping attack is less likely.

Only valid for first login Because these generated passwords are often issued and created by administrators the user has no confidence in the security of their password up until the point they receive it. Maintaining a rule that passwords must be changed at first login means that the user can now have complete control over the security of this new password. This advice then protects against previous duplication of the password.

C.2.5 Individual accounts

One account per user The alternative is multiple users using the one account. With multiple users using the same account one user could modify the authentication information without informing other users (unauthorized binding). In addition, if multiple users are recording or sharing with others the same credentials, then they are more likely to be duplicated and fall into the hands of an attacker. Social engineering and phishing attacks and endpoint compromise are also more likely if there are multiple points of access.

Each account password protected If there is no password we can likely consider the account to already be in a state of compromise. Password protecting an account increases the security of the account by necessitating one of the attacks to take place before an attacker can gain access. It obviously protects against both online and offline guessing. In addition having a password makes a side channel attack more complex. An attacker should be able to differentiate the difference between an account login where no password is used and when a password is used.

C.2.6 Input

Don't performed truncation Truncating passwords makes online and offline guessing easier. It can also affect social engineering attacks. If the user does not know that the password will be truncated they may reveal the first few characters of the password without realizing the true security extent of this action.

Accept all characters This increases the necessary search space of an attacker attempting online or offline guessing. Allowing all characters could give more scope for a SQL injection attack, but the hope is that there would be adequate string escaping in place to mitigate this fear.

C.2.7 Keep accounts safe

Implement Defense in Depth Defense in depth can be divided into three categories: physical controls, technical controls and administrative controls. The security defense in depth can provide depends on exactly what strategies are deployed. They have the potential to mitigate any of the eleven attack types but without knowing what is implemented we cannot say exactly what the security advantages or disadvantages are.

Implement Technical Defenses The same argument as above can be used for this advice; it is not specific enough for us to know it's benefits. Though it is unlikely to aid against physical theft and social engineering.

Apply access control systems Access controls make sure that only certain users have access to their required aspects of the system. With respect to authentication, this means that only the privileged administrators have the power to view and control the authentication procedures and modify the stored authentication data. This protects against a malicious employee "turning off" authentication or other security mechanics, duplicating the stored password dataset or downloading malware to attempt side channel or keylogging attacks. However, exactly what this advice protects against depends on which specific access controls are put in place.

Monitor and analyze intrusions Awareness of what an attacker is doing within the system and learning where the vulnerability is is important for

security. However this advice has no direct security affect unless the analysis is acted on. For example, if an administrator witnesses an attacker duplicating the plaintext password file, then a forced password change might need to be implemented. Else if an administrator witness an attacker binding an additional form of authentication to a user or changing the credentials for a user, then these actions would need to be reversed. Monitoring and analyzing intrusions could also guide user education.

C.2.8 Policies

Establish clear policies This advice does not directly increasing or decrease the probability of success of an attack type.

C.2.9 Storage

Encrypt password files Encrypting password files will protect against the theft of the hard drive. However, the password used for encryption could still be read from the RAM. If the system can access the password without manual intervention then the password is likely to be stolen if the encrypted file is stolen. An attacker will have more difficulty downloading the password file for offline guessing.

Restrict access to password files Restricting access to password files will protect against certain types of unauthorized binding. If an attacker does not have access to the stored authentication details then the attacker will find it difficult to change the password stored for the user or link additional passwords or authenticators to the account. Preventing read access to a password file could prevent offline guessing attacks.

C.2.10 Throttling

Throttling (or rate limiting) password guesses drastically reduces the number of guesses an attacker can make. The attacker can no longer continuously make guesses until the correct password is accepted. However, because of the right-skewed nature of password distribution, the attacker does still have a high probability of success with a small number of guesses [98][115].

C.2.11 Additional advice

Don't allow users to paste passwords There appears to be no security benefits to this advice [7] and indeed in our model we cannot find any attack type that it mitigates.

APPENDIX D

NIST Calculations

In Chapter 5, we created a model that calculates the costs of enforcing authentication advice and the benefits the advice provides. The model computes the net value of an authentication policy by comparing the costs and benefits. To examine these models, we applied them to different versions of the National Institute of Standards and Technology (NIST) authentication standards documents: the archived NIST 2003 Electronic Authentication Guidelines and the superseding NIST 2017 Digital Identity Guidelines. In this appendix, we detail the methods and statistics used for assigning benefits and costs to these policies. For each of the NIST guideline documents, the policies are graded according to the strength of the security they seek to provide. The NIST 2017 document offers 3 policies starting at Level 1 and finishing at Level 3. NIST 2003, has 4 security levels. We will look at Level 1 and Level 3 of NIST 2017 and Level 1 and 4 of NIST 2003. We also look at an adapted version of the NIST 2003 Level 2 policy which we find in NIST 2007 documents.

D.1 Introduction

This appendix is included in order to describe, in detail, the approximation methods that were used to quantify both the costs and benefits of the five authentication policies compared in Chapter 5. It is not intended as an interesting read and is likely best viewed as a reference guide that can provide insight into the difficulties that materialized and how they were dealt with in the different contexts of the authentication policies.

As mentioned in Chapter 5, we repeatedly encountered problems with sourcing accurate attack breach data. Even when looking at the statistics we did find, we found that the statistics from different sources were often not comparable. We acknowledge that the statistics approximated in this work could be off by as much as a factor of ten and potentially by more. This does not make them worthless, but instead we recommend considering them conceptually as weightings. Each policy benefit has been computed using statistics from the same source and therefore while a specific statistic may be unreliable, that same statistic will be used across all policy quantifications. Therefore, the comparisons offered in Chapter 5 are substantiated.

Here we describe the layout of this appendix. In Section D.2, we provide details corresponding to an arbitrary set up of a fictional company. The quantification of the policies described in this appendix are all provided for this fictional company. A list of the statistics we require values for are listed in Section D.3 as well as an indication of where these statistics have been sourced from. Section D.4 begins to describe the quantification of the first NIST policy: 2017 Digital Identity Guidelines Level 1. This first policy is used as the example to explicitly describe and justify each step in the quantification of a policy.

We begin, in Section D.4.1, with a summary of the NIST 2017 Level 1 policy. Then, in Section D.4.2, we describe how the security benefits are calculated for this policy. This involves computing the probability of attack when the NIST 2017 Level 1 policy is in place and the probability of attack when there is no policy in place. In Section D.4.3, the costs of implementing the NIST 2017 Level 1 policy are calculated. Finally, in Section D.4.4, the security benefits minus the implementation costs are computed to give the overall Value of the policy for this fictional company.

In the following sections, this same quantification is provided for the NIST 2003 Level 1 policy (Section D.5), NIST 2017 Level 3 policy (Section D.6), NIST 2003 Level 4 policy (Section D.7), and the NIST 2007 policy (Section D.8) respectively. The sections describing these subsequent policies largely provide the simple numerical values, only offering further information when it differs from the NIST 2017 level 1 policy descriptions.

D.2 Fictional company

To apply the policy value model, we begin by creating a fictional company. Our fictional company has 500 users. We will look at this company over a 1 year time-frame. The users are important to them as they are their customers, U=1. The users are external to the organization so a compromised user would not immediately be seen as a threat to the overall system; the saturation point $\alpha = 0.5$. That is, 50% of users would need to be compromised before the reputation and actual damage combine to bring down the organization.

We use the equation we define in Table 5.3 to calculate L_1, the loss when a single user is compromised. We suggested that an organisation can estimate their expected damages as a result of a single breach as the average loss per breach for that organization, or an organization of that type:

$$damages = \frac{Annual\ cost\ of\ breaches}{\#breaches}$$

The 2018 Cost of a Data Breach Study [132] finds that the average cost per lost or stolen record is \$148. We will take this as our value for \$DAMAGES. L_{system}, the loss as a result of full system shutdown, can be set as equal to the value of the organization, in our example we will set this as \$10,000,000. The full details for this organization are given in Table D.1.

We would like to know which authentication policy best suits this organization's security and usability needs.

D.3 Statistics needed for policy analysis

Our model is designed to quantify the benefits and costs of an authentication policy. In order to do this we need to be able to quantify the security benefits that a policy provides and the expected costs that a user and organisation will experience. In this section, we will provide an overview of what statistics we needed and where we sourced them.

D.3.0.1 Quantifying attacks

When an organisation wishes to make informed decisions about their security choices, they have the advantage of access to their own company data and log files. An organisation may even have data about how often they experience

Table D.1: Details for fictional organization.

L_1:	\$166
L_{system}:	\$10,000,000
U	1
\$ADMIN WAGES	\$18/(60*60) =\$0.005/second
\$USER WAGES	\$18/(60*60) =\$0.005/second
\$ORG PROFIT PER USER	\$15
\$COMP POWER COST	\$0.01/second
TIME ADMIN RESET	900 seconds
TIME HASH SALT ENCRYPT	2 seconds
#users	500
#logins: Number of logins by all users in 1 year	261*#users = 130500
$\mathbb{P}$[abandon| forget]	10/100 = 0.1
$\mathbb{P}$[abandon| created authenticator rejected]	10/100 = 0.1
$\mathbb{P}$[abandon during creation]	20/100 = 0.2
$\mathbb{P}$[forget]	15/100 = 0.15
$\mathbb{P}$[forget| new password]:	20/100 = 0.2

attacks of different types and how often those attacks are successful. We however do not have access to such data. We therefore will be relying on information published in publicly available data breach reports.

For our computations, we need to know: the general probability of a breach occurring and the probability that the breach was from a particular attack type.[1] Below we will describe the sources we found for these statistics.

The UK 2018 Cyber Security Breaches Survey [167] found that 72% of UK large businesses (64% for medium, 47% for small and 40% for micro businesses) experienced cyber security breaches or attacks in the last 12 months.[2] We take the large business value and include the probability of a breach as 0.72 in all our calculations.

To find the probability of a breach of each type occurring we take values from

[1] We do not want to take into account breaches from any attack type because these will not necessarily relate to authentication and, therefore, it is not reasonable to expect on authentication policy to protect against them.

[2] Micro businesses (1 to 9 employees), small businesses (10 to 49 employees), medium businesses (50 to 249 employees) and large businesses (250 employees or more)

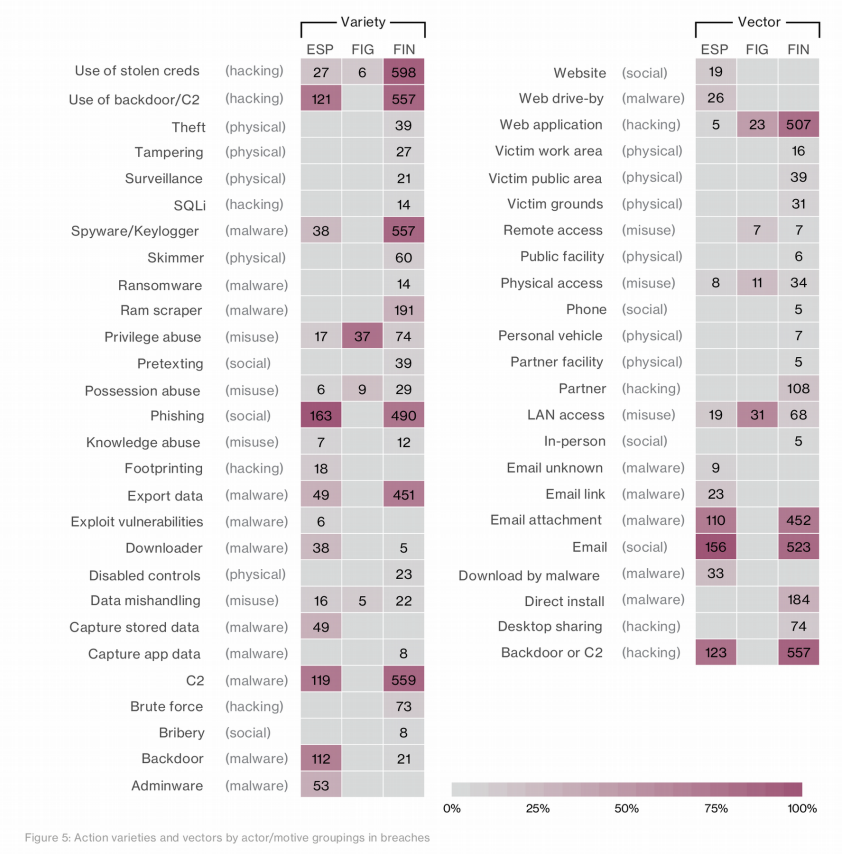


Figure D.1: 2017 Verizon's Data Breach Investigations Report (DBIR) table depicting the source actions for recorded breaches organised by vector of variety [169]

Figure 5 in the 2017 Verizon's Data Breach Investigations Report (DBIR) [169]. This figure (shown in Figure D.1) breaks breaches down into categories grouped by either vector or variety.[3] Depending on the statistic we are interested in, we will be looking at the proportion of breaches of a particular variety or vector.

To convert the statistics in Figure D.1) to a probability of attack, we can sum all the breaches that occur for the given vector or variety. This gives the number of breaches of each type which occurred in the given year to the organizations surveyed. Therefore the probability an organization experiences a breach of each type is: The probability a breach occurs $\times$ The probability of a breach due to this attack vector or motive, given a breach has occurred.

[3]The variety describes what type of attack it was, and vector describes how the attack was carried out. For example a particular attack could be of variety 'theft' and the vector is access to the 'victim work area'.

$$(0.72) * \frac{\#breaches\ of\ type\ a}{(total\ \#\ breaches)} \tag{D.1}$$

The total # of breaches will be 4788 if looking at an attack variety and 3231 if looking at an attack vector.

This is also the baseline value we use to estimate the probability of an attack occurring when there is no policy in place, i.e. no mitigation. This means that the expected Losses with no policy in place will be a significant underestimate of the actual expected losses. This is because the Verizon DBIR values are given for breaches, where a breach is defined as an incident that results in the confirmed disclosure of data. Whereas if there is no mitigation in place, every attempted attack of each type would very likely result in a successful breach.

D.3.0.2 Using the same data source for each statistic

The Verizon DBIR is not a perfect match for the values we require statistic data for. It does not include the same categories of attack that we have chosen to use from the NIST 2017 document. In addition, it is not directly reporting statistics from authentication breaches, but instead focuses on any breach to company data. It is, for this reason likely little concerned with the single compromise of one users' password.

However, we have chosen to use the Verizon DBIR data. The statistics in Figure D.1 were the most granular we could find from a single source. The values we take from this source can be conceptually thought of as weightings. While the specific figure may not reflect a true probability of attack. The weighting of the attack type in comparison to other attack types is the true importance of the statistics. Therefore, provided all the values come from the same data source, they should be appropriate relative to the other statistics.

D.3.0.3 Attack breach statistics needed

The NIST 2017 standards document [59] detailed 11 different types of attack that an authentication security policy should protect against. We chose to use these as the basis of our model (Chapter 2). We therefore need to be able to measure the affect each policy has on the success of each attack type.

Table D.2: Attacks success probabilities: statistics needed

P[Assertion Manufacture or Modification]
P[Physical theft]
P[Authenticator duplication]
P[Eavesdropping]
P[Password dataset leaked]
P[Offline guessing successful]
P[Side channel attack]
P[Phishing or Pharming]
P[Social engineering]
P[Online guessing]
P[Endpoint compromise]
P[Unauthorized binding]
P[targeted attack]

In Table D.2 we list the different probabilities used to compute the risk of attack. Note that for each statistic we will need to compute the probability of success when there is no policy in place and the probability of success when a particular NIST policy is in place.

D.3.0.4 Quantifying costs

For each piece of advice, we need to assign costs to it. These costs represent the affect enforcing this advice will have on users and the organisation. To determine which costs will occur, we can use the data gathered in our user study described in Chapter 2. However, in some cases there will also be some probability that the cost will not occur. For example, if we force users to create longer passwords, then only a certain proportion of the users will forget their password. That is, the cost of forgetting a password will be invoked with a certain probability. In Table D.3, we provide an indicative list of probabilities we were required to estimate in order to quantify the costs. In column 2 of this table, we provide examples of the sources we used to collect these statistics.

For each piece of advice in each policy, we also need to quantify the magnitude of each cost. For example, if resetting a password requires physically going to a help desk and providing identification then the time cost of forgetting a password will be much greater than if the password can be reset via email. In Table 5.3 in Chapter 5, we provided the equations that are used to compute these individual costs. We will not list the exact method for determining

Table D.3: Cost probabilities: statistic needed and source

P[Increased help desk/support time]	University IT Services data
P[User education required]	University IT Services data
P[Password more difficult to create]	[164, 85, 87]
P[Increased risk of forgetting password]	Komanduri et al. user study [85, 87]
P[User time and inconvenience]	Highly dependent on specific advice, [163]
P[Losing or misplacing a physical authenticator]	[159, 107, 131].

these magnitudes here as they are highly dependent on the specific advice. For examples and further details on this see Section D.4.3.

In the following section, we will describe the exact statistics and computations that were used to quantify the costs and benefits of the NIST 2017 Level 1 authentication policy.

D.4 NIST 2017 Level 1 Worked example

The NIST 2017 [59] policy allows a number of different authentication methods. We focus on subscriber chosen memorized secrets (typically referred to as 'passwords') which are the most common form of authentication [67].

D.4.1 Policy summary

General authentication rules

- Periodic re-authentication of subscriber sessions SHALL be preformed every 30 days.
- Access controls in place.
  - Specified by NIST 800-53 [120].
- The verifier SHALL limit consecutive failed authentication attempts on a single account to no more than 100.
- Communication between the claimant and verifier SHALL be via an authenticated protected channel.
- A CSP SHOULD bind at least two physical authenticators to the subscriber's credentials.

Authenticator 1: Memorized secrets

- Memorized secrets SHALL be at least 8 characters in length if chosen by the subscriber.
- Verifiers SHOULD permit subscriber-chosen memorized secrets at least 64 characters in length.
- No other complexity requirements for memorized secrets SHOULD be imposed.
- All printing ASCII characters as well as the space character SHOULD be acceptable in memorized secrets.
- Unicode characters SHOULD be accepted.
  - The verifier SHOULD apply the Normalization Process for Stabilized Strings using either the NFKC or NFKD normalization
- Truncation of the secret SHALL NOT be performed.
- Memorized secret verifiers SHALL NOT permit the subscriber to store a "hint" that is accessible to an unauthenticated claimant.
- Verifiers SHALL NOT prompt subscribers to use specific types of information (e.g., "What was the name of your first pet?") when choosing memorized secrets.
- When processing requests to establish and change memorized secrets, verifiers SHALL compare the prospective secrets against a list that contains values known to be commonly-used, expected, or compromised. For example, the list MAY include, but is not limited to:
  - Passwords obtained from previous breach corpuses.
  - Dictionary words.
  - Repetitive or sequential characters (e.g. 'aaaaaa', '1234abcd').
  - Context-specific words, such as the name of the service, the username, and derivatives thereof.
- Verifiers SHOULD offer guidance to the subscriber, such as a password-strength meter, to assist the user in choosing a strong memorized secret.
- Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily.

- The verifier SHOULD offer an option to display the secret - rather than a series of dots or asterisks - until it is entered.
- Verifiers SHALL store memorized secrets in a form that is resistant to offline attacks. Memorized secrets SHALL be salted and hashed using a suitable one-way key derivation function.
- The salt SHALL be at least 32 bits in length and be chosen arbitrarily so as to minimize salt value collisions among stored hashes. Both the salt value and the resulting hash SHALL be stored for each subscriber using a memorized secret authenticator.
- The iteration count for the PBKDF2 function SHOULD be as large as verification server performance will allow, typically at least 10,000 iterations.
- Verifiers SHOULD perform an additional iteration of a key derivation function using a salt value that is secret and known only to the verifier. The secret salt value[4] SHALL be stored separately from the hashed memorized secrets.

D.4.2 Benefits: NIST 2017 Level 1 policy

The expected benefit of the NIST 2017 Level 1 example, will be measured by the expected losses when no policy is in place minus the expected losses when the NIST 2017 Level 1 policy is in place.

In Section D.4.2.1 we will compute the probability of compromise, and the expected Loss due to compromise, when the NIST 2017 Level 1 policy is in place. In Section D.4.2.3 we complete the same computation for when there is no policy in place. Finally, in Section D.4.2.5 these two results are compared to give the overall security benefit offered by the NIST 2017 Level 1 policy.

D.4.2.1 Quantifying attacks: NIST 2017 Level 1

We will now discuss the impact of the NIST 2017 Level 1 advice on each of the 11 cost categories that we identified in Chapter 2.

Assertion Manufacture or Modification *The attacker generates a false assertion or the attacker modifies an existing assertion.*

[4]Often known as a "pepper".

From Figure D.1, in the 2017 Verizon's Data Breach Investigations Report (DBIR) [169] we find that that a breach due to the variety privilege abuse occurs 128 times. This value can be used as an estimate for an assertion being modified or manufactured. The probability of this occurring is mitigated by the access controls specified by NIST 800-53 [120]. To model this we account for the percentage of users who have access to the assertion mechanisms. We take this value as 10% of those who would have access if this advice was not in place.

$$\begin{aligned} \text{P[assertion manu or mod]} &= (0.1)\left(\frac{(128)(0.72)}{4788}\right) \\ &= 0.00192481203007365 \end{aligned} \tag{D.2}$$

Physical Theft *A physical authenticator is stolen by an Attacker.*

For the NIST 2017 Level 1 policy there is no physical authenticator that can be stolen. However a laptop or PC could be taken. Verizon DBIR reports 39 breaches due to theft. Reauthentication is required once every 30 days. If the user is only required to login every 30 days, then in a year time frame, a login will only be required approximately 12 times. Therefore, if an attacker steals a computer, there are only approximately 12 days in the year when the device will be locked. The probability the computer is unlocked when stolen is $1 - \frac{12}{365} = 0.96712387$.

If the disc is not encrypted, rebooting and rewriting the stored secret will compromise the authentication. NIST Level 1 does not specify encrypting the computer disc. This will require theft and tampering.

$$\begin{aligned} &\text{P[theft]} = \\ &\text{P[stolen]P[tampering]} + \text{P[stolen]P[unlocked]} \\ &= \left(\frac{(39)(0.72)}{4788}\right)\left(\frac{27}{4788}\right) + \left(\frac{(39)(0.72)}{4788}\right)(0.96712387) \\ &= 0.00570492567549227 \end{aligned} \tag{D.3}$$

Duplication *The subscriber's authenticator has been copied with or without their knowledge.*

Duplication of the memorized secret can occur when a subscriber willingly divulges the secret or when it has been copied without their knowledge after being recorded electronically or physically. Shay et al. [149] found that over a quarter of their respondents reported they had shared a password. Because we need the person the password was divulged to be the same as the attacker, we introduce the Verizon reported statistic that the vector in 108 of 3231 breaches is a partner. While this likely implies a business partner, we can generalize to someone who trust was placed in. The attack vector in 16 breaches was the victim work area. This is likely an upper bound, however it does account for copying a written password, copying a password from an electronic file and gaining access if the password is saved to the victim's browser. There is nothing in the 2017 NIST policy that mitigates the duplication of the password.

$$\begin{aligned} &\text{P[duplication]} = \text{P[divulge password]P[partner exploits]} \\ &+ \text{P[recorded password compromised]} \\ &= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) \\ &= 0.00958217270195177 \end{aligned} \tag{D.4}$$

Eavesdropping *The authenticator secret or authenticator output is revealed to the attacker as the subscriber is authenticating. Or an out-of-band secret is intercepted by the attacker by compromising the communication channel.*

Eavesdropping can take the form of shoulder surfing, keylogging, pass-the-hash attack, or the secret being intercepted by an attacker as it travels over a compromised communication channel.

