Strategies and Applications for Creating More Memorable Passwords

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Brent Alexander MacRae

Faculty of Business and I.T.

University of Ontario Institute of Technology

This thesis is submitted for the degree of

M. Sc. in Computer Science

ABSTRACT

As we continue to learn and grow in an ever evolving technological age, we deepen our understanding of the importance of authentication. There are many different types of authentication, each exhibiting their own strengths and weaknesses. Each authentication mechanism serves the same purpose: to verify a user's identity. In this thesis, we explore two authentication mechanisms aimed at helping users remember stronger authentication tokens: one aimed at creating a secure, memorable token, and the other aimed at strengthening a previous token (known as a password strengthening technique). The first is GeoPassNotes, a geographic location-based authentication scheme. GeoPassNotes requires users to select a location on a digital map and then annotate it in order to authenticate. The combination of the location and the annotation is the authentication token. GeoPassNotes allows users to select a location that is tied to a significant event / memory, which is very memorable to that person. The other system we design and explore is PassMod, a system designed to help users create more secure versions of their password. This system separates itself from other password strengthening techniques because it interprets and attempts to preserve the original meaning behind the user's password. We demonstrate that it is possible to create a more secure password without compromising the memorability of the original password. Both GeoPassNotes and PassMod help users produce a more secure, yet memorable authentication token.

Table of contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Despite many weaknesses, passwords remain the most common form of authentication. Passwords authenticate legitimate users to access assets. At one point in time, passwords were well suited to this task. However, everything is online now: photos, banking information, life stories, etc. We are becoming a "cloud-centric" civilization, and are still relying on old habits to protect our identities. Passwords still remain the best option, despite the apparent weaknesses with their use in authentication [33]. A lot of research has been done on the memorability and security of passwords, but there is not enough evidence against them to stop using them. Therefore, they remain the best option.

Now that most of our resources are accessed and authenticated through the web, many security risks are present that were not previously possible. A dedicated hacker can try to crack passwords from the comfort of his own home. Much research has been done on leaked password datasets (RockYou, Yahoo) which has led to many insights about how users choose passwords. Users tend to reuse passwords wherever possible, and when a different password is required (as the result of an updated password policy) users will only slightly modify their password to meet the policy (e.g., add a number to the end of their existing password) [35]. Users choose passwords in predictable ways due to the constraints of their memorability; attackers know this, and use this to their advantage.

The results are not pleasant; passwords are vulnerable and easily guessed [33, 36, 59]. This motivated us to help users create a more memorable authentication token, one that is more difficult for an attacker to crack yet still very memorable to the user.

1.1 Motivation

As the tools of the attacker continue to grow, security administrators compensate by enforcing more stricter password composition policies. Many password policies now have length requirements, special character requirements, letter, number, and case requirements. It has come to a point that in order for a password to be considered "secure", it must be a random, non-pronounceable sequence of letters, characters and numbers. This increases the burden on the memorability of the average user. Users cannot be expected to remember unique, 12 character random passwords for every account.

The increased burden of having to remember multiple passwords leads to dangerous coping strategies for users. To name a few, people will write down their passwords, store all of their passwords in one file, and reuse their passwords for multiple accounts. Some password policies enforce a password change at set intervals (e.g., 3 months). This is a major nuisance for most users, but it is required for security purposes. Something as simple as this makes it more difficult, albeit not impossible, for a determined hacker to crack a password within the time constraints of the expiration policy. However, changing the password to the same thing with the addition of a number or letter does not increase the security [67]. This, among other slight modifications, is a common coping strategy for users dealing with the issue of having to remember multiple passwords.

One potential solution to this problem is the use of a Password Manager, such as LastPass [38]. Password managers are a central repository for the storage of account credentials. It allows users to store their passwords, removing the need to remember them. Typically, password managers are secured by a strong password, but it is much easier for a user to remember one password as opposed to a unique one for all of their accounts. To some, this seems like a secure, memorable way to store and access account credentials. To others, however, it is a single point of failure for all of a user's accounts. If an attacker can crack the password securing the password manager, they will have all account information for the accounts in it.

The systems presented in this thesis are aimed at decreasing the burden of memorability on users while increasing security. We help users create a more secure and memorable authentication token, such that it is not as difficult for them to remember a single, secure password for use in e.g., a password manager. The systems presented still take advantage of the deployability and ease-of-use of passwords, but yield greater memorability and security.

1.2 Thesis Summary

Our research focuses on helping users successfully authenticate to their accounts using a secure, unique, memorable authentication token. We do this by establishing a relationship between the users' memories and their passwords. In one approach, instead of having a user create a non-memorable, random sequence of characters, we get them to select a location on a digital map of the Earth and then annotate that location with a word or sequence of words they can associate with the location they chose. We call this system GeoPassNotes (which is an extension of the GeoPass system from Thorpe et al. [56]). The main intuition behind GeoPassNotes is that significant events and experiences are extremely memorable [65] and users also have a stronger memory for pictures and images over words and strings [43]. GeoPassNotes first requires users to think of a location that is easy for them to remember but difficult for others to guess as in GeoPass, then it asks users to create an associated annotation. We conducted a 30 person multi-session user study that spanned the course of 2 weeks. Our goal was to see whether an annotated location-password (location plus annotation) was more memorable, as memorable, or less memorable than traditional text-based passwords. The results indicate that annotated location-passwords are an extremely memorable way for users to authenticate, even when being asked to recall their annotated location-password one week after creation. The results of our study indicate that the majority of places the users chose were locations they had been, or would like to visit, and one that not many others would know about (similar to GeoPass [56]). These locations and events were memorable to them, and easy for them to authenticate with, but not easy for an attacker to discover without knowing anything about the target user. Most users reported the system was very easy to use, some indicating it was a "fun" or "cool" way to authenticate. When compared with GeoPass, GeoPassNotes were similarly more memorable since no user forgot their annotated location-password (compared with 2 forgotten location-passwords in GeoPass [56]). We propose that GeoPassNotes is most suitable for environments where logins are infrequent or as secondary authentication. This is mainly due to the longer login times (~25-35 seconds) when compared with traditional passwords (under 10 seconds) [4].

While annotated locations proved to be a highly memorable way to authenticate, login times are high, limiting their use in every day situations. We were also interested in improving the memorability of traditional passwords, ones that could be entered quickly and used for frequently accessed accounts. Users apply a particular thought-process before first creating a password [11]. Whether the password is related to a memory, a significant other, a historic / memorable date, etc., users make decisions about what components and themes to add to their passwords. We conjecture that these decisions are most likely based on what the users deem memorable and the perceived security for the account they are creating the password for. For example, a user creating a password for a financial institution may be more inclined to add more components (security) to the password to make it more secure [19]. In any case, the password the user ends up with is assumed to be memorable to them. Veras et al. explored a large variety of user-chosen passwords with the goal of determining how users choose passwords [59]. If we assume a password chosen by a user is memorable to that user, maybe we can make it more secure while preserving the memorabilia contained in the original password. Previous attempts at strengthening passwords contained many issues [27] which PassMod fixes. For example, if the attacker was able to gain access to the strengthening algorithm or the database used for strengthening, a significant amount of strengthened passwords could be cracked. Furthermore, since edits were made at the character level, in order for the resulting password to be memorable, a maximum of 1-2 edits could be performed [27]. The system we developed differs by taking a users password and adding / modifying components to make it more secure, all while keeping its original semantic components intact so that it makes sense to the user. The reason it makes sense to the user is because our strengthening algorithm operates by adding / modifying at the word level, not the character level. If it makes sense to the user, the user will be able to remember it. One of the reasons secure text passwords are not memorable is because they are not presented in a way that is easy for the user to remember. For example, the password "<=Clt?zO1Y:96d]}Y8M" is very secure, but few users can remember it as they can't break it up into memorable components. That is, each character has to be remembered individually as opposed to remembering whole words. However, our system is capable of taking a password the user originally thought was memorable, such as "iloveschool1999" and inserting a semantically sensible component such that the resulting password is "iloveschoolmemories1999". The memorability of the original password remains intact, but a new component has been added to enhance the user's password by making it more secure. Another example is the password "barkingdog5". When entered into our system, a resulting password suggestion was “quitbarkingdog5”. As demonstrated by these two examples, this system makes semantically sensible insertions that follow the English language, and hopefully makes them more memorable than random insertions.

We conducted a 43 person user study to test the usability and memorability of passwords strengthened by PassMod. The study was comprised of 3 sessions spanning the course of 8-9 days. Users were required to login with their suggested passwords and complete the questionnaires where appropriate. The results indicate that users had similar memorability for their new passwords in practice. The login rate was very high for each session, and only 6 people had to reset their password throughout the course of the study. When compared to traditional text passwords, the median login times were similar. Some users even left comments at the end of the study stating that the system “helped them think of a password they wouldn’t have otherwise thought of” or “the suggested password was long but surprisingly memorable”. We propose that PassMod is well-suited for everyday environments where logins are frequent (e.g., a password mamnager) and possibly also when users have multiple sites they must use a strong password for.

1.3 Thesis Statement

This thesis explores the possibility that password security can be improved through the use of systems and strategies that are more compatible with human memory. Such compatibility is important as it may reduce the use of insecure password coping strategies [35].

The following questions will be answered through this research:

1. Is it possible to increase the security of location-passwords by annotating them in a way that preserves the memorability of their unannotated counterparts?

2. Is it possible to automatically increase the security of a user's text password in a meaningful way such that the resulting password remains memorable?

3. What is the difference in terms of usability versus security of GeoPassNotes and PassMod? Also, how does this compare to passwords with a standard 8-character password policy in place?

1.4 Contributions

Our contributions are as follows: (1) We propose two novel user authentication schemes called GeoPassNotes and PassMod. (2) We design, implement, and pilot test these systems to refine their interfaces. (3) We measure their usability through two separate user studies. (4) We estimate the security of the systems using the user study data collected. (5) We perform an in-depth comparison between these two systems and compare with traditional passwords and policies where appropriate.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 describes literature relating to GeoPassNotes in the field of geographic authentication schemes and graphical hybrid schemes. We also describe work relating to PassMod such as passwords in general, password modification systems, and attacks against passwords. Chapter 3 describes the GeoPassNotes system including the interface design, user study conducted, usability analysis, security analysis, and discussion of interesting results. Chapter 4 details the PassMod system including the interface design, usability and security analyses, and discussion of results. In Chapter 5 we compare the GeoPassNotes and PassMod systems to each other and to traditional passwords. Finally, in Chapter 6 we summarize our findings and provide concluding remarks.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

It has been known for years that traditional passwords have flaws. Many researchers have researched ways to make passwords more secure, ways to make passwords more memorable, and alternative authentication methods to traditional passwords. Our research focuses on password memorability, including alternative authentication schemes that boast high memorability, and ways to create more secure, yet memorable passwords.

This chapter first explores previous research into an alternative authentication mechanism known as graphical passwords. More specifically, we explore geographic authentication schemes as they relate most closely to GeoPassNotes. Authentication through a digital map can be seen as a type of graphical password. GeoPassNotes is a system that contains both a graphical and textual element. As such, we also review graphical-text hybrid schemes. While graphical password schemes might be more memorable, we are also interested in their drawbacks. We investigate some state-of-the-art attacks against graphical password schemes. We also explore graphical-text hybrid schemes as GeoPassNotes combines the location with an annotation (see Section 2.2). Next, we explore traditional password approaches. More specifically, we look into previous research on password strengthening techniques and automatic strengthening of passwords. We focus on the flaws with this research and the weaknesses demonstrated by previous literature (see Section 2.3).

2.2 A Survey of Graphical Passwords

For over 14 years, various graphical password schemes have been proposed as viable alternatives to text-based passwords [17, 29, 61, 64]. Graphical passwords were created with the intention of fixing the weaknesses that plague regular text passwords. It is well known that password authentication is becoming less secure as the tools of an adversary grow [33]. Computers are now able to perform millions of computations per second. This is made possible as more processors and memory can inexpensively fit onto a single motherboard [42]. The end result is a computer that an adversary can use to crack passwords at high speeds. As a direct result of an attacker's ability to crack passwords with ease, many system administrators enforce password composition policies that are unrealistic and frustrating for the average user. Inglesant and Sasse [28] explored the burden complex password policies have on users. One password policy enforced 7 or 8 characters, must contain three classes of characters (classes being uppercase, lowercase, digits, special), must not contain any words or proper names. One user's remark about the system was "so it's got to the point where it's ... so difficult to make one up, and difficult to remember, that I have to write it down.". This results in unsafe coping strategies by the user, such as password reuse over multiple accounts, minimally unique password changes, or writing down passwords [50]. This issue represents the classic usability-security trade-off. The average users want a more usable system, mainly because they are unaware of the dangers of weak passwords, or what constitutes a password as being weak. The system administrators that are aware of this problem err on the side of security.

Unfortunately, despite these weaknesses in text-based passwords, they are still the most prominently used authentication mechanism for virtually every account.

Graphical passwords aim to remedy this situation. A graphical password can be defined as an authentication mechanism that utilizes a graphical user interface [29]. Contrary to what one would think, not all graphical passwords utilize a background image (the user performing an action on a picture). Some involve the user simply doodling on a blank grid. Graphical passwords were created with usability and memorability in mind, to directly benefit the end user. Many graphical password approaches also contain a relatively large theoretical password space, making them viable alternatives to the problems that text passwords contain. A complete overview of graphical passwords is out of the scope of this thesis; see a survey [8] for a comprehensive overview. Instead, we focus on geographic authentication schemes, which can be viewed as a subset of graphical passwords that involve a digital map as a background image. Geographic authentication is a form of a cued-recall graphical password scheme. Cued-recall systems involve the user remembering specific locations on an image. During password creation, the user will have to select a predetermined number of points and remember them for future login attempts [61].

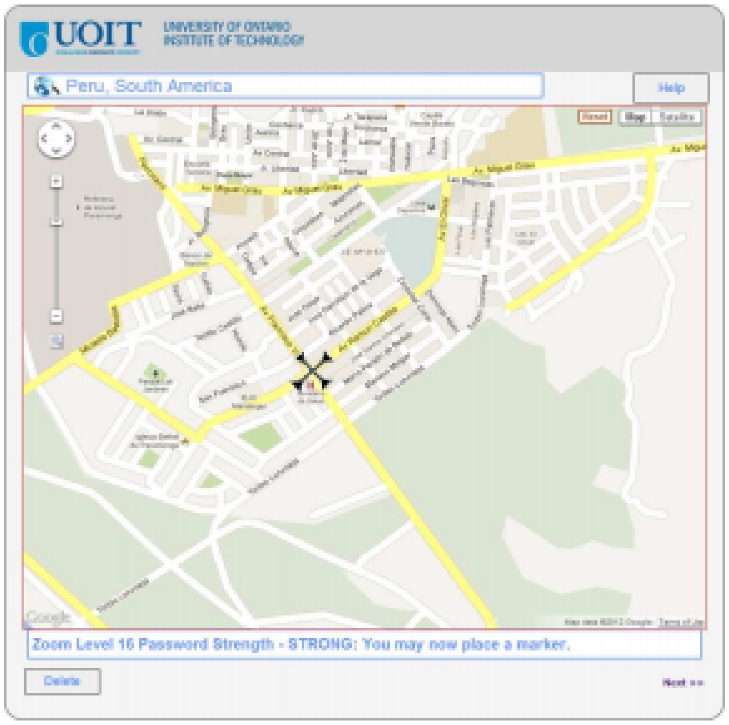
2.2.1 Geographic Authentication Schemes

Geographic authentication schemes require users to select a location on a digital map to authenticate with. GeoPassNotes takes advantage of what's known as "event-specific" memory [35]. It has been shown that users have greater memorability for pictures over words, and geographic schemes extend this one step further by exploiting event memory, or memory of significant moments in life or history. Users are able to remember significant moments in life (e.g., where they got engaged, first date, historic war site, etc.) and use this as an authentication token [11]. This is not a heavy burden for the user to remember as it is something they have already committed to memorize. After this point we change our focus from geographic authentication schemes in general to more specific examples and implementations of geographic authentication schemes.

Spitzer et al. [52] presented a cued-recall scheme that uses the user-friendly nature of Google Maps to authenticate users. Users are presented with a map of the United States with an overlaid grid on top. The user is asked to enter his / her username and select whether they want to zoom in 5 or 7 levels. The user then clicks on a grid square on the map. When they click, they are zoomed into that grid square and shown a blown up area of that square. A new 16x16 grid is then overlaid and they must select another square. They are zoomed in again and must repeat this process for the number of zooms they selected. Spitzer et al. tested the usability of their system in a 50 person user study. In total, each level contained 256 possible grid locations that the user could select. Users were tasked with remembering the grid square they selected at each zoom level. Usability ratings at the end of the study indicated that it was a lengthy process, longer than typing in a normal password.

Sun et al. [55] present PassMap: a map based graphical password scheme. PassMap allows users to login using a series of 2 points chosen on a digital Google Maps interface. Users have the ability to zoom in to any zoom level they choose to set their passwords. There are two phases to PassMap, registration and login. During registration, the user navigates the map and selects his / her two locations. The user is required to confirm the two chosen points. For the login phase, the user is asked to input the same two points they chose. In total, there were 27 university students that participated in the user study. The usability results were very high for PassMap (login accuracy of 92.59% after one week and 81.13% after six weeks), indicating that it is a good alternative to text passwords.

Thorpe et al. [56] presented GeoPass which uses the graphical interface of Google Maps to authenticate users. Users must zoom in to zoom level 16 before setting their



Peru, South America

Zoom Level 16 Password Strength - STRONG: You may now place a marker.

Fig. 2.1 A snapshot of interface of the GeoPass scheme [56].

location-passwords to increase the variability over the limited 5 or 7 levels as in Spitzer et al. [52]. The authors conducted a multi-session in-lab / at-home user study to test the usability and memorability of GeoPass. 35 university students were recruited to take part in the user study. The study was broken into 3 sessions that spanned the course of 8-9 days. The study demonstrated very high memorability for location-passwords. Users self-perceived memorability and failed login attempts per session support that the system was highly memorable. This was seen as 97% of users were able to remember their location-password over the span of 8-9 days. The median login times for sessions 1, 2, and 3 were 25s, 30s, and 25s respectively. While login times were high, most users indicated they would use this method every day, or at least if they knew it was more secure than text passwords. Some users even commented and mentioned that they prefer the increase in security and don't mind the long times. The security of location-passwords is also comparable to text passwords. The maximum security for the system is 37 bits, and for the weakest 11% it was estimated to be 16.75, which would still be enough security to hold up to an online guessing attack [20].

Al-Ameen et al. [2, 3] recently ran a 66-day long field study [2] with GeoPass, finding a 96.1% login success rate and that 100% of participants logged in successfully within five attempts on average. They also conducted two separate three week long studies [3] to test the interference of multiple location passwords (4 per user) for both the GeoPass scheme and GeoPass with modified instructions. The modified instructions were to ask users to make a meaningful association between their location password and corresponding account. Their results indicate that in the absence of mental associations, GeoPass suffers from interference effects of multiple location passwords; however, by leveraging mental associations, the login success rates were 98% after one week.

GeoPass appears as the most user friendly geographic authentication scheme and has extremely positive memorability and usability results. The main motivation for the addition of notes to location passwords is to increase security. GeoPassNotes has at least the security of GeoPass since in an online attack, the attacker needs to first guess the location and then the associated note. As such, we chose to use this as the base of our research in creating GeoPassNotes (see Chapter 3).

Graphical-text Hybrid

As GeoPassNotes can also be seen as a hybrid graphical-text scheme, we review related literature on such hybrids below.

Marasim [32] is a graphical-text hybrid authentication scheme. During enrollment, the user creates tags for a personal image of their choice. Using the tags created, four random images are found on Google. The four random tag-related images are then mixed with 4 decoy images and the user is asked to correctly identify the four images related to their tags.

GridWord [7] is a hybrid scheme as well. During enrollment, the user selects a set of three words. The system stores a one-to-one mapping of words to cells on a 2D grid. The user can then enter their password by selecting the three grid cells or selecting the three words from drop-down menus.

Inkblot authentication [54] is another hybrid scheme that is based on cueing users with a set of inkblot images. During enrollment, the user is asked to create a tag for each inkblot, and then type the first and last letters of the tag. For example, a set of 10 inkblot cues produces a 20-character password.

These hybrid schemes, like GeoPassNotes, involve a graphical and associated text element. To the best of our knowledge, GeoPassNotes is the first hybrid system that uses digital maps for text-location associations.

2.2.2 Attacks on Geographic Authentication Systems

As with any authentication scheme, whether it be passwords, graphical passwords, biometrics, etc., active enthusiasts will always try to "crack" the authentication mechanisms. Since they are the most prominently used authentication scheme to date, passwords have been the central target of attacks for years. More on password authentication can be found in section 2.3. Van Oorschot and Thorpe [58] investigate the notion of "hotspotting" in cued-recall graphical password schemes. Hotspotting refers to areas of the image that users attention is naturally drawn to (e.g., points, areas with contrasting colours, etc.) and as such, they are more likely to select a point there. They then developed an algorithm that can crack a significantly large number of cued-recall graphical passwords using the notion of hotspotting. While this has not been applied to geographic authentication schemes, this concept is still relevant as a certain amount of hotspotting can be seen in the results presented in GeoPass [56].

Shoulder surfing is an inherent weakness with most graphical authentication schemes. Shoulder surfing is the act of an adversary observing password creation or login. By observing the user as they enter their password, an adversary can observe some or all of the users password [16]. Alphanumeric passwords have some resistance to this weakness as their ASCII characters are typically hidden (stars or dots) as the user types in their password.

2.3 A Survey of Passwords

Traditional alphanumeric passwords have been the most widely used authentication for over 50 years [66]. We use passwords every day from logging into Facebook to see what our friends are doing to making a credit card payment online through our banking website. We have relied on this form of authentication for years, despite growing concerns.

2.3.1 Password Attacks

Over the years, passwords have been one of the primary subjects of research due to the increasing numbers of authentication-related attacks [25, 34, 53]. Financial concerns are a major motivator as well as privacy concerns. The goal of passwords is to authenticate a legitimate user to a resource. However, if non-legitimate users are able to obtain authentication credentials of a user, they can successfully authenticate as that user.

John the Ripper

For a while, the most popular password cracking tool was John the Ripper (JtR) [45]. JtR took a password file containing hashes and attempted to crack the hashes using a variety of guessing attacks. The primary operation of JtR is dictionary mode which uses a list of common passwords to try and guess weak passwords in the password file. This method works well if the passwords JtR is trying to crack are very weak (i.e., contained in the list). Another mode of JtR is referred to as "incremental mode". In incremental mode, JtR tries to crack passwords without an input dictionary. It does this by using frequency tables and trying the most frequent characters first. It will also try variations of these passwords. While this mode may crack some passwords that were not in the dictionary, it is much slower than the dictionary mode.

Probabilistic Context-free Grammars

Over the years, more sophisticated attacks against passwords have taken place. Traditional, dictionary style attacks are no longer the state-of-the-art method for cracking passwords. Now, cracking a password that has never been seen before is more likely. But how can a password that has never been seen before possibly be guessed by a deterministic machine such as a computer?

Weir et al. [63] develop a system called Probabilistic Context-Free Grammars (PCFG) that generates guesses in the highest probability order. The goal is to generate guesses in decreasing probability order in order to try and crack the most probable passwords first. Password collections needed to be divided up into two sets, the training set and the test set.

When parsing a training set, simple structures and base structures are created. A simple structure is just a description of the password (*SLD* where *S* = special string, *L* = alpha string and *D* = digit string). The base structures are the same but have lengths of each of the strings as well. The first pre-processing technique is to derive all of the base structures and their associated probabilities from the training set. A pre-terminal structure is a base structure that has values substituted for *S* and *D*. The pre-terminals define mangling rules that can be directly applied in cracking trials. Given a pre-terminal structure, a dictionary is used to derive a terminal structure. In an example where an alpha word of length 3 was needed, a dictionary word of length 3 is required to make a guess. Thus, a comprehensive input dictionary is required. Pre-terminals have an associated probability, so one approach is to fill in all words for a specific, highly probably pre-terminal, then move on to the next one, etc.

Three dictionaries were used to test PCFG on: MySpace, SilentWhisper and the Finnish list. In order to train the PCFG and generate the grammars, half of each dataset was taken and used for training (except SilentWhisper as this set was very small). Passwords not in the training set were used as a target. Popular password cracker John the Ripper [44] and PCFG require an input dictionary to operate, thus, six publicly available dictionaries were used. The first test was against the MySpace passwords. 3 trials were conducted: John the Ripper with default mangling rules (leetifying words, prepending / appending characters, case substitutions) applied, PCFG with pre-terminal probabilities applied, and PCFG using the probabilities of the terminals (guesses). PCFG with terminal probabilities was the most effective cracker. Another trial was conducted to see the effect that the training dictionary size has on the effectiveness of PCFG. The results show that the larger the set, the more effective PCFG is as a password cracker. At the time of it's inception, PCFG-based password cracking was the very best, able to crack significantly more passwords than JohnTheRipper. The reason for this is the abstraction to the structure form of passwords, rather than the password itself.

Semantic Cracker

Veras et al. [59] present the first framework for segmentation, semantic classification and semantic generalization of passwords. By segmenting a password into its components, Veras et al. were able to analyze the semantic structure contained in the password; the essence of the password. Using natural language processing techniques, Veras et al. were able to generate guesses used to crack passwords. This framework was able to crack significantly more passwords than the previous best method (67% more passwords from LinkedIn and 32% more passwords from MySpace). This method was also reported to be the best at cracking English language passwords when compared with 11 other popular password crackers [30].

Veras et al. make use of a variety of English corpora consisting of source corpora (collection of raw word lists, used as base for building the segmentation candidates) and reference corpora (a collection of part-of-speech tagged N-grams with frequency of use information, used for selection the most probable segmentation). The main corpus used was the Contemporary Corpus of American English (COCA), as this is a very large, general-purpose corpus containing part-of-speech tagged unigrams, bigrams, and trigrams as well as the observed frequencies in the English language. This is used as the reference corpus, and a trimmed version is used as part of the source corpus. The trimmed version removes words with three characters that have a frequency of less than 100, only used the top 37 two character words and only used ‘a’ and ‘I’ for single letter words. The reason for the trimmed corpus is to reduce the occurrence of short, rare words in an effort to speed up parsing and improve accuracy. COCA is a very useful general-purpose corpus, but it is not useful for names entities. Thus, a names list, cities list, surnames list, months list, and countries list were also used.

Veras et al. assume that every password is a combination of words and/or gap segments. A word is a string found in a corpus whereas a gap can be a number, space, symbol, etc. Using this method, segmentation’s of the password can be created. Segmentation’s are ranked based on coverage (the presence of gaps) and frequency of use. The frequency of use is determined by the reference corpora, since this contains ranked n-grams which can be used to rank segmentation’s based on likelihood. The next step was to use the segmentation method discussed above to extract words from the RockYou passwords. Veras et al. report that a limitation of their segmentation algorithm is that it is limited to English passwords. This is due to the fact that English corpora were used to train the system.

Part-of-speech tagging is required for the semantic classification of nouns and verbs. POS tagging works better when context is provided, however there are many free tools that provide reasonable part-of-speech tagging results. The algorithms used was the POS tagging module of the NLTK. The results indicate that the POS tagger does a good job at tagging given the context. For example, for the passwords *gangsterlove* and *ilovestacy*, the tagger was able to distinguish between the word *love* being a noun and verb, respectively. The algorithm takes a string of passwords and outputs for each an array $K = [s, t, c]$ where s is the string, t is the POS tag, and c is the semantic category. First, WordNet was used to classify strings. Only nouns and verbs were assigned a semantic category. If s is a gap, it is classified using regular expressions. If it is a proper noun, the source corpora is used to tag it as a month, female name, male name, surname, country, or city. The ability to generalize semantic categories is desirable. Thus, Li and Abe's [39] tree cut model is used to select the best generalization for the sample. Some of the most popular semantic categories are names and dates (unsurprisingly), but more interesting ones are love, places, sexual terms, royalty, profanity, animals, food, alcohol, and money.

The intuition behind the usefulness of semantic patterns is that some words tend to be paired up with specific classes of words. For example, the verb *eat* is typically followed by the name of a food. This is significant in terms of security because only semantically probable guesses will be made first which will result in a large reduction of the guess space. The guess generator was modeled after PCFG. PCFG works well when the training set is the same as the target set. Veras et al. used their semantic generator to output guesses in the highest probability order and input them directly into JtR for cracking. When semantic guesser was trained on RockYou, approximately 67% more passwords from LinkedIn were cracked. 32% more passwords were cracked when using MySpace.

Since this is the most effective password cracking algorithm to date, we decided to use the same semantic PCFG to inform the PassMod system. The PassMod system was designed with the same algorithm as presented by Veras et al. [59].

Since this was known as the best attack, other researchers looked into the opposite side of things. If PCFG's can be used to crack passwords with positive results, could they be used to create a better password to begin with?

2.3.2 Password Modification Systems

Abadi et al. [1] attempted strengthening passwords as early as 1997. They state that a password P that is supplied by a user could be strengthened using a password supplement Q which is provided by the system. Traditional authentication uses a password P and a salt S which is fed through a collision-free hashing function H as demonstrated by the function $H = f(P+S)$. Abadi et al.'s strengthening system can be demonstrated by the function $H = f(P+Q)$. They state the difference between a salt and a password supplement is storage of a salt on the system where the user is authenticating. For example, typically, hashes are stored with a plaintext salt to provide randomness to the output hashes. The password supplement seen in Abadi et al.'s system is not stored on the system. The password supplemnt is randomly generated on password creation, and then discarded. When a user attempts to login with their password, the system tries to regenerate the password supplement. This is essentially a brute-force method until it gets the right one. Once it gets the right one, the user is granted access. In every case, after the password supplement is used, it is discarded, never being stored on the system. The system stores all possibilities of the password supplement Q and essentially tries to bruteforce the correct one every time the user tries to logon. The benefit of this approach is the password supplement is not stored on the system, but is computed when the user enters their password. Since the password supplement is generated by the system at the time the password is input, this means that if an attacker were to gain a copy of the password database for use in an offline attack, they would be unable to regenerate the password supplement.

Forget et al. [21, 22] demonstrate a method of strengthening passwords by randomly inserting randomly chosen characters at random intervals in the users password. The user then has the option to "shuffle" the inserted characters and to be presented with a password that has newly inserted characters. The user has the option to shuffle as many times as they like. Forget et al. conducted a user study to test the memorability and security of the suggested passwords. The results indicate that the theoretical security of passwords was able to be improved significantly. However, multiple insertions into a users already strong password resulted in them starting with a weaker password. This lowered the overall security of the system, as users intentionally selected a weaker password due to the memory constraints.

Houshmand et al. present an approach that analyzes and modifies users passwords at the time of creation if they are considered weak. Weak passwords are determined based on the probability of the password being cracked using a probabilistic context-free grammars approach as seen by Weir et al. [63]. If the password is determined to be weak, the password is slightly modified to be stronger while still preserving the structure of the original password so that it is still memorable to the user. Passwords are determined weak if they fall below a certain *threshold* value. The threshold is calculated as a function that takes into account hash type and system cracking speed. Based on a system with a 2.4GHz Intel Core 2 Duo using MD5 hashing, the authors selected 2.96×10^{-13} as a threshold. With this threshold, it can be guaranteed an attacker using this machine would take 1 day to crack the password. The system works by taking the users' password, and converting it to a base structure. The base structure of a password is the tagged segments of the password all put together to make one long structure. The base structure should have a probability associated with it from training. The probability is then compared with the threshold value to see if the password is weak or not. One main goal of this system is that when a weak password is modified, it must still remain as memorable as the original password. The authors make the same assumption as in our research; the original password the users typed is a memorable password to that user. Operations that can be done to modify the base structure are insertion, deletion, and transposition. Operations that can be done to the components of the password are insertion, deletion, substitution, and case changes.

Houshmand et al. use three stages in the process of strengthening passwords:

Step 1 Used to generate all passwords at or below a certain threshold. This stage is a password cracking algorithm that outputs passwords in highest probability order. This algorithm takes a password's grammar structure, and sends it as input to step 2 to generate the guesses.

Step 2 A guess generator for a particular structure. This step searches for valid terminals to fill the components of the structure. The most probable terminals are tried first. The output is a list of passwords that were created from the input structure. If this step was called from step 1, this list will be used in a cracking attack.

Step 3 This is the actual strengthening algorithm. The input to this step is a list of passwords (from step 2). This step references a strengthening database, which is a collection of password structures, Markov chains, and dictionaries, each with associated probabilities. Passwords can either be fully processed or partially processed into the database. When passwords are fully processed into the database, the Markov chains probabilities are updated and the dictionaries are updated. When partial processing is being used, the Markov chain probabilities are updated but the dictionaries are not; this becomes relevant later when we discuss attacks against PCFG-based strengthening systems.

In order to test Houshmand et al.'s system, three separate password sets were used: RockYou (32 million plaintext passwords), MySpace (61,000 passwords), and Hotmail (9,000 plaintext passwords). For each of these sets, half of the passwords were taken and used for training the PCFG. The system takes either one password or a set of passwords at a time. It checks the probabilities of the passwords against the threshold value. If the password is determined to be weak, it is modified. A password is determined weak if it could be cracked in one day. After all of the passwords were strengthened, the authors ran two password cracking algorithms to try to guess the newly formed passwords. John the Ripper and PCFG were both used. The passwords were grouped into one of four categories: originally strong, originally weak - not able to make stronger, originally weak - able to make stronger, and strengthened passwords. The results indicate that passwords that were originally strong and strengthened passwords were very resistant to cracking. JtR cracked 1% of these passwords and PCFG cracked 5%.

2.3.3 Weaknesses of Password Modification Systems

The methods described in Section 2.3.2 are a step in the right direction. However, after a more thorough analysis of the modifications made and how they were made, other researchers were able to conclude that in previously proposed methods, automatic strengthening of passwords may be less than optimal, and make more issues than it fixes.

When passwords are automatically strengthened using probabilistic techniques such as by Houshmand et al. [27], specifically with an edit distance of 1, it is possible that the strengthened passwords are susceptible to an attack [47]. Schmidt et al. describe an edit distance of 1 single character as one modification (insertion, deletion, replace) that is performed on the original password. The equivalent of this in the semantic system would be one modification of an existing component or insertion of a new component. We note this method of attack as it is the only known comparable attack against PassMod. The authors make the assumption that the strengthening algorithm is known through repetitive use. There are two attacks an attacker can mount against strengthening schemes that do modifications as in our system and others (e.g., [21, 22, 27]):

1. The attacker could build a library of passwords that are statistically similar to the strengthened passwords. This means that if the training data used to create this library is similar to the guesses made in the original system, the output would be able to guess a significant number of passwords. This could then be used in a PCFG attack.
2. The second approach involves the attacker employing a guided brute-force attack against the original, weak password, and then trying all known strengthening variants.

In general, a way to prevent (1) above is to have the system be adaptive in terms of password suggestions. With regards to modification systems, if the users' password is not strong enough, one or more modifications are made to it until it achieves a high enough score. When passwords are modified, the initial password, and any strengthened variants, are added to whatever database the scoring algorithm requires for its use. This will keep the scoring algorithm adaptive, and take into account strengthened passwords as well.

With regards to Houshmand et al.'s system described in Section 2.3.2, passwords are inserted into the database in order to make the process adaptive. By having a collection of strengthened and unstrengthened passwords, the system can ensure that once the occurrence of a certain structure becomes too high, it can be assigned a lower probability and not suggested as modification. For example, over time, one type of structure may start out as being strong, but get suggested so many times that it becomes weak. This is the "adaptive" part of the system. This step takes the password and applies a modification to it based on the number of modifications allowed (edit distance of 1 or 2).

Resistance to PCFG-Based Attacks

Houshmand et al.'s [27] algorithm 3 ensures that every password output meets the required threshold at the time of creation. However, due to the algorithm's adaptive nature, when processed with a different dataset at a later point in time, the guess probability will vary. Therefore, it's possible that at the time, a password is secure, but when using a different dataset to measure strength, the password could be weak (e.g., strengthen with RockYou, try to crack with LinkedIn).

Houshmand et al. [27] found that using the original passwords to try to crack the strengthened passwords produced by algorithm 3, the attack is very ineffective (1.3% in a day, 2.2% in a week). However, when using the strengthened passwords produced by algorithm 3 to train the algorithm, the attack (within one edit) is more effective, cracking 2.5% in a day and 4.6% in a week. When passwords are strengthened with an edit distance of 2 edits, the strengthening algorithm works well, resulting in only 1.3% of the passwords being cracked in one week, even when using algorithm 3.

Due to the nature of algorithm 3, only the strengthened passwords are partially processed into the database (the Markov chains probabilities are updated, but the dictionaries are not). If the full strengthening database is leaked (fully processed original and strengthened passwords), attacks against strengthened passwords are much better. Three types of attacks are performed by Schmidt et al. [47] against Houshmand et al.'s system:

Attack 1 - PCFG-based attack where the strengthening database is leaked (partially processed original and strengthened passwords). This would be the effect of the database after algorithm 3 has been run (since in algorithm 3, each password and suggestion is partially processed).

Attack 2 - Step 3 where all original and modified passwords are fully processed. Every string, and structure in both passwords enters the dictionaries. Furthermore, Markov Chains are updated.

Attack 3 - Step 3 where all original passwords are fully processed (every string, structure enters dictionaries, Markov Chain probabilities are updated) and all modified passwords are partially processed (only Markov Chain probabilities are updated).

Attack 1 as presented provides no information to an attacker. This is attributable to the partial processing of both the original and strengthened passwords in the database (i.e., only the Markov chain probabilities are updated). Attack 2 with fully processed original and strengthened passwords (i.e., Markov chain probabilities are updated and the dictionaries are updated) provided the most information to the attacker. However, within 10^{-15} guess probability (PCFG probability of the structure multiplied by the probabilities of all terminals), 75.6% of passwords could be cracked with an edit distance of 1. For an edit distance of 2, this number is reduced to 50%. Attack 3 saw 28.7% of passwords guessed within 10^{-15} guess probability (GP) for an edit distance of 1, and 10.3% for an edit distance of 2. All three attacks perform similarly in a PCFG cracking attack when the resulting databases are used as input to PCFG (~18% after 10^{-15} GP).

Resistance to Guided Brute-force Attacks

In this attack scenario, Algorithm 1 is used to generate passwords below a certain GP, and a brute force attack is used against the original, unstrengthened passwords. Once guessed, PCFG's can be used to quickly discover all possible variations of passwords. It was also shown that this attack was quite successful, with the only limiting factor being the time taken to compute guesses.

One possible way to thwart a GBF attack is to increase edits. However, as shown in previous work [47], going beyond two random edits significantly impairs memorability. Another possibility is to enforce that the original password is stronger. Requiring that the original passwords' GP to be at most 10^{-12} completely thwarted the GBF attack all together [47].

2.3.4 User Choice In Passwords

Up to this point, all attacks reported in the literature on PCFG grammars were based on the PCFG algorithm of Weir et al. [63]. When a user creates a password, they do not do so randomly. There are though processes and memories that play into the creation of every password. We explore user choice in passwords to better understand the thought processes and procedure of selecting a password. If we can better understand how a user selects a password, perhaps we can find a way to help them create a more secure password that makes sense to them. As such, in order to truly make an effective cracking algorithm, the meaning of passwords must be taken into account.

Veras et al. [60] develop a way to explore passwords graphically through the use of a visualization. They hypothesize that semantic patterns exist in passwords, but are difficult to classify computationally. Thus, they create a visualization that allows a user to explore a dataset and discover semantic patterns. Veras et al. are primarily interested in dates and numbers in passwords. The RockYou dataset was used as the basis for this experiment.

Since dates are the primary concern, the everyday use of dates is important. For example, certain characters are generally used to delimit elements of a date (year, month, day). Similarly, the format of a date varies from person to person (e.g., MM/DD or DD/MM). In the RockYou dataset (32 million passwords) 25% contained sequences of 4 or more digits. Of these sequences, 62% contained 5 to 8 digits. Of all passwords that contain numerical components, 37% contained date-like number sequences.

The visualization contains multiple views that contain different information. The radial plot shows the distribution of dates parsed from passwords along with years and decades. The radial plot represents years via circles, positioned in a radial layout. All years of a certain decade are distributed throughout the ring, in a clockwise order. The rings, which represent decades, are organized in ascending order from the center. The frequency of patterns can be observed via the colour intensity. The Tile map depicts the distribution of passwords across days and months. The colour code is consistent with that of the radial plot; the darker regions are the most prominent. Clicking on a specific day will update the word cloud to contain passwords associated with that specific day. The raw passwords are shown in the word cloud view. The word cloud represents raw passwords of different sizes according to their frequency. The word cloud is interactive in that a researcher can click and remove prominent passwords, updating the graph to discover new patterns that were previously unknown. The radial plot indicates that recent years (after 1969) were the most popular. It was noted that of all of the dates, 86% were after 1969. Some possible reasons are that these dates represent birthdays, significant events, and dates that the account was created. Using the word cloud, it was seen that single characters and initials are the most popular text strings that co-occur with dates. When full words are used, they are almost always months of the year with the date. Another interesting finding was that familiar dates were prominent, such as Valentines Day, New Years Day, New Years Eve, and Christmas Day. Other days that were interesting are the first day of spring, Indonesian Independence Day, and the day the Titanic sank.

Dates are just one form of semantic structure contained in a password. Using a dictionary composed of only dates, approximately 22% of the date based passwords could be guessed using a dictionary of only 15% of the total possible dates. If all possible dates were able to be guessed (a dictionary size of 206,658), 4.16% of the RockYou passwords could be guessed. This is an extremely effective attack considering the initial dictionary is only comprised of dates.

2.3.5 Other Approaches to Improve Password Security

Password policies are aimed at guiding a user to create a more secure password by enforcing a set of requirements the user must abide by to create their password. Password policies can be relaxed, only enforcing a minimum character length (e.g., minimum 5 characters in length) or be very complex consisting of special character, length, case, and digit requirements. The complexity of the password policy is up to the individual organization implementing it. Good practice would have us seeing high value websites (such as banking, government, etc.) employing stricter password composition policies and less important sites having weaker ones. There has been much criticism in the area of password composition policies in the past. For example, Weir et al. [62] found that most password policies do very little in preventing online attacks. The reason for this is users take the “path of least resistance” when creating a password. Users already know what password they want to use, most likely due to password reuse. They may try that password on a site, only to be informed that their password is not “secure” enough because it does not meet the complexity requirements of that website. Therefore, the user appends the missing character to the end of the password, and they now can use that password. An example of this can be seen by a user that uses the password "password" for every site. One site they come across requires digits in their password, so they append a 1 to the end of their password to make "password1" which satisfies the password policy. Attackers know this behaviour exhibited by users and can leverage it in an online attack [62].

Egelman et al. [18] investigate the influence that password meters have on password creation and password changes. The authors conducted a field study to test the effects password meters of different types and orientations have on important and unimportant accounts. Their results indicated that passwords created with a meter present yielded no statistical difference in terms of password length, strength, entropy distributions, and login success rate. They were also able to conclude that browser features to remember user passwords were not a factor in their experiment. The results of their study indicated that password meters did not impact password creation in a statistically significant way, and that in some cases they were unnoticed. Through a post questionnaire, they found that 63.8% of the users in their study reused their passwords, and 22% of them reverted to using their old passwords when the study was completed. They also mention that the entire reason behind password meters in the first place is to "nudge" users into creating a more secure password. However, from their study and research, it was found that "Weakness was not a problem of which they were unaware, but one of which they were aware but insufficiently motivated to fix".

CHAPTER 3

GEOPASSNOTES

3.1 Introduction

It is well known that traditional text passwords have flaws. Weak passwords are easy to crack, so many security administrators compensate for this by enforcing stricter policies. This usually results in passwords that are difficult for users to remember without the use of additional coping strategies (e.g., password managers, writing down passwords, or password reuse [19]). This motivates new authentication approaches for accounts that have infrequent logins as they are more likely to be forgotten and fallback authentication methods in the event the primary authentication method is forgotten.

Location-password schemes, where a user chooses a specific place on a map instead of a word, appear to have high memorability and provide sufficient security against online attacks [56]. If an authentication scheme produces credentials that are highly memorable, it is possible for users to remember more than one token, which may reduce their need to reuse password [35]. As most users appear to choose places based on events that occurred in familiar places, we hypothesize that it may be possible to incorporate other event-related information with the location-password. Event-specific information, such as the "what" and "who" of events (which tend to be recalled along with the "where"

[35]), could be expressed in the form of an *annotation* (i.e., a word or sequence of words). Therefore, we aim to enhance the security of location-passwords by asking users to choose an annotation that can associate with their chosen location; we call this combination of the location-password and its annotation an *annotated location-password*.

We implement an annotated location-password system called "GeoPassNotes", which is based upon the GeoPass location-password system [56]. We evaluate the security and usability of GeoPassNotes through a user study involving 30 participants over 8-9 days to allow comparison to unannotated location-passwords. The results of our study indicate that annotated location-passwords are more secure and as memorable as unannotated location-passwords (*all* participants recalled their annotated location-password after one week). Many users seemed interested to login to their accounts using this scheme and were curious if it would ever be used in practice. Other users indicated that it was a fun, secure take on traditional password authentication. Given that the median login times are 33 and 36 seconds for sessions 2 and 3 respectively, we suggest the most appropriate use for GeoPassNotes is accounts where logins are infrequent (e.g., once per week). It is also useful for fallback authentication (e.g., in place of personal knowledge questions) [46]. This work has been published as a part of a journal paper [40].

The remainder of this Chapter is organized as follows. Section 3.2 describes the interface components of our GeoPassNotes system. Section 3.3 details our study methodology and procedure. Section 3.4 contains two main sections: a security analysis (see Section 3.4.1) and a usability analysis (see Section 3.4.2). In Section 3.5, we explain resulting policy recommendations. In Section 3.6, we discuss interesting findings, and finally, in Section 3.7, we conclude and discuss potential avenues for future work.

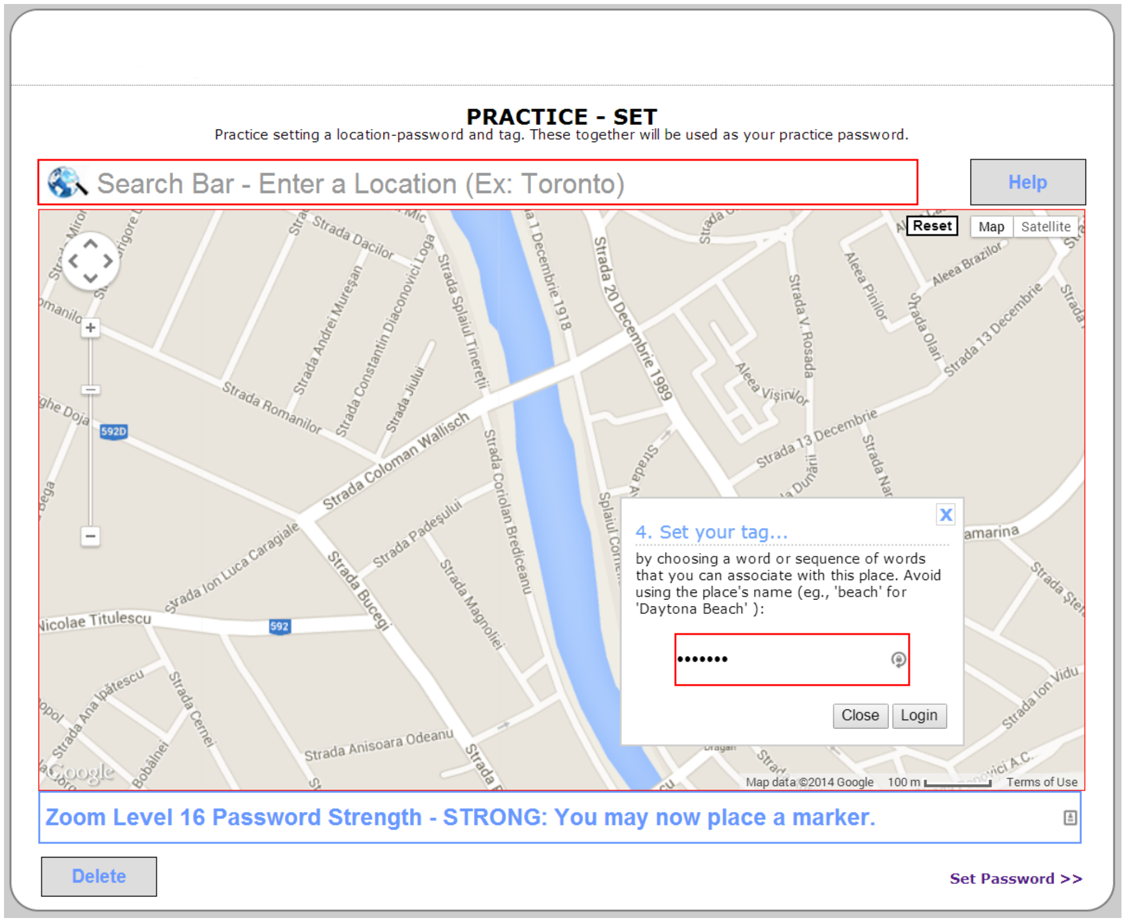
3.2 System Design

Our GeoPassNotes system is the first to combine notes (a word or sequence of words the user can associate with the location) with user chosen locations. The combination of the location and the note is what makes the credential. GeoPass [56] was chosen as the basis for the system as it is the least complex (the user only needs to remember one location, as opposed to a sequence of locations or more than one location) and is thus most conducive to eliciting event-specific memories. We implement the same functionality as GeoPass for searching locations, zooming in, and panning the map. Users choose their location by right-clicking on the map, which places an ‘X’ marker on the map for visual feedback. One improvement over the GeoPass scheme that could be implemented is the use of a locally hosted Google Maps API. Locally hosting Google Maps provides a number of benefits including shorter load times, offline support, and not having search traffic / geographic coordinates being sent in plaintext through the public Internet [24].

Users register by first selecting a location on the map as in GeoPass and then they create an annotation. For a login to be successful, both the same location (error tolerance accepted within a 10 pixel radius at zoom level 16 are tolerated) and the exact same annotation must be re-entered. There are a few design differences from GeoPass based on the introduction of the annotation:

- **Annotation pop-up** Once the user sets their ‘X’ marker, a pop-up box appears that allows them to enter their annotation (see Figure 3.1). After typing the annotation, the user can press the “Enter” key or “Login” button to login. This field is like a regular password field in that the typed characters appear as circles and the user can press the delete key as needed.
- **Clickable ‘X’ marker** To enable users to change the location entered before logging in, they can close the annotation popup and then move the ‘X’ marker by

right-clicking elsewhere on the map. If they chose to keep the chosen location after closing the annotation pop-up, they could re-open the annotation pop-up by left-clicking on the 'X' marker.


PRACTICE - SET

Practice setting a location-password and tag. These together will be used as your practice password.

Search Bar - Enter a Location (Ex: Toronto)

4. Set your tag...

by choosing a word or sequence of words that you can associate with this place. Avoid using the place's name (eg., 'beach' for 'Daytona Beach'):

•••••••

Zoom Level 16 Password Strength - STRONG: You may now place a marker.

Fig. 3.1 Snapshot of the GeoPassNotes interface.

3.3 User Study

We evaluated the usability and security of the system through a 30-person user study that was approved by our university's Research Ethics Board. The system was first pilot tested with 3 experienced and 4 casual computer users, allowing us to iron out any obvious design flaws before the user study began. To enable comparison with unannotated location-passwords, we use the same study design as for GeoPass [56].

3.3.1 Sessions

The study had three sessions over 8-9 days:

Session 1 (day 1, in-lab). Participants were shown a demo video that explained how to create an annotated location-password and the various interface features. Users were recommended to avoid directly labelling their location with the name of the location itself (e.g., if the location was in Daytona Beach, avoid choosing labels along the lines of “daytona beach” or “beach”). They were also recommended to avoid choosing a location that they had lived or worked (as in GeoPass [56]). Of course, in a field implementation, these recommendations should be enforced as discussed in Section 3.5. Participants then practiced creating and confirming an annotated location-password. After successfully confirming a practice annotated location-password, users were asked to create and confirm the annotated location-password that they will use for the remainder of the study. Some participants asked if they were allowed to reuse their practice annotated location-password as their main GeoPassNote. In this case, participants were told that they were allowed to reuse their practice annotated location-password. Participants were then distracted with a background questionnaire for approximately 10-15 minutes. To conclude the session, participants were asked to login with their chosen annotated location-password.

Session 2 (day 2, online). Session 2 was 24-48 hours after successful completion of session 1. Participants were asked to login once to our online system. All 30 participants completed session 2.

Session 3 (day 8-9, in-lab). Session 3 was held approximately one week after session 2. Participants were asked to login and complete a feedback questionnaire. All 30 users returned to complete session 3.

3.3.2 Environment

The in-lab sessions were conducted in an isolated environment at our university, in a room dedicated to running user studies. Participants were asked to bring their own personal laptops to complete the sessions. One user at a time completed the in-lab sessions to allow the investigators to qualitatively observe how the user interacted with the system.

3.3.3 Participants

30 participants were recruited from a university campus through an email sent to all students and also through posters. Participants were given $10 to participate in the study and were also entered into a draw for $50.

A background questionnaire was used to gather demographic information. For all questions, users were given the option to not answer. Our participants (19 male, 11 female) were all university students between the ages of 18 and 30. Most (28/30) were between the ages of 18 and 25.

3.3.4 Limitations and Ecological Validity

As two sessions were performed in the lab (for qualitative observation), we only collected data from 30 users. If our data set were much larger, it is possible that further patterns in notes and locations than what we describe in the following sections could appear. Web-based studies have advantages including the ability to gather larger sample sizes;

however, they do not allow observing participants to collect qualitative data. Given that this is the first study of GeoPassNotes, a larger study would be premature at this time.

Since we recruited our users from a university campus, our population is not fully representative of who the users of this system might be. However, we only recruited participants that had self-declared that they were not in IT programs, and had never taken a course in security before. Despite this criteria, they most likely still had a heightened awareness of computers and are probably more skilled than the average user, which might positively impact e.g., our findings for login times.

3.4 Results

Here we discuss the usability and security results of our study. In Section 3.4.1 we analyze the security offered by the notes alone and in combination with the location component. In Section 3.4.2 we discuss usability results, which we found to be comparable to unannotated location-passwords.

3.4.1 Security Analysis

The main reason for the addition of notes to location-passwords is to increase the security. Thus, we aim to estimate the increased security as a result of adding notes. We do this by (1) categorizing the notes collected from our study to observe their relationship to the location, (2) estimating the security of the location component alone, (3) analyzing the potential security impact of notes by categorizing them to determine common patterns in user choice, (4) running popular password cracking programs against the notes to estimate the security they alone provide, and (5) comparing the estimated security of GeoPassNotes to that of regular passwords. Our analysis primarily focuses on resistance to offline attacks, as that is a common threat model (distinct from shoulder surfing as discussed in Section 3.4.1). Offline attacks refer to an adversary who tries to guess a local copy of hashed passwords.

Notes-Location Relationship

We first analyze the notes for any relationships with the corresponding locations chosen. Through the use of a questionnaire, we asked participants why they chose their notes. Out of our 30 users, 24 users claimed their notes held significance to the location. Two users' reasons for choosing their note were difficult for us to classify, and to say whether they were related to the location or not, and four users' notes descriptions indicated they did not seem to be related to the location at all.