Verizon DBIR 2017 gives the frequency of a breach resulting from physical surveillance as 21 in 4788. We take this for the frequency of a shoulder surfing compromise. This is also helped by the NIST 2017 policy allowing a toggle for displaying the password at entry.

A pass-the-hash attack should not be possible under the NIST hash-salt method since the hash is not being passed over the channel.

The NIST 2017 Level 1 guidelines require the use of approved encryption and an authenticated protected channel when requesting memorized secrets. This means the channel is secure from eavesdropping unless the encryption is broken. As with all the pieces of advice, we assume that the organization follows it to the fullest degree. Therefore the probability that an attacker compromises the encryption is negligibly small.

In reality, the likelihood that an attacker can break the encryption to access the data transmitted is dependent on the type of encryption used. In 2017, the paper [89] showed that 28 administrators struggled to set up HTTPS. At the end of a two hour lab study they found that 18.5% of their participants failed to set up a secure HTTPS connection. However, services such as Let's Encrypt have made this set-up easier [92].

We were unable to find a statistic for how often eavesdropping attacks take place. This is likely a reflection of the difficulty/impossibility of detecting an eavesdropper.

The NIST 2017 Level 1 policy does not mitigate a keylogger attack and we can take these values as the frequency reported in DBIR [169].

$$\begin{aligned} &\text{P[eavesdropping]} = \\ &\text{P[physical surveillance]} + \text{P[keylogger breach]} \\ &= \left(\frac{(21)(0.72)}{4788}\right) + \left(\frac{(595)(0.72)}{4788}\right) \\ &= 0.0926315789473664 \end{aligned} \tag{D.5}$$

Offline Guessing attacks *The authenticator is exposed using analytical methods outside the authentication mechanism. E.g. A trial and error guessing attack against an offline dataset of passwords.*

For offline guessing to take place a dataset or subset of encrypted passwords must be leaked. In the case of the NIST 2017 Level 1 policy, the passwords will be individually hashed and salted before being stored and then an additional

iteration of a key derivation function with a separately stored salt (often called a "pepper") will take place against the whole dataset.

This means offline guessing for this policy can only occur if both the password database and the separately stored salt (or pepper) is leaked [99].

$$\begin{aligned} &\text{P[user password cracked offline]} \\ &= \text{P[passwords leaked} \cup \text{pepper leaked]P[user password cracked]} \end{aligned} \tag{D.6}$$

We saw already, in Section 5.4.1.1, that our benefits model has been adjusted to take into account the dependence of offline guessing on the dataset already being leaked before cracking can occur. But now we include the requirement for the pepper to also be leaked before cracking can occur.

It is worth noting that there are situations when brute force guessing will still be feasible even if the pepper is not leaked [40]. However, in these scenarios it is likely that other aspects of the NIST 2017 Level 1 requirements have not been adhered to. For example, salting individual passwords and not storing hints.

$$\begin{aligned} &\mathbb{E}[\text{Benefits}] = \\ &\mathbb{E}[\text{Benefits}|(\text{no password leak} \cup \text{no salt leak})]\mathbb{P}[\text{no leak} \cup \text{no salt}] \\ &+ \mathbb{E}[\text{Benefits}|(\text{password leak} \cap \text{salt leak})]\mathbb{P}[(\text{password leak} \cap \text{salt leak})] \end{aligned} \tag{D.7}$$

The 2017 Verizon's Data Breach Investigations Report [169] states that, from 4788 breaches, 49 breaches resulted in the capture of stored data.

We need two separately stored data files to be leaked. However we only need a breach to occur once:

$$\begin{aligned} &\text{P[database of passwords is leaked} \cap \text{salt leaked]} \\ &= \left(\frac{(49)(0.72)}{4788}\right)\left(\frac{49}{4788}\right) \\ &= 0.0000754078177923466 \end{aligned} \tag{D.8}$$

We can see that the Level 1 NIST 2017 policy makes offline guessing very difficult for an attacker. Given that both the password dataset and the salt have been leaked, we look for the probability a password in that database is cracked.

Because the passwords are also individually salted, some of the attackers' computing power is consumed by combining each guess with all the possible salts. Therefore instead of the usual 10^{13} guesses we would allow an attacker, this time the attacker will make $10^{13}/\#unique\ salts$. NIST 2017 requires that the salts are randomly generated for each user. Therefore $\#unique\ salts = \#users$.

There have been many suggestions by researchers for ways to model password guessing; Entropy [20] which has now been disproved [83], Zipf law [98] [171] [143] and Loss analysis in comparison to optimal rates [116].

However, the best way to determine how susceptible your password set is to cracking is to attempt password cracking or an equivalent such as Kelley at al.'s guess-number calculator [83]. Kelley et al.'s paper "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms", gives the result for the number of guesses required to crack passwords created under a variety of password composition policies. For example, for their BlacklistHard policy they required their study participants to create a password which contained "at least 8 characters," did "not contain a dictionary word" and did not exist in "a five-billion-word dictionary created using the algorithm outlined by Weir et al. [178]". One of their guessing experiments showed that they could guess 23% of the passwords in this dataset in 10^{13} guesses. Only 5.5% were guessed in 10^{10} guesses. We use this value as the probability an offline guessing attack successfully guesses a password since the NIST 2017 level 1 policy requires passwords to be created using similar

constraints.

$$\begin{aligned}&\text{P[user password is guessed in } (10^{13}/\#\textit{unique salts}) \text{ guesses]} \\ &= \text{P[user password is guessed in } \ 10^{10} \text{ guesses]} \\ &= 0.055\end{aligned} \tag{D.9}$$

Side Channel Attack *The authenticator secret is exposed using physical characteristics of the authenticator.*

While side channel attacks are generally designed against cryptographic keys, there is at least one attack which will work against password [158]. The attack involves changing one character at a time to one that requires a longer processing time. Thus when the password is typed by the user the attacker can identify whether that character occurred in the password. There is some probability that the user will detect the changes occurring as their password will be rejected if the changed key does exist in their password. We set the probability that the attack goes ahead undetected at 0.8. The attacker will need to force the user to download malware onto their machine.

We presume that this attack would be targeted at an individual. Hackmageddon [128] finds that 15.2% of attacks are targeted. Of those 22.3% target individuals. Giving our overall probability of a user being targeted as: 3.3896%.

$$\begin{aligned}&\text{P[side channel attack] =} \\ &\text{P[targeted attack].P[not detected].P[malware]} \\ &= (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451251986\end{aligned} \tag{D.10}$$

Phishing or Pharming *The authenticator output is captured by fooling the subscriber into thinking the attacker is a verifier or relying party.*

The NIST 2017 level 1 policy offers no mitigation for phishing or pharming, and therefore we consult the Verizon DBIR documents for the frequency breaches from such attacks occurring. We use the statistic provided for phishing as there is none available for pharming.

$$\begin{aligned} &\text{P[phishing} \cup \text{pharming]} = \\ &\text{P[phishing]} = \frac{(653)(0.72)}{4788} = 0.0981954887218051 \end{aligned} \tag{D.11}$$

Social Engineering *The attacker establishes a level of trust with a subscriber in order to convince the subscriber to reveal their authenticator secret or authenticator output.*

The mitigation suggested by the NIST 2017 guidelines is to avoid using authenticators that present a risk of social engineering of third parties. Memorized secrets do not fall into this category and we have no mitigation at level 1 against social engineering. The Verizon DBIR reports that 39 breaches are of the variety social pretexting. That is, a person presents a false motive for requiring information.

$$\text{P[social engineering]} = \text{P[pretexting]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429 \tag{D.12}$$

Online Guessing attacks *The attacker connects to the verifier online and attempts to guess a valid authenticator output in the context of that verifier. E.g. An attacker performs repeated login trials by guessing possible values for the password and username.*

For online guessing, the attacker does not need to have access to the password dataset. However, a knowledge of both the username and password is necessary.

Therefore for an online guessing attack an attacker can either target individuals whose username they know or can easily find out, and then guess their password. Or alternatively, they can attempt to guess both the username and the password of any person.

In both cases online guessing is hindered by rate limiting or throttling. An organization can increase the time allowed between guessing attempts or stop all attempts after a threshold number of wrong guesses. The NIST 2017 guidelines allows at most 100 consecutive incorrect attempts. The counter

will reset to zero every time there is a successful login. We assume a user will login once per working day so the counter will reset when this happens. This means in 1 year an attacker can try 99*261 guesses against a user's account. Kelley et al. show the percentage of passwords guessed in $99 * 261 = 25839$ guesses is approximately 1% [83].

Because the limit on wrong guesses is attached to a subscriber's account, an attacker can try many wrong usernames and they will not be tallied. Therefore we calculate the probability of guessing a username in 10^{13} guesses. In reality we hope that an organizations' administrator could prohibit such a large number of username guesses by blocking the IP address of the attacker, but this mechanism is not stated in the NIST 2017 policy. Because there are no blacklisting or composition requirements on the usernames for the NIST 2017 level 1 policy, we use Kelley et al.'s [83] guessing return for their Basic8survey password. A study of the strength of usernames could be an interesting future research project. 63% of these passwords could be guessed in 10^{13} guesses.

We would like to know the probability that each of these two types of online guessing attacks take place. place. To estimate the probability of an attacker targeting a user whose username they know we use the Hackmageddon statistics from 2017 [128]. These tell us that an individual has a probability of 0.033896 of having a targeted attack against them. Verizon DBIR tells us that breaches due to brute force or use of stolen credentials occur 73 and 631 times respectively.

$$
\begin{aligned}
&\text{P[online guessing successful]} = \\
&\text{P[targeted attack].P[password in (99*261) guesses]} \\
&+ \text{P[brute force or using stolen creds]P[(username guessed in } 10^{13} \text{ guesses]} \\
&* \text{P[password guessed in (99*261)]} \\
&= (0.033896)\,(0.01)(0.72) + \left(\frac{(73+631)(0.72)}{4788}\right)(0.63)(0.01) \\
&= 0.000910998568420505
\end{aligned}
\tag{D.13}
$$

Endpoint compromise *Malicious code on the endpoint proxies remote access to a connected authenticator without the subscriber's consent or causes*

authentication to other than the intended verifier or compromises a multi-factor software cryptographic authenticator.

The NIST Level 1 guidelines do not include mitigations for endpoint compromise. Therefore, finding a frequency in the Verizon report we use the statistic for the number of breaches which leverage a backdoor or C2 servers. C2, command and control servers are used by attackers to maintain communications with compromised systems within a target network.

$$\begin{aligned} \text{P[endpoint compromise]} &= \text{P[breach using backdoor or C2]} \\ &= \frac{(678)(0.72)}{4788} = 0.101954887218044 \end{aligned} \tag{D.14}$$

Unauthorized binding *An attacker is able to cause an authenticator under their control to be bound to a subscriber's account.*

The unauthorized binding of the authenticator to an attacker could take the form of an unauthorized password reset. In order for an attacker to successfully reset a subscriber's password they would need to already have access to their email account and be interested in targeting that individual since it is a time consuming attack. Alternatively they could reset a password by having physical access to the unencrypted computer disc and rebooting and resetting the password. We assume this would also have to be a targeted attack.

$$\begin{aligned} &\text{P[unauthorized binding]} = \Big(\text{P[use stolen credentials]} \cup \text{P[physical theft]}\Big) \\ &* \text{P[targeted attack]} \\ &= \left(\frac{(631)(0.72)}{4788} + \frac{(39)(0.72)}{4788}\right)(0.033896) = 0.00341508571428736 \end{aligned} \tag{D.15}$$

D.4.2.2 Calculation of NIST 2017 Level 1 Losses

Recall the following equation for computing the expected loss from password attacks:

$$\mathbb{E}[\text{Loss}] = \left(\mathbb{P}\left[\mathcal{N}\left(p_l, \frac{p_l(1-p_l)}{N}\right) > \alpha\right] L_{\text{system}} + N p_l L_1\right) \mathbb{P}[\text{leak}] \tag{D.16}$$
$$+ \left(\mathbb{P}\left[\mathcal{N}\left(p_{l'}, \frac{p_{l'}(1-p_{l'})}{N}\right) > \alpha\right] L_{\text{system}} + N p_{l'} L_1\right) \mathbb{P}[\text{leak}']$$

where $p_l = \mathbb{P}[\text{compromise}|\text{leak}]$, $p_{l'} = \mathbb{P}[\text{compromise}|\text{no leak}]$ and $\mathbb{P}[\text{leak}'] = \mathbb{P}[\text{no leak}]$.

So we calculate the probability of compromise separately for when a leak has occurred and when a leak has not occurred. P[offline guess] will equal zero for $p_{l'}$ but will equal 0.055 in p_l (Equation D.9). The probability of compromise is given in terms of each attack type:

$$\begin{aligned} p &= 1 - \prod_a (1 - P_a) \\ &= 1 - (1 - \text{P[assertion manu]})(1 - \text{P[theft]})(1 - \text{P[dup]}) \\ &\quad (1 - \text{P[eavesdrop]})(1 - \text{P[offline guess]})(1 - \text{P[side channel]}) \\ &\quad (1 - \text{P[phishing]})(1 - \text{P[social eng]})(1 - \text{P[online guess]}) \\ &\quad (1 - \text{P[endpoint]})(1 - \text{P[unauth bind]}) \end{aligned} \tag{D.17}$$

For the NIST 2017 Level 1 policy this computation with and without a leak occurring is:

$$p_l = \mathbb{P}[\text{compromise}|\text{leak}] = 0.324538796275133 \tag{D.18}$$
$$p_{l'} = \mathbb{P}[\text{compromise}|\text{no leak}] = 0.285226239444585 \tag{D.19}$$

Now, taking the values defined in Section D.1: $Lsystem = \$10^7$, $L_1 = \$166$, and $N = 500$ users. We calculate the expected loss due to compromise for the organization when the NIST 2017 Level 1 policy is in place.

$$\begin{aligned}
&\mathbb{E}[\text{Loss}|\text{NIST 2017 L1 policy}] = \\
&\Big(\mathbb{P}\left[\mathcal{N}\left(0.324538796275133, \frac{(0.324538796275133)(1-0.324538796275133)}{500}\right) > 0.5\right] \times \$10^7 \\
&+ (500)(0.324538796275133)(\$166)\Big) \times 0.0000754078177923466 \\
&+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.285226239444585, \frac{(0.285226239444585)(1-0.285226239444585)}{500}\right) > 0.5\right] \times \$10^7 \\
&+ (500)(0.285226239444585)(\$166)\Big) \times (1 - 0.0000754078177923466) \\
&= 2.10466041986588 + 24527.6078713633 = 24529.7125317831
\end{aligned} \tag{D.20}$$

D.4.2.3 Quantifying attacks: No policy

In order to determine the 'benefit' of a password policy, we compare the losses with the policy in place to losses that would occur without the policy.

Thus for each type of attack the authentication policy mitigates, we require a statistic indicating how probable this attack would have been without the policy. We will be using the DBIR [169] statistics as the baseline values for each attack type. Below are the baseline values for each attack type. Explanations are offered where they were not covered in Section D.4.2.1.

Assertion Manufacture or Modification

$$\text{P[assertion manufacture or modification]} = \left(\frac{(128)(0.72)}{4788}\right) = 0.0192481203007521 \tag{D.21}$$

Physical Theft

$$\text{P[theft]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429 \tag{D.22}$$

Duplication

$$\begin{aligned}
&\text{P[duplication]} = \text{P[divulge.password]} * \text{P[partner.exploits]} \\
&+ \text{P[recorded.password.compromised]} \\
&= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) = 0.00958217270195177
\end{aligned} \tag{D.23}$$

Eavesdropping Eavesdropping can take the form of shoulder surfing, key-logging, skimming, pass-the-hash attack, or the secret being intercepted by an attacker as it travels over a compromised communication channel.

$$\begin{aligned} &\text{P[eavesdropping]} = \text{P[network eavesdropping breach]} \\ &+ \text{P[keylogger breach]} + \text{P[physical surveillance breach]} \\ &= \left(\frac{(118)(0.72)}{3231}\right) + \left(\frac{(595)(0.72)}{4788}\right) + \left(\frac{(21)(0.72)}{4788}\right) = 0.1189268435713 23 \end{aligned} \tag{D.24}$$

Offline Guessing attacks

$$\text{P[database of passwords is leaked]} = \left(\frac{(49)(0.72)}{4788}\right) = 0.00736842105263307 \tag{D.25}$$

With no policy in place the passwords will be stored in plaintext. Therefore once the password set has been leaked, the passwords will be visible to all.

$$\text{P[user password is guessed in } 10^{13} \text{ guesses]} = 1 \tag{D.26}$$

Side Channel Attack

$$\text{P[side channel attack]} = \text{P[targeted attack].P[not detected].P[malware]} \tag{D.27}$$

$$= (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451253493 \tag{D.28}$$

Phishing or Pharming

$$\text{P[phishing} \cap \text{pharming]} = \text{P[phishing]} = \frac{(653)(0.72)}{4788} = 0.0981954887218045 \tag{D.29}$$

Social Engineering

$$\text{P[social engineering]} = \text{P[pretexting]} = \frac{(39)(0.72)}{4788} = 0.00586466165413534 \tag{D.30}$$

Online Guessing attacks It is reasonable to assume that any password with less than 8 characters can easily be guessed in an exhaustive search. In the Yahoo.com password set [15] (had no password composition restrictions) 22.428% of the passwords contained less than 8 characters. After 8 characters we know that 63% of the passwords can be guessed in 10^{13} [83]. This indicates that when no policy is in place, an attacker should be able to guess at least 71.29836% of the users' passwords in a year. Given that there are no restrictions on the username we use this same value to estimate the probability it is guessed by an attacker.

$$\begin{aligned}
&\text{P[online guessing successful]} = \\
&\text{P[targeted attack].P[password guessed in } 10^{13} \text{ guesses]} \\
&+ \text{P[brute force or using stolen creds].P[username guessed in } 10^{13} \text{ guesses]} \\
&* \text{P[password guessed in } 10^{13}] \\
&= (0.033896)(0.7129836)(0.72) + \left(\frac{(73+631)(0.72)}{4788}\right)(0.7129836)(0.7129836) \\
&= 0.0712162867316309
\end{aligned} \tag{D.31}$$

Endpoint compromise

$$\text{P[endpoint compromise]} = \frac{(678)(0.72)}{4788} = 0.101954887218044 \tag{D.32}$$

Unauthorized binding

$$\begin{aligned}
&\text{P[unauthorized binding]} = \\
&\Big(\text{P[use stolen credentials]} \cup \text{P[physical theft]}\Big)\text{P[targeted attack]} \\
&= \left(\frac{(631)(0.72)}{4788} + \frac{(39)(0.72)}{4788}\right)(0.033896) = 0.00341508571428736
\end{aligned} \tag{D.33}$$

D.4.2.4 Calculation of Losses with No policy

$$p_l = \mathbb{P}[\text{compromise}|\text{leak}] = 1 \tag{D.34}$$

$$p_{l'} = \mathbb{P}[\text{compromise}|\text{no leak}] = 0.366081483597361 \tag{D.35}$$

$$\mathbb{E}[\text{Loss}|\text{No policy}] = \tag{D.36}$$
$$\Big(\mathbb{P}\left[\mathcal{N}\left(1, \frac{(1)(1-1)}{500}\right) > 0.5\right] \times \$10^7 + (500)(1)(\$166)\Big) \times 0.00736842105263307$$
$$+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.366081483597361, \frac{(0.366081483597361)(1-0.366081483597361)}{500}\right) > 0.5\right]$$
$$\times \$10^7 + (500)(0.366081483597361)(\$166)\Big) \times (1 - 0.366081483597361)$$
$$= \$74317.89 + \$31254.01 = \$105571.91$$

D.4.2.5 Quantifying benefit of the NIST 2017 Level 1 policy

The expected benefit of the NIST 2017 Level 1 policy is measured by the expected losses when no policy is in place minus the expected losses when the NIST 2017 Level 1 policy is in place.

$$\mathbb{E}[\text{Benefits}] = \mathbb{E}[\text{Loss without policy}] - \mathbb{E}[\text{Loss with policy}] \tag{D.37}$$
$$= \$105571.90 - \$24529.71 \tag{D.38}$$
$$= \$81042.19 \tag{D.39}$$

This tells us that the overall benefit for this organization of implementing the NIST 2017 Level 1 policy is \$81042.19.

D.4.3 Costs: NIST 2017 Level 1 policy

Table 5.3 defines the twelve categories of costs. To quantify each equation we must input values which will depend on the individual organization and the individual pieces of advice. As mentioned, the lowercase items in these equations are variables dependent on the advice and the capitals are constants known or set by the organization. We identify each cost, probability and number of repetitions with $\{i, j\}$ where i denotes the cost category $1 \ldots 12$ and j denotes the piece of advice the cost belongs to.

In this section we will quantify the cost associated with each piece of advice in the NIST 2017 Level 1 policy. We use input from our user administrator surveys described in Section 2.6 as well as our own insights and analysis.

In order to keep the survey concise, we allowed respondents to *optionally* include the rationale behind their assigned costs. This means that in some cases we do not have a clear quantifiable idea of what the costs are. We therefore include an arbitrary scaling for the assigned costs where we do not know the exact values. These are described below. In all other cases we are able to give more accurate quantifications.

Minor user education costs that are periodic we assign as 1 hour of administrative time every 6 months and 15 minutes of every users' time every 6 months. For Major user education costs this becomes 1 hour every month for the organisation and 15 minutes every month for each users. Minor organisation time to implement is assigned as 1 day, Major is three weeks (120 hours), except in cases when we know it to be longer or shorter. Finally, minor help desk time is assigned as 15 minutes per user per year while major help desk time is assigned as 15 minutes per user every 3 months. These values were formulated using input from system administrators, developers and log data from our university's implementation of multi-factor authentication as well as from feedback we received from the administrator surveys (Section 2). These values can be inputted into the quantified model directly according to Table 2.5. Where more insightful values can be provided we have detailed these below.

j=1: Composition Enforcing these composition restrictions will take minor $C_{4,1}$*: Organizations' time to implement.* According to the above rules we assign the task 1 day. This time may involve the time to approve the use of the system and establish an effective blocklist tailored to the organisation. This cost will occur with probability 1 and will be repeated once in time frame T, $R_{4,1} = 1$.

Administrators in our study agreed that this advice does not require additional computing power for organisations. However they did say it would require user education. On the Organisation side of the table we assign 1 hour every 6 months for the organisation to provide this user education and on the User side in $C_{12,1}$ we assign 15 minutes every 6 months for the users to receive this information.