	High estimate		**Low estimate (per tripadvisor [57])**	
Guessing attack model	attacker guesses	# locations guessed	attacker guesses	# locations guessed
Unknown adversary (all land surface area)	$2^{36.88}$	30/30 (100%)	$2^{24.07}$	4/30 (13.33%)
Known adversary (places lived and visited)	$2^{29.17}$	17/30 (56.67%)	$2^{19.69}$	3/30 (10%)
Local adversary (Greater Toronto Area)	$2^{22.52}$	7/30 (23.33%)	$2^{16.75}$	2/30 (7%)

Table 3.1 Security estimates of the location portion of GeoPassNotes, based on guessing attacks under different threat models, including that of a known adversary that knows where the target lives and has vacationed.

However, it is interesting to note that even though the users saw a relationship, it was difficult in most cases for us to determine. We manually analyzed each location and note pair; the only relationship we could find was in 2 of the user's notes where they directly labelled their location. Direct location labels arguably provide the worst case for note security. When a user labels a location, and the location is compromised, the note can easily be guessed by compiling a small dictionary of location-specific terms.

One user created his/her annotated location-password prior to the recommendation. The other user was labelling a school. In this case, if an attacker discovers the location part of the password (the school), some of the first location-specific words might be “school”, “highschool”, “education”, etc. In this case, the note was “highschool”, and was placed nearby the school. When we noticed the first user of our study labelling the location, we decided to implement the “no labelling” recommendation for future users. In Section 3.5, we discuss how proactive checking by the system can be used to prevent direct labels.

Location Security Estimates

Working towards our goal of estimating the security of GeoPassNotes, we must estimate the security offered by the location component. For the location component of each annotated location-password, we consider the same adversary models as used by GeoPass [56]: local, known, and unknown. In the local adversary model, the adversary knows the system’s deployed location (i.e., a specific building) and guesses its surrounding area (i.e., a single city’s metropolitan area). If the location the user chose was in the same geographic region as our deployed system, it was considered “guessed” under the local adversary model. In the known adversary model, the adversary knows where the target lives and has vacationed, and in the unknown adversary model, the adversary does not have any information. The known adversary model success was estimated based on the users’ responses to a question in the background questionnaire asking what the significance of their chosen location was. If their answer indicated that they had been there before, we categorized the location as “known”. High estimates are based on all possible pixel locations on land being guessed (accounting for the system’s 21x21 error tolerance), and low estimates are based on well-known points of interest (POI) found on TripAdvisor [57]. We note that there is overlap between these adversary models. For example, all of the low estimates are subsets of the corresponding high estimates (e.g., "Low estimate" local is a subset of "High estimate" local). Then for each estimate, $Local \subset Known \subset Unknown$.

We categorize each of the locations that users chose as part of their annotated location-password in Table 3.1. The number of attacker guesses presented in Table 3.1 represent the estimated number of guesses an attacker would need to successfully guess a location that falls under each threat model [56].

To measure the actual security of the location component of annotated location-passwords, we must determine whether there exist patterns in user choice that might allow an adversary (unknown to the user, or someone who the user may know) to guess the user's secret location. We first plot the locations that our participants selected to determine geographic patterns (see Figure 3.2). Figure 3.2 indicates that the distribution of locations chosen is fairly well spread-out. No two users chose the same location. In general, the more populated areas of the Northern Hemisphere appear to be more popular. The most popular area was Southern Ontario. In the questionnaire, we asked participants to characterize the locations they chose by selecting what best described it. The results indicate that 29/30 users from our user study followed our recommendation of avoiding a place they lived or worked, and the most popular category was a place the participants had visited or wanted to visit. Many users also stated the location was one they didn't think anyone else but them would know about.

To further categorize the location component of the participant's annotated location-passwords, we asked them whether the place had any personal memory or attachment; 43% (13/30) reported yes. This indicates a large amount did not report any significance for the location they chose. Further free-form comments indicated that for most of these users, their location password was a place they have been before, but not a place they have been very often.

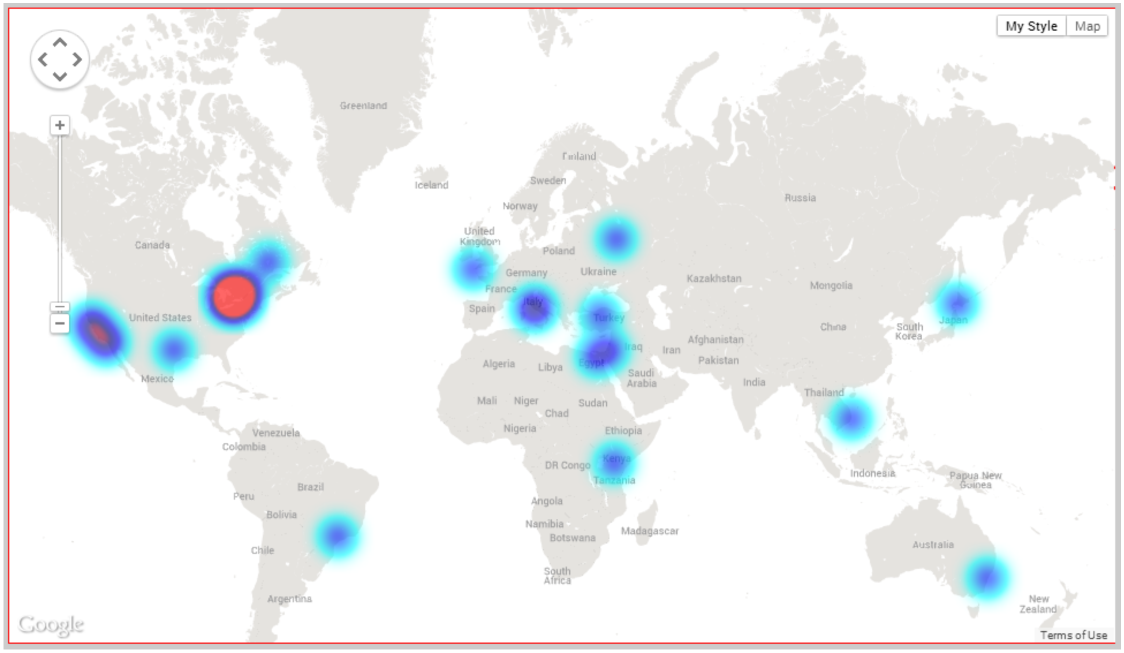


Fig. 3.2 Heat map of the locations chosen for GeoPassNotes.

Note Categorization

We noticed that a few notes appeared to be names, and a significant number were simple words. Surprisingly, a fairly large number looked like a password (containing mixed case, numbers, special characters, etc.). Based on our informal observations, we categorized the notes in terms of whether they exist in one of the following dictionaries:

- COCA (all). This dictionary should capture most words in the English language. We use the Contemporary Corpus of American English (COCA) [14], which contains 497,186 words.
- COCA (frequent). This dictionary should capture what we expect to be the most common words, i.e., the most common nouns, adjectives, and verbs in COCA [14]. As a vocabulary of 15,851 words was found to cover 97.8% of the Brown Corpus [48], we consider this as a likely vocabulary size. We sort the COCA unigram database and find the frequency of the $15,851^{st}$ word is 15,41. We only use the

nouns, adjectives, and verbs that have frequency higher than 1,541, which produces a dictionary of 14,674 words.

- Names. The dictionary contains common names, based on a list of babies born in the USA since 1960 [51]. There are 82,386 unique names in this list.
- Location-specific terms. The adversary could use custom information about the specific location chosen to guess the note. For example, if the location was a golf course, then one might assume that the name of the golf course (or some variation of it) could be chosen. To estimate the dictionary size, we consider the worst-case scenario for a street intersection: the user chooses the two street names, in both orders, with no spaces in between, and with all possible combinations of capitalization, resulting in 8 distinct notes.
- Low # characters. The dictionary contains all combinations of three or fewer lower-case characters. There are 17,576 unique notes in this list.

These dictionaries were created based on what we observed at least some participants associating with a location. If a note does not fall into one of the above categories, it is categorized as "Unclassified".

The results of classifying both the user's chosen notes and locations are summarized in Table 3.2. Notice that the column for "High estimate - Unknown" shows the categorization for the notes alone (since this location-password adversary model contains all possible locations). In Table 3.2, we can see that 16 of the user's notes fall into the "Unclassified" category, meaning that they do not fall into any of the defined dictionaries. We observed that some users chose phrases, sequences of words, mangled words (containing digits), and even one user chose a random string. Surprisingly few names were chosen; only 3 of the notes could be classified as names.

Note that there is some overlap between some of the note dictionaries in Table 3.2. COCA (frequent) is a subset of COCA (all). Many entries from the Names set are also

Note dictionary	**Location-password adversary models**					
	High estimates			**Low estimates**		
	U‡	**K**	**L**	**U**	**K**	**L**
COCA (all)	12	7	5	1	1	1
COCA (frequent)	5	3	3	1	1	1
Names	3	3	2	0	0	0
Location-specific	2†	1	1	0	0	0
Low # characters	1	1	1	1	1	1
Unclassified	16	10	2	2	1	0

Table 3.2 Categorization of the annotated location-passwords chosen by study participants. Note that there is overlap between some of the note dictionaries (see text). † One was prior to our recommendation to not directly label the location. ‡ U - Unknown, K - Known, L - Local. Refer to Section 3.4.1 for a description of these location-password adversary models.

contained in COCA. One of the location-specific words was “highschool”, which is also contained in COCA.

Security of Notes Alone

We simulated a password cracking attack against the notes collected for the purpose of (a) estimating the additional security they provide, and (b) comparing the notes to regular passwords. We used two popular password crackers:

- John the Ripper (JtR [45]), a popular open-source password cracker. JtR was configured to use a large password dictionary [15] while operating in wordlist mode using default word mangling rules, followed by incremental mode.
- Probabilistic Context-Free Grammars (PCFG) [62], which generates a guessing order that JtR can use. This method is considered to be better than JtR [31], but we found was inferior to the other algorithms we used, and as such we do not report on its results.

- Semantic Guesser [59], which generates guesses based on the semantic and grammar patterns of passwords. We run this guesser with and without word mangling (which modifies capitalization on word boundaries). To date, this is the most effective method at cracking passwords.

We used a large comprehensive password list as a dictionary available from Dazzlepod [15]. JtR was configured to use this dictionary while operating in wordlist mode using default word mangling rules. For comparison to the semantic guesser [59], we ran JtR until it made 3 billion guesses. Wordlist mode would not reach this alone, so once the wordlist had been exhausted we configured JtR to continue in incremental mode. We also used a common wordlist called DIC-0294 as used by Weir et al. [59]. However, we found that the Dazzlepod list outperformed this wordlist when cracking our notes. We discuss potential explanations for this in Section 3.6.

PCFG required training prior to its use. In order to obtain the best results, PCFG needed to be trained on a very large password list. We trained PCFG on the RockYou dataset [49], since this is the largest available plaintext password list (32 million). We configured PCFG to use the Dazzlepod list to generate our guesses based on the grammar rules generated through training. PCFG generated guesses which JtR used to crack the notes. The other two guessing algorithms far outperformed PCFG in terms of guessing results, as such we do not ficus on these results.

As demonstrated by Figure 3.3, both JtR using the Dazzlepod dictionary and the semantic guesser with word mangling rules applied (semantic_mangle) were the most effective at cracking the most notes, each correctly guessing 63.33% (19/30 notes). However, semantic_mangle started cracking much sooner than JtR (first cracked note was at 1,000 guesses for semantic_mangle as opposed to 707,000 for JtR).

We compare the security of the notes alone (see Figure 3.3) with the security of passwords in terms of percentage guessed within 3 billion guesses (see Figure 3.4). The

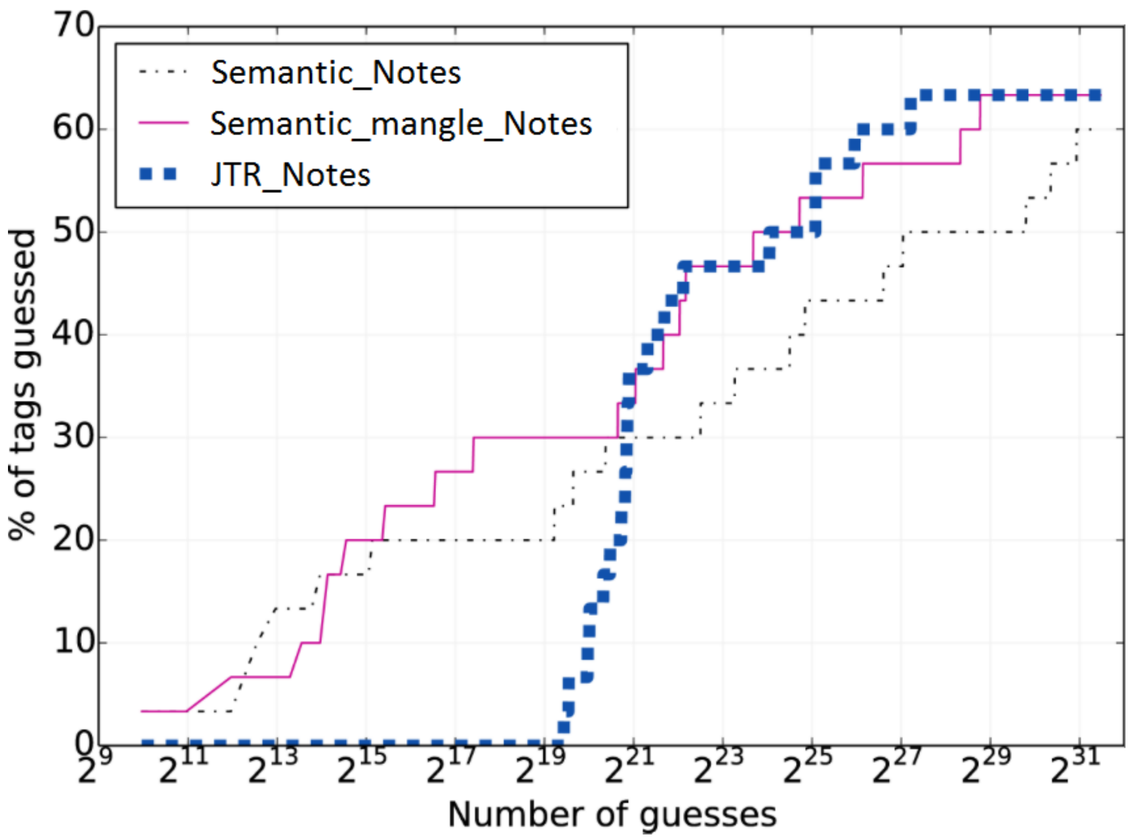


Fig. 3.3 JtR and semantic cracking results against the notes.

security of the notes is comparable to that of the MySpace passwords (63% guessed for notes vs. 65% for MySpace). However, the notes are not as secure as the LinkedIn passwords (only 25% were cracked), which is not surprising as users were told that the *combination* of note and location was their password.

We also used our note dictionaries (described in Section 3.4.1) to "guess" the notes. However,we found the password crackers outperformed our note dictionaries, which is why we focus on these results.

Security Estimate of GeoPassNotes

Next we evaluate the total combined security of GeoPassNotes, that is, the security of the location and the note together.

To perform a reasonable security estimate, we must consider how an attacker would approach guessing an annotated location password. For optimal security, a GeoPassNotes system should hash the combination of location and note (since both of them together make up the credential), and not provide any feedback to the user until both have been entered. This way, the attacker must then correctly guess *both* the location and note without any feedback to indicate they have guessed the correct location.

A sensible approach for an attacker attempting to guess an annotated location password would be to guess the location component using the location dictionaries discussed in Section 3.4.1, ordered based on their relative size and guessing efficiency, as follows: (1) POI–local, (2) All land–local, (3) POI–known, (4) All land–known, (5) POI–unknown, (6) All land–unknown. Of course, for each of the location guesses in this ordering, the attacker must guess the note as well. An attacker should choose a maximum number of annotation guesses to make per location (e.g., 3 billion), meaning that for each location guess, they would guess at most this maximum number of notes. For each failed location guess, they would need to guess exactly this maximum number of notes, even if one of the note guesses were correct (as it is the combination of location and note that form a password).

A sensible approach for an attacker attempting to guess a annotated location-password would be to first guess the location component using the adversary models discussed in Section 3.4.1 based on their relative size and guessing efficiency. This means they should guess in the following order: low estimate local, high estimate local, low estimate known, high estimate known, low estimate unknown, and end with high estimate unknown. When combining the security of the notes with the locations, we take into account that the attacker has exhausted all of the previous entries in this ordering.

The estimates in Figure 3.4 represent the total estimated security for all of the annotated location passwords gathered in our study. The number of total guesses for an

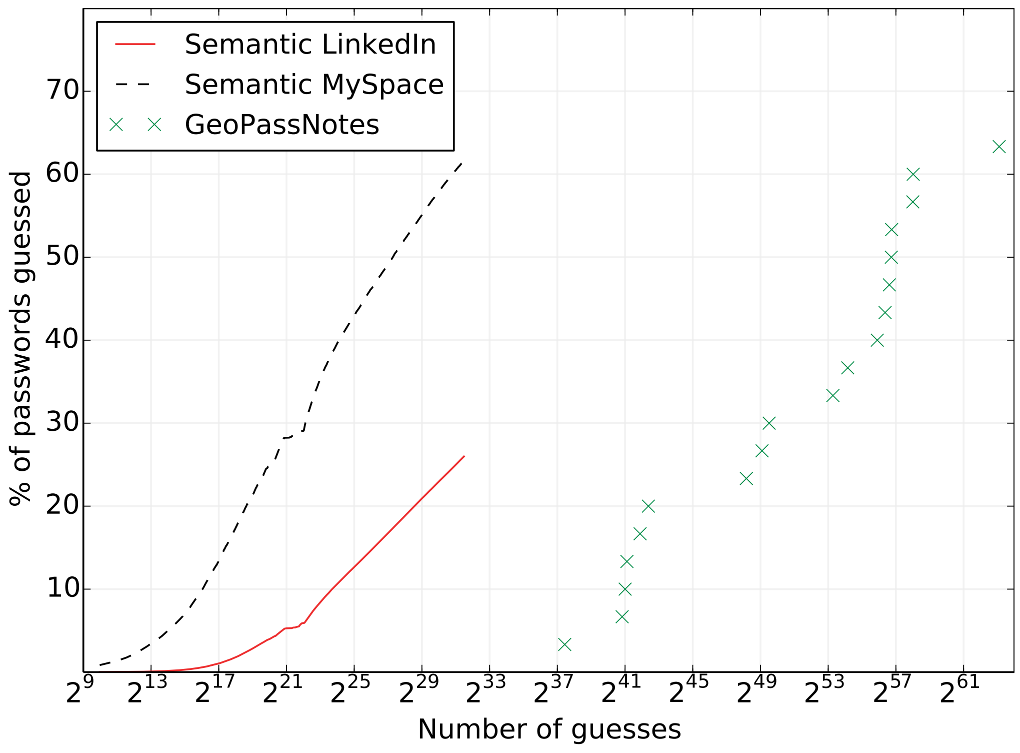


Fig. 3.4 Results of GeoPassNotes security estimate compared with the semantic (with mangling) attack against the MySpace and LinkedIn password sets.

annotated location password is computed by multiplying the number of guesses required by semantic (with mangling) to guess the annotation with the sum of the sizes of all location password dictionaries exhausted and half of the size of the last dictionary used. We consider half of the last location dictionary as the annotated location password can be anywhere within it (as the entries within each are not ordered). For example, to guess an annotated location password whose location component exists in the All land - local dictionary, the dictionary size would be the size of the POI - local dictionary plus 50% of the size of the All land - local dictionary, times the number of the semantic attack's attempts to guess the corresponding note.

We compare our estimates for GeoPassNotes with the results of password guessing with semantic with mangling against the MySpace and LinkedIn leaked password sets. This will provide a baseline for the strength of text passwords in practice as compared to our conservative estimate of the security of GeoPassNotes. For the location passwords that would be guessed by each of the POI dictionaries, only 3 notes were guessed. The first POI annotated location password was not guessed until approximately 2^{37} guesses had been exhausted whereas $> 30\%$ of the MySpace passwords and $> 4\%$ of the LinkedIn passwords were correctly guessed within 2^{20} attempts.

The weakest two annotated location passwords are estimated to be guessed after 2^{41} guesses (these two fell into our POI dictionaries), which is still much better than the weakest passwords from both MySpace and LinkedIn data sets. A comparable number of the LinkedIn passwords are guessed within approximately 2^{23} guesses.

In total, 18 users chose places that they have been before, and thus their location would be vulnerable to the known adversary attack without the note; however, only 8/18 of these would be guessed by our note dictionaries (refer to Table 3.2). Using the password cracking approach, an attacker could successfully guess 13/18 notes for the high estimate known threat model.

Other Security Considerations

As with many graphical password schemes, shoulder surfing is an inherent and apparent weakness. We did not specifically conduct a shoulder-surfing study as it would only serve to confirm the obvious; that the location element of the annotated location-password is observable. Although adding notes to location-passwords strengthens their resistance to shoulder surfing attacks, they seem as observable as a standard password. If an attacker observes the location, they must still guess the note. Thus we have modelled this type of attack in Figure 3.3, which represents the situation where the location part of the annotated location-password has been compromised, and all that remains is the note. Whether this is because the attacker guessed the location or shoulder surfed it is irrelevant, the end result is still the same; the attacker must still guess the note. Through our analysis of note security (recall Section 3.4.1) we were able to conclude that the weakest note is guessed in approximately 2^{13} guesses. The strongest note that was cracked by JtR was guessed in approximately 2^{27} guesses. Some notes were not guessed in 2^{32} (3 billion) guesses.

Comparison to GeoPass Security

Through our analysis of locations, notes and the variety of data collected, we have been able to distinguish what makes a good annotated location-password. One question we are particularly interested in is "How much *more* security do annotated location-passwords offer compared to the location-passwords of GeoPass [56]". Table 3.1 provides the security estimates for the location portions alone (from the annotated location-passwords) under different adversary models (the same as in GeoPass [56]). The maximum security offered by these estimates is the case where the attacker has no knowledge of the system or user, and the location has no known or observable relationship to the note. In this case, the security offered is 36.88 bits. However, when adding a note to the location, we were able to increase the maximum security of the aforementioned case to approximately 65 bits. For the most insecure low estimate location-passwords, the security increased from ~16 bits to ~38 bits. Thus, it is evident that the introduction of notes greatly increases the security offered by location-passwords. The benefit of having a note and password is that they are both required to successfully login. If an attacker were to gather one of the two (e.g., the note or the location, but not both), they would not be able to authenticate.

3.4.2 Usability Analysis

We discuss the memorability, acceptability, usability, and login times of the system, comparing where appropriate to non-annotated location-passwords.

Memorability

Our study demonstrated that annotated location-passwords have similarly high memorability to non-annotated location-passwords [56]. We asked participants questions with Likert-scale responses from 1 (strongly disagree), to 5 (strongly agree). The majority of users (93%) reported no trouble in remembering their annotated location-password (as demonstrated by Figure 3.5).

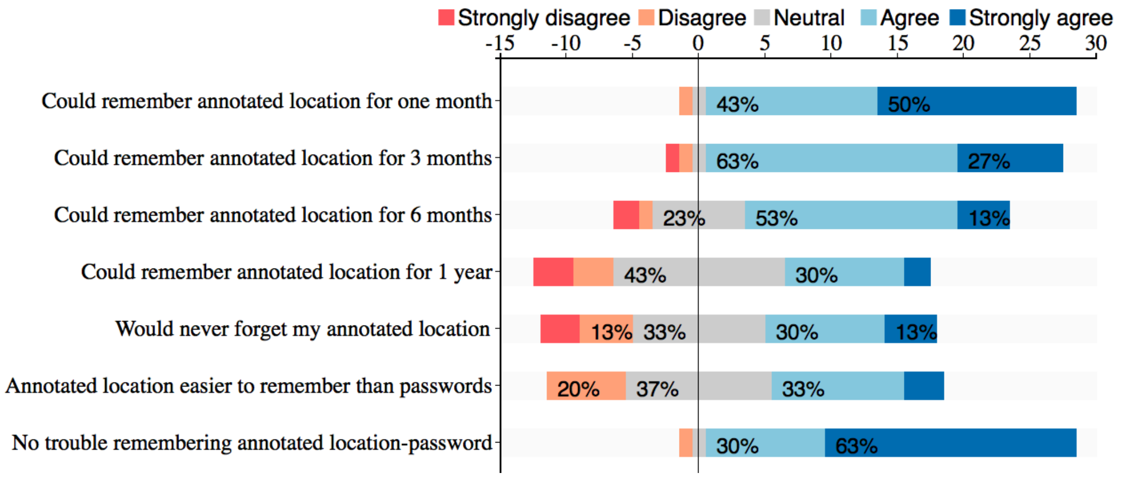


Fig. 3.5 Likert scale questionnaire responses to the associated questions pertaining to memorability.

This response complements our analysis of failed logins, which indicated that over the course of all three sessions, only 4/30 users had a failed login; overall, there were very few failed logins (see Figure 3.6). Compared to GeoPass (which can be easily done, as our study design, population, and population size mirrors that of the GeoPass study [56]), GeoPassNotes have a a slightly higher number of login failures (22 vs. 19) but no password resets (0 vs. 2).

No users forgot/reset their annotated location-password. When asked if they could remember their annotated location-passwords for up to 3 and 6 months, most users (see Figure 3.5) feel they would have no trouble remembering. In future work, it might be interesting to include an additional session to test the memorability of annotated location-

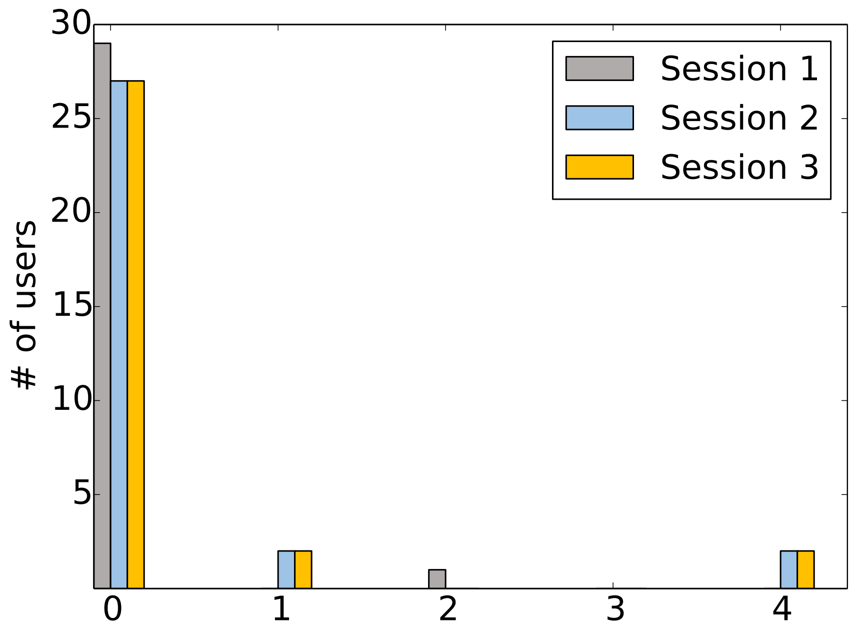


Fig. 3.6 Failed login attempts per session.

passwords over longer periods of time. We also asked participants if, at any point in time, they wrote down a part of their annotated location-passwords. Four participants out of 30 (13%) mentioned they wrote down some part of their annotated location-password. All four participants wrote down information pertaining to the name of the location they chose. Two users wrote down the note they chose; one exactly and the other in his native language. Another wrote down the countries and cities he saw on the map as he zoomed into his location. The results of GeoPass do not indicate whether or not any participants wrote down their location-passwords, although one user referred to their recording in session 3 after experiencing a problem setting his/her marker [56].

Acceptability

We asked participants questions with Likert-scale responses from 1 (strongly disagree), to 5 (strongly agree). Most users agreed that they would use this method for some of their accounts if they knew it was more secure than passwords (see Figure 3.7c).