We would like to know how the probability of *forgetting* is affected by different

composition policies. We could find no literature identifying the distribution of the probability of forgetting. We would like to know whether the probability is correlated with the guessability of the password. Some researchers have described forgettability in terms of the entropy of a password [148]. Komanduri et al. record the number of participants in each of their password composition groups who forgot their passwords [87]. However they found no significant difference in the rates across different conditions. They did find that participants in the stricter composition groups wrote down their passwords more frequently than those in the other groups. We believe the correlation between forgettability and strength would be an interesting study for researchers and would be very beneficial for further understanding user behavior.

For the purpose of our example we will use statistics sent to us by Saranga Komanduri [85, 86]. The probability that a user forgets a password that was chosen according to the rules of the BlacklistHard Policy is $P_{7,1} = 0.02044025$.

We are interested in the probability that the advice *makes it more difficult for a user to create a password*. Ur et al. found that of their 49 study participants, 43 (88%) said they had a well-defined process for creating passwords [164] and similarly in our user study, 90% of participants said that this advice would make it more difficult for them to create a password.

The probability of a conflict occurring can be estimated using a leaked or past password dataset:

$$P_8 = \frac{\text{Number of passwords which do not abide by the policy}}{\text{Total number of passwords in database}} \tag{D.40}$$

Or alternatively

$$P_8 = \text{average number of password rejections at creation} \tag{D.41}$$

This second equation can give a value greater than 1 but while this makes it no longer a probability it does give a much better estimate for the inconvenience to the user.

The likelihood that these composition rules *makes it more difficult for a user to create a password* can also be taken from the statistics for Komanduri's BlacklistHard policy. They report users making an average of 0.98 attempts before

successfully creating a password which satisfied the BlacklistHard restrictions. $P_{8,1} = 0.98$. The time to choose a new password under this composition policy is 85.4 seconds; $C_{8,1}(85.4)$. We also include the basic cost *need to pick a new password* since a new password will need to be chosen as a result of a new policy being put in place; $C_{9,1}(85.4)$. This is repeated once for each user.

The help desk cost of a user forgetting their password is covered under the cost of forgetting function. Recall the cost function for forgetting is:

$$C_7() : \text{Cost of forgetting} = \text{P[abandon|forget]} * (\$\text{ORG PROFIT PER USER}) + C_1(\text{TIME ADMIN RESET}) + C_{11}(\text{TIME ADMIN RESET})$$

j=2: Input The costs associated with allowing the full input types by the subscriber falls with the organization authenticating. The system must be able to accept all ASCII and Unicode characters. In our administrator survey, administrators were asked about the costs of allowing all ASCII characters. Administrators said there was a minor cost associated with this. In order to accept both ASCII and UNICODE characters we expect there to be a major implementation cost. Accepting all UNICODE characters introduce complications because different keyboards in different locales will potentially give you different encoding. What works on one computer may not work on another computer. However, its implementation is certainly possible and involves applying a normalization process for stabilized strings. NIST suggests using either the NFKC or NFKD normalization [59].

Password hint Not allowing a hint allows for the same forgetability normally associated with a password policy; $P_{7,1} = 0.02044025$. However, this is already accounted for in $j = 1$ *composition rules* so there are no other costs associated with this piece of advice.

Specific knowledge based questions Using specific information questions for authentication allow for a high success rate for both brute force guessing and targeted attacks. The information required to answer the questions can often be found very easily through online searches or can be known by a peer. There is no real cost associated with not allowing these question-answer

authentication methods since an alternative, email or other, can still be used for password recovery. Therefore we leave it out of our calculations.

j=3: Password strength meter Ur et al. [163] present research relating to study participants' reaction to password meters. They found that "the majority of participants who saw the most stringent meters changed their mind partway into password creation, erasing what they had typed and creating a different password". From Table 3 in Ur's paper we find that the percentage of participants who changed their password while entering it when there was no meter was 14.4% and the number of participants who changed their password when there was a meter was an average of 32% over all the different stringency password meters. The difference of $32\% - 14\% = 18\%$ is what we take as the probability a password meter "inconveniences a user's personal system for password creation", $P_{8,3} = 0.18$, though, notably, in the study, participants voluntarily chose to change their password. Ur et al. also report that between 27% and 40% of participants in the four stringent password meter conditions found the meter annoying. This is in comparison to 13% of the lenient/baseline meter participants. We use the low 13% as the annoyance of using a password meter, $P_{11,3} = 0.13$. To assign a cost to the inconvenience of the password meter we take the difference between the average time taken by those using the password meter ($33.6s$) and those creating passwords with no meter ($19.9s$). $C_{11,3} = 33.6s - 19.9s = 13.7seconds$. Including a password strength meter will take organisation time to implement and will also involve user education.

j=4: Toggle to display entered secret Administrators in our survey indicated that there was minor cost to the organisation involved to implement this. We therefore assign it 1 day for set-up, $C_{4,4}$(1 day). It also involved user education. This toggle helps users who would have made a typo in their password on entry. Therefore we take the costs to be 'negative' to show that the advice is offering an improvement in these areas. To measure how likely a typo is we take Komanduri's statistics [85] which record an average of 0.08 confirmation failures for participants using the BlacklistHard policy. So when the secret is able to be toggled visible a user no longer needs to wait for the authentication process to reject the password before realizing it must have been typed wrong. So the probability is $P_{2,5}$=$P_{3,5}$=0.08 per login that a user saves 3 seconds of authentication time $-C_{2,5}(3)$.

j=5: Hash and salt passwords Administrators said this was a minor implementation cost. We have assigned 1 day of organizations time to setting up a system which will hash and salt a password according to the requirements of the NIST 2017 policy. There is an additional 2 seconds of computing time needed for the organization during the authentication procedure.

The disadvantage of hashing and salting passwords over using reversible encryption is that if the password is forgotten it must be created anew. The user cannot be simply reminded of their old password as the organization should not have access to it. We therefore determine that a user will have to choose a new password with the same probability that they forget their password with $P_{9,5}$ = the probability of forgetting = 0.02044. The cost $C_9(85.4)$ will therefore occur with probability $P_{9,5}$ for each user.

Password expiry Not requiring expiry has no costs associated with it. If expiry was enforced multiple costs would need to be reoccurring. For example, all costs associated with composition of the password would reoccur every time the password needs to be recreated.

j=6: Throttling The policy states that:

- The verifier SHALL limit consecutive failed authentication attempts on a single account to no more than 100.

While throttling protects against online guessing it means that an adversary can intentionally lock users out of their accounts [51]. The probability of an adversary seeking to leverage throttling to conduct a denial of service attack depends on the organization. We were unable to find any statistics on its occurrence despite it being an easily measurable attack. We suspect this type of attack would very rarely be used to target an organization, but it is easily used against an individual. To calculate the probability of this we look at the probability of a breach due to a brute force guessing attack [169] and the probability of a targeted attack [128]. The probability affects each user.

$$P_{2,4} = \text{P[targeted attack} \cup \text{ brute force attacker]} = \tag{D.42}$$
$$(0.033896)\left(\frac{(73)(0.72)}{4788}\right) = 0.00037209142$$

Though it is not stated in the policy, we assume the lockout time at 100 consecutive incorrect guesses is 12 hours or until an administrator can unlock the account. We calculate the cost as 12 hours of the users' time $C_{11,6}$(12 hours), and assign the minor help desk, user education and implementation costs as specified by respondents in our administrator survey (Section 2).

j=7: Access to password files should be restricted In our administrator survey we found that this advice incurred minor help desk costs periodically, a small amount of user education periodically and that it is a minor cost to implement.

j=8: Access controls should be applied to access to particular features or systems System wide access controls incur major help desk/user support time and minor user education. It also takes major periodic time to implement. we see this as an initial major set up cost and then a minor per user cost so that each user can be set up with the access controls applicable to them.

Administrators assigned a major help desk cost to users and users assigned a minor increase in their time or effort. We therefore assume these two are related and add in a 15 minute user time or effort for each user as a result of this policy.

In Table D.2 we reference this information using a second set of rows as we have multiple inputs to some cost categories.

j=9: Two physical authenticators The policy states that:

- The CSP SHALL bind at least one, and SHOULD bind at least two, physical (something you have) authenticators to the subscriber's online identity, in addition to a memorized secret or one or more biometrics.

- While all identifying information is self-asserted at Level 1, preservation of online material or an online reputation makes it undesirable to lose control of an account due to the loss of an authenticator. The second authenticator makes it possible to securely recover from an authenticator loss. For this reason, a CSP SHOULD bind at least two physical authenticators to the subscriber's credential at Level 1 as well.

At creation, either a user will need to provide the physical authenticators or the organisation will need to. We assume the organisation provides the devices. We assign 15 minutes of the organization's time per user to distribute the two physical authenticators. We also estimate each authenticator will cost the organization $6 and the postage of these authenticators will cost $0.50.

We estimate the probability a user loses one of the authenticators as 0.01. In this instance the organization will need to send a replacement.

We also assign a minor user education for teaching users the purpose of these physical authenticators. There is a help desk cost should a user lose an authenticator.

j=10: Authenticated protected channel Setting up an authenticated channel could at one time have high costs for the organization in both time and money. Now free and efficient certificates and pre-written code is available to make the task accessible to any organization or administrator. Administrators marked this as a minor cost under the advice statement "don't transmit in cleartext". We assign 1 day for the implementation. Administrators also marked it as having user education and help desk time.

j=11: Re-authentication This requirement affects the amount of times each user must authenticate. We have generally assumed that the user will login daily. Here however we assign a cost to the number of mandatory logins. The user must re-authenticate every 30 days. So in a 1 year time frame, the number of mandatory logins is:

$$\#logins = \left(\frac{T}{\textit{re-authentication requirement}}\right)(\#users) = \frac{365}{30} = 12 \quad \text{(D.43)}$$

We assign 20 minutes organization time to automate this. Using Komanduri's statistics we estimate that the authentication process takes 31.85 seconds for the Blacklist Hard policy.

D.4.3.1 Calculation of NIST 2017 Level 1 costs

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=11}} P_{i,j}.C_{i,j}.R_{i,j} \quad \text{(D.44)}$$

Table D.2 shows the costs associated with each of the $j = 11$ pieces of advice in NIST 2017 Level 1 policy.

Figure D.2: Costs of implementing the NIST 2017 Level 1 password advice.

		Organisation costs					User costs						
		$C_1(t)$: Increased help desk/user support time	$C_2(t_o)$: User education provided	$C_3(r_o)$: Additional organization resources	$C_4(t_o)$: Organization time taken to implement	$C_5(tc_o)$: Increased organization computing power needed	$C_6(t_u)$: Makes it more difficult to create a password	C_7: Increased risk of forgetting	$C_8(r_u)$: Additional user resources needed	$C_9(t_u)$: Need to pick a new password	$C_{10}(tc_u)$: Increased user computing power need	$C_{11}(t_u)$: User time and inconvenience	$C_{12}(t_u)$: User education time required
j=1: Composition	*p(i,1)*		1		1		0.98	0.02044		1			1
	C(i,1)		C_2(1hr)		C_4(1 day)		C_6(85.4 secs)	C_7		C_9(85.4 secs)			C_{12}(15mins)
	R(i,1)		2		1		#users	#users		#users			2*#users
j=2: Input	*p(i,2)*	1	1		1								1
	C(i,2)	C_1(15mins)	C_2(1hr)		C_4(3 week)								C_{12}(15mins)
	R(i,2)	#users	2		1								2*#users
j=3:Password strength meter	*p(i,3)*		1		1		0.18					0.13	1
	C(i,3)		C_2(1hr)		C_4(1 day)		C_6(85.4 secs)					C_{11}(13.7 secs)	C_{12}(15mins)
	R(i,3)		2		1		#users					#users	2*#users
j=4: Toggle to display entered secret	*p(i,4)*		1		1							0.08	1
	C(i,4)		C_2(1hr)		C_4(1 day)							-C_{11}(3 secs)	C_{12}(15mins)
	R(i,4)		2		1							#logins	2*#users
j=5: Hash and salt passwords	*p(i,5)*				1	1				0.02044			
	C(i,5)				C_4(1 day)	C_5(2 secs)				C_9(85.4 secs)			
	R(i,5)				1	#logins				#users			
j=6: Throttling	*p(i,6)*	1	1		1							0.000071576	1
	C(i,6)	C_1(15mins)	C_2(1hr)		C_4(1 day)							C_{11}(12hrs)	C_{12}(15mins)
	R(i,6)	#users	2		1							#users	2*#users
j=7: Access to password files	*p(i,7)*	1	1		1								
	C(i,7)	C_1(15mins)	C_2(1hr)		C_4(1 day)								
	R(i,7)	#users	2		1								
j=8: Access controls applied to features & systems	*p(i,8)*	1	1		1							1	1
	C(i,8)	C_1(15mins)	C_2(1hr)		C_4(3 weeks)							C_{11}(15 mins)	C_{12}(15mins)
	R(i,8)	4*#users	2		1							#users	2*#users
j=8: continued	*p(i,8)*				1								
	C(i,8)				C_4(15mins)								
	R(i,8)				#users								
j=9: Two physical authenticators	*p(i,9)*	0.01	1	1	1							0.01	
	C(i,9)	C_1(15mins)	C_2(1hr)	C_3($12.50)	C_4(15mins)							C_{11}(15 mins)	
	R(i,9)	#users	2	#users	#users							#users	
j=9 continued	*p(i,9)*			0.01	0.01								
	C(i,9)			C_3($6.50)	C_4(15mins)								
	R(i,9)			#users	#users								
j=10: Authenticated protected channel	*p(i,7)*	1	1		1								1
	C(i,7)	C_1(15mins)	C_2(1hr)		C_4(1 day)								C_{12}(15mins)
	R(i,7)	#users	12		1								12*#users
j=11: Reauthentication	*p(i,8)*				1							1	
	C(i,8)				C_4(1 day)							C_{11}(31.85 secs)	
	R(i,8)				1							12*(#users)	

We take the constant values defined in Section D.2 for the organization and first fill these into the cost categories. For example, $ADMIN WAGES = $\$18/(60 * 60) = \$0.005/second$.

The cost category equations then become:
$C_1(t_0) = supportTimeTaken * 0.005$
$C_2(t_o) = EducatorTime * 0.005$
$C_3(r_o) = \$orgResources$
$C_4(t_o) = timeToImplement * 0.005$
$C_5(t_{c_o}) = orgCompTime * 0.01$
$C_6(timeChoosePwd) = (10/100)*15+C_{11}(timeChoosePwd)+(15/100)*C_7()$
$C_7() = (10/100) * 15 + C_1(900) + C_{11}(900)$
$C_8(r_u) = 1 * \$userResources$
$C_9(timeChoosePwd) = (20/100)*15+C_{11}(timeChoosePwd)+C_5(2)+(20/100)*$
$C_7()$
$C_{10}(t_{c_u}) = 1 * userCompTime * 0.01$
$C_{11}(t_u) = 1 * userTime * 0.005$
$C_{12}(t_u) = 1 * EducationTime * 0.005$

Now for each piece of advice j we can fill the input values from the table into these cost categories. Finally, we multiply each $C_{i,j}$ by the corresponding $P_{i,j}$ and $R_{i,j}$.

$$\begin{aligned}
\mathbb{E}[Costs]_{j=1} &= ((1)(60*60*0.005)(2)) + ((1)(60*60*8)(0.005)(1)) \\
&+ ((0.98)((10/100*15) + (1*85.4*0.005) + (15/100)((10/100)*15 + (900*0.005) \\
&+ (1*900*0.005)))(500)) + ((0.02044)((10/100)*15 + (900*0.005) \\
&+ (1*900*0.005))(500)) + ((1)((20/100)*15 + (1*85.4*0.005) + (2*0.01) \\
&+ (20/100)*((10/100)*15 + 900*0.005 + (1*900*0.005)))(500)) \\
&+ ((1)(1*900*0.005)(2*500)) = 9276.79 \\
\mathbb{E}[Costs]_{j=2} &= ((1)(900*0.005)(500)) + ((1)(60*60*0.005)(2)) \\
&+ (1*(60*60*8*5*3*0.005)*1) + ((1)(1*900*0.005)(2*500)) = 8946 \\
\mathbb{E}[Costs]_{j=3} &= ((1)(60*60*0.005)(2)) + ((1)(60*60*8)(0.005)(1)) \\
&+ ((0.18)((10/100)*15 + (1*85.4*0.005) + (15/100)((10/100)*15 + (900*0.005) \\
&+ (1*900*0.005)))(500)) + ((0.13)(1*13.7*0.005)*500) \\
&+ ((1)(1*900*0.005)(2*500)) = 4999.6325 \\
\mathbb{E}[Costs]_{j=4} &= ((1)(60*60*0.005)(2)) + ((1)(60*60*8)(0.005)(1)) \\
&+ ((0.08)(1*3*0.005)*130500) + ((1)(1*900*0.005)(2*500)) = 4836.6 \\
\mathbb{E}[Costs]_{j=5} &= ((1)(60*60*8)(0.005)(1)) + ((1)(2*0.01)(130500)) + ((0.02044)((20/100)*(15) \\
&+ (1)(85.4)(0.005) + ((2)(0.01)) + (20/100)*((10/100)*15 + 900*0.005 \\
&+ (1*900*0.005)))(500)) = 2810.69034 \\
\mathbb{E}[Costs]_{j=6} &= ((1)(900*0.005)(1)) + ((1)(60*60*0.005)(2)) + ((1)(60*60*8)(0.005)(1)) \\
&+ ((0.000071576)(1*60*60*12*0.005)(500)) + ((1)(1*900*0.005)(2*500)) \\
&= 6937.730208 \\
\mathbb{E}[Costs]_{j=7} &= ((1)(900*0.005)(1)) + ((1)(60*60*0.005)(2)) + ((1)(60*60*8)(0.005)(1)) \\
&= 2430 \\
\mathbb{E}[Costs]_{j=8} &= ((1)(900*0.005)(4*500)) + ((1)(60*60*0.005)(2)) \\
&+ (1*(60*60*8*5*3*0.005)*1) + (1(1*900*0.005)*500) \\
&+ ((1)(1*900*0.005)(2*500)) + ((1)(900*0.005)(500)) = 20196 \\
\mathbb{E}[Costs]_{j=9} &= ((0.01)(900*0.005)(500)) + ((1)(60*60*0.005)(2)) \\
&+ ((1)(12.5)(500)) + ((1)(900*0.005)(500)) + ((0.01)(1*900*0.005)(500)) \\
&+ ((0.1)(6.5)(500)) + ((0.01)(1*900*0.005)(500)) = 8636 \\
\mathbb{E}[Costs]_{j=10} &= ((1)(900*0.005)(1)) + ((1)(60*60*0.005)(12)) + ((1)(60*60*8)(0.005)(1)) \\
&+ ((1)(1*900*0.005)(12*500)) = 29610 \\
\mathbb{E}[Costs]_{j=11} &= ((1)(60*60*8)(0.005)(1)) + ((1)(1*31.85*0.005)(500)) = 1099.5
\end{aligned}$$

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=11}} P_{i,j}.C_{i,j}.R_{i,j} = \$99{,}778.94 \tag{D.45}$$

D.4.4 Value of the NIST 2017 Level 1 policy

$$\mathbb{E}[Value] = \mathbb{E}[Benefits] - \mathbb{E}[Costs] \tag{D.46}$$

$$\mathbb{E}[Value] = \$81{,}042.19 - \$99{,}778.94 = -\$18{,}736.75 \tag{D.47}$$

D.5 NIST 2003 Level 1 Worked example

We now look at the NIST 2003 Level 1 policy and highlight the costs and benefits of this policy. We will not repeat our explanations already described for the NIST 2017 policy so this will be a briefer account.

One of the suggestions by the NIST 2003 policy [20] for authentication is a challenge response protocol. This involves a subscriber creating a password and sending it to the organization. When the subscriber wants to login the organization sends them a challenge, for example 'xY253N'. The subscriber then hashes their password with this value, $\mathcal{H}[challenge, password]$, and sends it back to the verifier. The verifier hashes their stored value of the password with the same challenge. If the results match the subscriber is successfully authenticated. This is the method we will consider when looking at their authentication rules.

D.5.1 Policy summary

General authentication rules

- Plaintext passwords or secrets shall not be transmitted across a network.
- This level does not require cryptographic methods that block offline analysis by eavesdroppers.
- There is no requirement at this level to use approved cryptographic techniques.
- Long term shared secrets may be revealed to verifiers.

- There is no stipulation about the revocation or lifetime of credentials at Level 1.
- Files of shared secrets used by verifiers shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.

Authenticator: Memorized secrets

- There are no min-entropy requirements for Level 1.
- The probability of success of a targeted online password guessing attack by an attacker who has no a priori knowledge of the password, but knows the user name of the target, shall not exceed 2^{-10} (1 in 1024), over the life of the password.
  - Lock the password for 1 minute after 3 incorrect guesses.
- Shared secret files shall not contain the plaintext password.
  - Typically they contain a one-way hash or "inversion" of the password.

D.5.2 Benefits: NIST 2003 Level 1 policy

Benefits, as mentioned, are affected by the probability of successful attack. Looking at the pieces of advice from the NIST 2003 policy we can see which of the identified attacks the policy protects against.

D.5.2.1 Quantifying attacks: NIST 2003 Level 1

Assertion Manufacture or Modification

- Files of shared secrets used by verifiers shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.

Discretionary access controls limit the number of people who have the authority to manipulate assertions. If 10% of the people in an organization have access to the authentication mechanism then we model this by taking this fraction of the probability of a breach.

$$\begin{aligned} \text{P[assertion manufacture or modification]} &= \left(\frac{(128)(0.72)}{4788}\right)(0.1) \\ &= 0.00192481203007365 \end{aligned} \tag{D.48}$$

Physical theft No protection

$$\text{P[theft]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429 \tag{D.49}$$

Duplication No protection

$$\begin{aligned} \text{P[duplication]} &= \text{P[divulge password]*P[partner exploits]} \\ &\quad + \text{P[recorded password compromised]} \\ &= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) \\ &= 0.00958217270195177 \end{aligned} \tag{D.50}$$

Eavesdropping

- Plaintext passwords or secrets shall not be transmitted across a network.
- This level does not require cryptographic methods that block offline analysis by eavesdroppers.
- There is no requirement at this level to use approved cryptographic techniques.
- Long term shared secrets may be revealed to verifiers.

The NIST 2003 policy recommends a challenge response protocol whereby the challenge is sent to the subscriber, the subscriber hashes their password with this challenge ($\mathcal{H}$[secret,challenge]) and sends the hashed version back to the verifier. The verifier/organization hashes their stored value of the users' password with the same challenge and verifies that both hashes match. An

eavesdropper on the connection can; see the plaintext password when it is sent to the subscriber at creation, or decrypt the hash sent back to the verifier by the user.

We were unable to find a statistic for how often eavesdropping attacks take place. This is likely a reflection of the difficulty/impossibility of detecting an eavesdropper. For lack of a better option we take the probability of a remote eavesdropper attack as the statistic in the Verizon DBIR for breaches from the vector LAN access.