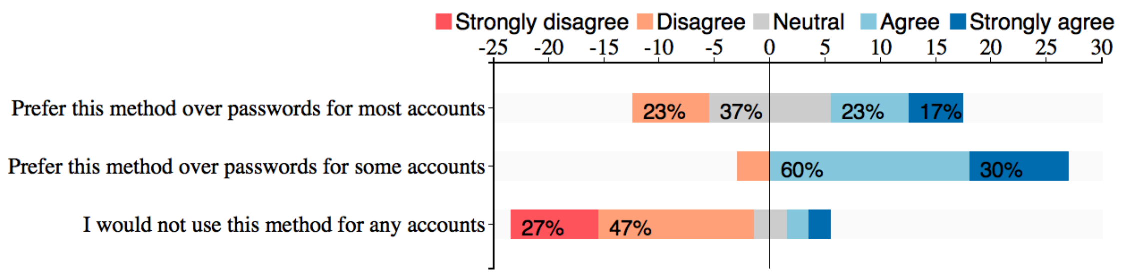


Fig. 3.7 Likert scale questionnaire responses to the associated questions pertaining to acceptability.

When asked if they would use this method for most of their accounts, most remained neutral, neither agreeing or disagreeing (see Figure 3.7a). However, most (90%) users agreed they would use it for some accounts (see Figure 3.7b). One user mentioned that while the login times are longer, she would still prefer to use the system because of the security that it offers. There are other users that commented on the security as well, stating that it is a “better and more secure way of creating passwords”.

We also asked participants a number of questions relating to their experience with GeoPassNotes. When asked how difficult it was for them to use the system, the majority of users reported not having any difficulty (see Figure 3.8c). Despite the increased time to login, users seemed accepting of the system, one of which mentioned “Even though it takes longer, I would use it because it is more secure”. Other comments included “fun” and “a cool way to authenticate”.

Usability

Further complementing our use-case of infrequently used accounts (e.g., once per week), the majority of users (90%) indicated that they could easily use this method every week (see Figure 3.8b). Very few reported that the method was difficult to use (see Figure 3.8c). More users indicated they could easily use the method every week than every day most likely because 40% of users reported that this method was too time-consuming (see Figure 3.8d). Furthermore, most of the users (76%) reported not having any difficulty navigating back to the location they chose (see Figure 3.8f). We plan to explore ways to make annotated location-passwords less time-consuming.

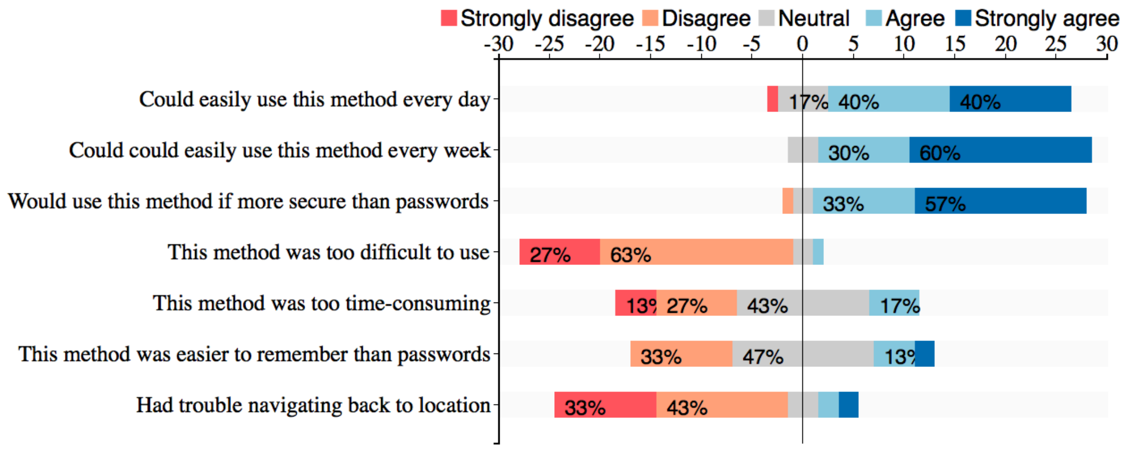


Fig. 3.8 Likert scale questionnaire responses to the associated questions pertaining to usability.

Login Times

On average, login times were 32, 37, and 47 seconds for session 1, 2, and 3, respectively. These averages are raised by five users with long login times (> 150 seconds). As such, we also examined the median login times which were 26, 33, and 36 seconds for sessions 1, 2, and 3, respectively. The login times for GeoPass (Figure 4 from [56]) can be directly compared to Figure 3.9; GeoPass median times are lower than that of GeoPassNotes (by 1, 3, and 11 seconds for session 1, session 2, and session 3 respectively).

We also note that there were interesting qualitative observations from our study that may have contributed to some of the outliers in our times analysis (see Figure 3.9). One participant answered a phone call at the beginning of the login phase, which led to increased login time. The network in the laboratory that we ran our experiments in had a great deal of latency; it took a long time for some users to load Google Maps, resulting in increased login times. Network usage on the campus meant that there was a great deal of jitter in the amount of latency, so unfortunately it is difficult to quantify a specific delay. For the future, a wired internet connection opposed to wireless will be used.

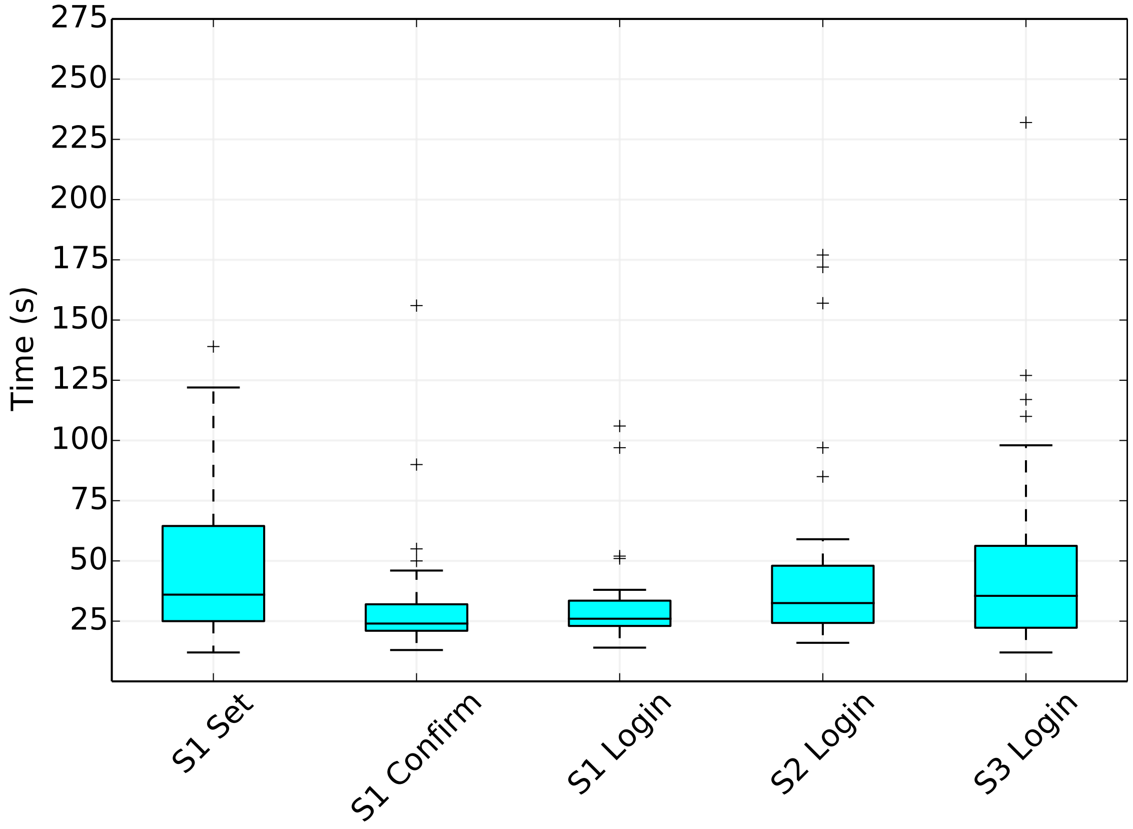


Fig. 3.9 Login times for each session.

3.5 Resulting System and Policy Recommendations

Based on our findings, we suggest a set of recommended policies for future implementations of annotated location-passwords. These recommendations are as follows:

- *Avoid choosing a location that you have previously worked or lived.*

This recommendation is intended to prevent users from choosing locations that are easily guessed (i.e., the known and local adversary models). The results of our study indicate that all users followed the recommendation not to choose a location previously lived or worked at. This was a positive result, and it did not seem to affect the overall usability of the system. As mentioned in Section 3.4.1, the first user directly labelled a location (the names of two intersecting streets). As such, we also recommend the following policy:

- *Avoid directly labelling a location in the note (e.g., do not note Daytona Beach with "Daytona" or "beach").*

We used this recommendation for the remainder of the study, which prevented the majority of users from labelling their locations. Similar to our previous recommendation, the usability of the system did not appear to be affected in any way, while the security was vastly increased. One user of our study also chose a three character note. We did not disallow this as part of the study was to collect a distribution of notes and see what kinds of things users note locations with. As such, we recommend the following policy:

- *Avoid choosing a note with a low number of characters (e.g., less than 4).*

We plan to implement proactive checking for these policies in future versions of GeoPass-

Notes. Disallowing short notes is easy, but the others might not seem as obvious. Using Google API's built in tools, we can perform on-the-fly checks of the location (e.g., suppose a note is set on the CN Tower, and the user tries to note it with "CN Tower", we can prohibit this by performing a reverse geolocation of the coordinates of the marker and seeing that the CN Tower in Toronto, Canada was selected). This is easily done as Google Maps API allows extensive customization with regards to geolocation. The response to a reverse geolocation request would be an array consisting of different address components (e.g., formatted address, short name, long name, postal code, etc.). In our example of the CN Tower, the note “CN Tower” would exist in the long name of the array returned by the reverse geolocation request. Alternatively, the user’s note could be entered in the Google Maps Search API to see if the location is the same as the note text.

Enforcing that a user does not choose a place lived or worked before is more challenging. Relevant information could be collected as part of the registration/enrollment process such as a user’s city of birth. Then, it would be possible to restrict markers placed in that location. The local adversary model could be easily prevented through a coordinate lookup based on IP geolocation results.

3.6 Discussion

We found that annotating a location seems to slightly improve memorability, (GeoPass had 19 login failures [56], vs. 22 for GeoPassNotes); also, GeoPass had 2 passwords reset over the study, vs. 0 for GeoPassNotes). This raises the question of whether the notes might actually help users remember their locations. Our initial thoughts were that an event-specific memory combined with a note would create a highly memorable way for users to authenticate. However, after gathering results from our study, we noticed that some participants randomly chose places that looked interesting on the map. This could be due to the formation of the land, unique or funny words, or just randomly zooming into a place until they found one they liked. Therefore, it may be possible that it is not the significance of the location, but rather locations in general that people remember. This leads us to consider whether annotating a randomly generated location might yield the same positive results that we have seen with GeoPassNotes? This is an interesting future area of research we plan to explore.

In order to determine the security that notes alone provided, we ran various password cracking algorithms against them using two different input dictionaries: a comprehensive wordlist[63] and Dazzlepod's dictionary of the most common passwords [15]. Contrary to our expectations, using the Dazzlepod dictionary in these cracking algorithms cracked the most of the notes (63% compared to 43% with the DIC-0294 wordlist). Many notes were also observed to have a strong similarity to passwords. While we did not give any password policies or guidelines relating to the notes, it seems that many of the notes resembled more of a password than a word. This is an interesting topic to further examine in the future.

In our questionnaire, we asked users why they chose the note they did. We also analysed if there were any observable relationships between the note and the location. From this, we were able to say that 24 out of 30 users' notes held significance for them to the location in some way. However, it is interesting to look more deeply into user's individual associations between notes and places as they are generally not apparent to others; only 2 notes had an observable association to the location. This implies that while the note is somehow associated, it is not easily tied to the location by anyone but the users themselves.

We also noticed that the majority of users seemed to navigate to their chosen locations the same way every time. In the interest of determining whether a user's journey could be used to enhance the strength of GeoPass and its variants, we analyzed each user's navigation strategies across every session to see if this could be an added security requirement.

To evaluate this, we considered the sequence of navigation events (e.g., click, drag, double-click, search, or scroll) as the *journey*. We then tested the journey from the location password confirmation for equality with the journeys collected in each session. If we were to require the exact journey to be used for logins, then 70%, 63%, and 60% of users would have successful logins. Even though we did not enforce this in our study, we were interested if it could potentially be added in the future as another login requirement. Our findings indicate it is not consistent enough to be a usable login requirement (there is too much variance in the way users navigate back to their location from login to login). Next, we evaluated how many users would successfully authenticate if the subsequent journey was off by at most one navigation event. For example, if the journey was (drag, double click, search), then (drag, search) would also be accepted. This changed the results to be 98% successful login rates across all three sessions. However, the security impact of this requirement would be very little.

After all of the research we have conducted on using locations to authenticate, it is not clear why locations are so memorable. Is it the location itself, a special event and/or memory that occurred, or something else? One interesting idea is that users might be remembering parts of the journey associated with their location. Sometimes, when humans are driving, searching for a location, they can recall where they are going based on their experiences along the way (e.g., turn right at the big tree, then left at the second stop sign, and finally it's the third house on the right). Since the majority of users use the same journey for each login, it might be the key factor in making GeoPassNotes so memorable. It would be very interesting to research this more, and discover if the user's journey effects the memorability of the location they end up choosing.

3.7 Conclusions and Future Work

We implemented and evaluated a security enhancement for location-passwords called GeoPassNotes. Through a 30 person user study, we found that annotated location-passwords remain as memorable as their non-annotated counterparts, however yield much more security against attacks when our recommendations are applied. Compared to unannotated location-passwords [56], we found notes only slightly increase login time. Although annotated location-passwords are highly memorable, and may have stronger security than text passwords, it is still only a viable option for accounts where logins are infrequent (e.g., to replace the role of text passwords or secondary/fallback authentication in financial accounts, where logins have been found to average 1.3 times/week [26]). This is because login times for GeoPassNotes are high (median login time 26-36 seconds) vs. under 10 seconds for traditional passwords. The results of our study also show that users are most open to using this system once/week.

Future work includes studies to gain a better understanding of why annotated location-passwords have such high memorability as the reasons may potentially help inform password guidelines in other authentication systems.

CHAPTER 4

PASSMOD

4.1 Introduction

Despite their inherent weaknesses, password usage has not declined in the past few years [12]. They are still the most widely used form of authentication due to their deployability and simplicity [35]. However, we have entered an age of evolution with regards to computers and the Internet. Every year, computers become faster, smarter, and smaller, allowing attackers to have more tools at their disposal. Many organizations are finally becoming more aware of the dangers of dealing with passwords. However, they still continue to be used. The current state of Internet security is a tradeoff between security, usability, and deployability [10]. Every decision that is made is a balance between how easy it is for the end user, how much protection is offered, and how easy it is to deploy. Typically, systems will excel in one type of benefit, but lack in others (e.g., passwords excel in deployability but not as much in security [10]).

In order to counter against attackers, many sites now implement strict password composition policies and procedures. Some of these include multiple password segments required, password history restrictions, and password length requirements. It is also recommended that participants use a different password for every site they visit. These recommendations make it nearly impossible for participants to create and remember all of their passwords (up to 30 in some cases).

We have also seen pushes to develop alternative authentication schemes that are more memorable than passwords. Some of these are graphical password schemes, like the one we presented in Chapter 3. The goal of these systems is to authenticate easily using different authentication tokens for each site. The interference of these different tokens does not appear to be an issue when distinct background images are used for each account [13]. Graphical passwords are very memorable, but their use is limited due to exaggerated login times compared with passwords. They also suffer from shoulder surfing attacks, an issue that isn't as much of an issue with passwords. Passwords are the most widely used [35], but are not a good option going forward. Password cracking approaches continue to become better and more efficient, promoting administrators to enforce more strict password policies [20]. This results in unsafe coping strategies such as password reuse, minimally different passwords, and writing passwords down. As such, we were interested in increasing the security of passwords with little or no reduction in their memorability such that participants can remember more passwords without needing to reuse the same one.

We designed and implemented a system called PassMod to strengthen participants' passwords in meaningful ways to increase memorability. The purpose of PassMod is to take a user's initial, weak password, and suggest a modified version of that password to the user. The modifications performed are not just character or case changes, but rather word-level changes. PassMod analyzes and interprets the meaning of the participants original password, and tries to make suggestions that are meaningful and sensible. We hypothesize that if the suggestions make sense to the user, they will be easier to remember. This system is our effort at trying to increase security of passwords without sacrificing the memorability of the original password. The semantic attack of Veras et al. [59]

is currently the best known cracking method, capable of cracking previously unseen passwords. PassMod is an effort at trying to create a password that is secure against the attacks described by Veras et al. [59].

We conducted an online user study to test the usability and security of PassMod to try to recreate how participants use passwords in an everyday situation. The study contained three sessions held over the course of 8-9 days. In total, 43 participants completed session 1, 42 participants completed session 2, and 35 participants completed all three sessions. Participants were also invited to partake in an optional session 4 held one month after the completion of session 3; 26 participants completed this session. Of the original set of participants in session 1, 72% were female and 27% were male with the majority (~52%) being between the ages of 20-25. The results indicate PassMod is not only effective at increasing the security of passwords, but also at retaining the memorability of the original password. Of the 35 participants that completed all 3 sessions, 2 (5%) forgot their passwords and chose to reset after 3 failed attempts. Out of the 43 total participants, 6 (14%) forgot their passwords and had to reset. In total, 8/43 (19%) admitted to writing down some or all of their passwords. Of the 26 participants that returned one month later, 21 remembered their password and 5 forgot (8 of these participants had previously secure passwords). The majority of participants (52%) agreed that they could easily use this method every day, and 64% would if they knew it was making their accounts more secure. More than half of participants (54%) also found this system easy to use. 58% of participants thought this system would make their accounts more secure.

PassMod differs itself from other password modification systems through its use of password analysis. PassMod first parses participants' passwords to learn their semantic structure. Using this structure, PassMod will make one or more modifications, but tries to preserve the memorability of the original password by modifying it in a way that preserves the semantic structure, which hopefully makes sense to the user. Previous approaches were solely focused on increasing security and resistance to guessing attacks; our method focuses on security and memorability, not just one or the other.

We also conducted an analysis to test the security of the passwords output by PassMod. The system was able to significantly strengthen participants' passwords. Previous attempts at strengthening passwords have been known to be conducive to certain types of attacks [47]. Previous attempts have also been seen to have more of a focus on security rather than memorability. Our approach focuses on maintaining the memorability of the password while also increasing its security. Houshmand et al. [27] did not conduct a usability study; as such, we cannot compare usability metrics between PassMod and their strengthening approach. Forget et al. [21, 22] conducted a user study to test their password modification system. Their system contained five different experimental conditions representing different types of modifications performed. Participants were tasked with creating and confirming a password, completing a questionnaire, and then logging in with their password. This is very similar to the structure of our session 1, as such, we only compare those results. The results indicate that memorability was very high, with login rates of 98%, 93%, 99%, 98%, and 94% for all 5 of their experimental conditions. These success rates are higher than those collected in PassMod which saw login rates of 86% for session 1. Forget et al. indicated there were very few login failures throughout the course of their study (as indicated by their high success rates) but do not explicitly state how many password resets there were. Furthermore, they do not mention if they collected data on whether participants wrote down their passwords or not. When we look at the security of the passwords output by their system, none were able to be cracked by John the Ripper operating in dictionary mode within 40 million guesses; PassMod was able to strengthen passwords to the point that most popular password crackers were only able to crack, at most, 5% of strengthened passwords within 3 billion guesses. While slightly more strengthened passwords were able to be cracked in PassMod, we used a larger number of cracking techniques, which are currently considered the best [30]. The initial passwords input by the PassMod system were on average 10 characters long and 33% percent were able to be cracked, whereas the strengthened passwords collected from our study were, on average, 15 characters long and none could be cracked using the same method of Veras et al. [59]. The best attack against the strengthened passwords came from PCFG, cracking 5% of them within 3 billion attempts.

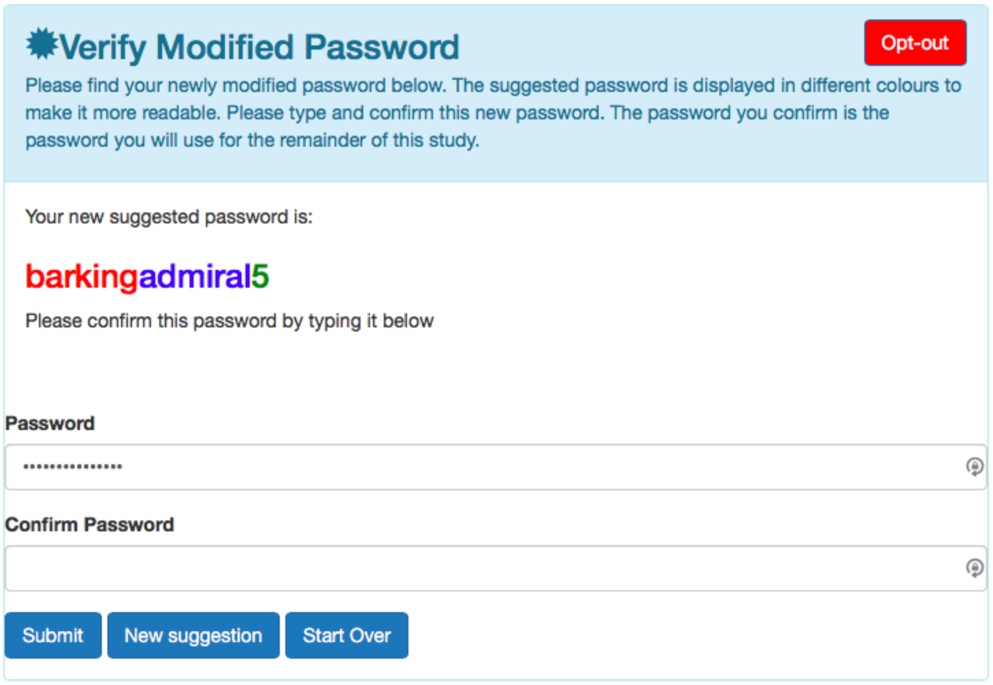
The rest of this Chapter is organized as follows: Section 4.2 provides a detailed overview of the system such as how it analyzes and interprets passwords, modifies a password, and evaluates modifications; Section 4.3 describes the user study we conducted to collect real use data from participants that tested our system; Section 4.4.1 analyzes the theoretical and practical security of the system; Section 4.4.2 describes the usability results of our user study; and Section 4.5 talks about reasons for design choices that were made, implementation decisions, and usability-security tradeoffs.

4.2 System Design

PassMod was designed with the purpose of making a password more secure without sacrificing memorability. When a user chooses a password, they go through a thought process, an internal selection process that combines the user's perceived security of the site and a password they can remember [11]. We aim to preserve the memorability of the original password. Our system interprets the password that was typed, and looks for other words, structures, or variations of the original password that are less probable among a large set of collected passwords. The goal is to keep the newly suggested password as similar to the original as possible, while increasing its security. The following sections provide a brief overview of the interface and design choices, explain how the underlying system works at a high level, how we analyze the user's password, followed by a description of the exact modifications we make to the password to make it more secure.

4.2.1 Interface Design

Through preliminary pilot testing with experienced computer participants, visualization specialists, as well as non-technical participants, we were able to refine the interface to be as user-friendly and approachable as possible. Figure 4.1 demonstrates the interface we created as well as an example of a modification output by the system after a user has entered a password.


Verify Modified Password

Opt-out

Please find your newly modified password below. The suggested password is displayed in different colours to make it more readable. Please type and confirm this new password. The password you confirm is the password you will use for the remainder of this study.

Your new suggested password is:

barkingadmiral5

Please confirm this password by typing it below

Password

•••••••••••••••

Confirm Password

Fig. 4.1 Modification of the password “barkingdog5” displayed to the user in the PassMod system.

As shown in Figure 4.1, the most notable design choice is the use of different, distinct colours to represent the different segments of a participants suggested password. Also present are three buttons allowing the user to submit their passwords, generate a new suggestion from the original password they entered, or start over from scratch (taking them back to the start).

4.2.2 Grammar Generation

The underlying system is based on the semantic cracking system and grammar of Veras et al. [59]. Their system is capable of cracking previously unseen English passwords with greater success rates than the previously best known methods (PCFG [27], JtR [45]) [30, 59]. As such, we were interested in creating a system that could resist the cracking efforts of the semantic cracker. PassMod uses the same underlying password grammar as the semantic system of Veras et al. [59]. The password grammar is trained on a large collection of passwords (RockYou, MySpace, Yahoo, LinkedIn, Gawkr, phphBB, and eHarmony). The grammar is generated by segmenting the passwords into constituent terminals (words, digits, and symbols), classifying segments into non terminal semantic categories (e.g., NN_animal, VB_movement), then training a shallow PCFG to learn probabilistic grammar rules which will produce the observed structures. We also use some external resources for assisting in strengthening participants' passwords in the event that the grammar is not sufficient. We use WordNet to find synonyms of words in the event that no synonyms can be found in the grammar (further described in later sections). We also use COCA to ensure the suggestions we make are part of the top 75,000 most common words (three times humans average vocabulary size [41]). This ensures that the suggestions we make are at least common enough in the English language to make sense to the user. There are many words that participants might not have ever seen before, and we do not want to make suggestions using those words. The process of generating the grammar is detailed below.

The first step in generating the grammar is finding a very large collection of passwords to train it on. For our system, we decided the more passwords we could acquire, the better.

As such, we collected RockYou [49], Yahoo [5], LinkedIn [6], eHarmony, MySpace[49], Gawkr [37], and phpBB [49]. This resulted in a total of 40,558,327 passwords. Having more passwords from different websites is beneficial as it provides a larger, more comprehensive grammar. If our grammar was only generated from one password set (i.e., RockYou), the grammar would be influenced by that that collection's password policy. As such, any password created with a different policy in mind would not have its strength accurately assessed using the password grammar. We opted for a larger password collection using a multitude of password sets containing different password policies so we can accurately assess the strength of new passwords. The choice of using these specific password collections is because they were available in clear text (i.e., not hashed / encrypted).

We use the same password grammar generation process as used by Veras et al. [59], as outlined in the following paragraphs.

For each password, the first step in generating a grammar that would be beneficial to us was to parse the passwords. Parsing is the process of determining the most probable structure of a password. It takes place in two phases. First, the password is segmented into the most probable syntactic components (words, numbers, symbols, etc.), then the grammar is used to determine the most probable rule which could generate the observed password. The initial segmentation step works by computing all possible segmentations of the input password and then by using COCA unigrams, bigrams, and trigrams to find the most probable segmentation. We refer to all of the segments that make up any given password as the password segment set. Each segment set is stored in a database and linked to the original password it was produced from.

The next step is to part of speech (POS) tag each segment. Part of speech tagging is used in creating structures of passwords. A base structure (or structure) is a grammatical representation of the password and its segments. For example, the password "ilovebob" can be divided into its segments “i, love, and bob”. Once each of these segments are tagged, they can be put together again to form a base structure. In order to create a structure, we must know whether each segment is a noun, verb, or gap. A gap segment refers to a non-word segment, such as numbers or symbol. Segments are tagged using a combination of taggers including Claws [23], WordNet [9], and a backoff tagger [9]. The POS tags are stored in the database and also linked with the original password and segments.

After we have the segments and POS tags for all of the passwords, we must generate semantic categories and eventually, base structures. This stage takes the POS-tagged segments and builds a rule for the password. This stage also uses dictionaries of words to try to assign more specific categories to groups of words. For example, we used a dictionary of cities to assign cities this category. Suppose the password “ilovebob2012” was input. This password would be divided into its segments “i, love, bob, 2012”. Next, each non-word segment (referred to from hereon as a “gap” segment) would be POS tagged “{i->ppis1}{love->vvz}{bob->mname}”. All of the results are stored in a database and linked with the original password. Once all of the structures are stored in a database, a generalization function is performed to generalize more specific words. For example, the word “husky” would most likely not become its own category. This word can be encompassed by a more general term such as “dog” or “canine”. This generalization is user-specified, and is used to influence a tree cut using Li and Abe’s Tree Cut model [39]. We conducted tests to determine which cut would be the most appropriate for our purposes. In order to do this, we generated four different grammars based on four different tree cuts: specific, normal, general, and very general. We then ran password cracking attacks against the LinkedIn and MySpace passwords to test which grammar would conduct the best cracking attack. Our motivation behind using a password cracking attack is that an attacker would likely use the grammar that performed the best for cracking purposes. As such, we want to use the same grammar for creating more secure passwords. It was seen that the very general grammar performed the best, so this is the one we went with. This resulted in more general structures (e.g., canine instead of husky). It also resulted in extremely common words such as “love” having their own category.