To find the probability an attacker sees the plaintext password sent to the verifier at creation we say that of the number of logins by a user in time frame T, one is at creation. So the probability, given an eavesdropper is on a users' connection, that they are on the creation connection is 1/#logins. For our organization, the #logins per user in time frame T is one per working day; #logins = 261. So the probability of an eavesdropper on the line at creation is: P[eavesdropper sees plaintext at creation] = P[eavesdropper(stat:LAN access)].P[connection is at creation] = P[eavesdropper(stat:LAN access)](1/261).

The probability an attacker decrypts the hash sent back to the verifier by the user is the probability of an offline guessing attack being successful by the probability the password was eavesdropped. We estimate that 72.07408% of the passwords can be guessed in 10^{13} guesses (described further below).

A shoulder surfing eavesdropping attack is slightly more difficult for the NIST 2003 policy than the NIST 2017 policy. Because a password can not be toggled as visible, a physical eavesdropper must be able to see and record the keys typed by the user on their keyboard. For this reason, we include physical access to the victim's work area as a requirement for a physical eavesdropping attack. Keylogger attacks are still not mitigated. A pass-the-hash attack should still not be possible.

$$\begin{aligned}
&\text{P[eavesdropping]} = \\
&\text{P[eavesdropping attempted(stat:LAN access)].P[connection is at creation]} \\
&+ \text{P[eavesdropping attempted(stat:LAN access)].P[offline guessing]} \\
&+ \text{P[keylogger breach]} + \text{P[physical surveillance breach} \cup \text{access to victim work area]} \\
&= \left(\frac{(118)(0.72)}{3231}\right)\left(\frac{1}{261} + 0.64\right) + \left(\frac{(595)(0.72)}{4788}\right) + \left(\frac{(21)(0.72)}{4788}\right)\left(\frac{16}{4788}\right) \\
&= 0.106413954407714
\end{aligned} \tag{D.51}$$

(D.52)

Offline guessing attack

- Shared secret files shall not contain the plaintext password.
  - Typically they contain a one-way hash or "inversion" of the password.
- There are no min-entropy requirements for Level 1.

For offline guessing to take place first a dataset of passwords must be leaked. This time there is no requirement for a global salt so this does not also need to have been leaked.

$$\text{P[database of passwords is leaked]} = \frac{(49)(0.72)}{4788} = 0.00736842105258628 \tag{D.53}$$

The passwords in this case are stored as a one-way hash or "inversion" and can therefore be reversed with a key. Attackers can either steal the key with the dataset, or brute force guess the key. Brute forcing the key should only be possible if the key space is small. In some cases, such as the Adobe leak [40] even when passwords are cracked, the key is never discovered. Because the NIST policy sets no requirements on the size of the key we take the probability it is compromised as 0.01 [64].

The passwords can be guessed directly using brute force especially since salting is not enabled. There are no composition or minimum entropy requirements

on the password. As with the no policy set up (Section D.4.2.3), we assume passwords less than 8 characters can be brute force guesses and for those above 8 characters, we take Kelley et al.'s value for percentage of passwords cracked when they were made using their basic8survey password rules. This gives 72.07408% of passwords guessed in 10^{13} guesses.

$$\begin{aligned}&\text{P[user password revealed]} = \\ &\text{P[key leaked]} + \text{P[key brute forced]} + \text{P[user password in } 10^{13} \text{ guesses]} \\ &= \frac{49}{4788} + 0.01 + 0.7207408 = 0.740974718128655\end{aligned} \tag{D.54}$$

Side Channel Attack No protection.

As we discussed in NIST 2017 Level 1, there are side channel attacks which exist for compromising passwords.

$$\begin{aligned}\text{P[side channel attack]} =& \text{P[targeted attack].P[not detected].P[malware]} \\ =& (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451251986\end{aligned} \tag{D.55}$$

Phishing or Pharming No protection

$$\text{P[phishing} \cup \text{pharming]} = \text{P[phishing]} = \frac{(653)(0.72)}{4788} = 0.0981954887218051 \tag{D.56}$$

Social Engineering No protection

$$\text{P[social engineering]} = \text{P[pretexting]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429 \tag{D.57}$$

Online guessing

- There are no min-entropy requirements for Level 1.

- Lock the password for 1 minute after 3 incorrect guesses.

Locking the account for 1 minute after 3 incorrect guesses means that an attacker can make no more than $525600 * 3 = 1576800 \approx 10^6$ guesses in the year time frame.

There is no minimum entropy requirement for Level 1 passwords. In the NIST 2003 policy, this means that there is no minimum length. Again, we assume passwords less than 8 characters long can easily be brute force guessed (Section D.4.2.3).

We use [83] to estimate the probability of guessing the users' password eight characters and above in length in 1576800 guesses. We take their basic8survey composed passwords, of which 14% are guessed in 1576800 guesses. Combined with the simplicity of guessing the shorter passwords, this gives a probability of compromise of 0.3328808.

Because the passwords are not salted, once a password is cracked all other users' who have used the same password will also be compromised.

$$\begin{aligned}
&\text{P[online guessing successful]} = \text{P[targeted attack].P[password in 1576800 guesses]} \\
&+ \text{P[(username guessed in } 10^{13} \text{ guesses]P[password guessed in 1576800]} \\
&= (0.033896)(0.3328808)(0.72) + \left(\frac{(73+631)(0.72)}{4788}\right)(0.7207408)(0.3328808) \\
&= 0.0335231274432347
\end{aligned} \tag{D.58}$$

Endpoint Compromise No protection

$$\begin{aligned}
&\text{P[endpoint compromise]} \\
&= \text{P[breach using backdoor or C2]} = \frac{(678)(0.72)}{4788} = 0.101954887218044
\end{aligned} \tag{D.59}$$

Unauthorized Binding No protection

$$\begin{aligned}&\text{P[unauthorized binding]} = \\ &\Big(\text{P[use stolen credentials]} \cup \text{P[physical theft]}\Big)\text{P[targeted attack]} \\ &= \left(\frac{(631)(0.72)}{4788} + \frac{(39)(0.72)}{4788}\right)(0.033896) = 0.00341508571428736\end{aligned} \tag{D.60}$$

D.5.2.2 Quantifying benefit of the NIST 2003 Level 1 policy

We quantify benefits by splitting the equation to look at the scenario when the password dataset has been leaked and when it has not (Equation D.16). For the NIST 2003 Level 1 policy this gives the following probabilities of compromise:

$$p_l = \mathbb{P}[\text{compromise}|\text{leak}] = 0.823647757902002 \tag{D.61}$$

$$p_{l'} = \mathbb{P}[\text{compromise}|\text{no leak}] = 0.319169770518423 \tag{D.62}$$

Now, taking the values defined in Section D.1: $Lsystem = \$10^7$, $L_1 = \$166$, and $N = 500$ users. We calculate the expected loss due to compromise for the organization when the NIST 2003 Level 1 policy is in place.

$$\begin{aligned}&\mathbb{E}[\text{Loss}|\text{NIST 2003 L1 policy}] = \\ &\Big(\mathbb{P}\left[\mathcal{N}\left(0.8236477579, \frac{(0.8236477579)(1-0.8236477579)}{500}\right) > 0.5\right] \times \$10^7 \\ &+ (500)(0.8236477579)(\$166)\Big) \times 0.001417397660819 \\ &+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.3191697705, \frac{(0.3191697705)(1-0.3191697705)}{500}\right) > 0.5\right] \times \$10^7 \\ &+ (500)(0.3191697705)(\$166)\Big) \times (1 - 0.001417397660819) \\ &= \$74206.1431050767 + \$27246.3697187516 = \$101452.512823828\end{aligned} \tag{D.63}$$

Taking the value for expected losses with no policy from Section D.4.2.3, our

computation of the NIST 2003 L1 benefits is:

$$\begin{aligned}\mathbb{E}[\text{Benefits}] &= \mathbb{E}[\text{Loss}|\text{No policy}] - \mathbb{E}[\text{Loss}|\text{NIST 2003 L1 policy}] \\ &= \$105571.91 - \$101452.51 = \$4119.39\end{aligned} \tag{D.64}$$

D.5.3 Costs: NIST 2003 Level 1 policy

In this section we will quantify the costs associated with each piece of advice in the NIST 2003 policy

Composition

- There are no min-entropy requirements for Level 1.

The NIST 2003 policy has no composition requirements and therefore incurs no costs relating to it.

j=1: Throttling

- Lock out the claimant for a minute after three successive failed authentication attempts.

One scheme suggested by the NIST 2003 Level 1 policy is to lock out the claimant for a minute after three successive failed authentication attempts.

Brostoff and Sasse [18] show that 31% of their study participants who would have been able to login with unlimited number of attempts, failed to login successfully within 3 attempts.

Brostoff and Sasse also found that approximately 7% of these failed logins led to password reminder requests: $P_{(7,1)} = 0.31 * 0.07 = 0.0217$

Because it only requires 3 guesses, both a brute force attacker or a peer could also lock the account.

$$P_{2,1} = \text{P[targeted.attack]}\,(\text{P[brute force.attacker]} + \text{P[partner]}) \qquad \text{(D.65)}$$
$$= (0.033896)\left(\frac{(73)(0.72)}{4788} + \frac{(108)(0.72)}{3231}\right) = 0.00118786145$$

The 60 second inconvenience to the user will occur whether an outsider or the user themselves have exceeded the 3 guess limit: $0.31 + 0.00118786145 = 0.31118786145$.

Administrators in our survey also indicated that it required minor held desk time, user education and time to implement.

Password expiry

- There is no stipulation about the revocation or lifetime of credentials at Level 1.

Not requiring expiry has no costs associated with it.

j=2: Password storage

- Shared secret files shall not contain the plaintext password.
  - Typically they contain a one-way hash or "inversion" of the password.

We have assigned 1 day of organizations time to setting up a system which will create a one-way hash or "inversion" for each user password. There is an additional 0.5 seconds of computing time needed for the organization to store the password "inversion" at creation.

j=3: Authenticated protected channel

- Plaintext passwords or secrets shall not be transmitted across a network.

The NIST 2003 level 1 policy recommends the use of a challenge-response protocol. The password is hashed with the challenge before being sent across the network. This requires an additional 0.5 second of user computing time to preform the hash on the password and challenge. It will take 1 second of the organizations computing time to generate a unique challenge for the authentication and hash this with their stored users' password. We assign 1 day to the organization to set this process up and also include the need for user education and help desk time as per the "Don't transmit in cleartext" advice (Table 2.5).

Cryptographic methods

- This level does not require cryptographic methods that block offline analysis by eavesdroppers.
- There is no requirement at this level to use approved cryptographic techniques.

These incur no user or administrator direct costs.

Shared secrets revealed to verifier

- Long term shared secrets may be revealed to verifiers.

In the challenge-response protocol this refers to when the secret is initially created by the user, it may be sent to the verifier for their storage. This is a key component of the challenge-response system and while it has security impacts it incurs no additional costs.

j=4: Access to password files

- Files of shared secrets used by verifiers shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.

D.5.3.1 Calculation of NIST 2003 Level 1 costs

Table D.3 shows the costs associated with pieces of advice in NIST 2003 Level 1.

Figure D.3: Costs of implementing the NIST 2003 Level 1 password advice.

		Organisation costs					User costs						
		$C_1(t_o)$: Increased help desk/user support time	$C_2(t_o, t_u)$: User education required	$C_3(r_o)$: Additional organization resources	$C_4(t_o)$: Organization time taken to implement	$C_5(tc_o)$: Increased organization computing power needed	$C_6(t_u)$: Makes it more difficult to create a password	C_7: Increased risk of forgetting	$C_8(r_u)$: Additional user resources needed	$C_9(t_u)$: Need to pick a new password	$C_{10}(tc_u)$: Increased user computing power need	$C_{11}(t_u)$: User time and inconvenience	$C_{12}(t_u)$: User education time required
j=1: Throttling	*p(i,1)*		1		1			0.0217				0.31	1
	C(i,1)		C_2(1hr)		C_4(1 day)			C_7				C_{11}(60 secs)	C_{12}(15mins)
	R(i,1)		2		1			#users				#logins	2*#users
j=2: Password storage	*p(i,2)*				1	1							
	C(i,2)				C_4(1 day)	C_5(0.5 secs)							
	R(i,2)				1	#users							
j=3: Authenticated protected channel	*p(i,3)*	1	1		1	1	0.18				1	0.13	1
	C(i,3)	C_1(15mins)	C_2(1hr)		C_4(1 day)	C_5(1 secs)	C_6(85.4)				C_{10}(0.5)	C_{11}(13.7)	C_{12}(15mins)
	R(i,3)	1	2		1	#logins	#users				#logins	#users	2*#users
j=4: Access to password files	*p(i,4)*	1	1		1								
	C(i,4)	C_1(15mins)	C_2(1hr)		C_4(1 day)								
	R(i,4)	#users	2		1								

Using this table and the values defined in Section Section D.2 we have the following results for the cost of the NIST 2003 L1 policy:

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=4}} P_{c_{i,j}}.C_{i,j}.R_{i,j} = \$26{,}468.56 \tag{D.66}$$

D.5.4 Value of the NIST 2003 Level 1 policy

$$\mathbb{E}[Value] = \mathbb{E}[Benefits] - \mathbb{E}[Costs] \tag{D.67}$$

$$\mathbb{E}[Value] = \$4{,}119.39 - \$26{,}468.56 = -\$22{,}349.16 \tag{D.68}$$

D.6 NIST 2017 Level 3 Worked example

Level 3 is the highest authentication assurance level in the NIST 2017 policy [59]. It requires multi-factor authentication. To satisfy Level 3 we have chosen to authenticate using a Single-Factor Cryptographic device plus a memorized secret. A single factor cryptographic device is a hardware device that performs cryptographic operations using protected cryptographic key(s) and provides the authenticator output via direct connection with the user endpoint. NIST describes the authenticator as operating by signing a challenge nonce presented through a direct computer interface (e.g. a USB port). For our evaluation we assume such a device will take the form of a USB.

The rules for using a memorized secret for authentication are the same as for the NIST 2017 Level 1 policy (Section D.4). Therefore we will list the additional rules which apply to the use of the single-factor cryptographic device.

D.6.1 Policy summary

General authentication rules

- Periodic re-authentication of subscriber sessions SHALL be preformed after 12 hours or 15 minutes inactivity; SHALL use both authentication factors.
- Communication between the claimant and verifier SHALL be via an authenticated protected channel.

- Access controls in place.
  - Specified by NIST 800-53 [120].
- Store the expected authenticator output in hashed form.
- A verifier impersonation resistant authentication protocol SHALL establish an authenticated protected channel with the verifier.
- The CSP SHALL bind at least one, and SHOULD bind at least two, physical (something you have) authenticators to the subscriber's online identity, in addition to a memorized secret or one or more biometrics.
  - Binding of multiple authenticators is preferred in order to recover from the loss or theft of the subscriber's primary authenticator.
  - One example of a verifier impersonation resistant authentication protocol is client authenticated TLS, because the client signs the authenticator output along with earlier messages from the protocol that are unique to the particular TLS connection being negotiated.

Authenticator 1: Memorized secrets

- Same as NIST 2017 Level 1 policy (Section D.4)

Authenticator 2: Single factor cryptographic device

- The single factor cryptographic device authenticators encapsulate one or more secret keys unique to the device that SHALL NOT be exportable (i.e., cannot be removed from the device).
- The challenge nonce SHALL be at least 64 bits in length, and SHALL either be unique over the authenticator's lifetime or statistically unique.
- Tamper detection and response should be in place for covers and doors [48].
- Single-factor cryptographic device authenticators SHOULD require a physical input (e.g., the pressing of a button) in order to operate.
- The secret key and its algorithm SHALL provide at least the minimum security length specified in the latest revision of SP 800-131A (112 bits as of the date of this publication).
- Use a cryptographic authenticator that requires the verifier store a public key corresponding to a private key held by the authenticator.

D.6.2 Benefits: NIST 2017 Level 3 policy

The NIST 2017 Level 3 policy requires two factor authentication and Client authenticated TLS. Therefore the probability of successfully compromising a user is marked by the probability of compromising the single factor cryptographic device, the memorized secret and the client certificate and private key. A compromise, for example, could result from: the password (pwd) compromised via social engineering, the cryptographic device (USB) stolen and the certificate and private key ($cTLS$) gained via malware. To model this, we quantify the probabilities of compromise separately for each of the two factors and also for the client authenticated TLS.

$$p_{pwd} = 1 - \prod_{a}(1 - \mathrm{P}_{pwd,a}) \tag{D.69}$$

$$p_{USB} = 1 - \prod_{a}(1 - \mathrm{P}_{USB,a}) \tag{D.70}$$

$$p_{cTLS} = 1 - \prod_{a}(1 - \mathrm{P}_{cTLS,a}) \tag{D.71}$$

where p is the probability of the $pwd/USB/cTLS$ being compromised and P_a is the probability of compromise due to each individual attack type a.

Once we have calculated p_{pwd}, p_{USB} and p_{cTLS} we can calculate the probability a user is compromised, p, by:

$$p = p_{pwd} * p_{USB} * p_{cTLS} \tag{D.72}$$

As for the previous policies, we must again calculate this p when there has been a leak of the password file and when there has not: $\mathbb{P}$[compromise|no leak] and $\mathbb{P}$[compromise|database leak]

D.6.2.1 Quantifying attacks: NIST 2017 Level 3

Assertion Manufacture or Modification Assertion manufacture or modification can only for implemented by a privileged user. These, in our example, allow access to 10% of the administrators. The risk of this attack occurring is

mitigated for all three factors by having access controls in place.

$$\mathrm{P}_{pwd,cTLS,USB}[\text{assertion manufacture or modification}] = \left(\frac{(128)(0.72)}{4788}\right)(0.1) = 0.00192481203007519 \tag{D.73}$$

Theft

- Periodic re-authentication of subscriber sessions SHALL be preformed after 12 hours or 15 minutes inactivity; SHALL use both authentication factors.

If the computer is unlocked when it is stolen then the certificate and private key will be compromised. Given a typical 8 hour working day and a 5 day working week, the computer will lock at lunch time and at the end of the working day. Meaning that the computer is locked for approximately 16.75 hours 5 days a week and all day during the weekend. The probability that an attacker stealing a laptap finds it unlocked is $1 - \left((\frac{5}{7})(\frac{16.75}{24}) + \frac{2}{7}\right) = 0.21577380952$.

$$\begin{aligned}&\mathrm{P}_{pwd,cTLS}[\text{theft}] = \\ &\mathrm{P}[\text{stolen}]*\mathrm{P}[\text{tampering}] + \mathrm{P}[\text{stolen}]*\mathrm{P}[\text{unlocked}] \\ &= \left(\frac{(39)(0.72)}{4788}\right)\left(\frac{27}{4788}\right) + \left(\frac{(39)(0.72)}{4788}\right)(0.21577380952) \\ &= 0.0012985117869626\end{aligned} \tag{D.74}$$

The USB is susceptible to theft.

$$\mathrm{P}_{USB}[\text{theft}] = \frac{(39)(0.72)}{4788} = 0.00586466165413534 \tag{D.75}$$

Duplication

The Secret key on the USB is not exportable.

$$\mathrm{P}_{USB}[\text{duplication}] = 0 \tag{D.76}$$

Password is only duplicable if revealed.

$$\begin{aligned} &\mathrm{P}_{pwd}[\text{duplication}] = \mathrm{P}[\text{divulge password}]*\mathrm{P}[\text{partner exploits}] \\ &+ \mathrm{P}[\text{recorded password compromised}] \\ &= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) = 0.00958217270194984 \end{aligned} \tag{D.77}$$

Theft of the certificate's private key must be a targeted attack as each individual user will have their own stored private key and certificate. We assume that if the private key is captured then the certificate will also be taken with little extra effort. The value for a targeted attack comes from [128].

$$\begin{aligned} &\mathrm{P}_{cTLS}[\text{duplication}] = \mathrm{P}[\text{targeted attack}].\mathrm{P}[\text{private key} \cap \text{certificate leaked}] \\ &= 0.033896\left(\frac{(49)(0.72)}{4788}\right) = 0.00024976 \end{aligned} \tag{D.78}$$

Eavesdropping

The communication of the password over the network is encrypted.

$$\begin{aligned} \mathrm{P}_{pwd}[\text{eavesdropping}] &= \mathrm{P}[\text{physical surveillance}] + \mathrm{P}[\text{keylogger breach}] \\ &= \left(\frac{(21)(0.72)}{4788}\right) + \left(\frac{(595)(0.72)}{4788}\right) \\ &= 0.0926315789473684 \end{aligned} \tag{D.79}$$

The USB cryptographic device and client authenticated TLS are not susceptible to eavesdropping attacks. They use public private key pairs and therefore the private key is never transmitted.

$$\mathrm{P}_{USB,cTLS}[\text{eavesdropping}] = 0 \tag{D.80}$$

Offline Guessing attacks The password hash and salt can be cracked offline if it is leaked by the organization. The USB and client authenticated TLS public keys stored on the organizations' system could be leaked and a brute force attack against the public keys to identify the private key is possible. However, the probability of success is negligibly low for a well implemented asymmetric cryptographic algorithm.

$$\text{P[database of passwords is leaked} \cup \text{salt leaked]} \tag{D.81}$$
$$= \left(\frac{(49)(0.72)}{4788}\right)\left(\frac{49}{4788}\right) = 0.0000754078177900894$$

$$P_{pwd}[\text{user password is guessed in } 10^{13}/\#users \text{ guesses}] = 0.055 \tag{D.82}$$

$$P_{USB,cTLS}[\text{brute force guess private key}]=0 \tag{D.83}$$

Side Channel Attack Side channel attacks are possible against all three security methods: password, TLS and USB cryptographic device. However, measuring the probability of them occurring is a difficult problem. We make an attempt at rough estimates here but due to the lack of sufficient data these numbers are not reliable.

While side channel attacks are generally designed against cryptographic keys, there is at least one attack which will work against passwords [158]. This attack was described in Section D.4 and applies again here.

$$\text{P}_{pwd}[\text{side channel attack}] = \tag{D.84}$$
$$\text{P[targeted attack].P[not detected].P[malware]}$$
$$= (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451253482$$

The NIST guidelines state that the USB device must be resistant to power and timing analysis attacks. Other attacks are possible though, particularly those that can be carried out with physical access [96].

$$\text{P}_{USB}[\text{side channel attack}] = \tag{D.85}$$
$$\text{P[targeted].P[physical access].P[exploit vulnerabilities]}$$
$$= (0.033896)\left(\frac{(53)(0.72)}{3231}\right)\left(\frac{6}{4788}\right) = 0.00000050166865632045$$

Encryption for TLS using the private key is susceptible to a side channel attack over the network [2]. Side channel attacks to determine the private key can also be done via physical access or malware.

$$\begin{aligned}
&\mathrm{P}_{cTLS}[\text{side channel attack}] = \\
&\mathrm{P}[\text{targeted attack}].\,(\mathrm{P}[\text{physical access}] \cup \mathrm{P}[\text{malware}]) \\
&\cup\,(\mathrm{P}[\text{eavesdropper(stat:LAN access)}])\,(\mathrm{P}[\text{exploit vulnerabilities}]) \\
&= (0.033896)\left(\frac{(53)(0.72)}{3231} + \frac{(33)(0.72)}{3231}\right) + \left(\frac{(118)(0.72)}{3231}\right)\left(\frac{6}{4788}\right) \\
&= 0.000682546111239098
\end{aligned} \tag{D.86}$$

Phishing or Pharming A password will only be able to be phished if the client authenticated TLS connection is compromised first.

$$P_{pwd}[\text{phishing} \cap \text{pharming}] = \tag{D.87}$$

$$\mathrm{P}[\text{phishing}] = \frac{(653)(0.72)}{4788} = 0.0981954887218045$$

$$P_{USB}[\text{phishing} \cap \text{pharming}] = 0 \tag{D.88}$$

It is not reasonable to assume that the private key of the USB or client authenticated TLS can be compromised via phishing. Since it should never be transmitted. However we do consider it to be possible to compromise it via social engineering.

$$P_{cTLS}[\text{phishing} \cap \text{pharming}] = 0 \tag{D.89}$$

Social Engineering The mitigation suggested by the NIST 2017 guidelines are to avoid using authenticators that present a risk of social engineering of third parties. The password alone does not fall into this category.

$$P_{pwd}[\text{social engineering}] = \mathrm{P}[\text{pretexting}] = \frac{(39)(0.72)}{4788} = 0.00586466165413534 \tag{D.90}$$

We consider that a person masquerading as an IT assistant could relatively easily convince a user to send them the files which include the certificate and private key. It is slightly more difficult than revealing a password as the certificate and key cannot be easily stated over the phone. The Verizon statistics tell us that phones are the attack vector 5 of 3231 times.

$$P_{cTLS}[\text{social engineering}] = \text{P}[\text{pretexting}]\&\text{P}[\text{not phone}] \tag{D.91}$$
$$= \left(\frac{(39)(0.72)}{4788}\right)\left(1 - \frac{5}{3231}\right) = 0.00585558604031134$$

Convincing a user to part with a USB decide would be more difficult than either of the above. Since the user would not have the use of the device once they part with it as it cannot be duplicated there would be an inconvenience factor for the user in this. If we take the P[pretexting] to get someone's password as requiring 1 minute of the users' time. Then for a USB we set the time as 10 minutes for someone with physical proximity to the user and 3 working days (1440 minutes) for a remote attacker.

$$P_{USB}[\text{social engineering}] = \tag{D.92}$$
$$\left(\frac{\text{P}[\text{pretexting}]}{10}\right)(\text{P}[\text{partner exploits}]) + \left(\frac{\text{P}[\text{pretexting}]}{1440}\right)(\text{P}[\text{remote attacker}])$$
$$= \left(\frac{(39)(0.72)}{(4788)(10)}\right)\left(\frac{108}{4788}\right) + \left(\frac{(39)(0.72)}{(4788)(1440)}\right)\left(1 - \frac{108}{4788}\right)$$
$$= 0.000017209376827148$$

Online Guessing attacks

$$P_{pwd}[\text{online guessing successful}] \tag{D.93}$$
$$= \text{P}[\text{targeted attack}].\text{P}[\text{password in (99*365) guesses}]$$
$$+ \text{P}[\text{brute force guessing}].\text{P}[(\text{username guessed in } 10^{13} \text{ guesses}]$$
$$* \text{P}[\text{password guessed in (99*365)}]$$
$$= (0.033896)\,(0.01)(0.72) + \left(\frac{(73+631)(0.72)}{4788}\right)(0.63)(0.01)$$
$$= 0.000910998568420505$$
$$P_{USB,cTLS}[\text{online guessing successful}] = 0 \tag{D.94}$$

Endpoint compromise

$$P_{pwd}[\text{endpoint compromise}] = \frac{(678)(0.72)}{4788} = 0.101954887218044 \tag{D.95}$$

Because the private key never leaves the USB device, a compromised endpoint should not be able able to compromise the device.

$$P_{USB}[\text{endpoint compromise}] = 0 \tag{D.96}$$

A compromised endpoint could compromise the private key and certificate.

$$P_{cTLS}[\text{endpoint compromise}] = \frac{(678)(0.72)}{4788} = 0.101954887218044 \tag{D.97}$$

Unauthorized binding

$$P_{pwd}[\text{unauthorized binding}] = \text{P[use stolen credentials].P[targeted attack]} \tag{D.98}$$

$$= \left(\frac{(631)(0.72)}{4788}\right)(0.033896) = 0.00321629714285714$$

To compromise the USB or client TLS the attacker would need access to the database and need to have a USB, and certificate and private key created. This would need to be a targeted attack.

$$P_{USB,cTLS}[\text{unauthorized binding}] = \left(\frac{(49)(0.72)}{4788}\right)(0.1)(0.033896) \tag{D.99}$$

$$= 0.000024976$$

D.6.2.2 Calculation of NIST 2017 Level 3 losses

We quantify benefits by splitting the equation to look at the scenario when the password dataset has been leaked and when it has not (Equation D.16). For the NIST 2003 Level 3 policy this gives use the following probabilities of

compromise:

$$\mathbb{P}_{pwd}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{pwd,a}) = 0.3214100261 \tag{D.100}$$

$$\mathbb{P}_{USB}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{USB,a}) = 0.0078205399 \tag{D.101}$$

$$\mathbb{P}_{cTLS}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{cTLS,a}) = 0.1109406998 \tag{D.102}$$

$$\mathbb{P}_{pwd}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{pwd,a}) = 0.2819153715 \tag{D.103}$$

$$\mathbb{P}_{USB}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{USB,a}) = 0.00782053993 \tag{D.104}$$

$$\mathbb{P}_{cTLS}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{cTLS,a}) = 0.1109406998 \tag{D.105}$$

$$\begin{aligned} p_l &= \mathbb{P}[\text{compromise}|\text{leak}] \qquad (D.106)\\ &= \mathbb{P}_{pwd}[\text{compromise}|\text{leak}] * \mathbb{P}_{USB}[\text{compromise}|\text{leak}] * \mathbb{P}_{cTLS}[\text{compromise}|\text{leak}] \\ &= 0.0002788605356 \\ p_{l'} &= \mathbb{P}[\text{compromise}|\text{no leak}] \qquad (D.107)\\ &= \mathbb{P}_{pwd}[\text{compromise}|\text{no leak}] * \mathbb{P}_{USB}[\text{compromise}|\text{no leak}] * \mathbb{P}_{cTLS}[\text{compromise}|\text{no leak}] \\ &= 0.0002445943347 \end{aligned}$$

Now, taking the values defined in Section D.1: $Lsystem = \$10^7$, $L_1 = \$166$, and $N = 500$ users, we calculate the expected loss due to compromise for the organization when the NIST 2017 Level 3 policy is in place.

$$\begin{aligned} &\mathbb{E}[\text{Loss}|\text{NIST 2003 L1 policy}] = \qquad (D.108)\\ &\Big(\mathbb{P}\left[\mathcal{N}\left(0.0002788605356, \frac{(0.0002788605356)(1-0.0002788605356)}{500}\right) > 0.5\right] \times \$10^7 \\ &+ (500)(0.0002788605356)(\$166)\Big) \times 0.0000754078177900894 \\ &+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.0002445943347, \frac{(0.0002445943347)(1-0.0002445943347)}{500}\right) > 0.5\right] \times \$10^7 \\ &+ (500)(0.0002445943347)(\$166)\Big) \times (1 - 0.0000754078177900894) \\ &= \$0.00180843073953175 + \$21.0335265280851 = \$21.0353349588246 \end{aligned}$$

D.6.2.3 Quantifying benefit of the NIST 2017 Level 3 policy

Taking the value for expected losses with no policy from Section D.4.2.3, our computation of the NIST 2017 L3 benefits is:

$$\begin{aligned}\mathbb{E}[\text{Benefits}] &= \mathbb{E}[\text{Loss}|\text{No policy}] - \mathbb{E}[\text{Loss}|\text{NIST 2017 L3 policy}] \quad \text{(D.109)}\\ &= \$105571.91 - \$21.04 = \$105550.87\end{aligned}$$

D.6.3 Costs: NIST 2017 Level 3 policy

In this section we will quantify the costs associated with the pieces of advice in the NIST 2017 policy. Because the advice relating to Memorized Secrets is the same as for the Level 1 policy we will not repeat the details of the calculations here. They are included in Table D.2 as $j = 1 \ldots 5$. The advice relating to Rate limiting, $j = 6$, Access controls, $j = 7$ & $j = 8$, are also the same as for NIST 2017 Level 1. Therefore, these will also not be re-discussed.

j=9: Client authenticated protected channel

- A verifier impersonation resistant authentication protocol SHALL establish an authenticated protected channel with the verifier.
  - One example of a verifier impersonation resistant authentication protocol is client authenticated TLS, because the client signs the authenticator output along with earlier messages from the protocol that are unique to the particular TLS connection being negotiated.

The organization must set up a system to allow Client authenticated TLS to be used by all of it's users. We suggest that this is a major implementation cost: 3 working weeks of the organisation's time. The organization will need to distribute a certificate to each of it's users. Options for this include physically handing certificates out, posting them, or allowing them to be downloaded from a website. Because we have decided the users in this company are external, we opt for the latter option. This can be done using open source certificate software. The user has the non trivial task of downloading and setting up the certificate. We estimate that on average 15 minutes help

desk time will be required for users to help them if they run into difficulties with this task. Major user education will also be required.

The user needs 30 minutes to set up the certificate when they create an account. They will also need to set up a new certificate every time it expires. We say that in our year long time frame a user will need to install a certificate twice.

It will take 3 seconds for the organization to generate a public private key pair for the user's certificate. It will also take approximately 0.1 seconds extra to authenticate each user at login.

If the user's computer is lost, broken or stolen they will need to authenticate themselves in another way and generate a new certificate. We estimate that this process would take an hour for the user and will require another 15 minutes help desk time. It will happen with the Verizon probability of a physical theft $P_{9,9} = \frac{39}{4788} = 0.081453634$.

An inconvenience of the client authentication is that it is linked to a device. Google has reported that 6 in 10 users switch between devices while online shopping [159]. This statistic could be applied to tell us that each user will be inconvenienced by this lack of portability with a probability of 0.6. We estimate the inconvenience as 10 second per login.

In Table D.4 we reference this information using a second set of rows as we have multiple inputs to some cost categories.

j=10: Reauthentication

- Periodic re-authentication of subscriber sessions SHALL be preformed after 12 hours or 15 minutes inactivity; SHALL use both authentication factors.

Komanduri's statistics [85] tell us that with this password policy a user will take 31.85 seconds to authenticate. We double this when a USB must be used as well.

If we estimate that a user moves away from their computer, for more than 15 minutes, three times per day. Then, given 261 working days in a year, each

user will need to authenticate 783 times in time frame T.

j=11: Single factor cryptographic device

- The single factor cryptographic device authenticators encapsulate one or more secret keys unique to the device that SHALL NOT be exportable (i.e., cannot be removed from the device).
- The challenge nonce SHALL be at least 64 bits in length, and SHALL either be unique over the authenticator's lifetime or statistically unique.
- Tamper detection and response should be in place for covers and doors [48].
- Single-factor cryptographic device authenticators SHOULD require a physical input (e.g., the pressing of a button) in order to operate.
- The secret key and its algorithm SHALL provide at least the minimum security length specified in the latest revision of SP 800-131A (112 bits as of the date of this publication).
- Use a cryptographic authenticator that requires the verifier store a public key corresponding to a private key held by the authenticator.
- A CSP SHOULD bind at least two physical authenticators (something you have) to the subscriber's credentials.
  - Binding of multiple authenticators is preferred in order to recover from the loss or theft of the subscriber's primary authenticator.

The different pieces of advice relating to the use of single factor cryptographic devices overlap largely with each other. Therefore we will discuss them all together. This means that in Table D.4 we will have multiple costs in each category under $j = 11$.

The type of single factor cryptographic device we are considering is a dedicated USB authenticator. We allow three working weeks of organization time to set up this method of authentication. This allows for finding a supplier, briefing customers and integrating the protocols into their software. We assign 15 minutes of the organization's time per user for distribution of the USBs. Also

if a user loses the device the organization will need to have an alternative method for identifying and authenticating the user and distribute a new USB to them. `Mozy.ie` in 2012 [107] found that 70% of people have lost a data storage device. They also found that the average person loses 1.24 items a year and that the more frequently you carry an item, the more likely you are to lose it. We will say that 2.5% of users will lose their USB device in the year long time frame. The probability that they lose the second physical authenticator bound to their account is: 0.01. So losing either of these will be is: $0.01 + 0.025 = 0.035$. This is the probability that the organization will need to send a replacement.

The user will need to wait approximately 15 working hours for their device to arrive when they create their account. If they lose their two physical authentication devices they will also need to wait for a replacement to arrive; probability is $0.01 \times 0.025 = 0.00025$. To demonstrate this the probability value is 1 for the certainty that at creation they will need to have the device sent to them, plus 0.00025 to represent the probability of being locked out while a replacement device is issued. $P_{11,11} = 1 + 0.00025$. At each login the user will need to take the time to press a button or insert the USB in order to authenticate. We assign an average of 31.85 seconds to this; the same amount of time as to enter the password. An inconvenience for the user is that it is now necessary to carry this authenticator around with them. Pixie Technology Inc. [131] found that 28% of people misplace their car keys at least once a week. Since car keys are another form of physical authenticator we can equate the likelihood of misplacing them to the likelihood of misplacing the USB device. So with $P_{11,11} = 0.28$, it will take 10 minutes for the user to find their USB, $C_{11,11} = C_{11}(10mins)$, $R_{11,11} = \#logins/\#weeks = \#logins/52$.

The organization will use additional computing power. For each login, they will need to create a statistically random 64 bit nonce, they will also then need to verify that the signature did originate from the user matching the saved public key. $C_{5,11} = C_5(3secs)$

We presume that the organization purchases the USB devices from a dedicated supplier before distributing them to their users. We estimate each device costing $6 and the organization will need to purchase two of them per user to satisfy the advice to distribute backup authenticators. If a user loses their

USB device, it will need to be replaced. We use the same probability of loss of 0.035% as before. We also assign \$0.50 per user for the cost of delivery each time a USB needs to be posted.

Administrators also assigned major user education and help desk costs to this advice.

D.6.3.1 Calculation of NIST 2017 Level 3 costs

Table D.4 shows the costs associated with the NIST 2017 Level 3 advice. Note that because $j = 1, \ldots, 8$ are included in Table D.2 under the NIST 2017 Level 1 advice, we do not repeat them here.

Figure D.4: Costs of implementing the NIST 2017 Level 3 password advice.

		Organisation costs					User costs						
		$C_1(t_o)$: Increased help desk/user support time	$C_2(t_o, t_u)$: User education required	$C_3(r_o)$: Additional organization resources	$C_4(t_o)$: Organization time taken to implement	$C_5(tc_o)$: Increased organization computing power needed	$C_6(t_u)$: Makes it more difficult to create a password	C_7: Increased risk of forgetting	$C_8(r_u)$: Additional user resources needed	$C_9(t_u)$: Need to pick a new password	$C_{10}(tc_u)$: Increased user computing power need	$C_{11}(t_u)$: User time and inconvenience	$C_{12}(t_u)$: User education time required
j=9: Client authenticated protected channel	*p(i,9)*	1	1		1	1					1	1	1
	C(i,9)	C_1(15 mins)	C_2(1hr)		C_4(3 weeks)	C_5(3 secs)					C_{10}(0.5 sec)	C_{11}(30mins)	C_{12}(15mins)
	R(i,9)	2*#users	12		1	2*#users					#logins	2*#users	12*#users
j=9 continued	*p(i,9)*	0.081453634				1						0.6	
	C(i,9)	C_1(15 mins)				C_5(0.5 sec)						C_{11}(10 secs)	
	R(i,9)	#users				#logins						#logins	
j=9 continued	*p(i,9)*											0.081453634	
	C(i,9)											C_9(1hr)	
	R(i,9)											#users	
j=10: Reauthentication	*p(i,10)*				1							1	
	C(i,10)				C_4(1 day)							C_{11}(63.7)	
	R(i,10)				1							783*#users	
j=11: Single factor cryptographic device	*p(i,11)*	1	1	1	1	1						1.00025	1
	C(i,11)	C_1(15mins)	C_2(1hr)	$12.50	C_4(3 weeks)	C_5(3 secs)						C_{11}(15hrs)	C_{12}(15mins)
	R(i,11)	4*#users	12	#users	1	#logins						#users	12*#users
j=11 continued	*p(i,11)*			0.035	1							0.28	
	C(i,11)			$6.50	C_4(15 mins)							C_{11}(10mins)	
	R(i,11)			#users	#users							#login/52	
j=11 continued	*p(i,11)*											1	
	C(i,11)											C_{11}(31.85 secs)	
	R(i,11)											#logins	

Using this table and the values defined in Section D.2 we have the following results for the cost of the NIST 2017 L3 policy:

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=11}} P_{c_{i,j}}.C_{i,j}.R_{i,j} = \$438{,}787.67 \tag{D.110}$$

D.6.4 Value of the NIST 2017 Level 3 policy

$$\mathbb{E}[Value] = \mathbb{E}[Benefits] - \mathbb{E}[Costs] \tag{D.111}$$

$$\mathbb{E}[Value] = \$105{,}550.87 - \$438{,}787.67 = -\$333{,}236.80 \tag{D.112}$$

D.7 NIST 2003 Level 4 Worked example

Level 4 is the highest authentication policy in the NIST 2003 standards. It requires authentication using a hard cryptographic token and a password.

The NIST 2017 Level 3 policy gave the option of using a password to "unlock" the hard cryptographic token and then authenticating with the verifier using the cryptographic key. Or using a password and also the cryptographic key to authenticate with the verifier. These are similar to the two authentication options described in NIST 2017 Level 4: A multi-factor cryptographic device, or a single factor cryptographic device used in conjunction with a memorized secret. To maintain consistency with the resources used in our NIST 2017 Level 3 analysis, we will assume both the hard cryptographic device and the password are communicated to the verifier during authentication.

The language used in the two standards documents differ, but we consider the "hard cryptographic token" in NIST 2003 to be identical to the "cryptographic devices" described in NIST 2017.

D.7.1 Policy summary

General authentication rules

- Remote registration is limited to Levels 1 through 3.

- At this level sensitive data transfers shall be cryptographically authenticated using keys bound to the authentication process.
  - Either public key or symmetric key technology may be used.
  - Example implementation: Client authenticated TLS.
- Re-authentication shall be required after not more than 24 hours from the initial authentication.
- Files of shared authentication secrets shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.

Authenticator 1: NIST 2003 Level 2 requirements for Memorized secrets

- Shared secret files shall not contain the plaintext passwords or secret; two alternative methods may be used to protect the shared secret:
  1. Passwords may be concatenated to a salt and/or username and then hashed with an Approved algorithm.
  2. Store shared secrets in encrypted form using Approved encryption algorithms and modes and decrypt the needed secret only when immediately required for authentication.
  - To contrast the NIST 2017 policy, we choose the encryption option.
- Password authentication systems can make targeted password guessing impractical by:
  - Requiring use of high-entropy passwords. NIST offers a number of examples to satisfy the Level 2 password requirements. The most familiar one to us is:
    - Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
    - Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
    - Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.
  - Limiting the number of unsuccessful authentication attempts, or by controlling the rate at which attempts can be carried out.

* Consider a system that required passwords to be changed every two years and limited trials by locking an account for 24 hours after 6 successive failed authentication attempts.

Authenticator 2: Hard token - a hardware device that contains a protected cryptographic key.

- Authentication is accomplished by proving possession of the device and control of the key. Hard tokens shall:
  - require the entry of a password or a biometric to activate the authentication key; or must also use a password in a secure authentication protocol, to establish two factor authentication.
  - Authentication keys may not be exportable.
  - Tamper detection and response should be in place for covers and doors [48].

D.7.2 Benefits: NIST 2003 Level 4 policy

The NIST 2003 Level 4 policy recommends two factor authentication and Client authentication TLS. Therefore, similar to the NIST 2017 Level 3 policy, the probability of successfully compromising a user is marked by the probability of compromising the single factor cryptographic device, the memorized secret and the client certificate and private key.

Therefore, we will require this computation again:

$$p_{pwd} = 1 - \prod_{a}(1 - \mathrm{P}_{pwd,a}) \tag{D.113}$$

$$p_{USB} = 1 - \prod_{a}(1 - \mathrm{P}_{USB,a}) \tag{D.114}$$

$$p_{cTLS} = 1 - \prod_{a}(1 - \mathrm{P}_{cTLS,a}) \tag{D.115}$$

where p is the probability of the $pwd/USB/cTLS$ being compromised. P_a is the probability of compromise due to each individual attack type a. As for all other policies, we must again calculate this p when there has been a leak of the password file and when there has not:$\mathbb{P}$[compromise|no leak] and $\mathbb{P}$[compromise|database leak]

D.7.2.1 Quantifying attacks: NIST 2003 Level 4

Assertion Manufacture or Modification The risk of this attack occurring is mitigated for all three factors by having access controls in place.

$$\begin{aligned} &\mathrm{P}_{pwd,cTLS,USB}[\text{assertion manufacture or modification}] \\ &= \left(\frac{(128)(0.72)}{4788}\right)(0.1) = 0.00192481203007519 \end{aligned} \tag{D.116}$$

Theft

- Re-authentication shall be required after not more than 24 hours from the initial authentication.

The password and the certificate and private key will be compromised if the computer is unlocked when it is stolen.

Given a user logs in each day, working a 5 day working week, there is a $\left(\frac{6}{7}\right) = 0.85714285714$ probability that the computer is unlocked when it is stolen.

$$\begin{aligned} &\mathrm{P}_{pwd,cTLS}[\text{theft}] \\ &= \mathrm{P}[\text{stolen}]*\mathrm{P}[\text{tampering}] + \mathrm{P}[\text{stolen}]*\mathrm{P}[\text{unlocked}] \\ &= \left(\frac{(39)(0.72)}{4788}\right)\left(\frac{27}{4788}\right) + \left(\frac{(39)(0.72)}{4788}\right)(0.85714285714) \\ &= 0.00505992424668823 \end{aligned} \tag{D.117}$$

The USB is susceptible to theft.

$$\mathrm{P}_{USB}[\text{theft}] = \frac{(39)(0.72)}{4788} = 0.00586466165413534 \tag{D.118}$$

Duplication

The Secret key on the USB is not exportable and we assume the issuer of the USBs is trustworthy.

$$\mathrm{P}_{USB}[\text{duplication}] = 0 \tag{D.119}$$

Password is only duplicable if revealed.

$$\begin{aligned} &\mathrm{P}_{pwd}[\text{duplication}] \\ &= \mathrm{P}[\text{divulge password}].\mathrm{P}[\text{partner exploits}] + \mathrm{P}[\text{recorded password compromised}] \\ &= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) = 0.00958217270195177 \end{aligned} \tag{D.120}$$

$$\begin{aligned} &\mathrm{P}_{cTLS}[\text{duplication}] \\ &= \mathrm{P}[\text{targeted attack}].\mathrm{P}[\text{private key} \cap \text{certificate leaked}] \\ &= 0.033896\left(\frac{(49)(0.72)}{4788}\right) = 0.000249759999999422 \end{aligned} \tag{D.121}$$

D.7.2.2 Eavesdropping

- At this level sensitive data transfers shall be cryptographically authenticated using keys bound to the authentication process

The communication of the password over the network is encrypted.

$$\begin{aligned} &\mathrm{P}_{pwd}[\text{eavesdropping}] = \mathrm{P}[\text{physical surveillance}] + \mathrm{P}[\text{keylogger breach}] \\ &= \left(\frac{(21)(0.72)}{4788}\right) + \left(\frac{(595)(0.72)}{4788}\right) = 0.0926315789473664 \end{aligned} \tag{D.122}$$

The USB cryptographic device and client authenticated TLS are not susceptible to eavesdropping attacks. They use public private key pairs and therefore the private key is never transmitted.

$$\mathrm{P}_{USB,cTLS}[\text{eavesdropping}] = 0 \tag{D.123}$$

D.7.2.3 Offline Guessing attacks

- Store shared secrets in encrypted form using approved encryption algorithms and modes and decrypt the needed secret only when immediately required for authentication.
- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;

- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.