Each category is referred to as a “nonterminal”, and is based on the level of tree cut used. The nonterminal is essentially a placeholder for words, numbers, characters, etc. in the password’s rule. For each nonterminal, a list of possible substitutions (i.e., terminals) is compiled. These are the words that can be substituted in for the given nonterminal, each containing probabilities relative to one another. Since the probabilities of the terminals are only relative to others within the same nonterminal category, the grammar is known as “context-free”. For example, suppose we have a noninterminal $vvi_s.love$. The only word within this nonterminal set is the word “love” with a probability of 1.0. The reason the probability is 1.0 is because there are no other words in the training data that share the same nonterminal category.

A parser is then trained on the base structures, learning a PCFG which generates passwords. A “structure” here can be thought of as a single level grammar rule (password$\rightarrow$structure) in a traditional PCFG. Structures are all segments of a password, in order, assigned a probability that is relative to all other structures (i.e., $total\_structures$). Any time a structure is generated more than once (e.g., from two different passwords), the probability of that structure increases. The more probable a structure is, the weaker the password that uses that structure will be. This process is done for all sets of segments (once for every single input password). The same algorithm also creates lists of words (i.e., terminals) for each given category (i.e., nonterminal). The words are parsed out of all of the input passwords. These words are then assigned a probability relative to all other words in the same list. For example, one list may contain all nouns, whereas the probability of each noun is relative to the total number of nouns in that list. In another example, we have the semantic category "vvz_s.love". In this case, the word "love" was seen so many times in the password grammar that it has been assigned its own category. The only word in this list is "love" and it is assigned a probability of "1.0". The total number of words contained in each list is referred to as n_{terms}.

The collection of all rules, nonterminals, terminals, and associated probability information is referred to as the "password grammar".

4.2.3 High-level Description

Once the password grammar has been generated, the next step is to assess a participants' password and determine if it needs strengthening (see Figure 4.2).

The password grammar described in Section 4.2.2 is used in this work to:

Assess Password Strength. Determine if a user's password is weak or strong by computing the probabilities of all its segments and comparing to a predetermined threshold.

Modify Existing Password. If a user's password is deemed weak, modify it by using a sequence of insertions / modifications until the overall probability falls below a prespecified threshold.

The high-level modification algorithm operates in the following way:

Stage 1a,b,c. User enters their password for strength to be assessed. The password grammar is loaded for use in determining password probability and generating suggestions. WordNet and COCA resources are loaded for use in later stages.

Stage 2. Check the initial password against a blacklist of ~3,500 of the most common passwords (described further in Section 4.4.1).

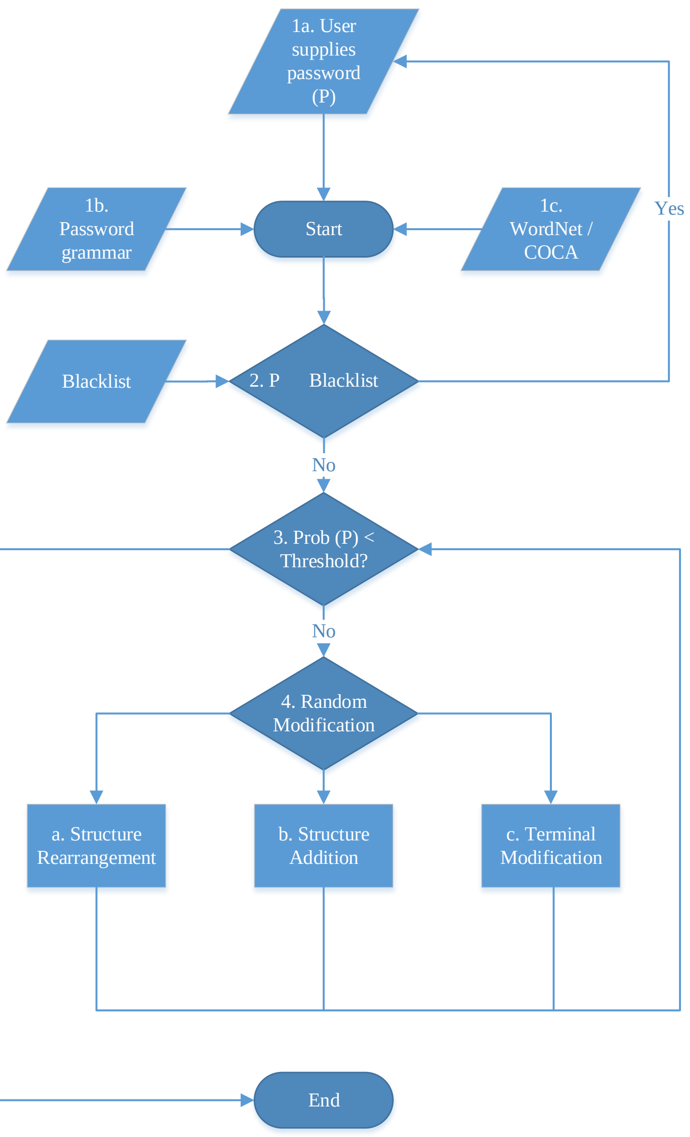


Fig. 4.2 Flowchart describing the PassMod strengthening algorithm.

Stage 3. Check the probability of the password against a threshold to see if it is above the threshold and requires modification.

Stage 4a,b,c. Perform a randomized modification to the original input password to get a modified password.

Stage 5. Recheck the probability of the modified password. If the password's strength does not meet the required threshold, call the main function again using the modified password as the input password.

The result is a modified password with a guaranteed probability lower than the threshold. This password is suggested to the user.

The following sections describe the stages of the strengthening algorithm as they appear in Figure 4.2:

Stage 1.

The user first enters their password to be assessed. During the process of assessment, various resources are required by the system which are also loaded during this time. Theses resources include WordNet synsets, COCA unigram, bigram, and trigram lists, and the password grammar itself. In this case, only the "very general" grammar is loaded as we evaluated it to provide the best information for our purposes (recall Section 4.2.2). The password grammar does not need to be loaded into memory in order to function, however, we found that we were able to significantly speed up the process if all of the resources were available in memory rather than on disk.

Stage 2.

The user's password is first checked to see if it is contained on a blacklist of common passwords. Through pilot testing and our own trials, we determined that very weak initial passwords required a great deal of modification in order to make them secure. This resulted in unusable password suggestions. As such, we implemented a blacklist check to the initial passwords input by the participants. This blacklist contains the top 3,500 most common passwords (see Section 4.4.1 for a description of why this list was used). If the user's password is found on this blacklist, the password is rejected and the user is asked to select a new password.

Stage 3.

Individual participants' passwords are analyzed in much the same way the grammar was generated. All of the same steps must be performed such as parsing the password into its n segments, classifying the segments, and generating a base structure from the classified segments. The total probability of a user's password is represented by the following formula:

$$Prob(password) = Prob(structure) * \prod_{i=1}^{n} Prob(n)$$

The first step is to parse the password grammar for the base structure that was created to see if it exists. If it does exist, the probability of that base structure is used for structure probability (where $totalstructures$ is the total number of structures generated by the password grammar). If it does not exist, a probability of $1/(totalstructures+1)$ is used for structure probability. This effectively gives the structure a probability slightly lower than the lowest probability seen in the password grammar. Since the structure in question was not found, it means it was not present in any of the passwords in our password collection. As such, since it was not seen, it should have a lower probability than something that was only seen once. This is why we divide it by the total number of structures + 1. Next, each segment of the parsed password (referred to as a "terminal") is searched for in its corresponding nonterminal set in the password grammar. Similar to base structures, if the word is found in a list of all words ($totalterminals$), the probability associated with that word is used. If it is not found, the probability is $1/(totalterminals+1)$. The total probability of the user's password is computed as the product of the individual terminal probabilities and the probability of the base structure.

The probability of the password is then checked against a predetermined threshold of 1.080935×10^{-12}. This value was computed as the result of a non-linear regression model (see Figure 4.3) using the total number of guesses an attacker could make in a 4 month effort (see Section 4.4.1 for a detailed explanation of where this value came from and our threat model). If the probability of the participants password is below this threshold it is deemed "previously secure", as in no modifications are required. If the password is above the threshold then a strengthening technique must be performed to lower the probability of the password.

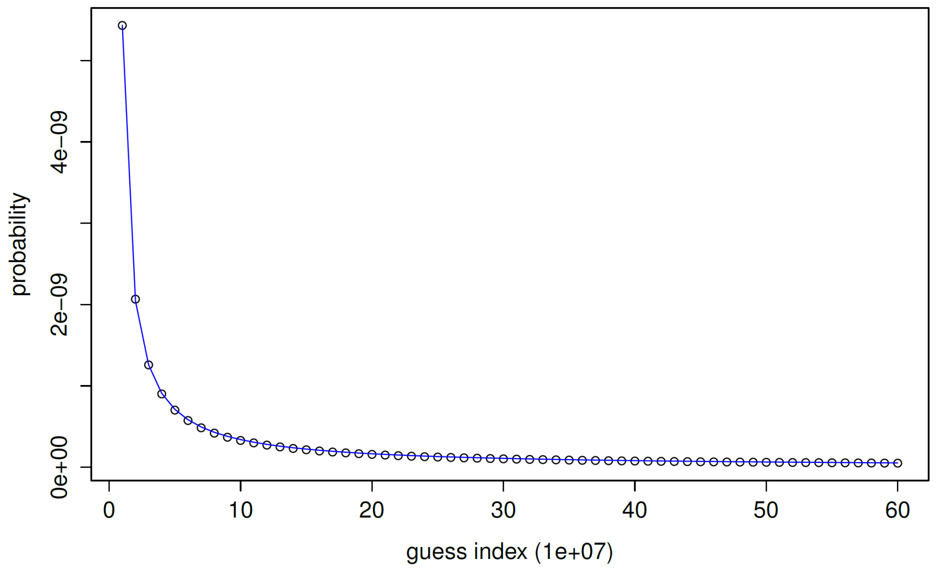


Fig. 4.3 Non-linear regression model to forecast guess probability [59].

In an effort to make passwords more secure, PassMod makes any one of three modifications (chosen at random) to the original password. These are structure rearrangement, structure addition, or terminal modification. The only constraint is the same modification technique will not be performed twice in a row (a unique modification is performed at each iteration). Each of these is described in their own sections below. See Table 4.1 for examples of each modification and a brief description of the modification.

Type of Modification	**Example**	**Description**
Structure Addition from Grammar	iloveschool1999 → iloveschool**memories**1999	Inserts a segment found in a structure that was seen in password grammar
Structure Addition from Bigram	barkingdog5 → **quit**barkingdog5	Backoff method that uses a common bigram and adds a new segment
Structure Rearrange	ilovebill2012 → 2012billilove	Takes the original structure and rearranges it
Terminal Modification from Grammar	ironman76 → iron**patriot**76	Swaps a weak word with a more secure one found in grammar
Terminal Modification from WordNet	dancinginrain → **parading**inrain	Swaps a weak word with a more secure one found in WordNet

Table 4.1 Examples of each modification type performed by PassMod.

Stage 4a: Structure Rearrangement

One of the potential modifications performed is structure rearrangement. The primary form of structure rearrangement uses the existing grammar. There is no backoff method for structure rearrangement. The only constraint for structure rearrangement is the probability of the structure entering this function must not be the minimum probability for structures. The probabilities of the structures are computed based on how many of that structure was seen in the original password dataset. If the structure in question was only seen once, it would have a probability of 2.48×10^{-08} (1 / $totalstructures$). Structure rearrangement is only useful if there was more than structure comprised of the same categories. In this case, structure rearrangement has the potential to rearrange the existing password to a different, less probable structure. Thus, if the probability of the structure of the participants' password is the lowest, the function returns early.

Structure rearrange from grammar attempts to find a rearranged form of the original password already in the grammar. For example, the password "ilovebob2012" has a structure of {ppis1}{vvz_s.love}{mname}{number4}. This structure is present in the grammar and has an associated probability. However, there could be other different structures with the same set of categories in the grammar as well, such as {ppis1}{number4}{vvz_s.love}{mname}. This would yield the password "i2012lovebob", which could have a lower probability than the original. The main motivation behind this is if we can find a rearranged form of the password somewhere in the grammar, it means at least one other user thought this was a good password structure (since the grammar was trained on passwords). If this is the case, we can assume this newly rearranged password is more memorable, semantically meaningful, and syntactically sensible than randomly rearranging the structure. The first operation performed is to compute all permutations of the original structure. This is done by breaking the password into its segments (e.g., "ilovebob2012" becomes "i, love, bob, 2012"). All permutations of these segments are then computed. The base structures for these permutations are found, and then parsed. Of all of the matching structures that are found in the grammar, a random one is selected. This is the new structure that is to be used for rearrangement. The user's original password is then rearranged according to this structure.

One potential security issue with this function is if one (or very few) rearranged structures were found in the grammar. Whenever this function is called for that password (or one with the same structure), the same rearranged structure(s) would be used. This is being avoided by requiring at least 5 previous matching structures being found in the grammar before one is selected. If there are fewer than 5 possible suggestions, structure rearrangement is not used. If our rearrange function uses the same structure every time for a password, the suggestions are just as weak as the original.

Stage 4b. Structure Addition

Structure addition has the largest impact on the overall password and reduces the guessing probability the most (most of the time, we are adding entire words at a time). The primary method tries to expand a password using structures that were seen in the grammar previously. For example, the structure for “ilovebob” would find “ilovebob2012”, or “ilovebobsummers”, as potential matches. The reason is because structure addition parses structures in the grammar where the original structure is seen as a substructure of the new password. In our example, the original structure for “ilovebob” (**{ppis1}{vvz_s.love}{mname}**) is seen in both of the other suggestions “ilovebob2012” (**{ppis1}{vvz_s.love}{mname}**$\{number4\}$), or “ilovebobsummers” (**{ppis1}{vvz_s.love}{mname}**$\{nn2\_season\}$). The primary method only works when the original structure was found as a subset of another in the grammar. If it does not, there is an alternative method for structure addition that is used. The backoff method adds structures from a nonterminal bigram model. For example, one structure from the passwords rule is selected (“love” in our example of “ilovebob”). Bigrams containing the word “love” are searched, and one is randomly selected from that set. One potential addition could be “reallylove”, in which case the resulting password suggestion would be “ireallylovebob”. The added structure can occur after the original word (as in the above example) or before the original word. There are no constraints for structure addition, it will operate as expected for all passwords.

Structure addition from grammar only works if the original structure was found as a subset of another structure in the grammar. The reason is structure addition parses all occurrences of the original structure in the grammar. It tries to find structures where the original one is a substructure of another one (e.g., the structure for ilovebob would find ilovebob2012). It is worth noting that the substructure must occur exactly in the new structure; the structure cannot be split up. This means that the structure for “ilovebob” can only find structures where words / gaps are added to beginning, end, or both beginning and end of the substructure. Each structure found is added to a list, and a structure is randomly selected from that list to be used. Similar to rearrangement, structure addition suffers from the same weakness if only one (or very few) structures are found. For this reason, if 5 or fewer structures are found, the backoff method is used (described below). In line with our goal, we want to make passwords more secure, and less guessable. If we add the same structures every time, we are limiting our resulting password space to be very small, and very guessable, which is not what we want. This would allow an attacker to exercise whats known as a “guided bruteforce (GBF) attack”. A GBF occurs when the resulting password space is so small that the attacker can simply just bruteforce all possible modifications and guess them all. We enforce this requirement such that the resulting password space is large enough to resist a GBF attack.

There are a few restrictions in place for selecting structures from grammar. One, is that only special character insertions of length one are used. More than one special character adds complexity to the password and can make it very difficult to remember. Furthermore, more than one gap segment will not be used. For example, only a structure that contains at most one added special, char, or number will be used. From our initial testing, we found that multiple gap segments make the resulting passwords very unintuitive to the end user.

After a structure is selected, the original password is substituted into the correct parts of the structure (e.g., {ppis1}{vvz_s.love}{mname}{number4}{aa}{nn1} would become ilovebob2012{aa}{nn1}). Now we must find specific words for the newly added structures ({aa} and {nn1} in this example). In order to find a word that is a part of the structure in question, we need to look in the corresponding lists for each rule generated by the grammar. These lists contain all of the words that were seen in the original password set that are a part of different nonterminal categories. For example, the list for mname (male name) will hold all words that have mname as a categorization, each with a probability of occurrence. The relative probability of the word in question can be used to determine the overall probability of the password.

Structure addition from grammar is a very useful enhancement to the user's password because the guess probability is reduced significantly with very little change to the original password. Most of the time, the new structure makes sense, and works with the existing password. All of the structures that are added come from previous participants' passwords, so we assume the structure is memorable since it was used at least once before.

The backoff method for structure addition is structure addition from a nonterminal bigram model of frequencies. The nonterminal bigram model was created from all of the password sets used in training the grammar. All structures in this set were parsed into their bigram segments and input into a frequency distribution. The result is a frequency distribution of nonterminal bigrams, which can be used for structure addition. When this method is required, a random structure is selected from the current participants' password. For example, if the user's password is "ilovedogs", the random structure chosen could be $\{ppis1\}$. A list of all bigrams of the nonterminal bigram model that contains the unigram $\{ppis1\}$ is created. From this list, a random bigram is selected to be used for structure addition. The bigram that was selected is parsed into its two unigram segments. The unigram that did not exist in the original password is inserted into the structure of the password. We ensure the inserted word is within the top 75,000 most frequent words according to COCA so that it is a human readable word. A terminal is then selected for the added structure. Similar to structure addition from grammar, special character structures are restricted from suggestions. Also, number and char structures are restricted to fewer than 4 characters. Since any special or char insertions are allowed, there were cases where up to 20 random sequences of characters could be added. As such, we needed a way to keep the insertions to a reasonable length, a length that could be remembered with little difficulty. It is worth noting that the COCA filter is only used in the cases where a word is being added. If a number or special character is added, this is not checked for in the COCA list.

Stage 4c. Terminal Modification

Terminal modification is the final modification type we make use of in the system. The primary method is terminal modification from grammar, whereby the new terminal must appear in the grammar, in the same list of nonterminals as the original word. The backoff method is WordNet terminal modification, where the original word is swapped with a related word in WordNet found through a hyponymy relation. The word selected by WordNet must also pass the COCA filter of the top 75,000 words.

Terminal modification first precomputes a list of all children of the grandparents (from WordNet) of the most probable word and the second most probable word of the participants password. The reason for spanning up to the grandparent level is to try to keep the modification semantically relevant whilst also creating a larger list of terminal modifications to choose. These lists are used to verify future suggestions as being semantically relevant to the original word. Terminal modification uses two words of the participants original password, the most probable and the second most probable. To prevent the most probable word in each password being chosen for modification every time (therefore making the modifications deterministic), there is a 50% chance the second word will be chosen for modification instead of the first. The reason for this is to increase the total suggestion space of the resulting suggested passwords.

For example, if a user inputs the password “ironman76” instead of always selecting the most common word for modification, either one of the top two most common words can be selected. In our example, the resulting password suggestion could be “galliumman76” if the word “iron” was modified, or “ironpatriot76” if the word “man” was selected. By randomly deciding to either use the most probable or second most probable word, we effectively increase the resulting suggestion space for this input password. This is another reason why we selected the very general treecut. With this cut, it is less likely we will encounter more of these situations where a terminal is the only one in its category. This is because a more general cut will produce fewer categories with more words in each category.

Once the words for modification are decided, the nonterminal file for the new most probable word is read. Each line of this file is added to a list only if the word appears in the WordNet set that was precomputed. At the end, we have a list of all terminals that appeared in the WordNet set. A random modification is selected, inserted into the password in the location of the bad terminal, and then the probability is recomputed. This will happen 20 times, or until a secure password is found. 20 was chosen as the number of modifications to try since the process of modifying and testing the security of the new password is time-consuming. Through our own trials, it was found that 20 iterations could be performed in a reasonable amount of time. The 20 iterations only applies to trying to find a suitable terminal. There is no limitation to the total number of rounds for the entire program – only the threshold check at the beginning of each iteration. Computing the new probabilities is not as simple as multiplying the probability of the new terminal into the password. The reason for this, is with a different word in the password, the parsing of the password could change entirely, changing the overall probability of the password. As such, the newly modified password must be input into the probability checker system to have a probability assigned. If no secure password is found, the second most probable word goes through the same process. If no suggestion is viable, the backoff method is used.

The backoff method selects a terminal from a set of all children of the grandparent (according to WordNet) of the original word and tries this in the probability checker system. This is also tried 20 times, or until a secure password is found.

Stage 5. Reassess Probability of Password

After a modification has been performed, the probability of the overall password is reassessed (this time with the modification included). If the probability of the password is below the specified threshold, the password is suggested to the user. If the probability is still higher than the threshold, another modification is selected and performed on top of the previous one. Eventually, the probability of the password will be below the threshold and the password can be suggested to the user.

4.3 User Study

We conducted a 43-person multi-session online user study to test the memorability and usability of PassMod. First, we pilot tested the system with 7 experienced and 3 casual computer users. The main motivation behind the pilot test was to iron out any obvious design flaws in the system and to improve the user experience prior to running a full study. Aside from fixing minor bugs in the system, two obvious design choices came from the results of our pilot study:

- Colour-coded Password Suggestions. After a couple of trial cases, we began to notice that when strings were inserted into the middle of the password, and the entire password was displayed in black, the word boundaries were harder to see, making it more difficult to see the exact insertion / modification. As such, we presented each segment of the password in a new colour.

- The Shuffle Button. Multiple participants from our pilot test mentioned not liking their initial password and wanting to "shuffle" or select a new modification (without starting over). As such, we added a button that provides this functionality.

One user from our study even commented in the post-questionnaire (at the end of the study) about the colour-coding, stating "the color coding helped a lot". This confirmed to us that presenting each segment in a different colour was a useful design decision. In order to compare the usability and memorability of our study with annotated location-passwords and regular passwords, we used the same study design as seen by Thorpe et al. [56] and Al Omari et al. [4].

4.3.1 Sessions

This study consisted of three required online sessions that spanned the course of 8-9 days and one optional session one month later:

Session 1 (day 1, online). Participants were tasked with creating and confirming a password of their choice. Participants were not told this password would be strengthened at the time of creation, and were not given any password policies to adhere to. The password they entered was subject to a blacklist check to determine if the password was too weak to modify (discussed further in Section 4.4.1). In the event the password was present on the blacklist, participants were informed their password was too weak and told to enter a new password. If the password was not present on the blacklist, it was analyzed by the PassMod system. If the guess probability of the password, as determined by the system, was below a prespecified threshold, the password was deemed "Previously Secure". In this case, the system did not need to make any modifications to the password. If the password was above the threshold, the system made as many modifications as needed to bring the strength of the password below the threshold. In this case, participants were then required to type and confirm the password they were suggested. Following a brief background questionnaire regarding participants' password habits, participants were asked to login with their password. Failure to do so resulted in participants being given the choice to to reset their password (after 3 failed login attempts) or being forced to (after 10 failed attempts). Successfully logging in with their password completed session 1. Fourty-three participants completed this session. One user opted out after failing three logins in session 1. This user did not comment on their reason for opting out.

Session 2 (day 2, online). Participants were asked to login with their password one day later. The purpose of this session was to model the frequency of logging into a frequent account (one that is accessed approximately once per day). If the participants failed their login and had to reset their password in session 2, they were asked to come back again the following day to login again. A successful login completed session 2. Fourty-two participants completed this session.

Session 3 (day 8-9, online). Participants were then tasked with logging into the system with their password approximately one week (7-8 days) after successful completion of session 2. The purpose of this session was to model the frequency of logging into an infrequently used account (approximately once per week [35]). After successfully logging in with their password, participants were asked to complete a short feedback questionnaire asking them about their experience with the system. Thirty-four participants completed this session. It is worth noting that 5 participants communicated that they were still interested in continuing the study and logging in for session 3, but they simply forgot or did not see the reminder email in time.

This indicates they did not drop out because of a flaw with the system, but that they were more likely preoccupied and did not have the time to complete the session.

Session 4 (day 38+). Participants were invited to participate in an optional login at least one month later (30+ days after session 3). Participants were not compensated for this optional session, but instead were entered into a separate draw for another $50. 26 participants completed this session.

4.3.2 Environment

All sessions of the user study were held completely online, from wherever the participants wished to complete them. Participants were sent a reminder email informing them of when they were allowed to login and how long they had to complete the session.

4.3.3 Participants

Fourty-four participants were recruited to partake in this study via an email broadcast to all students. Only non-IT students were selected for participation in the study. This was to avoid a heightened awareness of what makes a secure password. We were primarily concerned with getting results that are representative of the average computer user. Participants were given $10 to complete the study and entered into a draw to win $50. Through the use of a background questionnaire, we were able to collect demographic information about the participants. Our study consisted of 12 male and 31 female participants, all university students between the ages of 18 and 40. The majority were between the ages of 20 and 25.

4.3.4 Limitations and Ecological Validity

One issue with the parsing algorithm is it is based on COCA unigram, bigram, and trigram models as well as coverage of the English language. Most of the time, the parser correctly identifies the segments of the password. However, there are anomalies in which the correct segmentation is not found. This results in this parsing being used for POS tagging and for semantic classification. Since the probability of the password comes from the segments individual probabilities, the probability may be incorrect. More work will need to be done on the parsing algorithm to try and perfect it. However, give that this is a computer application trying to parse passwords into English segments, some error is likely to be expected. In our study, we did not experience a scenario where the incorrect parsing was used. We did notice this a few times in pilot testing, however.

Online studies have their advantages such as the ability to collect data from larger sample sizes. However, given that this is the first iteration of PassMod, we opted for a smaller study in order to identify any areas of the system that could be improved before using a larger sample size. Fourty-three participants gives us enough data to enhance the system and its features before running a larger scale study. One drawback to a fully web-based user study is the lack of qualitative data we can collect. We can openly report what participants tell us as we were not there to observe their behaviour / interaction with the system. We did notice abnormally high login times (with no failures) for a couple of participants. We can hypothesize that the user might have been distracted with something else at the time they were completing the study, which contributed to the long login time (>8 minutes). We wanted to collect results that were as representative of typical password habits as possible, and did not want the participants to feel the pressures of a lab environment where they were being watched.

Since our participants were recruited from a university campus, even though they were not directly enrolled in IT programs, they may have a heightened awareness of what makes a secure password. Due to this reason, it is possible that our study collected slightly more secure passwords than those of the average computer user.

4.4 Results

This section presents the results of our study. First we discuss the theoretical security of PassMod. This includes our theoretical threat model and how we have designed our system to resist the attacks mentioned in Section 2.3 (see Section 4.4.1). Next, we analyze the empirical security of PassMod by running a guessing attack against the sample of passwords we collected in our user study. In Section 4.4.2 we discuss the usability, memorability, and acceptability results collected from our user study.

4.4.1 Security Analysis

A secure password can be defined as providing a sufficient amount of security such that all common attack vectors are thwarted. The main attacks we focus on preventing in this discussion are the semantic attack described in Chapter 2 and guided brute-force attacks. We limit our focus to these attacks as they are the most comparable to the types of attacks that would affect the system we have designed and implemented.