First we have the probability of a leak:

$$\text{P[database of passwords is leaked]} = \frac{(49)(0.72)}{4788} = 0.00736842105263307 \quad \text{(D.124)}$$

Attackers can either steal the key with the dataset, or brute force guess the key.

Because of the complex password composition requirements we take Kelley et al.'s value for percentage of passwords cracked when they were made using their comprehensive8 password rules (22% of passwords guessed in 10^{13} guesses). This asked for passwords with "at least 8 characters including an uppercase and lowercase letter, a symbol, and a digit. It may not contain a dictionary word".

$$\begin{aligned} &P_{pwd}[\text{user password revealed}|\text{leak}] \quad \text{(D.125)} \\ &= \text{P[key leaked]} + \text{P[key brute forced]} + \text{P[user password in } 10^{13} \text{ guesses]} \\ &= \frac{49}{4788} + 0.01 + 0.22 = 0.240233918128655 \end{aligned}$$

The USB and client authenticated TLS public key stored on the organizations' system could be leaked and a brute force attack against the public keys to identify the private key is possible. However the probability of success is negligibly low for a well implemented asymmetric cryptographic algorithm.

$$P_{USB,cTLS}[\text{brute force guess private key}]=0 \quad \text{(D.126)}$$

Side Channel Attack

- No protection

Side channel attacks are possible against all three security methods; password, TLS and USB cryptographic device.

$$\mathrm{P}_{pwd}[\text{side channel attack}] = \mathrm{P}[\text{targeted attack}].\mathrm{P}[\text{not detected}].\mathrm{P}[\text{malware}] \tag{D.127}$$

$$= (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451251986$$

There is no stipulation in the NIST 2003 guidelines for protection against power or timing analysis attacks. Side channel attacks can be carried out either with physical access or via malware. We still assume that a side channel attack is going to be targeted.

$$\mathrm{P}_{USB}[\text{side channel attack}] \tag{D.128}$$

$$= \mathrm{P}[\text{targeted}].\left(\mathrm{P}[\text{physical access}] \cup \mathrm{P}[\text{malware}]\right)$$

$$= (0.033896)\left(\frac{(53)(0.72)}{3231} + \frac{(33)(0.72)}{3231}\right) = 0.000649594651809964$$

Encryption for TLS using the private key is susceptible to a side channel attack over the network [2]. Side channel attacks to determine the private key can also be done via physical access or malware.

$$\mathrm{P}_{cTLS}[\text{side channel attack}] \tag{D.129}$$

$$= \mathrm{P}[\text{targeted attack}].\left(\mathrm{P}[\text{physical access}] \cup \mathrm{P}[\text{malware}]\right)$$

$$\cup \left(\mathrm{P}[\text{eavesdropper(stat:LAN.access)}]\right)\left(\mathrm{P}[\text{exploit vulnerabilities}]\right)$$

$$= (0.033896)\left(\frac{(53)(0.72)}{3231} + \frac{(33)(0.72)}{3231}\right) + \left(\frac{(118)(0.72)}{3231}\right)\left(\frac{6}{4788}\right)$$

$$= 0.000682546111241588$$

Phishing or Pharming A password will only be able to be phished if the client authenticated TLS connection is compromised first.

$$P_{pwd}[\text{phishing} \cap \text{pharming}] = \tag{D.130}$$

$$\mathrm{P}[\text{phishing}] = \frac{(653)(0.72)}{4788} = 0.0981954887218051$$

$$P_{USB,cTLS}[\text{phishing} \cap \text{pharming}] = 0 \tag{D.131}$$

Social Engineering

$$P_{pwd}[\text{social engineering}] \tag{D.132}$$
$$= \text{P[pretexting]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429$$

We consider that a person masquerading as an IT assistant could relatively easily convince a user to send them the files which include the certificate and private key.

$$P_{cTLS}[\text{social engineering}] = \text{P[pretexting]\&P[not phone]} \tag{D.133}$$
$$= \left(\frac{(39)(0.72)}{4788}\right)\left(1 - \frac{5}{3231}\right) = 0.00585558604031134$$

As with NIST 2017 Level 3, we set the time a USB needs to be taken from the user for 10 minutes for someone with physical proximity to the user and 3 working days (1440 minutes) for a remote attacker.

$$P_{USB}[\text{social engineering}] \tag{D.134}$$
$$= \left(\frac{\text{P[pretexting]}}{10}\right)(\text{P[partner exploits]}) + \left(\frac{\text{P[pretexting]}}{1440}\right)(\text{P[remote attacker]}) \tag{D.135}$$
$$= \left(\frac{(39)(0.72)}{(4788)(10)}\right)\left(\frac{108}{4788}\right) + \left(\frac{(39)(0.72)}{(4788)(1440)}\right)\left(1 - \frac{108}{4788}\right) = 0.000017209376827148 \tag{D.136}$$

Online Guessing attacks

- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.
- Limit trials by locking an account for 24 hours after 6 successive failed authentication attempts.

An attacker can make 6 guesses every 24 hours. Therefore in a year, the attacker can make 2190 guesses against each valid account in the year time frame. According to Kelley et al., because of the complex password compo-

sition restrictions none of the users' passwords are successfully cracked with this number of guesses [83].

$$P_{pwd}, USB, cTLS[\text{online guessing successful}] = 0 \tag{D.137}$$

Endpoint compromise

$$P_{pwd}[\text{endpoint compromise}] = \frac{(678)(0.72)}{4788} = 0.101954887218044 \tag{D.138}$$

Because the private key never leaves the USB device, a compromised endpoint should not be able to compromise the device.

$$P_{USB}[\text{endpoint compromise}] = 0 \tag{D.139}$$

A compromised endpoint could compromise the private key and certificate.

$$P_{cTLS}[\text{endpoint compromise}] = 0.101954887218044 \tag{D.140}$$

$$\tag{D.141}$$

Unauthorized binding The unauthorized binding of the authenticator to an attacker could take the form of an unauthorized password reset. In order for an attacker to successful reset a subscribers password they would need to already have access to their email account and be interested in targeting that individual since it is a time consuming attack.

$$P_{pwd}[\text{unauthorized binding}] = \text{P[use stolen credentials].P[targeted attack]} \tag{D.142}$$

$$= \left(\frac{(631)(0.72)}{4788}\right)(0.033896) = 0.00321629714285458 \tag{D.143}$$

To compromise the USB or client TLS the attacker would need access to the database and need to have a USB, and certificate and private key created. This would need to be a targeted attack.

$$P_{USB,cTLS}[\text{unauthorized binding}] = \left(\frac{(49)(0.72)}{4788}\right)(0.1)(0.033896)$$
$$= 0.0000249759999994224$$

D.7.2.4 Calculation of NIST 2003 Level 4 losses

We quantify benefits by splitting the equation to look at the scenario when the password dataset has been leaked and when it has not (Equation D.16). For the NIST 2003 Level 3 policy we have the following probabilities of compromise:

$$\mathbb{P}_{pwd}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{pwd,a}) = 0.4559828624 \tag{D.144}$$

$$\mathbb{P}_{USB}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{USB,a}) = 0.0084645569 \tag{D.145}$$

$$\mathbb{P}_{cTLS}[\text{compromise}|\text{leak}] = 1 - \prod_a (1 - \mathrm{P}_{cTLS,a}) = 0.1142891665 \tag{D.146}$$

$$\mathbb{P}_{pwd}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{pwd,a}) = 0.2839675914 \tag{D.147}$$

$$\mathbb{P}_{USB}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{USB,a}) = 0.0084645569 \tag{D.148}$$

$$\mathbb{P}_{cTLS}[\text{compromise}|\text{no leak}] = 1 - \prod_a (1 - \mathrm{P}_{cTLS,a}) = 0.1142891665 \tag{D.149}$$

$$\begin{aligned} p_l &= \mathbb{P}[\text{compromise}|\text{leak}] \\ &= \mathbb{P}_{pwd}[\text{compromise}|\text{leak}] * \mathbb{P}_{USB}[\text{compromise}|\text{leak}] * \mathbb{P}_{cTLS}[\text{compromise}|\text{leak}] \\ &= 0.0004411210827 \end{aligned} \tag{D.150}$$

$$\begin{aligned} p_{l'} &= \mathbb{P}[\text{compromise}|\text{no leak}] \\ &= \mathbb{P}_{pwd}[\text{compromise}|\text{no leak}] * \mathbb{P}_{USB}[\text{compromise}|\text{no leak}] * \mathbb{P}_{cTLS}[\text{compromise}|\text{no leak}] \\ &= 0.0002747122791 \end{aligned} \tag{D.151}$$

Now, taking the values defined in Section D.1: $Lsystem = \$10^7$, $L_1 = \$166$, and $N = 500$ users, we calculate the expected loss due to compromise for the organization when the NIST 2017 Level 3 policy is in place.

$$\begin{aligned}
&\mathbb{E}[\text{Loss}|\text{NIST 2003 L1 policy}] = \qquad (D.152)\\
&\Big(\mathbb{P}\left[\mathcal{N}\left(0.00044112108, \frac{(0.00044112108)(1-0.00044112108)}{500}\right) > 0.5\right] \times \$10^7\\
&+ (500)(0.00044112108)(\$166)\Big) \times 0.00736842105263307\\
&+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.0002747122791, \frac{(0.0002747122791)(1-0.0002747122791)}{500}\right) > 0.5\right] \times \$10^7\\
&+ (500)(0.0002747122791)(\$166)\Big) \times (1-0.00736842105263307)\\
&= \$0.279531466191297 + \$23.4511752646053 = \$23.7307067307966
\end{aligned}$$

D.7.2.5 Quantifying benefit of the NIST 2003 Level 4 policy

Taking the value for expected losses with no policy from Section D.4.2.3, our computation of the NIST 2017 L3 benefits is:

$$\begin{aligned}
\mathbb{E}[\text{Benefits}] &= \mathbb{E}[\text{Loss}|\text{No policy}] - \mathbb{E}[\text{Loss}|\text{NIST 2017 L3 policy}] \qquad (D.153)\\
&= \$20993.78 - \$0.18 = \$20993.60
\end{aligned}$$

D.7.3 Costs: NIST 2003 Level 4 policy

In this section we will quantify the cost associated with the pieces of advice in the NIST 2003 Level 4 policy.

j=1: In-person registration

- Remote registration is limited to Levels 1 through 3.

At creation, a user must bring two IDs and proof of address to the organization in person. We assign 15 minutes per user for the organization help desk and 6 hours for the user. We also assign organisation resources for the storage of this information as 20 cent per user.

j=2: Client authenticated protected channel

- At this level sensitive data transfers shall be cryptographically authenticated using keys bound to the authentication process.

- Either public key or symmetric key technology may be used.
- Example implementation: Client authenticated TLS.

This has very similar costs to NIST 2017 Level 3. The difference is that the user can be handed a physical certificate when they register in person. So this will not be an addition cost at creation for the user. However certificates expire so the user will need to return to the organization to update it or update it themselves at least once a year. We assume the update is done remotely by the users. We assign 30 minutes for an average user to update their certificate and 15 minutes organization help desk time to guide the user through the process.

The organization of course also has the time it will take to set up the client authenticated procedures in the first place, $C_{4,2} = C_4(43200)$

If the user loses their computer/laptop, P[computer lost] $= \frac{39}{4788} = 0.081453634$, they will need to return to the organization to get a new certificate. This will take 6 hours of the user's time and another 15 minutes of the organization's time.

It will take 3 seconds for the organization to generate a public private key pair for the user's certificate; at creation and at expiry $P_{5,2} = 1$, $R_{5,2} = 2{*}\#users$.It will also take approximately 0.1 seconds extra to authenticate each user at login. As before, there is a cost for the user associated with needing to always use the same device at login.

j=3: Reauthentication

- Reauthentication shall be required after not more than 24 hours from the initial authentication.

Komanduri's statistics [85] tells us that with the comprehensive8 password policy a user will take 104.86 seconds to authenticate. We add on 31.85 seconds when a USB must be used as well. A user will need to authenticate each working day during the year.

j=4: Access to password file

- Files of shared secrets used by verifiers shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.

This advice incurs minor help desk costs periodically, a small amount of user education periodically and it is a minor cost to implement.

j=5: Password storage

- Shared secret files shall not contain the plaintext passwords or secret
  - Store shared secrets in encrypted form using Approved encryption algorithms and modes and decrypt the needed secret only when immediately required for authentication.

We have assigned 1 day of organizations time to setting up a system which will encrypt the passwords. There is an additional 1 second of computing time needed for the organization to decrypt and re-encrypt the password at each user login.

j=6: Composition

- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.

Setting up the composition rules is a minor implementation cost to the orgnaisation. Kelley et al. recorded that users will take 242.56 seconds to create passwords of this type [85].

Kelley et al. also provide statistics that allowed us the calculate the probability of forgetting this type of password as: $P_{7,5} = 0.02834645669$. The 'Probability'

that this advice makes it more difficult to create a password can also be taken from these statistics '$P'_{7,6} = 2.12$.

We assign 1 hour every 6 months to the organisation for user education and similarly on the User side, we assign 15 minutes every 6 months for the users. The help desk cost of a user forgetting their password is covered under the cost of forgetting function.

j=7: Rate limiting

- Limit trials by locking an account for 24 hours after 6 successive failed authentication attempts.

Brostoff and Sasse [18] show that 31% of their study participants failed to login successfully within 3 attempts whereas only 7% failed within 10 attempts.

If we assume there is a linear relationship between the number of attempts allowed and the failure rate then we can extrapolate that 6 attempts will result in a failure rate of 21%.

Brostoff and Sasse also found that approximately 7% of these failed logins led to password reminder requests: $P_{(7,1)} = 0.21 * 0.07 = 0.0147$

We calculate the probability and cost of an attacker targeting a user to lock their account the same as we did for the NIST 2003 Level 1 policy.

$$\begin{aligned} \text{P[malicious lockout]} &= \text{P[targeted.attack]}\,(\text{P[brute force.attacker]} + \text{P[partner]}) \\ &= (0.033896)\left(\frac{(73)(0.72)}{4788} + \frac{(108)(0.72)}{3231}\right) = 0.00118786145 \end{aligned} \tag{D.154}$$

Therefore, the total probability that the user's account is locked for 24 hours is: $0.21 + 0.00118786145 = 0.21118786145$.

There is no stipulation in the advice that an administrator can unlock the account.

j=8: Password expiry

- Requires passwords to be changed every two years

This means that within our time frame approximately half the users' passwords will expire. It is a minor cost for organisations to implement. It involved major user education and major help desk time. The help desk time will be linked to users not changing passwords on-time and becoming locked out of systems and difficulties involved with password changes. This is in additional to the inevitable help desk time needed when users forget their new password which is covered under the cost *Increased risk of forgetting*. The additional help desk time will occur with each password expiry deadline within timeframe T.

With each password expiration, a user is unable to do their work until the have reset their password. A 2014 study by NIST [26] estimated that, they can spend up to 12.4 hours per year changing passwords on a 90-day expiration schedule, or 18.6 hours changing passwords on a 60-day expiration schedule. If we take these statistics and assume they are linear, then for expiry every two year, this will take 1hr 33 minutes out of a work day for 9 accounts. If we assume this policy applies to just one account rather than nine, then this tells us that it can take 10 minutes in our time frame T for users to reset their password.

The other inconvenience for the user is the repetition of the password creation policy and the increased risk of forgetting a newly created passwords. In this way we take the user costs identified as part of the composition requirement and reapply them here.

j=9: Single factor cryptographic device

- Require the entry of a password or a biometric to activate the authentication key; or must also use a password in a secure authentication protocol, to establish two factor authentication.
- Authentication keys may not be exportable.
- Tamper detection and response should be in place for covers and doors [48].

The type of single factor cryptographic device we are considering is a dedicated USB authenticator. We allow 3 weeks for the organization to set up the method of authentication. This allows for finding a supplier, briefing customers and integrating the protocols into their software. Users may collect the USBs at registration.

Also if a user loses the device ($P = 0.025$) they need to return to the organization, identify themselves and receive a new authenticator. We set this as 15 minutes of the organizations time and 6 hours of the user's time.

At each login the user will need to take the time to press a button or insert the USB in order to authenticate. We assign 31.85 seconds to this; the same amount of time as to enter a basic password. An inconvenience for the user is that it is now necessary to carry this authenticator around with them. Pixie Technology Inc. [131] found that 28% of their study participants admitted to misplacing their carkeys at least once a week. We equate this to the likelihood of misplacing the USB device. So with $P_{11,9} = 0.28$, $C_{11,9} = C_2(10mins)$, $R_{11,9} = \#logins/\#weeks = \#logins/52$.

The organization will use additional computing power. Though there is no requirement for the security length of the nonce or keys $C_{5,9} = C_5(2)$

A USB device will need to be purchased for each user at creation and also if the user loses the device. We estimate each device costing $6.

Administrators also assigned major user education and help desk costs to this advice.

D.7.3.1 Calculation of NIST 2003 Level 4 costs

Table D.5 shows the costs associated with the NIST 2003 Level 4 advice.

Figure D.5: Costs of implementing the NIST 2003 Level 4 password advice.

		Organisation costs					User costs						
		$C_1(t_o)$: Increased help desk/user support time	$C_2(t_o, t_u)$: User education required	$C_3(r_o)$: Additional organization resources	$C_4(t_o)$: Organization time taken to implement	$C_5(tc_o)$: Increased organization computing power needed	$C_6(t_u)$: Makes it more difficult to create a password	C_7: Increased risk of forgetting	$C_8(r_u)$: Additional user resources needed	$C_9(t_u)$: Need to pick a new password	$C_{10}(tc_u)$: Increased user computing power need	$C_{11}(t_u)$: User time and inconvenience	$C_{12}(t_u)$: User education time required
j=1: In-person registration	*p(i,1)*	1		1								1	
	C(i,1)	C_1(15mins)		C_3($0.20)								C_{11}(6hrs)	
	R(i,1)	#users		#users								#users	
j=2: Client authenticated protected channel	*p(i,9)*	1	1		1	1					1	1	1
	C(i,9)	C_1(15 mins)	C_2(1hr)		C_4(3 weeks)	C_5(3 secs)					C_{10}(0.5 sec)	C_{11}(30mins)	C_{12}(15mins)
	R(i,9)	#users	12		1	2*#users					#logins	#users	12*#users
j=2 continued	*p(i,9)*	0.081453634				1						0.6	
	C(i,9)	C_1(15 mins)				C_5(0.5 sec)						C_{11}(10 secs)	
	R(i,9)	#users				#logins						#logins	
j=2 continued	*p(i,9)*											0.081453634	
	C(i,9)											C_9(1hr)	
	R(i,9)											#users	
j=3: Reauthentication	*p(i,3)*				1							1	
	C(i,3)				C_4(1 day)							C_{11}(136.71 secs)	
	R(i,3)				1							261*(#users)	
j=4: Access to password files	*p(i,4)*	1	1		1								
	C(i,4)	C_1(15mins)	C_2(1hr)		C_4(1 day)								
	R(i,4)	#users	2		1								
j=5: Password storage	*p(i,5)*				1	1							
	C(i,5)				C_4(1 day)	C_5(1 sec)							
	R(i,5)				1	#logins							
j=6: Composition	*p(i,6)*		1		1		2.12	0.0283465		1			1
	C(i,6)		C_2(1hr)		C_4(1 day)		C_6(242.56 secs)	C_7		C_9(242.56 secs)			C_{12}(15 mins)
	R(i,6)		2		1		#users	#users		#users			2*#users
j=7: Rate limiting	*p(i,7)*		1		1			0.0147				0.21	1
	C(i,7)		C_2(1hr)		C_4(1 day)			C_7				C_{11}(24 hours)	C_{12}(15mins)
	R(i,7)		2		1			#users				#logins	2*#users
j=8: Password expiry	*p(i,8)*	1	1		1		2.12	0.0283465		1		1	1
	C(i,8)	C_1(15mins)	C_2(1hr)		C_4(1 day)		C_6(242.56 secs)	C_7		C_9(242.56 secs)		C_{11}(10mins)	C_{12}(15mins)
	R(i,8)	#users	12		1		0.5*#users	0.5*#users		0.5*#users		#users	12*#users
j=9: Single factor cryptographic device	*p(i,9)*	0.025	1	1	1	1						0.025	1
	C(i,9)	C_1(15mins)	C_2(1hr)	$6	C_4(3 weeks)	C_5(2 secs)						C_{11}(6hrs)	C_{12}(15mins)
	R(i,9)	#users	12	#users	1	#logins						#users	12*#users
j=9 continued	*p(i,9)*	1		0.025								1	
	C(i,9)	C_1(15mins)		$6								C_{11}(31.85 secs)	
	R(i,9)	4*#users		#users								#logins	
j=9 continued	*p(i,10)*											0.28	
	C(i,10)											C_{11}(10mins)	
	R(i,10)											#login/52	

Using this table and the values defined in Section D.2 we have the following results for the cost of the NIST 2003 L4 policy:

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=9}} P_{c_{i,j}}.C_{i,j}.R_{i,j} = \$12{,}155{,}278.21 \tag{D.155}$$

D.7.4 Value of the NIST 2003 Level 4 policy

$$\mathbb{E}[Value] = \mathbb{E}[Benefits] - \mathbb{E}[Costs] \tag{D.156}$$

$$\mathbb{E}[Value] = \$105{,}548.18 - \$12{,}155{,}278.21 = -\$12{,}049{,}730.03 \tag{D.157}$$

D.8 NIST 2007 policy Worked example

The 2003 NIST policy is the beginning of the password security craze which introduced complex composition restrictions and short validity periods for user-chosen passwords.

The 2003 Level 2 guidelines contain recommendations that will be familiar to most web users:

- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.

However, regarding the expiration of the password, it simply recommends that the password is changed every two years [20]. It is in the 2007 NIST "Guidelines on Securing Public Web Servers" and the 2007 NIST "Guidelines on Electronic Mail Security " that we see the first NIST documentation specifying very short password validity periods [121, 122], though it was used in practice much earlier [151, 125].

Both NIST documents give the following advice:

- Administrator or root level passwords should be changed every 30 to 120 days. User level passwords should also be changed periodically, with the period determined by the enforced length and complexity of the password combined with the sensitivity of the information protected. When considering the appropriate aging duration, the exposure level of user passwords should also be taken into account

This advice was rapidly taken on board by most organizations and was generally applied to the passwords of anyone in the organisation [49, 82].

In this section we will show the workings for the costs and benefits associated with this typical authentication policy. We will take the same general authentication rules as for the NIST 2003 Level 2 policy with the addition of the strict expiration requirement.

D.8.1 Policy summary

General authentication rules

- Files of shared authentication secrets shall be protected by discretionary access controls that limit access to administrators and only those applications that require access.
- Approved cryptography is required to prevent eavesdroppers.

Authenticator: Memorized secrets

- Shared secret files shall not contain the plaintext passwords or secret; two alternative methods may be used to protect the shared secret:
  1. Passwords may be concatenated to a salt and/or username and then hashed with an Approved algorithm.
  2. Store shared secrets in encrypted form using approved encryption algorithms and modes and decrypt the needed secret only when immediately required for authentication.
  - To contrast the NIST 2017 policy, we choose the encryption option.

- Password authentication systems can make targeted password guessing impractical by:
  - Requiring use of high-entropy passwords. NIST offers a number of examples to satisfy the Level 2 password requirements. The most familiar one to us is:
    * Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
    * Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
    * Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.
  - Limiting the number of unsuccessful authentication attempts, or by controlling the rate at which attempts can be carried out.
    * Limit trials by locking an account for 24 hours after 6 successive failed authentication attempts.
- Ageing - **Passwords should be changed every 90 days.**
  - If possible, ensure that users cannot change their password by merely appending characters to the beginning or end of their original password (e.g., original password was "mysecret" and is changed to "1mysecret" or "mysecret1").

D.8.2 Benefits: NIST 2007 policy

D.8.2.1 Quantifying attacks: NIST 2007

Assertion Manufacture or Modification The risk of this attack occurring is mitigated for all three factors by having access controls in place.

$$\begin{aligned} &\text{P[assertion manufacture or modification]} \\ &= \left(\frac{(128)(0.72)}{4788}\right)(0.1) = 0.00192481203007365 \end{aligned} \tag{D.158}$$

Theft

- Re-authentication shall be required after not more than 24 hours from the initial authentication.

The password and the certificate and private key will be compromised if the computer is unlocked when it is stolen.

Given a user logs in each day, working a 5 day working week, there is a $\left(\frac{6}{7}\right) = 0.85714285714$ probability that the computer is unlocked when it is stolen.

$$\begin{aligned} &\text{P[theft]} \\ &= \text{P[stolen]}*\text{P[tampering]} + \text{P[stolen]}*\text{P[unlocked]} \\ &= \left(\frac{(39)(0.72)}{4788}\right)\left(\frac{27}{4788}\right) + \left(\frac{(39)(0.72)}{4788}\right)(0.85714285714) \\ &= 0.00505992424668823 \end{aligned} \tag{D.159}$$

Duplication

Password is only duplicable if revealed.

$$\begin{aligned} &\text{P[duplication]} = \\ &\text{P[divulge password].P[partner exploits]} + \text{P[recorded password compromised]} \\ &= (0.25)\left(\frac{(108)(0.72)}{3231}\right) + \left(\frac{(16)(0.72)}{3231}\right) = 0.00958217270195177 \end{aligned} \tag{D.160}$$

Eavesdropping

- Approved cryptography is required to prevent eavesdroppers.

The communication of the password over the network is encrypted.

$$\begin{aligned} &\text{P[eavesdropping]} = \text{P[physical surveillance]} + \text{P[keylogger breach]} \\ &= \left(\frac{(21)(0.72)}{4788}\right) + \left(\frac{(595)(0.72)}{4788}\right) = 0.0926315789473664 \end{aligned} \tag{D.161}$$

Offline Guessing attacks

- Store shared secrets in encrypted form using approved encryption algorithms and modes and decrypt the needed secret only when immediately

required for authentication.

- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.
- Passwords should be changed every 90 days.

First we have the probability of a leak:

$$\text{P[database of passwords is leaked]} = \frac{(49)(0.72)}{4788} = 0.00736842105258628 \quad \text{(D.162)}$$

Attackers can either steal the key with the dataset, or brute force guess the key.

In all policies, we have assumed an attacker will make 10^{13} guesses against a users' password. Each guess takes an attacker approximately 10^{-11} seconds. Therefore, with no slow hash function in place, it will take an attacker guessing offline less than 2 minutes to make $10^{1}3$ guesses. Therefore, inline with current literature, we conclude that for offline attacks there is little to no security advantage of password expiry [24].

Because of the complex password composition requirements we take Kelley et al.'s value for percentage of passwords cracked when they were made using their comprehensive8 password rules (22% of passwords guessed in 10^{13} guesses). This asked for passwords with "at least 8 characters including an uppercase and lowercase letter, a symbol, and a digit. It may not contain a dictionary word".

$$\begin{aligned} &\text{P[user password revealed|leak]} \quad \text{(D.163)} \\ &= \text{P[key leaked]} + \text{P[key brute forced]} + \text{P[user password in } 10^{13} \text{ guesses]} \\ &= \frac{49}{4788} + 0.01 + 0.22 = 0.240233918128655 \end{aligned}$$

Side Channel Attack No protection.

As we discussed in NIST 2017 Level 1, there are side channel attacks which exist for compromising passwords.

$$\begin{aligned} \text{P[side channel attack]} =& \text{P[targeted attack].P[not detected].P[malware]} \\ =& (0.033896)(0.8)\left(\frac{(33)(0.72)}{3231}\right) = 0.000199410451251986 \end{aligned} \tag{D.164}$$

Phishing or Pharming No protection.

$$\text{P[phishing} \cup \text{pharming]} = \text{P[phishing]} = \frac{(653)(0.72)}{4788} = 0.0981954887218051 \tag{D.165}$$

Social Engineering No protection

$$\text{P[social engineering]} = \text{P[pretexting]} = \frac{(39)(0.72)}{4788} = 0.00586466165413429 \tag{D.166}$$

Online Guessing attacks

- Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,
- Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and;
- Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.
- Limit trials by locking an account for 24 hours after 6 successive failed authentication attempts.
- Passwords should be changed every 90 days.

An attacker can make 6 guesses every 24 hours. Therefore in a year, the attacker can make 2190 guesses against each valid account in the year time frame. According to Kelley et al., because of the complex password composition restrictions none of the users' passwords are successfully cracked with this number of guesses [83].

$$\text{P[online guessing successful]} = 0 \tag{D.167}$$

Endpoint Compromise No protection

$$\begin{aligned}&\text{P[endpoint compromise]} \\ &= \text{P[breach using backdoor or C2]} = \frac{(678)(0.72)}{4788} = 0.101954887218044\end{aligned} \tag{D.168}$$

Unauthorized Binding No protection

$$\begin{aligned}&\text{P[unauthorized binding]} = \\ &\Big(\text{P[use stolen credentials]} \cup \text{P[physical theft]}\Big)\text{P[targeted attack]} \\ &= \left(\frac{(631)(0.72)}{4788} + \frac{(39)(0.72)}{4788}\right)(0.033896) = 0.00341508571428736\end{aligned} \tag{D.169}$$

D.8.2.2 Calculation of NIST 2007 losses

We quantify benefits by splitting the equation to look at the scenario when the password dataset has been leaked and when it has not (Equation D.16). For the NIST 2007 policy this gives use the following probabilities of compromise:

$$p_l = \mathbb{P}[\text{compromise}|\text{leak}] = 0.456091355733257 \tag{D.170}$$

$$p_{l'} = \mathbb{P}[\text{compromise}|\text{no leak}] = 0.284110389704333 \tag{D.171}$$

Now, taking the values defined in Section D.1: $Lsystem = \$10^7$, $L_1 = \$166$, and $N = 500$ users. We calculate the expected loss due to compromise for the organization when the NIST 2007 policy is in place.

$$\begin{aligned}
&\mathbb{E}[\text{Loss}|\text{NIST 2007 policy}] = \qquad (D.172)\\
&\Big(\mathbb{P}\left[\mathcal{N}\left(0.4560913557, \frac{(0.4560913557)(1-0.4560913557)}{500}\right) > 0.5\right] \times \$10^7\\
&+ (500)(0.4560913557)(\$166)\Big) \times 0.00736842105258628\\
&+ \Big(\mathbb{P}\left[\mathcal{N}\left(0.2841103897, \frac{(0.2841103897)(1-0.2841103897)}{500}\right) > 0.5\right] \times \$10^7\\
&+ (500)(0.2841103897)(\$166)\Big) \times (1 - 0.00736842105258628)\\
&= \$2068.25138396931 + \$24253.4581390645 = \$26321.7095230338
\end{aligned}$$

D.8.2.3 Quantifying benefit of the NIST 2007 policy

Taking the value for expected losses with no policy from Section D.4.2.3, our computation of the NIST 2007 benefits is:

$$\begin{aligned}
\mathbb{E}[\text{Benefits}] &= \mathbb{E}[\text{Loss}|\text{No policy}] - \mathbb{E}[\text{Loss}|\text{NIST 2003 L1 policy}] \qquad (D.173)\\
&= \$105571.91 - \$26321.71 = \$79250.20
\end{aligned}$$

D.8.3 Costs: NIST 2007 policy

In this section we will quantify the costs associated with each piece of advice in the NIST 2007 policy.

j=1: Access to password files should be restricted Same as for other policies this incurs help desk, implementation and user education costs.

j=2: Authenticated protected channel

- Approved cryptography is required to prevent eavesdroppers.

This advice requires sending passwords over a protected channel. This should be set up by the organisation as part of the authentication platform. Administrators marked this as a minor task.

j=3: Password storage

- Shared secret files shall not contain the plaintext passwords or secret; two alternative methods may be used to protect the shared secret:

  1. Passwords may be concatenated to a salt and/or username and then hashed with an Approved algorithm.
  2. Store shared secrets in encrypted form using approved encryption algorithms and modes and decrypt the needed secret only when immediately required for authentication.

  – To contrast the NIST 2017 policy, we choose the encryption option.

Administrators assigned a minor organisation cost to setting up a system which will encrypt the passwords. There is an additional 1 second of computing time needed for the organization to decrypt and re-encrypt the password at each user login.

j=4: Composition

- Requiring use of high-entropy passwords. NIST offers a number of examples to satisfy the Level 2 password requirements. The most familiar one to us is:

  – Require a minimum of 8 character passwords, selected by subscribers from an alphabet of 94 printable characters,

  – Require subscribers to include at least one upper case letter, one lower case letter, one number and one special character, and,

  – Use a dictionary to prevent subscribers from including common words and prevented permutations of the username as a password.

The costs for this are the same as for the NIST 2003 Level 4 composition rules.

j=5: Rate limiting

- Limit trials by locking an account for 24 hours after 6 successive failed authentication attempts.

Same as for NIST 2003 Level 4 rate limiting costs.

j=6: Password expiry

- Passwords should be changed every 90 days.
  - If possible, ensure that users cannot change their password by merely appending characters to the beginning or end of their original password(e.g., original password was “mysecret” and is changed to “1mysecret” or “mysecret1”).

This means that with our time frame of one year users will need to change their password approximately four times. We assign minor organization time to set up the expiry system.

The inconvenience for the user lies in the repetition of the password creation policy and the increased risk of forgetting a newly crafted passwords. In this way we take the user costs identified as part of the composition requirement and reapply them here.

Choong et al.’s 2014 NIST study [26] estimated that, users can spend up to 12.4 hours per year changing passwords on a 90-day expiration schedule. This was for 9 accounts. For one account, this equates to 1.38 hours a year.

D.8.3.1 Calculation of NIST 2007 costs

Table D.6 shows the costs associated with pieces of advice in NIST 2007.

Figure D.6: Costs of implementing the NIST 2007 password advice.

		Organisation costs					User costs						
		$C_1(t_o)$: Increased help desk/user support time	$C_2(t_o, t_u)$: User education required	$C_3(r_o)$: Additional organization resources	$C_4(t_o)$: Organization time taken to implement	$C_5(tc_o)$: Increased organization computing power needed	$C_6(t_u)$: Makes it more difficult to create a password	C_7: Increased risk of forgetting	$C_8(r_u)$: Additional user resources needed	$C_9(t_u)$: Need to pick a new password	$C_{10}(tc_u)$: Increased user computing power need	$C_{11}(t_u)$: User time and inconvenience	$C_{12}(t_u)$: User education time required
j=1: Access to password files	*p(i,1)*	1	1		1								1
	C(i,1)	C_1(15mins)	C_2(1hr)		C_4(1 day)								C_{12}(15mins)
	R(i,1)	#users	2		1								2*#users
j=2: Authenticated protected channel	*p(i,2)*				1								
	C(i,2)				C_4(1 day)								
	R(i,2)				1								
j=3: Password storage	*p(i,3)*				1	1							1
	C(i,3)				C_4(1 day)	C_5(1 sec)							C_{12}(15mins)
	R(i,3)				1	#logins							2*#users
j=4: Composition	*p(i,4)*		1		1		2.12	0.0283465		1			1
	C(i,4)		C_2(1hr)		C_4(1 day)		C_6(242.56 secs)	C_7		C_9(242.56 secs)			C_{12}(15 mins)
	R(i,4)		2		1		#users	#users		#users			2*#users
j=5: Rate limiting	*p(i,5)*		1		1			0.0147				0.21	1
	C(i,5)		C_2(1hr)		C_4(1 day)			C_7				C_{11}(24 hrs)	C_{12}(15mins)
	R(i,5)		2		1			#users				#logins	2*#users
j=6: Password expiry	*p(i,6)*	1	1		1		2.12	0.0283465		1		1	1
	C(i,6)	C_1(15mins)	C_2(1hr)		C_4(1 day)		C_6(242.56 secs)	C_7		C_9(242.56 secs)		C_{11}(1.38hrs)	C_{12}(15mins)
	R(i,6)	#users	12		1		4*#users	4*#users		4*#users		#users	12*#users

Using this table and the values defined in Section Section D.2 we have the following results for the cost of the NIST 2007 policy:

$$\mathbb{E}[Costs] = \sum_{\substack{i=12 \\ j=6}} P_{c_{i,j}}.C_{i,j}.R_{i,j} = \$11{,}931{,}636.61 \tag{D.174}$$

D.8.4 Value of the NIST 2007 policy

$$\mathbb{E}[Value] = \mathbb{E}[Benefits] - \mathbb{E}[Costs] \tag{D.175}$$

$$\mathbb{E}[Value] = \$79{,}250.20 - \$11{,}931{,}636.61 = -\$11{,}852{,}386.41 \tag{D.176}$$

Bibliography

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999. 12, 16, 32, 33, 47, 156, 157

[2] Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of tls. Cryptology ePrint Archive, Report 2015/1129, 2015. `https://eprint.iacr.org/2015/1129`, Accessed: 2019-01-31. 303, 319

[3] Mashael AlSabah, Gabriele Oligeri, and Ryan Riley. Your culture is in your password: An analysis of a demographically-diverse password dataset. *Computers & security*, 77:427–441, 2018. 142

[4] Kemal Altinkemer and Tawei Wang. Cost and benefit analysis of authentication systems. *Decision Support Systems*, 51(3):394–404, 2011. 160

[5] APWG. Phishing activity trends report 1st quarter 2018. Technical report, APWG, 2018. `https://docs.apwg.org/reports/apwg_trends_report_q1_2018.pdf`, Accessed=2019-10-31. 57, 71

[6] Simon Arnell, Adam Beautement, Philip Inglesant, Brian Monahan, David Pym, and Angela Sasse. Systematic decision making in security management modelling password usage and support. In *International Workshop on Quantitative Aspects in Security Assurance. Pisa, Italy.* Citeseer, 2012. 16, 55, 160

[7] Sacha B. Let them paste passwords. `https://www.ncsc.gov.uk/blog-post/let-them-paste-passwords`, 2017. Accessed: 2017-01-20. 248

[8] Claudine Beaumont. Microsoft hotmail leak blamed on phishing attack, 2009. `https://www.telegraph.co.uk/technology/microsoft/6264539/Microsoft-Hotmail-leak-blamed-on-phishing-attack.html`, Accessed: 2020-03-22. 87

[9] Adam Beautement, M Angela Sasse, and Mike Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, pages 47–58. ACM, 2009. 13, 16

[10] Matt Bishop and Daniel V Klein. Improving system security via proactive password checking. *Computers & Security*, 14(3):233–249, 1995. 31

[11] Rainer Böhme and Tyler Moore. How do consumers react to cybercrime? In *2012 eCrime researchers summit*, pages 1–12. IEEE, 2012. 27, 67, 219

[12] Joseph Bonneau. *Guessing human-chosen secrets*. PhD thesis, University of Cambridge, 2012. 80

[13] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 553–567. IEEE, 2012. 14, 15, 160

[14] Boston University. How to choose a strong password. `http://www.bu.edu/tech/support/information-security/security-for-everyone/how-to-choose-a-strong-password/`. Accessed: 2016-12-05. 31

[15] Anna Brading. Yahoo voices hacked, nearly half a million emails and passwords stolen. july 2012. `http://nakedsecurity.sophos.com/2012/07/12/yahoo-voices-hacked`. 147, 271

[16] Charles Henry Brase and Corrinne Pellillo Brase. *Understanding basic statistics*. Nelson Education, 2013. 180

[17] Christina Braz, Ahmed Seffah, and David M'Raihi. Designing a trade-off between usability and security: a metrics based-model. In *IFIP Conference on Human-Computer Interaction*, pages 114–126. Springer, 2007. 16, 159

[18] Sacha Brostoff and M Angela Sasse. "Ten strikes and you're out": Increasing the number of login attempts can improve password usability. *Presented at: CHI 2003 Workshop on Human-Computer Interaction and Security Systems, Fort Lauderdale, Florida.*, 2003. 37, 234, 292, 326

[19] Schneier Bruce. *Beyond fear: thinking sensibly about security in an uncertain world*. Springer-Verlag New York, Inc, 2003. 42

[20] Dodson Burr and Polk. NIST special publication 800-63: Electronic authentication guidelines. `https://csrc.nist.gov/csrc/media/publications/sp/800-63/ver-10/archive/2004-06-30/documents/sp800-63-v1-0.pdf`, 2003. Accessed: 2017-08-25. 68, 79, 178, 263, 284, 330

[21] Claude Castelluccia, Abdelberi Chaabane, Markus Dürmuth, and Daniele Perito. When privacy meets security: Leveraging personal information for password cracking. *arXiv preprint arXiv:1304.6584*, 2013. 31, 78, 114

[22] Junaid Ahsenali Chaudhry, Shafique Ahmad Chaudhry, and Robert G Rittenhouse. Phishing attacks and defenses. *International Journal of Security and Its Applications*, 10(1):247–256, 2016. 57

[23] William Cheswick. Rethinking passwords. *Communications of the ACM*, 56(2):40–44, 2013. 240

[24] Sonia Chiasson and Paul C Van Oorschot. Quantifying the security advantage of password expiration policies. *Designs, Codes and Cryptography*, 77(2-3):401–408, 2015. 25, 68, 73, 334

[25] Yee-Yin Choong and Mary Theofanos. What 4,500+ people can tell you–employees' attitudes toward organizational password policy do matter. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 299–310. Springer, 2015. 156

[26] Yee-Yin Choong, Mary Theofanos, and Hung-Kung Liu. *United States Federal Employees' Password Management Behaviors: A Department of Commerce Case Study*. US Department of Commerce, National Institute of Standards and Technology, 2014. 327, 339

[27] Hsien-Cheng Chou, Hung-Chang Lee, Hwan-Jeu Yu, Fei-Pei Lai, Kuo-Hsuan Huang, Chih-Wen Hsueh, et al. Password cracking based on learned patterns from disclosed passwords. *IJICIC*, 9(2):821–839, 2013. 114

[28] Mark M Christiansen and Ken R Duffy. Guesswork, large deviations, and shannon entropy. *IEEE transactions on information theory*, 59(2):796–802, 2012. 79

[29] Jessica Colnago, Summer Devlin, Maggie Oates, Chelse Swoopes, Lujo Bauer, Lorrie Cranor, and Nicolas Christin. "it's not actually that horrible" exploring adoption of two-factor authentication at a university. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2018. 225

[30] Data Protection Commission. Guidance for controllers on data security. Technical report, An Coimisiún um Chosaint Sonraí (Irish Data Protection Commission), 2020. `https://www.dataprotection.ie/sites/default/files/uploads/2020-02/Data%20Security%20Guidance_Feb20.pdf`, Accessed: 2020-03-20. 108

[31] M Cover Thomas and A Thomas Joy. Elements of information theory. *New York: Wiley*, 3:37–38, 1991. 83

[32] Philip Cox. Password sanity: Thank you NIST. `https://www.linkedin.com/pulse/password-sanity-thank-you-nist-philip-cox`, 2016. Accessed: 2016-12-15. 24, 55, 190

[33] Lorrie Faith Cranor and Simson Garfinkel. *Security and usability: designing secure systems that people can use*. O'Reilly Media Sebastopol, CA, 2005. 39, 157, 189

[34] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The tangled web of password reuse. In *NDSS*, volume 14, pages 23–26, 2014. 15, 34, 38, 240, 242, 244

[35] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010. 114

[36] Paul Dixon, 2002. `https://www.pastebin.com`. 87

[37] Yadolah Dodge. *Central Limit Theorem*. Springer New York, New York, NY, 2008. 165

[38] Vincent Drury and Ulrike Meyer. Certified phishing: taking a look at public key certificates of phishing websites. In *15th Symposium on Usable Privacy and Security (SOUPS'19). USENIX Association, Berkeley, CA, USA*, pages 211–223, 2019. 27, 238

[39] Dublin City University. Baseline password policy. `https://www4.dcu.ie/sites/default/files/policy/83%20%20password_policy_iss_v1.pdf`. Accessed: 2016-12-02. 31

[40] Paul Ducklin. Anatomy of a password disaster - Adobe's giant-sized cryptographic blunder. `https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/`, 2013. Accessed: 2018-08-16. 262, 288

[41] John E Dunn. Why NIST's Bill Burr shouldn't regret his 2003 password advice, 2017. `https://nakedsecurity.sophos.com/2017/08/11/why-nists-bill-burr-shouldnt-regret-his-2003-password-advice/`, Accessed: 2019-02-19. 156, 170

[42] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Abdelberi Chaabane. Omen: Faster password guessing using an ordered markov enumerator. In *International Symposium on Engineering Secure Software and Systems*, pages 119–132. Springer, 2015. 80, 114

[43] Malin Eiband, Mohamed Khamis, Emanuel Von Zezschwitz, Heinrich Hussmann, and Florian Alt. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4254–4265. ACM, 2017. 35

[44] Scott R Eliason. *Maximum likelihood estimation: Logic and practice*. Number 96 in Quantitative Applications in the Social Sciences. Sage, 1993. 119

[45] Larry D Evans. A two-score composite program for combining standard scores. *Behavior Research Methods, Instruments, & Computers*, 28(2):209–213, 1996. 181