Threat Model

We assume the attacker is a computer user with high end, but affordable computer hardware. Our attacker has a workstation that contains an Intel Core(TM) i7-5820 CPU running at a clock speed of 3.30 GHz. This system contains 32 GB of 2333 MHz DDR4 RAM. According to previous literature [20] a best effort attacker would use clusters of

1000 GPU's to crack passwords with. However, we base our calculations on the above computer specs in a cluster of 1000 alike computers. Florencio et al. [20] also assume that for an offline attack, a password guessed in 4 months is considered weak. A realistic password policy could be to enforce a password change once every 4 months.

We assume the system deploying our software has adequate defences in place. As such, the hashing method used is Bcrypt [20], with a cost of 15 applied. Each workstation is able to compute 0.55 hashes / second. The entire 1000-pc cluster is thus able to compute 5,840,000,016 hashes during the 4-month campaign. In order to generate all possible guesses from our system, it would take a very long time and an extremely large amount of memory. As such, we generated a non-linear regression model that is capable of estimating the guess probability of the n-th password generated by the semantic cracker. We use this model to estimate the probability of the 5,840,000,016th password which yields 1.080935×10^{-12}, our probability threshold for determining a strong or weak password.

Prevention of Pitfalls in Automated Strengthening Algorithms

As discussed in Section 2.3.3, previous approaches to password strengthening had pitfalls of PCFG cracking attacks and guided bruteforce attacks. PCFG cracking attacks occur when the password cracker is trained on a leaked database of strengthened passwords. If the attacker was to try to crack strengthened passwords using passwords that were previously strengthened by our system, there would most likely be an increase in success rate. Guided bruteforce attacks occur when the resulting password space from suggestions is so small that an attacker can very quickly bruteforce all suggested passwords for a given input password. However, many measures are in place to ensure the password space of suggested passwords is large. In PassMod, each modification is chosen at random from a large pool of possible modifications, which means the same suggestions are very unlikely to reoccur. In previous literature, modifications are chosen at random based on the previous structures that were seen in the training data (insert digits at the beginning or end). Our system utilizes this same method, but also contains a backoff method of new structure addition. In this scenario, a structure is added to the previous structure, creating a new structure that was not seen in the grammar previously. Furthermore, previous literature contained systems where there was a limit on edit distance (1 or 2 characters), however, our system is inserting entire words, sequences of digits, chars, and numbers. Our preventative measures ensure the resulting password is more secure against an attack than in previous literature [47]. The fact that we are not making single character edits, but rather entire word-level / structure modifications coupled with no restrictions on number of edits, makes our system more resilient to PCFG-based attacks.

Prevention of PCFG-based Attacks. If an attacker was able to recover a leaked copy of the strengthening database, there would be no gain. The reason is that strengthened passwords are not entered into the database. A leak would only yield the structures and dictionaries of the weak passwords, which does not yield any additional information that the attacker would not already have from publicly available data. Given that leaked data of strengthened passwords could be used to mount an effective attack, it is probably best to not insert any strengthened data into the database. The main motivation for this in the approach of Houshmand et al. [27] was to keep the system adaptive, and always changing. By inserting data obtained from strengthened passwords or already secure passwords into the database, the criteria for a secure suggestion would change over time. For example, a password that is secure today might not be secure tomorrow, as it has been inserted into the database and assigned a probability. If anything is to be inserted based on the newly strengthened passwords, it should only be the structures. Schmidt et al. [47] demonstrated that partial data used in a PCFG-based guessing attack did not yield successful results. We did not make our system adaptive because of the number of possible modifications for each password. With an average of ~10^{48} suggestions for each password, even if an attack was able to learn the probability distribution of the passwords output by the system, it would be very difficult for them to crack a significant number of passwords.

Evaluation of Guided Bruteforce (GBF) Attacks. A guided bruteforce attack is made possible when the password space of the suggested passwords is so small, that an attacker could exhaust it in a reasonable amount of time. The GBF attack assumes that the original, unstrengthened password the user enters is inherently insecure (which is the motivation for using this system in the first place). We assume that in any case, the unstrengthened password in question is weak, and can be guessed in little to no time. Now that the attacker has the original password, they will try to brute force all of the suggestions that can be made for that password. This section focuses on findings from previous literature regarding systems' weaknesses to GBF attacks as well as preventative measures employed by our system.

Previous literature [20] has shown that an original password with a guess probability weaker than 10^{-9} could be guessed in less than 1 minute. An original password with a guess probability weaker than 10^{-12} could be guessed in less than 2.2 hours (using 12-core Intel i7 CPU 3.2 GHz and MD5 hashing). As such, the only resistance to guessing the participants password would be the time it takes to brute force all of the variants of that password (all of the suggestions the system could make with that password as the base). Schmidt et al. [47] state that in order to completely thwart a GBF attack, a stronger initial password is needed as well as a minimum of two edits. As more edits are performed, the run time for a GBF attack to execute increases. Our implementation does not just perform one edit. This system recursively calls itself, passing the previously strengthened password as input to the next iteration. The result is a password that has been recursively modified until a certain threshold has been reached. The general trend appears to be 1-2 random edits (see Table 4.2) that are expected to be different every time, further resisting GBF attacks with every additional edit made.

According to previous literature [47], it was found that a user's original password must have a guess probability of at least 10^{-12} in order to completely thwart GBF attacks. However, it was also mentioned that this is a very high hurdle to expect participants to overcome. Initial implementations of our system used a blacklist of the top 2 million passwords, which translates to 10^{-8} guess probability [15], however, we learned very quickly through initial testing this blacklist became tedious when participants needed to select an initial password. Passwords with multiple segments were blacklisted due to the comprehensiveness of the blacklist. This led us to investigate more deeply the issue of the blacklist and whether it was actually needed or not. We decided to run our own GBF attack against all of the suggestions our system can generate for the top 1 million RockYou passwords. This was done three separate times, once with no blacklist at all applied to initial passwords, once with a small blacklist applied (3,559 of the most common passwords provided by John the Ripper [45]), and once with our initial comprehensive blacklist applied (Dazzlepod [15]). Since a guided bruteforce attack would exhaust all possible suggestions for the input password, the user's resulting password would be cracked. We assume the expected case for the attacker's guessing effort would be to exhaust half of the possible suggested passwords before cracking the user's real password. The actual number of guesses could vary, sometimes being fewer or greater than the average. We evaluate the effectiveness of the guided bruteforce attack in each different condition based on how many initial RockYou passwords end up being cracked as a result of the cracking effort.

A large component in the effectiveness of a GBF attack is the initial ordering of the attacker's guesses. When running a GBF attack, attackers do not simply make arbitrary guesses, but rather make guesses that are highly probable in order to crack as many passwords early on as they can. As such, we initially ordered the guesses in our simulated GBF attack by password guess probability (as output by our system). However, we decided this was not representative of how an attacker might order their guesses. An attack would want to crack as many passwords as possible early on. In order to crack a password, the attacker must bruteforce all of the suggestions produced by that password. As such, we decided to order the attacker's guesses based on the total number of suggestions ascending. This means the attacker would try to crack passwords that only have a few possible suggestions first, which would take no time at all. In the end, the attacker will be trying to crack passwords that contain billions of suggestion possibilities, which will slow their attack down significantly near the end.

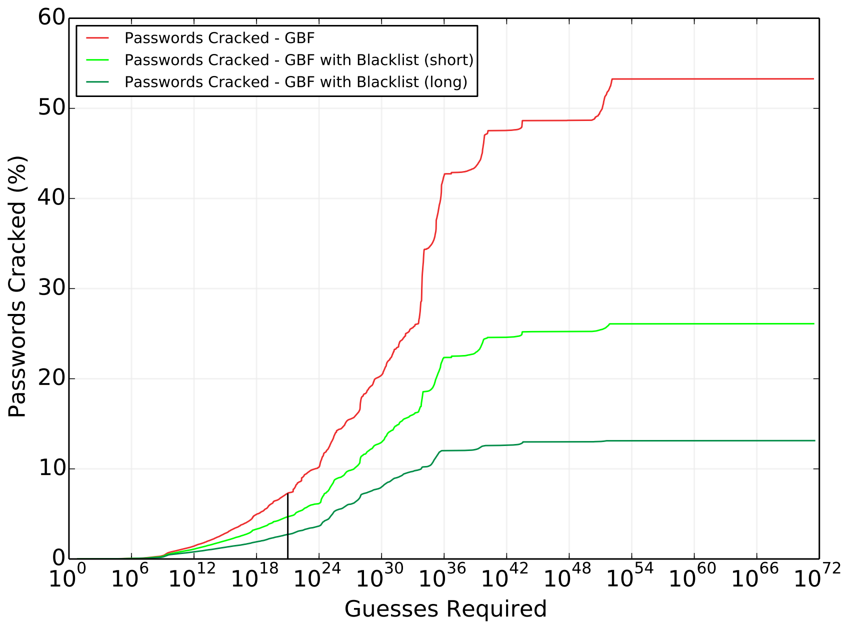


Fig. 4.4 RockYou passwords cracked as the result of a guided bruteforce attack using three different blacklists. The vertical line is drawn at 10^{21} to indicate the number considered crackable by an offline attack.

Obviously, the best case for security of the system and the participants' passwords was seen with the use of the large blacklist. However, it was not justified that this much of a blacklist was needed, for we might be able to achieve comparable security with the use of a much smaller, less comprehensive blacklist. Furthermore, since our system is different in the way it operates and makes modifications to a user's password, a blacklist might not even be needed.

The values we use for our results were chosen based on suggestions and conclusions drawn from Florêncio et al. [20]. Their results indicate that at 10^6 guesses, the effectiveness of an online attack is severely reduced whereas at 10^{21} the effectiveness of an offline attack is severely reduced. As such, we evaluate the effectiveness of our cracking effort at benchmarks of 10^6 and 10^{21} guesses. As seen in Figure 4.4, at 10^6 guesses, an insignificant number of passwords were cracked. Therefore, it is safe to assume our system provides adequate protections against an online guessing attack. At 10^{21} guesses, we start to see a difference in terms of a blacklist versus no blacklist. When no blacklist is applied, 8.48% of passwords have been cracked compared with 3.09% with a comprehensive one. At 10^{72} guesses, we really see the true strength of the blacklist, as only 13.1% (4,281,446 passwords) of passwords were cracked compared with 53.3% (17,369,986 passwords) without a blacklist. It is extremely unlikely an attacker would ever make it to this point given the computing time and memory stipulations of the system. Since Florencio et al. [20] state the efficacy of most offline attacks is reduced after 10^{21} guesses, we discuss that point. Based on these results, it would seem that a strict blacklist is not needed in all cases. At 10^{21} guesses, there is not a large enough difference in terms of overall passwords cracked to justify the decrease in usability of the system. We would like to note that in some cases, security may be desirable over a decrease in usability. As such, the security required by the type of account must be taken into account when determining the size of blacklist to be used. We decide to use the small blacklist (only containing ~3,500 entries) as it prohibits participants from using the weakest of passwords (e.g., password, 123456, etc.) while also limiting the efficacy of a guided bruteforce attack.

Another way PassMod prevents a GBF attack is by the number of suggestions possible for each password. We ran a simulated GBF attack against the top 1 million passwords of RockYou to see how many suggestions each contained. The results indicate the average number of suggestions for each password was ~10^{48}. The time and computing resources required to generate that many guesses indicates a GBF attack would not be effective. Figure 4.5 shows the total number of suggestions for each of the first 1 million passwords in RockYou. Given the large number of suggestions possible by the PassMod system, it would be interesting to analyze if providing participants with their original password as a password hint would be a security risk or not.

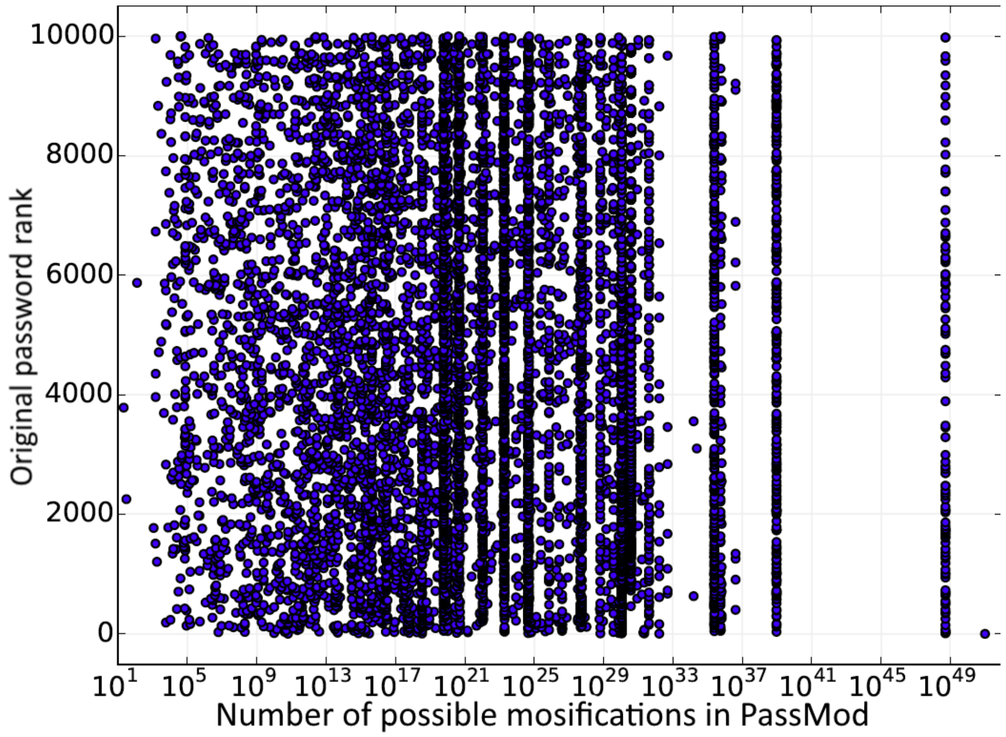


Fig. 4.5 Total number of suggestions for each of the top 1 million RockYou passwords and how many passwords in RockYou would have been cracked.

Practical Security

Through the use of a theoretical security analysis, we were able to show that this password modification system is more resilient to the types of attacks that made its predecessors weak. The design choices we made make this system more resistant to guided bruteforce attacks and guessing attacks. However, it is one thing to make this claim and another to demonstrate it. Figure 4.6 demonstrates the effectiveness of three different password cracking algorithms against the passwords collected in our study. The algorithms we chose were JtR in wordlist mode (followed by incremental mode) [45], PCFG with a dictionary of common passwords as input [63], and the semantic guessing system of Veras et al. [59]. In each case, the cracking algorithms ran until 3 billion guesses had been exhausted. As shown by Figure 4.6, the semantic cracking system was the strongest performer, cracking up to 33% of the original passwords collected in our study. This was followed by JtR which was able to crack 27% and PCFG cracking 19%. However, once the passwords were strengthened by our system, the efficacy of these attacks drops dramatically. The semantic system was unable to crack any of the strengthened passwords, which makes sense as it was the semantics underlying algorithm that strengthened the passwords. Even with JtR and PCFG, very few strengthened passwords were able to be cracked within 3 billion guesses. These results demonstrate that the system strengthens participants' passwords to levels that are able to withstand current state-of-the-art attacks against passwords.

The PassMod system was also able to significantly increase the length of participants' starting passwords. The average length of the original passwords in our study was 10 characters. Once modified, the average length of the passwords increased to 15. These extra 5 characters came using a variety of strengthening methods ranging from adding a single word to the beginning, middle, or end, up to adding a word, two symbols, and a

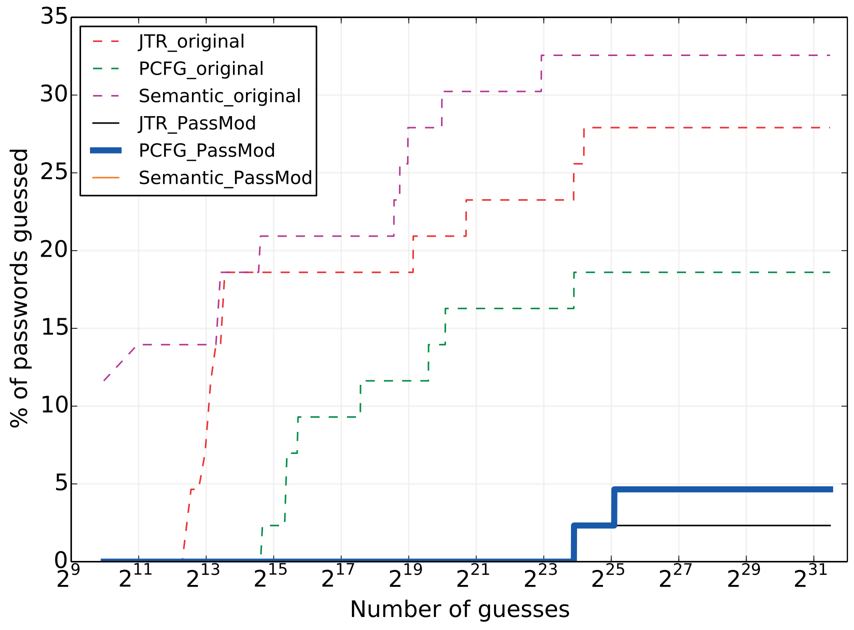


Fig. 4.6 Guessing results of three different algorithms against the original and modified passwords collected from our study.

digit (see Table 4.2 for a complete list of modifications performed and sample passwords). Table 4.2 indicates a potential bias towards structure addition over other strengthening methods. Structure rearrangement will only be performed if a rearranged form of that structure is available in the password grammar and has a lower probability of the current structure. Similarly, terminal modification will only be used if the word to be modified comes from a category that has other words with lower probabilities that would lower the overall probability of the password below the threshold. This is something that requires further investigation in the future.

4.4.2 Usability Analysis

In the section we discuss the usability, memorability, and login times of the PassMod system. 34 participants completed the third session of our user study; however, 12 of these participants entered passwords that were previously secure (i.e., the system did not need to make any modifications to the original). As such, we only report the

Modification	**% of Total**	**Example**
+ 1 word	49%	*ilovebob* → *ireallylovebob*
+ 1 word, digit	5%	*jellotime* → *jellotime*76
+ 1 word, symbol	8%	*barkingdog* → *quitbarkingdog*!
+ 1 word, symbol, digit	3%	*highschool*1 → *imhigh*19*school*1!
+ 2 words	21%	*jackolantern* → *jackolanternnightynight*
+ 2 words, 1 digit	3%	*schooltime* → *schooltime*29*downlow*
+ 3 words	3%	*mydogs*76 → *seemydogs*76*appletree*
terminal modification	5%	*ironman*76 → *ironpatriot*76
rearrangement	3%	*ilovedogs*2012 → 2012*dogsilove*

Table 4.2 Modification samples of the PassMod system.

usability questionnaire results of the 22 participants that actually needed a modification to their original password. For the optional session 4, we report on the memorability data collected from the 26 participants who completed the session; 8 had previously secure passwords and we differentiate where appropriate.

Usability

Participants were asked in the post questionnaire multiple questions pertaining to the usability of PassMod. Overall, participants self-perceived usability was good, with the majority stating that they would prefer to use this system if they knew it was more secure than regular passwords (see Figure 4.7c). Furthermore, the majority of participants disagreed that the system was too difficult to use and was too time-consuming.

We were also able to collect some useful usability data from the participants overall comments about the system. For example, a couple of participants stated “If the system provided more options, it would make it easier for me to pick which one i would like as I would pick the one i could remember the easiest” and “Perhaps one will stick out more to me from a list of options, and I like having the possibility personally”. Both of these comments indicate the participants would have preferred to see a list of multiple

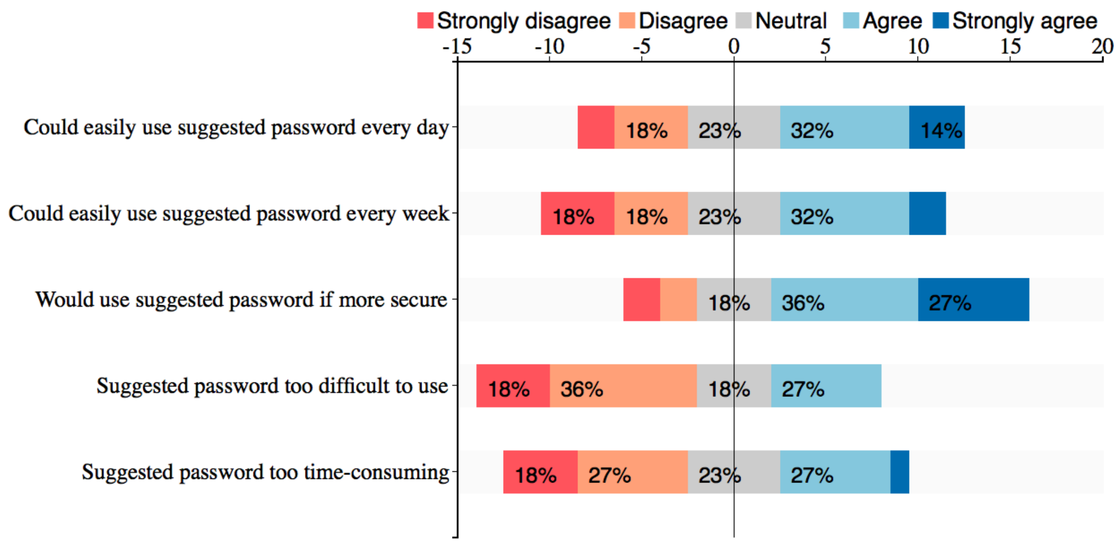


Fig. 4.7 PassMod questionnaire responses to the associated questions pertaining to usability.

password suggestions. Then they can choose the one they like the most instead of having to shuffle for a new password suggestion.

This led us to analyze how many participants actually used the shuffle button to acquire a new suggestion. Of the 42 participants that completed sessions 1 and 2, a total of 7 participants shuffled their suggested password: 6 in session 1 and 1 in session 2. We investigated the concerns of the two participants that commented stating they wished more suggestions were shown to them. Neither of these participants pressed the shuffle button. This indicates they either a) couldn't be bothered, or the more likely, b) they didn't know it was there. Another comment read "I think this system helped create a password that I wouldn't have otherwise thought of myself". One of the positive aspects of this system that separates it from other password suggestion / modification system is meaning. The system attempts to preserve the meaning behind the original password by parsing out its structure and then either inserting / modifying segments to create a stronger password. Since the modifications are based on previous password grammar, the suggestions are closer to password language than English language. As such,

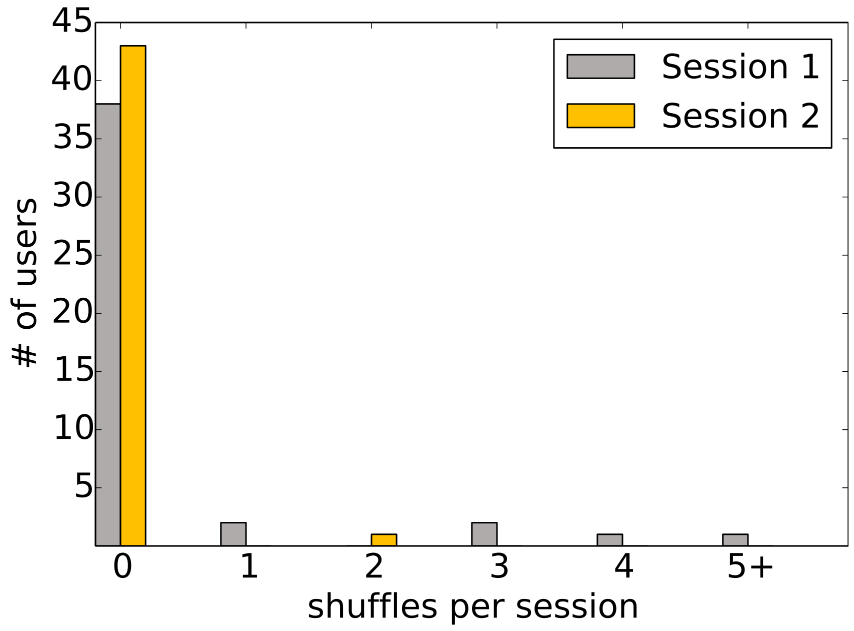


Fig. 4.8 Number of shuffles per session for the PassMod study.

one potential avenue of future work could be creating a grammar of English sentences and using those structures to strengthen passwords. Then, we may end up with more grammatically correct sentences, which may, in turn, be more memorable.

We asked participants whether they preferred password creation rules or the password suggestion system they used in the study. The majority (59%) responded they prefer password creation rules. We hypothesize the reasoning behind this is they have one or two passwords they use for every account (which two participants stated they did in the comments) and they did not want to have to remember a new one. We also asked participants if they wrote down any part of their password during the study. 24% stated they wrote down something about their password in the study. Of the participants that responded to our questionnaire question about what they wrote down about their password, 5 participants said the entire password, and 3 participants wrote down the modification. One user's remark "However I was surprised that even though I wrote down the password I only had to look at once, by the time the second survey came I knew it by heart which was shocking to me" indicates the user expected to forget the password, but it ended up being memorable to them. We were also interested to see if participants were content with the number of modifications required to make their original password secure or if they did not like it. The majority (71.4%) stated they were okay with the number of modifications because they knew it was making their password more secure.

Memorability

We measure the memorability of PassMod by examining the login rate for each session. As seen in Figure 4.9, the majority of participants did not experience any failed logins throughout the course of the study. Furthermore, only 6 participants forgot their password and had to reset throughout the course of the study. The login rates per session were 86%, 66%, 74.5%, and 72% for sessions 1, 2, 3, and 4 respectively. For session 4, we only report the login rate for the 18 participants that had a modified password; the total login rate for session 4 was 77%. Furthermore, we also collected data on participants who logged in on their first attempt each session (of those that successfully logged in). 84%, 74%, 78%, and 100% of participants successfully logged in on their first attempt for sessions 1, 2, 3, and 4, respectively (compared with 92%, 75%, and 81% for traditional password for sessions 1, 2, and 3, respectively [4]). The average number of login attempts before a successful login was 1.16, 1.17, 1.14, and 1.00 for sessions 1, 2, 3, and 4, respectively. This is slightly better than in traditional text passwords where the average number of attempts before a successful login was 1.23 [4].

We asked participants through the use of a feedback questionnaire, how many wrote down their passwords. If they did, we also asked them what about their password they wrote down (i.e., the entire thing, the modified segment, etc.). In total, 8 participants (24%) wrote down at least some part of their password. The majority of those participants (75%) wrote down the entire password with the other 2 just writing down the modified

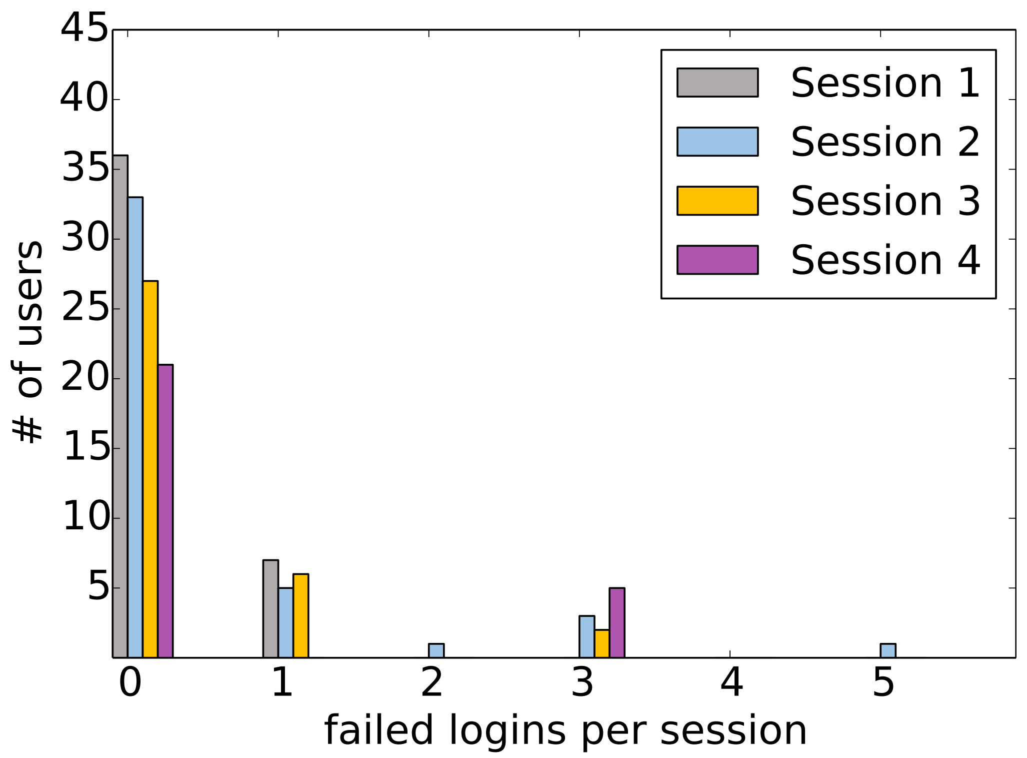


Fig. 4.9 Number of failed logins per session for the PassMod study.

component. This compares with traditional passwords where 25% of participants wrote down their passwords [4].

We were primarily interested with what makes a password memorable or forgettable. As such, we decided to investigate what the passwords looked like for participants that experienced failed logins compared with participants that did not experience any difficulty with the system. The results of this analysis indicate the majority of failed logins were due to what we could observe as typos. Further analysis indicates that all participants that had any number of modifications to their original password were able to successfully login. The majority of the failed logins (58%) can be attributed to slightly mistyped passwords (one or two characters swapped or incorrect) and participants that typed their original password instead of the modified one. Only 6 (19%) failed logins across all 3 sessions were arguably due to a memory lapse (see Table 4.3). Approximately 6% of passwords were "previously secure", meaning the password the user typed into the

Reason for Failure (sessions 1, 2, 3)	**% of total**
Previously secure	6.3% (2/31)
Typed original password	26% (8/31)
Typed only modification	6.3% (2/31)
Slight typo	35% (11/31)
Largely incorrect	19% (6/31)
Capitalization	6.3% (2/31)
Reason for Failure (session 4)	**% of total**
Previously secure	20% (3/15)
Typed original password	20% (3/15)
Largely incorrect	60% (9/15)

Table 4.3 Different types of failed logins for the PassMod user study.

PassMod system already met the threshold, and did not require any modifications. Also of note from Table 4.3 is the fact that 26% of participants attempted to login with their original password when they were provided a suggestion. This is one potential area of future work of trying to reinforce remembering the suggested password and not the original one. It is also possible that some of the login failures were due to the increased length of the strengthened passwords (~15 characters up from ~10). Session 4 saw 5 participants forget their passwords; 1 user incorrectly recalled their previously secure password (i.e., it did not undergo any modifications), 1 user typed their original password, and 3 participants typed incorrect passwords.

We also report on the memorability of PassMod through the use of questionnaire responses collected as well as login rate throughout the study. The responses we received from the questionnaire indicate that participants do not feel they could remember their passwords for a long time. This could be due to password reuse, and the introduction of a "different" password goes outside the participants comfort zone of memorability. Just over half of our participants indicated they could remember their passwords for another month. After that, the results drop down for 3, 6, and 12 months (see Figure 4.10defg). Almost all participants disagreed they would never forget their password (see Figure 4.10h).

However, we hypothesize that this is representative of any new password, not specifically the ones generated by our system. Session 4 was aimed at modelling the frequency of logging into an account one month later. Of the participants that completed session 4 that required a modification (18/26), 13 successfully recalled their passwords (72%). These participants also did not experience a failed login, successfully authenticating on their first attempt. 3 of these participants admitted to writing down their entire passwords. 3 of the participants that failed their session 4 login also reported writing down a part of their passwords (the first password, the modified component, and the entire password).

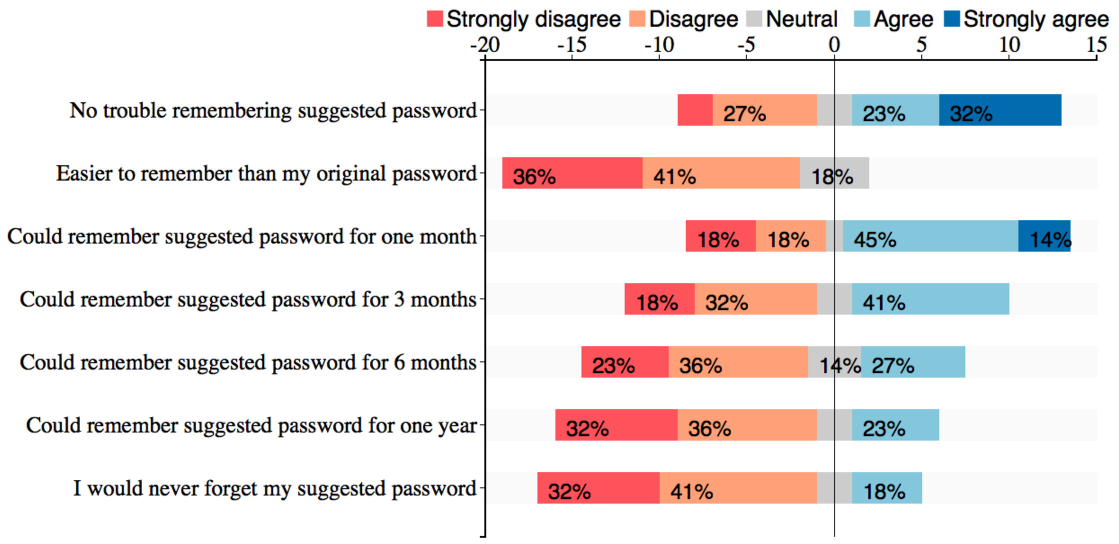


Fig. 4.10 PassMod questionnaire responses to the associated questions pertaining to memorability.

Contrary to the questionnaire results, many participants commented about the system, stating that it was memorable and a good experience. Some of the comments received from multiple participants were:

- I like this system, instead of me trying to come up with a password for the password creation rule, the format of entering a password and having it modify for me was quite convenient. Most generated passwords are too lengthy or generic to memorize, but modifying a password I already know makes remembering it much easier.

- I really liked this system since sometimes it's hard to think of a password and I know it's not good to use the same password for everything but I'm forgetful so I do it anyways. However I was suprised that even though I wrote down the password I only had to look at once, by the time the second survey came I knew it by heart which was shocking to me. Overall great survey experience.

- The suggested password was too long, although it did include a fairly memorable suggestion.

Interestingly, the general trend in the comments seemed to be the suggested passwords were memorable. It was also noted the system was able to suggest a modification that was interesting and memorable to the user, and was one they would not have thought up on their own. Another remark states that most generated passwords are lengthy or difficult to remember, but making a suggestion based on a password the user already knows is much more memorable. The last comment demonstrated a situation where this user thought the password was too long, but it was still memorable. The main purpose of this work is to strengthen a participants password in a way that makes sense to them without reducing the memorability. One user stated "I felt that the password generator was vague. They suggested a name to me that I had no connection with. I base my passwords on things that are meaningful to me (but could never be guessed). I was given a name that had no meaning to me, and that would make it hard to remember". This indicates that maybe there is some future work pertaining to the memorability of password structures. For example, what structures are the most memorable; what semantic categories are the most memorable? If we can determine which structures tend to be more memorable, we could modify the password suggestion system to only suggest memorable structures. Furthermore, we could also suggest a category and ask the user to come up with a meaningful word for that category (e.g., insert an animal at the end of the password). However, with this comes a reduction in security.

Login Times

Another way we can measure usability of the system is by the login times per session. The times collected for session 1 creation and modification are larger than the remaining sessions as this is when the user was first creating their password and viewing any modifications made. The average for creation and modification was 27 and 31 seconds, respectively. Creation time refers to the user first entering and confirming their original password. Modification time refers to when a user is presented a modified password and they must type and confirm it. Thus, the average time PassMod adds to the password creation process is estimated to be approximately 31 seconds. The login times for sessions 1, 2, 3, and 4 logins was 12, 11, 12, and 11 seconds, respectively. However, these times are slightly inflated due to outliers. This is also seen by the fact that the median login times per session is 11, 9, 9, and 9 seconds for sessions 1, 2, 3, and 4, respectively. Given the nature of an online study, it is impossible to ascertain whether or not there were any external factors contributing to login times. There are a couple of outlying login times (496s, 181s) where the user did not experience any failed logins. These indicate the user may have been distracted with something else at the time of completing the study.

4.5 Discussion and Future Work

As this is the first iteration of this work, there are many segments that still need to be researched. This section will explain the differences between PassMod and previous literature as well as briefly describe some of the areas we have identified that could be investigated further.

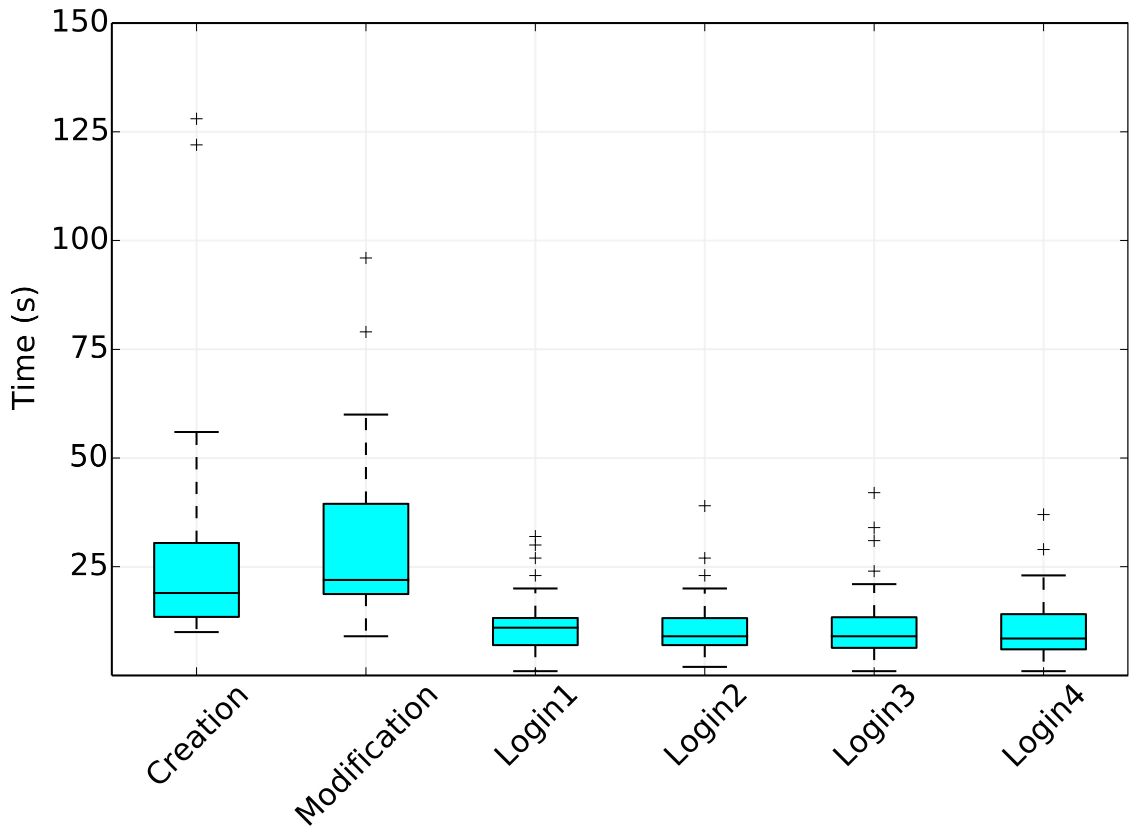


Fig. 4.11 PassMod login times per session.

4.5.1 Comparison to Previous Strengthening Techniques

In Chapter 2 we provided an overview in the area of password modification systems. The system we primarily compare ours with is that of Houshmand et al. [27]. Their approach takes the probabilistic context-free grammar of Weir et al. [62] and use it to try and create a more secure (i.e., less probable) password. Our method also uses PCFGs but has been expanded to include semantic categories as well. The following paragraphs differentiate our approach with that of Houshmand et al. [27]. We primarily focus on security as Houshmand et al. did not conduct a usability analysis of their system.

Houshmand et al. [27] evaluate the effectiveness of their password strengthening approach by simulating a password cracking attack against the strengthened passwords and already secure passwords collected in their study. The results indicate that JTR was able to crack 1% of these passwords and PCFG was able to crack 5% within 43.2 billion guesses (using MD5 hashing). The results from our study were very similar to this; JTR was able to crack 3.33% and PCFG was able to crack 5% within 3 billion guesses (using bcrypt). We also ran the semantic cracker against the same set of passwords and found that after 3 billion guesses it was completely ineffective (cracking no passwords).

Housmand et al.'s strengthening system contains much of the same functions our system uses. For example, the first step in their algorithm is to parse the password into its base structure. Next, an operation is either performed on the structure itself (similar to structure addition and structure rearrangement in our system) or on the individual segments (similar to terminal modification in our approach). The key difference is what is accepted as a modification. Houshmand et al.'s system operates a the character level, making 1 or 2 character modifications to segments. They refer to 1 and 2 edits as an edit distance of 1 and 2, respectively. Some examples they provide are inserting a number to the beginning, modifying the segment "## $\rightarrow$!#", or removing a segment entirely. Our approach operates at the segment level only. This means we add segments to the structure and then substitute in a word or number for that segment. Furthermore, we also analyze the base structure and try to insert a word that makes sense with the rest of the password. This process is further explained in Chapter 4.

One other key difference between our approach and that of Houshmand et al. is the cracking method. Houshmand et al. deemed a password to be secure if its guess probability fell below a given threshold. They determined that threshold by simulating a password cracking attack using a PC with a Core 2 Duo processor and MD5 hashing. As such, they can claim that if a passwords guess probability falls below the threshold, it would withstand one day of cracking. We do not believe this is representative of "passwords in the wild". The specifications for our cracking workstation was an Intel Core(TM) i7-5820 CPU running at a clock speed of 3.30 GHz with 32 GB of 2333 MHz

DDR4 RAM. The hashes we were cracking were Bcrypt hashes, which are currently the best known hashing method [20]. According to previous literature [20] for an offline attack, a password guessed in 4 months is considered weak. We created our threshold with these guidelines in mind. As such, if a password is strengthened using our system, we can guarantee it is secure against a semantic guessing attack that has been running for 4 months (assuming similar hardware as in our attack). Given that we used a more powerful system, a more realistic hashing method, and a realistic password change policy, we have tried to develop our system to be as realistic as possible given current best practices. Houshmand et al. did not conduct a usability analysis of their system and as such, we cannot know how usable their method is.

Aside from the practial security of Houshmand et al.'s system, there are two other attacks that automatically strengthened passwords are susceptible to; a PCFG-based attack and a guided brute-force attack. Schmidt et al. [47] found that Houshmand et al.'s system was susceptible to these attacks whereas PassMod has been modelled with these attacks in mind. PassMod is not adaptive, and does not enter any new information into the database after training (making it resistant to a PCFG-based attack). Furthermore, PassMod contains an extremely large possible suggestion space for each password (an average of 10^{48} suggestions per password) which makes a GBF attack more difficult to run (as time requirements is a large factor in the effectiveness of these attacks).

We also compare to the password strengthening method of Forget et al. [21, 22]. Forget et al. randomly insert characters into a user's password to make it more secure. They conducted a usability and memorability analysis and found that when participants started with an already strong password, the resulting "strengthened password" was very unusable. As a result, participants began to intentionally input weaker original passwords to get usable suggestions. The login rate for their study was very high, averaging well above 80%. For some of their more complex modifications, login times were longer than for PassMod (10.2 and 18 seconds for 3 and 4 additions, respectively). Even though the login success rate seems to be slightly better than exhibited with PassMod, the passwords strengthened by PassMod were able to withstand much more realistic password cracking approaches. The password cracking attacks conducted by Forget et al. indicate that no passwords could be cracked within 40 million guesses [21]. However, this was using John the Ripper in dictionary mode. The cracking techniques used in PassMod were state-of-the-art according to Ji et al. [30] and even then, only 5% of the strengthened passwords could be cracked within 3 billion guesses. Furthermore, our approach does not randomly insert components into the participants' passwords. Instead, we analyze the password and try to pick the most sensible location for addition. This is primarily based on the password grammar which is based off of passwords that participants have chosen. We believe that if a password was previously chosen by a user, it is assumed to be memorable to that user. If we can use that knowledge to make a more sensible addition, we can attempt to preserve the memorability in the original password moreso than randomly inserting a character. As indicated by Schmidt et al. [47], there are other attacks against automatically strengthened passwords that make them weak; a PCFG-based guessing attack and a guided bruteforce (GBF) attack. Forget et al.'s system would be vulnerable to a guided bruteforce attack because of the limited number of locations for an insertion as well as the limited character space for insertions. In their study, the minimum possible password length was 6 characters for the "Insert-2" condition, which appears to be the most promising for usability and security. They do not report the average lengths of the passwords collected in their study, but do report the security in terms of bits of their strengthened passwords. The mean estimated bits of security for the Insert-2 condition was 67.8 bits, which translates to $2.57x10^{20}$ guesses. We also computed the total possible password space of suggestions possible in their system. Given that the minimum size of passwords was 6 characters, there are 7 possible locations characters could be inserted.

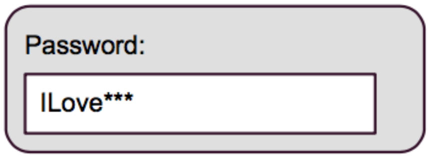
	Comparison Metric	Forget	Houshmand	PassMod
Security	Vulnerable to GBF	20%	54%	0%
	Vulnerable to PCFG (Weir)	Unknown	5.48%	5%
	Vulnerable to JtR	17.2%	0.10%	3.33%
	Vulnerable to Semantic attack	Unknown	Unknown	0%
	Vulnerable to leaked database	Unknown	18%	0%
Usability	% reset	0%	Unknown	5%
	% written down	Unknown	Unknown	24%
	% failed logins (S1)	10%	Unknown	15%

Table 4.4 Comparison of security and usability metrics of three different password strengthening schemes

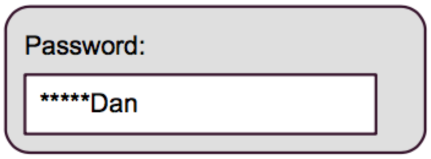
Since we are using the Insert-2 condition, we compute the total suggestion space using the following formula:

$$\binom{7}{2} \cdot (size\ of\ character\ set) \cdot (guesses\ to\ crack\ initial\ passwords)$$

The size of the character set would be 95 to cover all lowercase, uppercase, numbers, and special characters. The guesses to crack the original passwords would be 40 million (JtR attack from Forget et al. [21]). The resulting required number of guesses would be 79.8 billion to run a GBF attack and guess 20% of the improved passwords. This is a much smaller number of guesses required than seen in PassMod (~10^{48} guesses to crack just one). Our system utilizes a blacklist to make the initial password more difficult to crack. It also can generate an average of ~10^{48} suggestions per password, making it more resistant to a guided brute-force attack. See Table 4.4 for a comparison of the strengthening systems presented by Forget et al. [21], Houshmand et al. [27], and PassMod. For Forget et al.'s system, the value for GBF attacks was obtained based on the analysis above. The GBF value for Houshmand et al.'s system was obtained from Schmidt et al. [47] as they specifically conducted a GBF attack against the strengthened passwords of Houshmand et al. [27].



(a) Subject Hidden.



(b) Subject Shown.

Fig. 4.12 Design alternative hiding different parts of the password.

4.5.2 Design Alternatives

This section will explain some of the designs we considered as well as the pros and cons of each.

The first designs we discussed include hiding parts of the user's password when it is shown back to them (see Figure 4.12). Figure 4.12a demonstrates hiding the subject of the password. For example, if the user types the password "ilovedan", the system would analyze and determine that "dan" is the subject of the password, and that "ilove" needs to be changed. When the password is displayed to the user, the subject is starred out to keep it hidden and secure from shoulder surfing attacks. The user is then informed they must change everything they see (i.e., what isn't starred out). This strategy requires the user to make up and substitute in their own components rather than being "suggested" a password to use. Figure 4.12b is very similar except the subject is shown and everything else is hidden.

The next design involves not disclosing any part of the original password to a potential shoulder surfer. The user starts by typing their original password that needs modification. As shown in Figure 4.13, when the original password is displayed on the screen, the entire thing is starred out. Text would appear informing the user of potential structure modifications suggested by the system. For example, if the user types the password "ilovedan", the system could say "Please add a year to the end of your password", or "Please insert a city in the middle of your password". The structure suggestions made by the system would be based on a threshold and would make the overall password more secure.

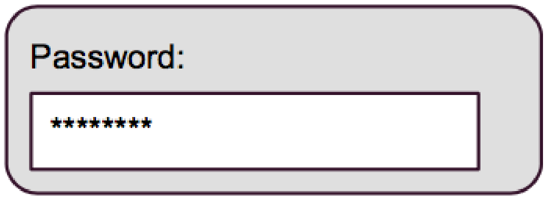


Fig. 4.13 Design alternative that suggests the user adds components to the original structure.

The final design decision we reviewed involves the system presenting the user with a password and asks the user to create a similar (but not the same) one (see Figure 4.14). In the example presented in Figure 4.14, the password presented to the user is “PurpleDonkeyNotepad67”. The user is then tasked with coming up with a similar password to use. The downside of this approach is the entire password structure is presented to the user (and a potential shoulder surfer). However, the user has the opportuity to be a bit more creative than in being suggested a password.

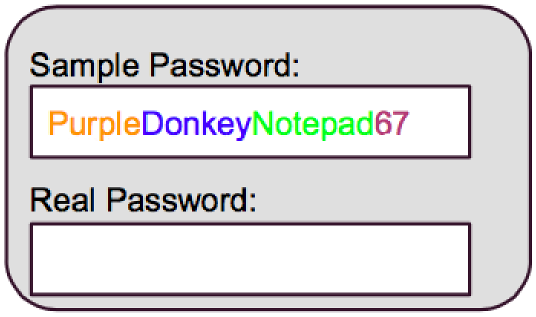


Fig. 4.14 Design alternative that presents a password and asks the user to create a similar one.

User Trust in Web Systems

Through preliminary testing of our system, it appeared as though there may have been some “doubt” in our system. Mainly, it seemed as though one user was uncomfortable inputting their passwords into our system. This sparked an interesting discussion on user trust in web-based systems. We decided to add two questions to our questionnaire to further gather user thoughts regarding trust. The first question pertained to whether participants trust entering their passwords into the system. Overall, 81% of participants trust the system enough to enter their passwords into it. We were also curious to know whether participants trusted the passwords being suggested to them were actually secure, just because the system was claiming them to be. 57% of participants trusted the passwords suggested to them are secure. This is interesting as the system did not make passwords less secure in any cases, only more secure.

4.5.3 Multiple Suggestions

Starting as early as our preliminary discussions on the design of the PassMod to pilot testing and our user study, the notion of presenting multiple suggestions to a user at one time has been around. As this is the first iteration of this work, we decided that one suggestion was enough to test the system, how it works, and if it was even feasible to present memorable suggestions to a user. As the user study was a positive experience, it is time to start thinking about presenting multiple passwords to participants (see Figure 4.15).

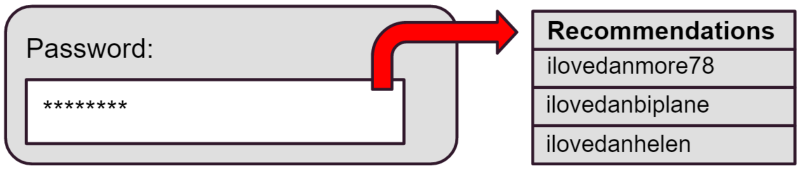


Fig. 4.15 Design alternative that suggests multiple passwords to the user based on the original structure.

This notion was further solidified when we asked participants in our feedback questionnaire whether or not they would have preferred multiple suggestions. 82% of participants agreed they would have preferred to see multiple suggestions so they can select the one they like the most. There were two participants that commented on the system stating they wish multiple suggestions were presented to them. We investigated how many times these participants utilized the "shuffle" button (which randomly generates a new suggestion). Surprisingly, neither of these participants used the shuffle button once. In any case, it seems multiple suggestions was a feature the majority of our participants would have preferred. Furthermore, we found that only 7/28 (25%) of participants opted to shuffle their password. The average number of shuffles among these participants was 3.3, indicating that providing multiple suggestions of 3 or 4 possibilities would be a good enhancement.

4.5.4 Alternative Method of Training

Our research stems from Veras et al.'s [59] design of a probabilistic password cracker. As such, our grammar was generated in a similar way (but using a larger password collection). This grammar is what is used to assess a password's strength and furthermore, suggest modified passwords to a user. It may be interesting to revisit the grammar generation in the future. For example, since we are suggesting passwords, it made sense to train the system on a large collection of passwords. This enabled the system to learn the structures and words that appear in different types of passwords that were created under different password policies. However, this results in some suggestions that don't quite make sense to the user. We have a few checks in place to ensure the suggestions make as much sense to the user as possible (e.g., limiting suggested words to the top 75,000 most common unigrams as per COCA). However, it may be beneficial to explore a different type of grammar, one build on the English language and English structures. This way, the system could be trained on entire sentences, slang sayings, movie titles, etc. The structures generated would be much more representative of how people live and speak, and could potentially enable the system to make more meaningful suggestions to participants. At the very least, we expect to see fewer suggestions that do not make any sense because they were modelled on password language, not English language.

Furthermore, given that PassMod was built using an English language password grammar, it is only capable of recognizing English language passwords. The strengthening algorithm is language-independent as it uses data from the grammar. If the grammar was trained using a password set from a different language, it would allow the system to correctly recognize passwords from that language.

4.5.5 Memorability of Structures

One of the most interesting aspects of this research to us is what makes a password memorable? Is it the combination of words? Is it something that is funny, rude, or serious? Is it something someone made up that makes them think of a vacation, or significant moment in their life? One potential avenue of future work would be to determine if there are such things as "memorable structures". All passwords in our system are parsed into their segments and a structure is built from the semantic categories. It would be interesting to see if there exists a structure that is more memorable than another one. For example, is the structure for "ilovebob" much more memorable than the structure for "barkingdog5". Currently, all we have to go off of is the frequency of structures that appear in our grammar (the number of times each structure was found), and we assume that a more probable structure is more memorable. It may be that there are much less probable structures (i.e., more secure) that are just as memorable. If we make suggestions using these structures, we could potentially increase the memorability of the passwords that are suggested while also increasing the security. The current version of PassMod appears to increase security while maintaining memorability.

4.6 Conclusion

In this chapter we have presented PassMod, a system capable of taking a password and semantically modifying it to make it more secure. We have designed the system with both memorability and security in mind. Unlike previous password modification systems, we attempt to preserve the meaning behind the original password and add segments that enhance or maintain the memorability, rather than hinder it (i.e., semantic modifications). In this way, we are able to increase the security of the password with no apparent reduction in memorability.

Through our user study, we have demonstrated that the passwords suggested by PassMod are memorable, and much more secure than their unmodified counterparts. Much work still has to be done in this area to enhance the user experience. For example, multiple suggestions presented at one time was a remark shared by many participants in our study. Participants were given the option to shuffle, but it seems many just opted to use the first suggestion, even if it wasn't one they particularly preferred. Other avenues of future work include trying to strengthen the memorability of the system even further by investigating what base structures (if any) are more memorable, and training the system on English grammar to try and create a more meaningful suggestion.

CHAPTER 5

COMPARISON OF APPROACHES

In this Chapter we discuss the differences between GeoPassNotes [40] and PassMod and compare both to traditional text passwords as seen in Al Omari et al. [4]. The password policy used by Al Omari et al. enforced a minimum of 8 characters in length containing at last one special character, number, and uppercase letter. In GeoPassNotes, the credential is the combination of a user-selected location on a digital map and an annotation for that location. The annotation can be a memory, significant event, or just a sequence of words the user can associate with the location. PassMod allows users to authenticate using a system-modified variant of a password the user provides. The system changes the password by adding or modifying components to make it more secure. The user must then authenticate using the newly modified password. This chapter aims at analyzing and comparing the usability (Section 5.2), memorability (Section 5.3), acceptability (Section 5.4), and security (Section 5.5) of these two approaches with one another and with text passwords to provide context and recommentdations.