[46] Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. On the ecological validity of a password study. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 13. ACM, 2013. 8

[47] Mohammed Farik and AS Ali. Analysis of default passwords in routers against brute-force attack. *International Journal of Technology Enhancements and Emerging Engineering Research*, 4(9):341–345, 2015. 24

[48] PUB FIPS. Federal information processing standard publication 140-2. *Security Requirements for Cryptographic Modules*, 25, 2001. (with Change Notices through December 3, 2002), Accessed: 2018-11-12. 298, 309, 315, 327

[49] Dinei Florêncio and Cormac Herley. Where do security policies come from? In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 10. ACM, 2010. 55, 178, 331

[50] Dinei Florêncio, Cormac Herley, and Baris Coskun. Do strong web passwords accomplish anything? *HotSec*, 7(6), 2007. 35, 38, 67, 242

[51] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. An administrator's guide to internet password research. In *LISA*, pages 35–52, 2014. 35, 36, 77, 277

[52] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In *USENIX Security*, pages 575–590, 2014. vi, 34, 46

[53] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. Pushing on string: The'don't care'region of password strength. *Communications of the ACM*, 59(11):66–74, 2016. 32, 109, 164

[54] Steven Furnell. An assessment of website password practices. *Computers & Security*, 26(7-8):445–451, 2007. 73

[55] Get Safe Online. Passwords. `https://www.getsafeonline.org/protecting-yourself/passwords/`. Accessed: 2016-12-01. 31

[56] Maximilian Golla and Markus Dürmuth. On the accuracy of password strength meters. In *CCS '18*, pages 1567–1582, 2018. 89, 123, 129

[57] Maximilian Golla, Miranda Wei, Juliette Hainline, Lydia Filipe, Markus Dürmuth, Elissa Redmiles, and Blase Ur. What was that site doing with my facebook password?: Designing password-reuse notifications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1549–1566. ACM, 2018. 14, 15

[58] Google. Creating a strong password. `https://support.google.com/accounts/answer/32040?hl=en`. Accessed: 2016-12-17. 34

[59] Paul A Grassi, Michael E Garcia, and James L Fenton. SP-800-63 Digital identity guidelines. *NIST special publication*, 800:63–3, 2017. `https://pages.nist.gov/800-63-3/`. 23, 29, 55, 69, 77, 108, 171, 254, 256, 275, 297

[60] Jeffrey Grobaski. You hate changing your password and it doesn't help. `https://epicriver.com/you-hate-changing-your-password-and-it-doesnt-help/`, 2016. Accessed: 2016-03-06. 25

[61] Kimmo Halunen, Juha Häikiö, and Visa Vallivaara. Evaluation of user authentication methods in the gadget-free world. *Pervasive and Mobile Computing*, 40:220–241, 2017. 15

[62] Weili Han, Zhigong Li, Lang Yuan, and Wenyuan Xu. Regional patterns and vulnerability analysis of chinese web passwords. *IEEE Transactions on Information Forensics and Security*, 11(2):258–272, 2015. 142

[63] Hashcat. `https://hashcat.net`. Accessed: 2020-06-09. 14, 113, 116

[64] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your Ps and Qs: detection of widespread weak keys in network devices. In *USENIX Security Symposium*, volume 8, page 1, 2012. 288

[65] Maria Henriquez, 2019. `https://www.securitymagazine.com/articles/91366-the-top-12-data-breaches-of-2019`, Accessed: 2020-03-20. 77

[66] Cormac Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on New security paradigms workshop*, pages 133–144. ACM, 2009. 16, 23, 28, 55, 160, 240

[67] Cormac Herley and Paul Van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy*, 10(1):28–36, 2012. 172, 256

[68] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. Passgan: A deep learning approach for password guessing. In *International Conference on Applied Cryptography and Network Security*, pages 217–237. Springer, 2019. 80, 115

[69] Roger A Horn and Charles R Johnson. Matrix analysis, 1985. 85

[70] S Houshmand and S Aggarwal. Using personal information in targeted grammar-based probabilistic password attacks. In *IFIP Int. Conf. on Digital Forensics*, pages 285–303. Springer, 2017. 79, 115

[71] Hang Hu and Gang Wang. End-to-end measurements of email spoofing attacks. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1095–1112, 2018. 27

[72] Troy Hunt. Have I Been Pwned. `https://haveibeenpwned.com/`. 228

[73] Troy Hunt. The "cobra effect" that is disabling paste on password fields. `https://www.troyhunt.com/the-cobra-effect-that-is-disabling/`, 2014. 17, 74

[74] Troy Hunt. The 773 million record "collection #1" data breach. 2019. `https://www.troyhunt.com/the-773-million-record-collection-1-data-reach`, 2019. Accessed: 2020-09-09. 142

[75] Philip G Inglesant and M Angela Sasse. The true cost of unusable password policies: password use in the wild. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, pages 383–392. ACM, 2010. 16, 33

[76] Intel. Password rules. `https://ssl.intel.com/reg-app/popup.aspx?Popuptype=PWD-RULES-CONF`. Accessed: 2016-12-02. 29

[77] Interactive brokers. Tips for creating a secure password. `https://gdcdyn.interactivebrokers.com/Universal/Application`. Accessed: 2016-12-02. 29

[78] Joe account dictionary definition | joe account defined. `https://www.yourdictionary.com/joe-account`. 31

[79] Audun Jøsang, Bander AlFayyadh, Tyrone Grandison, Mohammed AlZomai, and Judith McNamara. Security usability principles for vulnerability analysis and risk assessment. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 269–278. IEEE, 2007. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4412995`. 158

[80] John the ripper password cracking. `https://www.openwall.com/john/`. Accessed: 2020-06-09. 14, 101, 113, 116

[81] Joseph 'Jofish' Kaye. Self-reported password sharing strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2619–2622. ACM, 2011. 34

[82] Sarah Keller. Password changes to be required every 120 days. `https://thesimpsonian.com/13442/uncategorized/password-changes-to-be-required-every-120-days/`, 2008. Accessed: 2020-10-08. 331

[83] Patrick Gage Kelley, Saranga Komanduri, Michelle L Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 523–537. IEEE, 2012. 14, 24, 79, 114, 237, 238, 263, 266, 271, 290, 321, 336

[84] Daniel V Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, pages 5–14, 1990. 30

[85] Saranga Komanduri. personal communication. 256, 274, 276, 308, 324, 325

[86] Saranga Komanduri. *Modeling the adversary to evaluate password strength with limited samples*. PhD thesis, Carnegie Mellon University, 2016. 274

[87] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2595–2604. ACM, 2011. 24, 32, 240, 256, 274

[88] KoreLogic. Crack Me If You Can (CMIYC), 2020. `https://contest-2020.korelogic.com/`. 116

[89] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. "I have no idea what I'm doing"-on the usability of deploying https. In *Proceedings of the 26th USENIX Security Symposium, ser. USENIX Security*, volume 17, pages 1339–1356, 2017. 37, 261

[90] Manu Kumar, Tal Garfinkel, Dan Boneh, and Terry Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 13–19. ACM, 2007. 35

[91] Butler Lampson. Privacy and security usable security: how to get it. *Communications of the ACM*, 52(11):25–27, 2009. 157

[92] Let's encrypt. `https://letsencrypt.org/`. 37, 261

[93] John Leyden. Write down your password today. `http://www.theregister.co.uk/2005/07/19/password_schneier/`, 2005. Accessed: 2017-01-20. 32, 240

[94] Yue Li, Haining Wang, and Kun Sun. 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*, pages 1–9. IEEE, 2016. 114

[95] Holly Lockhart, Drew Nicholas, and Sarah Roberts. Demystifying password hash sync. `https://www.microsoft.com/security/blog/2019/05/30/demystifying-password-hash-sync/`, 2019. 234

[96] Victor Lomne and Thomas Roche. A Side Journey to Titan: Side-Channel Attack on the Google Titan Security Key (Revealing and Breaking NXPâĂŹs P5x ECDSA Implementation on the Way). *NinjaLab*, 2021. 302

[97] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy*, pages 689–704. IEEE, 2014. 114

[98] David Malone and Kevin Maher. Investigating the distribution of password choices. In *Proceedings of the 21st international conference on World Wide Web*, pages 301–310. ACM, 2012. 7, 14, 37, 79, 88, 108, 112, 142, 244, 247, 263

[99] Udi Manber. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security*, 15(2):171–176, 1996. 172, 262

[100] James L Massey. Guessing and entropy. In *Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on*, page 204. IEEE, 1994. 79, 81

[101] Robert McMillan. The Man Who Wrote Those Password Rules Has a New Tip: N3v$r M1ˆd! . `https://nakedsecurity.sophos.com/2017/08/11/why-nists-bill-burr-shouldnt-regret-his-2003-password-advice/`, 2017. Accessed: 2020-10-12. 170, 171, 190

[102] William Melicher, Blase Ur, Sean M Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate: Modeling password guessability using neural networks. In *25th*

USENIX Security Symposium (USENIX Security 16), pages 175–191, 2016. 80, 114

[103] Microsoft TechNet Magazine. Best practices for enforcing password policies. `https://technet.microsoft.com/en-us/library/ff741764.aspx`. Accessed: 2016-12-06. 25

[104] Martin Mihajlov, Borka Jerman Blazic, and Saso Josimovski. Quantifying usability and security in authentication. In *2011 IEEE 35th Annual Computer Software and Applications Conference*, pages 626–629. IEEE, 2011. 159

[105] Martin Mihajlov, Borka Jerman-Blažič, and Saso Josimovski. A conceptual framework for evaluating usable security in authentication mechanisms-usability perspectives. In *2011 5th international conference on network and system security*, pages 332–336. IEEE, 2011. 159

[106] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979. 14, 36, 79, 113

[107] Mozy.ie. Lost & found - data loss research. `https://mozy.ie/about/news/reports/lost-and-found/`, 2012. Accessed: 2019-02-02. 256, 310

[108] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. TOTP: time-based one-time password algorithm. `https://tools.ietf.org/html/rfc6238`, 2011. 225

[109] Alec Muffet. Cracklib: a proactive password sanity library. *Dec*, 14:1–5, 1997. 222

[110] Alec Muffett. "Crack Version 4.1": A sensible password checker for Unix. *disponible par ftp à l'adresse ftp. cert. org*, 1992. 30

[111] Hazel Murray. Password advice analysis. `https://github.com/HazelMurray/password_advice_analysis`, 2019. 6, 193

[112] Hazel Murray. Authentication costs administrator study. `https://github.com/HazelMurray/authentication_costs_admin_study`, 2020. 6, 45, 193

[113] Hazel Murray. Authentication costs user study. `https://github.com/HazelMurray/authentication_costs_user_study`, 2020. 6, 45, 193

[114] Hazel Murray and David Malone. Evaluating password advice. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6. IEEE, 2017. 12, 13, 40

[115] Hazel Murray and David Malone. Exploring the impact of password dataset distribution on guessing. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–8. IEEE, 2018. 37, 76, 109, 244, 247

[116] Hazel Murray and David Malone. Convergence of password guessing to optimal success rates. *Entropy*, 22(4):378, 2020. 76, 112, 123, 263

[117] Hazel Murray and David Malone. Multi-Armed Bandit Approach to Password Guessing. In *Who Are You?! Adventures in Authentication Workshop*, WAY '20, pages 1–6, Virtual Conference, August 2020. 111

[118] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 364–372. ACM, 2005. 78, 113

[119] Jakob Nielsen. Let's encrypt. `https://www.nngroup.com/articles/stop-password-masking/`, 2009. 35

[120] NIST Joint Task Force Transformation Initiative Interagency Working Group. Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800(53):8–13, 2013. `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf`. Accessed 2019-02-22. 256, 259, 298

[121] NIST, SP. 800-44 version 2. *Guidelines on Securing Public Web Servers*, 2007. 170, 330

[122] NIST, SP. 800-45 version 2. *Guidelines on Electronic Mail Security*, 2007. 170, 330

[123] Rishab Nithyanand and Rob Johnson. The password allocation problem: Strategies for reusing passwords effectively. In *Proceedings of the 12th*

ACM workshop on Workshop on privacy in the electronic society, pages 255–260, 2013. 14

[124] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Annual International Cryptology Conference*, pages 617–630. Springer, 2003. 77

[125] OSSG Documentation. OpenVMS System Manager's Manual. `https://www0.mi.infn.it/~calcolo/OpenVMS/ssb71/6015/6017p035.htm`, 1996. Accessed: 2020-10-08. 330

[126] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart. Beyond credential stuffing: Password similarity models using neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 417–434. IEEE, 2019. 115

[127] Dario Pasquini, Ankit Gangwal, Giuseppe Ateniese, Massimo Bernaschi, and Mauro Conti. Improving password guessing via representation learning. *arXiv preprint arXiv:1910.04232*, 2019. 115

[128] Paolo Passeri. 2017 cyber attacks statistics, 2018. `http://www.hackmageddon.com/2018/01/17/2017-cyber-attacks-statistics/`. Accessed: 2019-02-20. 264, 266, 277, 301

[129] Paypal. Tips for creating a secure password. `https://www.paypal.com/ie/selfhelp/article/tips-for-creating-a-secure-password-faq3152`. Accessed: 2016-12-02. 29, 73

[130] Roel Peeters, Jens Hermans, Pieter Maene, Katri Grenman, Kimmo Halunen, and Juha Häikiö. n-auth: Mobile authentication done right. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 1–15, 2017. 15, 38

[131] Pixie Technology Inc. Lost and found: The average american spends 2.5 days each year looking for lost items collectively costing u.s. households $2.7 billion annually in replacement costs. `https://www.prnewswire.com/news-releases/lost-and-found-the-average-american-spends-25-days-each-year-looking-for-lost-items-collectively-costing-us-households-27-billion-`

annually-in-replacement-costs-300449305.html, 2017. Accessed: 2019-02-02. 256, 310, 328

[132] Ponemon Institute LLC. 2018 cost of a data breach study. Technical report, Ponemon Institute LLC, 2018. https://www.ibm.com/downloads/cas/861MNWN2. Accessed 2018-07-25. 173, 251

[133] Paulo C Realpe, Cesar A Collazos, Julio Hurtado, and Antoni Granollers. A set of heuristics for usable security and user authentication. In *Proceedings of the XVII International Conference on Human Computer Interaction*, pages 1–8, 2016. 160

[134] Elissa M Redmiles, Michelle L Mazurek, and John P Dickerson. Dancing pigs or externalities? measuring the rationality of security decisions. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 215–232, 2018. 16

[135] Elissa M Redmiles, Noel Warford, Amritha Jayanti, Aravind Koneru, Sean Kross, Miraida Morales, Rock Stevens, and Michelle L Mazurek. A comprehensive quality evaluation of security and privacy advice on the web. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 89–108, 2020. 16

[136] General Data Protection Regulation. 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. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016. 184

[137] Karen Renaud. Quantifying the quality of web authentication mechanisms: a usability perspective. *Journal of Web Engineering*, 3(2):95–123, 2004. 157, 159

[138] Karen Renaud. Evaluating authentication mechanisms. *Security and usability*, pages 103–128, 2005. 157

[139] Karen Renaud. A process for supporting risk-aware web authentication mechanism choice. *Reliability Engineering & System Safety*, 92(9):1204–1217, 2007. 158

[140] Volker Roth, Kai Richter, and Rene Freidinger. A pin-entry method resilient against shoulder surfing. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 236–245. ACM, 2004. 35

[141] Von Christiane RÃijtten. Passwortdaten von flirtlife.de kompromittiert, 2006. `https://www.heise.de/security/meldung/Passwortdaten-von-Flirtlife-de-kompromittiert-126608.html`, Accessed: 2019-12-04. 88

[142] Software as a service (SaaS). `https://en.wikipedia.org/wiki/Software_as_a_service`. 220

[143] Cem S Sahin, Robert Lychev, and Neal Wagner. General framework for evaluating password complexity and strength. *arXiv preprint arXiv:1512.05814*, 2015. 263

[144] Stuart Schechter, AJ Bernheim Brush, and Serge Egelman. It's no secret. measuring the security and reliability of authentication via "secret" questions. In *2009 30th IEEE Symposium on Security and Privacy*, pages 375–390. IEEE, 2009. 22

[145] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security*, pages 1–8, 2010. 79

[146] Stuart E Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 51–65. IEEE, 2007. 27

[147] Bruce Schneier. The pros and cons of password masking. `https://www.schneier.com/blog/archives/2009/07/the_pros_and_co.html`, 2009. 35

[148] Richard Shay and Elisa Bertino. A comprehensive simulation tool for the analysis of password policies. *International Journal of Information Security*, 8(4):275–289, 2009. 16, 55, 159, 160, 274

[149] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Encountering stronger password requirements: user attitudes and behaviors. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 2. ACM, 2010. 25, 32, 243, 260

[150] David Silver, Suman Jana, Dan Boneh, Eric Chen, and Collin Jackson. Password managers: Attacks and defenses. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 449–464, 2014. 30, 239

[151] Randy Franklin Smith. Controlling user account logons. `https://www.itprotoday.com/windows-78/controlling-user-account-logons`, 2003. Accessed: 2020-10-08. 330

[152] Simple Network Management Protocol (SNMP). `https://csrc.nist.gov/glossary/term/simple_network_management_protocol`. 29

[153] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, and Meiko Jensen. On breaking SAML: Be whoever you want to be. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 397–412, 2012. 57

[154] Frank Stajano. Pico: No more passwords! In *International Workshop on Security Protocols*, pages 49–81. Springer, 2011. 14, 15

[155] Nick Statt. Best practices for passwords updated after original author regrets his advice. `https://www.theverge.com/2017/8/7/16107966/password-tips-bill-burr-regrets-advice-nits-cybersecurity`, 2017. Accessed: 2020-10-12. 170, 171, 190

[156] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 117

[157] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long "taile" of typosquatting domain names. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 191–206, 2014. 238

[158] Albert Tannous, Jonathan Trostle, Mohamed Hassan, Stephen E McLaughlin, and Trent Jaeger. New side channels targeted at pass-

words. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 45–54. IEEE, 2008. 264, 302

[159] Think with Google. How mobile has redefined the consumer decision journey for shoppers. `https://www.thinkwithgoogle.com/marketing-resources/micro-moments/mobile-shoppers-consumer-decision-journey/`, 2016. Accessed: 2019-02-01. 256, 308

[160] Daniel R Thomas, Sergio Pastrana, Alice Hutchings, Richard Clayton, and Alastair R Beresford. Ethical issues in research using datasets of illicit origin. In *Proceedings of the 2017 Internet Measurement Conference*, pages 445–462. ACM, 2017. 7

[161] Marc Toussaint. Lecture notes: Some notes on gradient descent. `https://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/gradientDescent.pdf`, 2012. Accessed=2020-10-20. 138

[162] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008. 83

[163] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, et al. How does your password measure up? the effect of strength meters on password creation. In *USENIX Security Symposium*, pages 65–80, 2012. 14, 79, 256, 276

[164] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. "I added '!' at the end to make it secure": observing password creation in the lab. In *Proceedings of SOUPS*, 2015. 24, 256, 274

[165] Blase Ur, Sean M Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L Mazurek, William Melicher, and Richard Shay. Measuring real-world accuracies and biases in modeling password guessability. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 463–481, 2015. 80

[166] USENIX Security '17 Poster Session. `https://www.usenix.org/conference/usenixsecurity17/poster-session`, 2017. Accessed: 2020-10-13. 12

[167] Rishi Vaidya. Cyber security breaches survey 2018. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/702074/Cyber_Security_Breaches_Survey_2018_-_Main_Report.pdf`, 2018. 174, 252

[168] Kami Vaniea and Yasmeen Rashidi. Tales of software updates: The process of updating software. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 3215–3226. ACM, 2016. 28

[169] Verizon. 2017 data breach investigations report. Technical report, Verizon, 2017. `http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/`. Accessed: 2018-01-25. 173, 253, 259, 261, 262, 269, 277

[170] Chun Wang, Steve TK Jan, Hang Hu, Douglas Bossart, and Gang Wang. The next domino to fall: Empirical analysis of user passwords across online services. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 196–203. ACM, 2018. 14, 15

[171] Ding Wang, Haibo Cheng, Ping Wang, Xinyi Huang, and Gaopeng Jian. Zipf's law in passwords. *IEEE Transactions on Information Forensics and Security*, 12(11):2776–2791, 2017. 263

[172] Ding Wang and Ping Wang. On the implications of zipf's law in passwords. In *European Symposium on Research in Computer Security*, pages 111–131. Springer, 2016. 7

[173] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. Targeted online password guessing: An underestimated threat. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1242–1254, 2016. 89, 114

[174] C. Warner. Passwords with simple character substitution are weak. `https://optimwise.com/passwords-with-simple-character-substitution-are-weak/`, 2010. Accessed: 2017-02-15. 33

[175] Miranda Wei, Maximilian Golla, and Blase Ur. The password doesn't fall far: How service influences password choice. *Who Are You*, 2018. 87, 108, 112

[176] M. Weir. The rockyou 32 million password list top 100. `reusablesec.blogspot.com/2009/12/rockyou-32-million-password-list-top.htmls`, 2009. Accessed: 2017-02-14. 32

[177] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 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*, pages 162–175. ACM, 2010. 79

[178] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 391–405. IEEE, 2009. 78, 80, 114, 263

[179] Dirk Weirich and Martina Angela Sasse. Persuasive password security. In *CHI'01 Extended Abstracts on Human Factors in Computing Systems*, pages 139–140. ACM, 2001. 34

[180] Zhiyang Xia, Ping Yi, Yunyu Liu, Bo Jiang, Wei Wang, and Ting Zhu. Genpass: a multi-source deep learning model for password guessing. *IEEE Transactions on Multimedia*, 22(5):1323–1332, 2019. 115

[181] XKCD. Password strength. `https://xkcd.com/936/`. 13

[182] Jianxin Jeff Yan. A note on proactive password checking. In *Proceedings of the 2001 workshop on New security paradigms*, pages 127–135. ACM, 2001. 30

[183] Targeted bruteforcing - mining patterns in passwords to make bruteforcing easy. `https://www.reddit.com/r/netsec/comments/5asjeu/targeted_bruteforcing_mining_patterns_in/`, Accessed: 2020-03-22. 89

[184] Shuo Zhang, Jianping Zeng, and Zewen Zhang. Password guessing time based on guessing entropy and long-tailed password distribution in the large-scale password dataset. In *Anti-counterfeiting, Security, and Identification (ASID), 2017 11th IEEE International Conference on*, pages 6–10. IEEE, 2017. 7

[185] Yinqian Zhang, Fabian Monrose, and Michael K Reiter. The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 176–186. ACM, 2010. 25, 68, 243, 244

[186] Leah Zhang-Kennedy, Sonia Chiasson, and Paul van Oorschot. Revisiting password rules: facilitating human management of passwords. In *Electronic Crime Research (eCrime), 2016 APWG Symposium on*, pages 1–10. IEEE, 2016. 32

[187] Moshe Zviran and William J. Haga. A comparison of password techniques for multilevel authentication mechanisms. *The Computer Journal*, 36(3):227–237, 1993. 25

[188] Daniel Zwillinger. *Standard mathematical tables and formulae*. CRC press, 2002. 85