5.1 Context and Limitations

As this chapter compares the results collected from three different studies, we discuss the limitations and merits of this comparison in terms of the three different studies that were conducted.

All three studies used the same study design; the participants were asked to create their credentials in session 1 (day one), login using their credentials one day later (session 2), and then login one week later (session 3). Participant demographic information was collected in session 1, and feedback on the studies and authentication scheme was collected at the end of session 3.

All three studies were conducted on a university campus and used students from UOIT as study participants. Al Omari et al.'s study also collected results from the general population and information technology professionals. We only compare the results of GeoPassNotes and PassMod studies to the students group of Al Omari et al.'s study as this is the most relevant.

One of the main differences between the three studies is the environment used for the study. Both PassMod and Al Omari et al.'s password study were fully online studies. For each session, participants were briefly told what they had to do, and were free to complete it from wherever they want. In these types of studies, participants may feel more comfortable and as such, may answer the questionnaires a little more honestly opposed to being watched in a lab setting. The downside of these studies is participants might be more distracted than in a lab setting; this may skew the results, especially the login times. GeoPassNotes was the only study that was conducted in a closed lab environment. As such, it is possible participants felt coerced into answering the questionnaires more positively as a member of the research team was observing them. Finally, the questions asked in each study had some differences; we compare only where the same question was asked.

5.2 Usability Comparison

The users from the GeoPassNotes study reported the system was user friendly and “fun” to use. When asked if they could easily use this method every day, 90% of users either agreed or strongly agreed (see Table 5.1). This bodes well for this system as its use is primarily in environments where accounts are logged into infrequently (i.e., once per week). This compares with PassMod where 47% of users reported being able to easily use this password every day. We would also like to mention that users might have interpreted this question as that they need to go through the modification process every time they login. In their everyday use after creation, the passwords generated by our system are no different than any other password in terms of their usability. PassMod was created to be quicker to use than GeoPassNotes, and as such, should be able to be used every day. We asked users of PassMod if they would use this method for some accounts if they knew it was more secure than passwords (see Table 5.4). 64% of users agreed to this which compares with 50% of users in Al Omari et al. [4]. Furthermore, Al Omari et al. found that 47% of the users in their study preferred an authentication system that was easier to use than passwords. This bodes well for GeoPassNotes given that users reported it was very user friendly and easy to use.

We also report on the usability in terms of login times. For GeoPassNotes, the average login times were 32, 37, and 47 seconds for sessions 1, 2, and 3, respectively. These times are relatively high for an authentication system, which could be in part due to the fact that it is a new approach and users were still getting used to it. We propose it would most appropriately be used in environments where logins are infrequent. PassMod on the

Usability Metric	**GeoPassNotes**	**PassMod**
I could easily use this method every day	80%	47%
I could easily use this method every week	90%	41%
I found this method too difficult to use	3%	27%
I found this method too time-consuming	17%	32%
I found this method easier than passwords	20%	N/A

Table 5.1 Usability comparison of authentication approaches.

other hand saw much more reasonable times. The average login times were 12, 11, and 12 seconds for sessions 1, 2, and 3, respectively. This is comparable to passwords without the PassMod system where the average login time for three sessions was 13.7 seconds [4]. The login time for PassMod is comparable to text passwords, but GeoPassNotes was ~3 times longer; however, user sentiment about the GeoPassNotes system appears to be more positive.

5.3 Memorability Comparison

We compare and contrast the memorability of GeoPassNotes and PassMod in this section as with both systems the study design was the exact same (three sessions spread across 8-9 days). The annotated location-passwords produced from GeoPassNotes proved to be memorable, both in practice and perceived. The majority of users from that study (66%) believed they would not forget their annotated location-password for another 3 months (see Table 5.3). The memorability of GeoPassNotes was very high, with some users commenting that the system was "an extremely memorable way to authenticate". In total, GeoPassNotes saw 21 failed login attempts over the course of the user study, but no forgotten annotated location-passwords (see Table 5.2). Al Omari et al. [4] asked users of their text password study if they would prefer a password that was slower to input but easier to remember for infrequently logged into accounts; 53% of users agreed. On the other hand, the passwords modified by PassMod received mixed results in terms of

Memorability Metric	GeoPassNotes	PassMod	Passwords (strict policy)
% users failed logins	13%	47%	N/A
% users password resets	0%	14%	11%

Table 5.2 Failed login comparison of authentication approaches.

Memorability Metric	GeoPassNotes	PassMod
I found this easier to remember than passwords	43%	0%
I had no trouble remembering my token	93%	55%
I think I could remember my token for one month	93%	59%
I think I could remember my token for three months	90%	41%
I think I could remember my token for six months	66%	27%
I think I could remember my token for one year	37%	23%
I think I would never forget my credential	43%	18%

Table 5.3 Memorability comparison of authentication approaches.

memorability. Users self-perceived memorability was low, with 41% stating they could remember their password for another 3 months. Throughout the course of the study, 6 users (4 in session 2, 2 in session 3) forgot their passwords and had to reset. This value is higher than in GeoPassNotes. In conclusion, we have found annotated location-passwords to be a memorable form of authentication when compared to traditional passwords and those modified by PassMod.

5.4 Acceptability Comparison

For the GeoPassNotes system, most users (90%) agreed they would use the system for some of their accounts. When asked if they would use it for most of their accounts, the majority remained neutral. We deduce this is most likely due to long login times. Almost all users reported they would use this method if they knew it was more secure than a password. Surprisingly, PassMod saw lower acceptance among users. 32% of users reported they would use this method for most accounts and for some accounts. Furthermore, 41% of users reported they would consider using this method for most

Acceptability Metric	GeoPassNotes	PassMod
I would prefer to use this method for most accounts	40%	32%
I would prefer to use this method for some accounts	90%	32%
I would use this method if I knew it was more secure than passwords	90%	64%
I would not use this method for any accounts	14%	27%

Table 5.4 Acceptability comparison of authentication approaches.

accounts; 55% of users agreed they would use this method. We were also interested to see if users preferred password creation rules (what they are used to) or the password modification method used in the user study. 41% of users prefer PassMod while 59% prefer the password creating rule method. Since few users had failed logins and password resets, and seem generally favourable of the system in their comments, we attribute the low reported acceptance due to users infrequently experiencing the nuisance of password creation rules because of password reuse. If users created separate passwords for every account that had to conform to each sites password composition policies, the password creation rules method may not be as plausible of a choice. We also note that this question is subject to user's perception of what typical password rules are; many sites apply different password policies, and Al Omari's study indicates that when a strict policy is applied, many users do not like it.

5.5 Security Comparison

In terms of security, we compare the resistance of GeoPassNotes, PassMod, and traditional passwords against popular password cracking attacks (see Table 5.5). The GeoPassNotes study was modelled off of the GeoPass system (see Chapter 3). We simulated a password cracking attack against the notes collected in our study and added their security to the related location based on which threat model (unknown, known, local) the location fell under. The results indicate the weakest annotated location-password contains ~37 bits

	% passwords vulnerable to	
	online attack	**offline attack**
GeoPassNotes	0%	10%
PassMod	0%	4.5%
Passwords (strict policy)	N/A	8.3%

Table 5.5 Different authentication schemes vulnerability to offline and online attacks.

of security, which is still enough to withstand an online attack. Florêncio et al. [20] recommend at least 10^{21} guesses in order to be considered secure against an offline attack. As such, the weakest annotated location-passwords would be vulnerable to an offline attack. We conducted a similar cracking attack against the original passwords and modified passwords collected throughout the PassMod user study. When trying to crack the original passwords, the most successful attack came from the semantic cracker, being able to crack 33.33% of the passwords. Surprisingly, the traditional passwords collected from Al Omari et al.'s [4] study held up surprisingly well against attacks, with only 8.3% being cracked using the semantic approach. We assume this is attributable to the fact that they enforced a password policy stating their password must be >= 8 characters in length, with at least one special character, one number, and one uppercase character. PassMod did not contain a password policy except that passwords must not be smaller than 5 characters and they must not be present on a blacklist of the top 3,500 most common passwords. The strengthened passwords created from PassMod were able to completely thwart the semantic attack, and limited JtR and PCFG to only cracking 3.33% and 5% of the strengthened passwords, respectively.

5.6 Comparison of Benefits Provided by Authentication Schemes

Bonneau et al. [10] present an evaluation criteria by which different authentication schemes can be evaluated. Note that this comparison is intended for web authentication and as such may not capture all desirable properties and trade-offs for other environments. Table 5.6 provides a comparison of the Usability (U1 - U8), Deployability (D1 - D6) and Security (S1 - S11) benefits that each authentication system contains.

Table 5.6 compares and contrasts the benefit that each authentication scheme provides with each other. It is meant to differentiate each authentication scheme. The rows are sectioned based on the different authentication schemes discussed. Finally, traditional text-based web passwords were provided for comparison with the approaches mentioned in this thesis.

The following sections explain each of these benefits and provide a comparative table according to Bonneau et al. [10].

5.6.1 Usability Benefits

This section explains the benefits as related to how easy the system is to use.

U1 - Memorywise-effortless

The scheme does not require that users remember anything at all.

U2 - Scalable-for-users

Scalable in terms of the user; multiple accounts does not increase the burden on the user.

U3 - Nothing-to-carry

Users do no need any extra device or materials to use the system. Quasi nothing-to-carry is granted if the user must use a device that they typically have on them at all times (Mobile Device).

U4 - Physically-effortless

No effort is required by the user aside from pressing a button. Quasi physically-effortless is granted if all the user has to do is speak or gaze at the screen.

U5 - Easy-to-learn

Users can learn how to use the system fairly easily.

U6 - Efficient-to-use

The login time for users is acceptably short.

U7 - Infrequent-errors

Legitimate users are granted access most of the time.

U8 - Easy-recovery-from-loss

User can regain the ability to authenticate if their credentials are lost.

5.6.2 Deployability Benefits

This section explains the benefits related to setting up and using the system in a real environment.

D1 - Accessible

Users are not prevented from using the system with a disability.

D2 - Negligible-cost-per-user

Low cost for entire deployment of the system.

D3 - Server-compatible

Providers don't need to change their existing authentication for it to work with the new scheme.

D4 - Browser-compatible

The system is expected to be compliant with current internet standards (HTML5 and Javascript-enabled browser).

D5 - Mature

System has been implemented by multiple parties and has had multiple studies done on it.

D6 - Non-proprietary

Anyone can use the system without having to pay royalties.

5.6.3 Security Benefits

This section explains the security benefits of each system.

	Spitzer	PassMap	GeoPass	GeoPassNotes	Passwords	PassMod
U1 - Memorywise-effortless						
U2 - Scalable-for-users						
U3 - Nothing-to-carry	■	■	■	■	■	■
U4 - Physically-effortless						
U5 - Easy-to-learn	■	■	■	■	■	■
U6 - Efficient-to-use					■	■
U7 - Infrequent-errors	■	■	■	■	□	□
U8 - Easy-recovery-from-loss					■	■
D1 - Accessible					■	■
D2 - Negligible-cost-per-user	■	■	■	■	■	■
D3 - Server-compatible					■	■
D4 - Browser-compatible	■	■	■	■	■	■
D5 - Mature			■		■	
D6 - Non-proprietary	■	■	■	■	■	■
S1 - Resilient-to-phyical-observation						
S2 - Resilient-to-targeted-impersonation				□		
S3 - Resilient-to-throttled-guessing			■	■	□	■
S4 - Resilient-to-unthrottled-guessing				■		■
S5 - Resilient-to-internal-observation	■	■	■	■	■	■
S6 - Resilient-to-leaks-from-verifiers						
S7 - Resilient-to-phishing						
S8 - Resilient-to-theft	■	■	■	■	■	■
S9 - No-trusted-third-party					■	■
S10 - Require-explicit-consent	■	■	■	■	■	■
S11 - Unlinkable	■	■	■	■	■	■

Table 5.6 Comparison of different authentication schemes according to Bonneau's categorization [10].

Legend
■ - contains benefit
□ - partially contains benefit (see Section 5.6 for details on column headers)
No value indicates absence of the benefit

S1 - Resilient-to-physical-observation

An adversary cannot observe login credentials and use them to authenticate themselves.

S2 - Resilient-to-targeted-impersonation

An adversary cannot login if they have knowledge of the users personal information (birth date, family, etc.)

S3 - Resilient-to-throttled-guessing

If the number of login attempts is limited by the system, a significant number of illegitimate users would be denied access.

S4 - Resilient-to-unthrottled-guessing

If the number of login attempts is not limited by the system, a significant number of illegitimate users would still be denied access.

S5 - Resilient-to-internal-observation

An adversary cannot intercept web traffic and capture login credentials.

S6 - Resilient-to-leaks-from-other-verifiers

The authentication server cannot leak anything that could compromise a user's account.

S7 - Resilient-to-phishing

An adversary cannot collect credentials by impersonating an authentication server.

S8 - Resilient-to-theft

An adversary cannot gain access to the system by stealing an object required for authentication.

S9 - No-trusted-third-party

The scheme does not rely on a third parties services.

S10 - Requiring-explicit-consent

The authentication process cannot start without a user specifically consenting to such.

S11 - Unlinkable

Multiple authentication servers cannot determine if the user is authenticating to both.

Based on the results presented, we see that GeoPassNotes allows users to create very memorable authentication tokens they feel they could remember for extended periods of time. However, the login times are longer than that of the average account. As such, GeoPassNotes is best suited in environments where logins are infrequent (i.e., once per week). The weakest form of GeoPassNotes is when the user directly labels a place that is local to the system. Even then, the security provided by GeoPassNotes is adequate to withstand an online attack [20]. PassMod on the other hand experienced much more reasonable login times when compared to passwords. The modification system was able to increase the length of passwords from an average of 10 characters to an average of 15 characters. Florêncio et al. state that after 10^6 and 10^{21} guesses, the efficacy of an online and offline attack, respectively, is significantly reduced. As such, we benchmarked our system at those levels and found that the system was completely resistant to online guessing attacks when throttling is being used. Given the high security and low login times, we suggest this system is most appropriate in everyday login situations. We also suggest that this system might be used to help users create more unique passwords they can remember for multiple accounts, reducing password reuse accross multiple accounts.

CHAPTER 6

CONCLUSION

In this thesis, we explore and evaluate two authentication schemes aimed at increasing the security of authentication while preserving / increasing memorability of credentials.

We implemented and evaluated a security enhancement for location-passwords called GeoPassNotes. Through a 30 person user study, we found that annotated location-passwords remain as memorable as their untagged counterparts, however yield much more security against attacks when our recommendations are applied. Compared to location-passwords [56], we found annotations only slightly increase login time. Although tagged location-passwords are highly memorable, and may have stronger security than text passwords, it is still only a viable option for accounts where logins are infrequent (e.g., to replace the role of text passwords or secondary/fallback authentication in financial accounts, where logins have been found to average 1.3 times/week [35]). This is because login times for GeoPassNotes are high (median login time 26-36 seconds) vs. under 10 seconds for traditional passwords. The results of our study also show that users are most open to using this system once/week.

We also implemented a password strengthening system we call PassMod. PassMod takes a user's password and strengthens it by adding components. These components are typically words, but can also be series of digits or symbols. In a 43 person user study, we evaluated the usability and memorability of the passwords produced by PassMod. The results indicate that, in practice, the usability of the passwords produced from PassMod are similar to traditional passwords. Login times are comparable (~10-13 seconds) to passwords, as are password resets. Overall, our system was able to increase the average character length of passwords from ~10 characters to ~15 characters. Furthermore, we conducted a password cracking attack using current best methods and found that at most, 5% of the strengthened passwords could be cracked (using PCFG). We propose this system is best suited for everyday environments where users have one important account they must remember a password for (e.g., a password manager master password). The reason is because PassMod can help users create and remember different, more secure variations of a password they already use. PassMod can help users remember more secure passwords since they are based on their original ones. This begs the question of whether passwords generated by the system are too similar to the originals, and the user could have just "slightly modified" the original. We analyzed the original and modified passwords collected from our study to see how different the system was making them. On average, the system needs to make 1.56 modifications to a password to make it secure. This also raises the average character count from 10 to 15 characters.

Through our research, it has been demonstrated that password security can be improved through the use of systems and strategies that are more compatible with human memory. With regards to our research questions presented in Chapter 1, we have found that:

1. Annotating a chosen place on a map can significantly increase the security of location passwords with very little reduction in memory.

2. Entering a memorable password and having it altered in a sensible way significantly improves the security of the password while preserving the memorability.

3. When the results of GeoPassNotes and PassMod are compared to text passwords, we see that both approaches are capable of generating a secure, memorable token. Due to login times, GeoPassNotes is most well-suited for environments where logins are infrequent. PassMod is most suited for everyday environments where a secure, memorable authentication token is required.

Our research into the memorability of passwords and alternative ways to help users remember credentials has demonstrated that other viable authentication approaches exist. GeoPassNotes is an memorable form of authentication. It was shown that it would be very difficult for an attacker, even if they know some information about the user, to crack a user's annotated location-password. GeoPass [56] has undergone more testing than GeoPassNotes, so a further study appears to be a good next step for the GeoPassNotes system. A larger field study that gathers usage statistics over a large period of time as well as interference between multiple GeoPassNotes would be an interesting next step. PassMod allows users to input their weak passwords and be suggested alternatives that have been deemed secure against a semantic guessing attack. It could also be used by users to help them come up with new passwords, before they create their accounts. This is the first iteration and study of the PassMod system. As such, there are many potential avenues of future work. Some include design modifications to the interface. Others include investigating the use of an English grammar opposed to a password grammar. This could potentially allow the system to make a grammatically correct password a larger percent of the time.

REFERENCES

[1] Abadi, M., Lomas, T. M. A., and Needham, R. (1997). Strengthening passwords. *Digital System Research Center, Tech. Rep*, 33:1–11.

[2] Al-Ameen, M. N. and Wright, M. (2014). A comprehensive study of the geopass user authentication scheme. In *arXiv preprint arXiv:1408.2852*.

[3] Al-Ameen, M. N. and Wright, M. (2015). Multiple-password interference in the geopass user authentication scheme. In *NDSS Workshop on Usable Security*.

[4] Al Omari, R. and Thorpe, J. (2016). Password user studies ecological validity. In Submission.

[5] Albanesius, C. (2012). Yahoo Voices Breach Exposes 453,000 Passwords. http://www.pcmag.com/article2/0,2817,2407015,00.asp, site accessed February 2016.

[6] BBC (2014). LinkedIn Passwords Leaked by Hackers. http://www.bbc.com/news/technology-18338956, site accessed May 2014.

[7] Bicakci, K. and van Oorschot, P. C. (2011). A multi-word password proposal (gridword) and exploring questions about science in security research and usable security evaluation. In *Proceedings of the New Security Paradigms Workshop*, pages 25–36.

[8] Biddle, R., Chiasson, S., and Van Oorschot, P. C. (2012). Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys*, 44(4):19.

[9] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

[10] Bonneau, J., Herley, C., Van Oorschot, P. C., and Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*, pages 553–567.

[11] Brown, A. S., Bracken, E., Zoccoli, S., and Douglas, K. (2004). Generating and remembering passwords. *Applied Cognitive Psychology*, 18(6):641–651.

[12] Cheswick, W. (2013). Rethinking passwords. *Communications of the ACM*, 56(2):40–44.

[13] Chiasson, S., Forget, A., Stobert, E., van Oorschot, P. C., and Biddle, R. (2009). Multiple password interference in text passwords and click-based graphical passwords. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 500–511.

[14] Davies, M. (1990–2015). Corpus of Contemporary American English. http://corpus.byu.edu/coca/, site accessed July, 2013.

[15] dazzlepod (2014). Password list. http://dazzlepod.com/site_media/txt/passwords.txt, site accessed February 2014.

[16] Dunphy, P., Heiner, A. P., and Asokan, N. (2010). A closer look at recognition-based graphical passwords on mobile devices. In *Proceedings of the Symposium on Usable Privacy and Security*.

[17] Dunphy, P. and Yan, J. (2007). Do background images improve draw a secret graphical passwords? In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 36–47.

[18] Egelman, S., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., and Herley, C. (2013). Does my password go up to eleven?: The impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2379–2388.

[19] Florencio, D. and Herley, C. (2007). A Large-Scale Study of Web Password Habits. In *Proceedings of the International World Wide Web Conference*.

[20] Florêncio, D., Herley, C., and Van Oorschot, P. C. (2014). An administrator's guide to internet password research. *Large Installation System Administration Conference*, pages 44–61.

[21] Forget, A., Chiasson, S., van Oorschot, P. C., and Biddle, R. (2008a). Improving text passwords through persuasion. In *Proceedings of the Symposium on Usable Privacy and Security*, pages 1–12.

[22] Forget, A., Chiasson, S., van Oorschot, P. C., and Biddle, R. (2008b). Persuasion for stronger passwords: Motivation and pilot study. In *Persuasive Technology*, pages 140–150. Springer.

[23] Garside, R., L. G. and Sampson, G. (1988). *The Computational Analysis of English: A Corpus-based Approach*, volume 57. Longman.

[24] Google (2016). Purchasing the google maps apis premium plan. https://developers.google.com/maps/premium/faq#purchasing-the-google-maps-apis-premium-plan, site accessed June, 2016.

[25] Harris, E. (2014). Data breach hurts profit at target. http://www.nytimes.com/2014/02/27/business/target-reports-on-fourth-quarter-earnings.html?_r=1, site accessed February, 2016.

[26] Hayashi, E. and Hong, J. (2011). A diary study of password usage in daily life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2627–2630.

[27] Houshmand, S. and Aggarwal, S. (2012). Building better passwords using probabilistic techniques. In *Proceedings of the Annual Computer Security Applications Conference*, pages 109–118.

[28] Inglesant, P. and Sasse, M. A. (2010). The true cost of unusable password policies: Password use in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 383–392.

[29] Jermyn, I., Mayer, A. J., Monrose, F., Reiter, M. K., Rubin, A. D., et al. (1999). The design and analysis of graphical passwords. In *Usenix Security*.

[30] Ji, S., Yang, S., Wang, T., Liu, C., Lee, W.-H., and Beyah, R. (2015). Pars: A uniform and open-source password analysis and research system. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 321–330.

[31] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and Lopez, J. (2012). Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *IEEE Symposium on Security and Privacy*, pages 523–537.

[32] Khot, R. A., Srinathan, K., and Kumaraguru, P. (2011). Marasim: A novel jigsaw based authentication scheme using tagging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2605–2614.

[33] Klein, D. V. (1990). Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the USENIX Security Workshop*, pages 5–14.

[34] Krebs, B. (2014). Banks: Credit card breach at home depot. http://krebsonsecurity.com/2014/09/banks-credit-card-breach-at-home-depot/, site accessed February, 2016.

[35] Kristo, G., Janssen, S. M., and Murre, J. M. (2009). Retention of autobiographical memories: An internet-based diary study. *Memory*, 17(8):816–829.

[36] Kuhn, B. T. and Garrison, C. (2009). A survey of passwords from 2007 to 2009. In *Information Security Curriculum Development Conference*, pages 91–94.

[37] Labs, D. (2010). Brief Analysis of the Gawker Password Dump. https://duo.com/blog/brief-analysis-of-the-gawker-password-dump, site accessed February 2016.

[38] LastPass (2008). LastPass Password Manager. https://lastpass.com, site accessed May 2016.

[39] Li, H. and Abe, N. (1998). Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, 24(2):217–244.

[40] MacRae, B., Salehi-Abari, A., and Thorpe, J. (2016). An exploration of geographic authentication schemes. *IEEE Transactions on Information Forensics and Security*, (to appear).

[41] McCrum, R., Cran, W., and MacNeil, R. (2003). *The story of English*. Penguin Group USA.

[42] Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8).

[43] Nelson, D. L., Reed, V. S., and Walling, J. R. (1976). Pictorial superiority effect. *Journal of Experimental Psychology: Human Learning and Memory*, 2(5):523.

[44] Openwall (2002a). John the Ripper Password Cracker. http://www.openwall.com/john/, site accessed January 2014.

[45] Openwall (2002b). JohnTheRipper. https://github.com/magnumripper/JohnTheRipper, site accessed January 2014.

[46] Schechter, S., Brush, A. B., and Egelman, S. (2009). It's no secret. measuring the security and reliability of authentication via "secret" questions. In *IEEE Symposium on Security and Privacy*, pages 375–390.

[47] Schmidt, D. and Jaeger, T. (2013). Pitfalls in the automated strengthening of passwords. In *Proceedings of the Annual Computer Security Applications Conference*, pages 129–138.

[48] Schmitt, N. and McCarthy, M. (1997). *Vocabulary: Description, acquisition and pedagogy*, volume 2035. Cambridge University Press Cambridge.

[49] Security, S. (2014). Leaked Passwords. http://downloads.skullsecurity.org/passwords/, site accessed May 2014.

[50] Shay, R., Komanduri, S., Kelley, P. G., Leon, P. G., Mazurek, M. L., Bauer, L., Christin, N., and Cranor, L. F. (2010). Encountering stronger password requirements: User attitudes and behaviors. In *Proceedings of the Symposium on Usable Privacy and Security*.

[51] Social Security Administration (1935). Beyond the Top 1000 Names. http://http://www.ssa.gov/oact/babynames/limits.html, site accessed September 2013.

[52] Spitzer, J., Singh, C., and Schweitzer, D. (2010). A security class project in graphical passwords. *Journal of Computing Sciences in Colleges*, 26(2):7–13.

[53] State of California Department of Justice (2014). Submitted breach notification sample. http://oag.ca.gov/ecrime/databreach/reports/sb24-47706, site accessed February, 2016.

[54] Stubblefield, A. and Simon, D. (2004). Inkblot authentication.

[55] Sun, H.-M., Chen, Y.-H., Fang, C.-C., and Chang, S.-Y. (2012). Passmap: a map based graphical password authentication system. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 99–100.

[56] Thorpe, J., MacRae, B., and Salehi-Abari, A. (2013). Usability and security evaluation of geopass: a geographic location-password scheme. In *Proceedings of the Symposium on Usable Privacy and Security*.

[57] Tripadvisor (2000). Kaufer, Stephen and Steinert, Langley . http://www.tripadvisor.com, site accessed March, 2014.

[58] van Oorschot, P. C. and Thorpe, J. (2011). Exploiting predictability in click-based graphical passwords. *Journal of Computer Security*, 19(4):669–702.

[59] Veras, R., Collins, C., and Thorpe, J. (2014). On semantic patterns of passwords and their security impact. In *Proceedings of the Network and Distributed System Security Symposium*.

[60] Veras, R., Thorpe, J., and Collins, C. (2012). Visualizing Semantics in Passwords : The Role of Dates. In *Proceedings of the International Symposium on Visualization for Cyber Security*, pages 88–95.

[61] Weidenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. (2005). Authentication using graphical passwords: Basic results. *Proc. Human-Computer Interaction International*.

[62] Weir, M., Aggarwal, S., Collins, M., and Stern, H. (2010). Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 162–175.

[63] Weir, M., Aggarwal, S., De Medeiros, B., and Glodek, B. (2009). Password cracking using probabilistic context-free grammars. In *IEEE Symposium on Security and Privacy*, pages 391–405.

[64] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. (2005). Authentication using graphical passwords: Effects of tolerance and image choice. In *Proceedings of the Symposium on Usable Privacy and Security*, pages 1–12.

[65] Wright, N., Patrick, A. S., and Biddle, R. (2012). Do you see your password?: Applying recognition to textual passwords. In *Proceedings of the Symposium on Usable Privacy and Security*.

[66] Yadron, D. (2014). Man behind the first computer password: It's become a nightmare. http://blogs.wsj.com/digits/2014/05/21/the-man-behind-the-first-computer-password-its-become-a-nightmare, site accessed May 2016.

[67] Zhang, Y., Monrose, F., and Reiter, M. (2010). The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proceedings of the ACM Conference on Computer and Communications Security*.