Personalized Typo-Tolerant Password Checking: A Tradeoff Between Usability and Security?

Ayo Osonowo
osono003@umn.edu

Jack Poehlmann
poehl083@umn.edu

Ben Smith
smit9524@umn.edu

Matt Strapp
strap012@umn.edu

Noah Zimmerman
zimm0508@umn.edu

ABSTRACT

A typo-tolerant password verification system, TypTop, is proposed that actively learns from users' typos and allows authentication of mis-typed passwords. Minimal security loss is reported as a part of this claim: TypTop is provably cryptographically secure, and thus threats to its security come in the form of offline guessing attacks. TypTop utilizes a cache of a user's common typos it deems are "safe," and with which it will allow users admittance. In this manner, there essentially becomes a pool of allowable passwords, each related to each other in that they are some sort of permutation of the true password. We consider the hypothetical in which the cache state is compromised and an oracle is obtained that can make requests to the cache to determine if a password is present or not. For simplicity, we consider a cache of plain text passwords.

We utilize OMEN, a modern password guessing system using ordered Markov enumeration to learn from a password list and generate a list of potential passwords in decreasing order of probability. We compare the generated passwords to those present in the cache to quantify the success of an offline guessing attack. We consider multiple cache sizes, as well as scenarios in which one password is compromised. In this case, OMEN can learn from this password and use it to guess other passwords present in the cache, including the original password. Our experimental analysis indeed supports the TypTop authors' claim of minimal security loss. Overall system security is dependent upon the strength of the password chosen, as opposed to the cache size or any form of cache leakage.

In addition to verifying the claims of minimal security loss, we wanted to determine whether or not users would actually want to use a system like TypTop. To test this, we sent out a survey asking several questions regarding the participant's perception of a typo tolerant password system, as well as passwords in general. The results showed mixed opinions– while a majority of responses indicated a lack of trust in the system, there was also a good amount of openness to trying the technology, citing convenience. A majority of users also said they would be more likely to use a more complex password with a system like TypTop in place, which could potentially offset the security loss inherent in the system.

1 INTRODUCTION

A survey of the modern internet will find passwords used extensively compared to other forms of authentication, such as biometrics or two-factor. Passwords remain prevalent due to their ease of implementation from both a user and a technical side; the average number of passwords an individual has in 2019 is 25 [12]. However, their ease-of-use leads to security flaws as well: often users will choose easy-to-remember passwords because the task of remembering 25 passwords is too much.

Strides in modern cryptography and security have also impacted "attacker" security research, and the general majority of user passwords are easily cracked [4] via brute-force, dictionary, and other more complex attacks. It is for this reason that users are encouraged to choose robust passwords that conform to a set of recommendations (for example, longer than 16 characters, use special symbols, use a combination of upper and lowercase characters, alphanumeric passwords). Longer, more complex passwords see a higher rate of typos. A study by Chatterjee et. al in 2016 showed that 9.3% of login attempts for Dropbox users were denied based on small, easy to correct typos [1]. Some typo tolerant password checking systems are already implemented, such as the system by Facebook that allows common typos like accidentally leaving caps lock on, or mis-capitalizing the first character. However, the most prevalent typos only account for 20% of password typos made by users, leaving the majority unaccounted for.

Enter TypTop (pronounced: "tip top"), a typo tolerant password checking system that learns users' most common password typos, and will overlook them. The cryptographic security of this system is well-documented, and we seek not to examine this aspect of security of the TypTop system, but examine instead the usability and customizability of the system to provide an optimally secure yet tolerant system. We will explore users' perceptions on such a system, as well as security implications of altering cache size and an attack in which a portion of the cache has been "leaked" to an attacker.

The easiest way to do this is through an offline guessing attack in which we can model the state of TypTop at a given point, and measure the ease in which a password guessing tool is able to either guess the password or one of its allowable typos. We can successfully model the cache state by following the caching policies laid out in Chatterjee et. al's paper [2]. The password guessing tool we use is called *OMEN* (**O**rdered **M**arkov **EN**umerator) [3], a probabilistic algorithm based on the Markov model used to generate passwords in decreasing order of likelihood of being used.

2 BACKGROUND AND RELATED WORK

In common password based authentication schemes, users are allowed access to a system only by inputting their exact username and password. Because of this unforgiving system, users often choose passwords that are easy to remember, such as names, dates, pet's names, etc. [6]. Even "complex" user-chosen passwords, such as those that contain varying capitalization, numbers, and special characters are still formatted in a way that can be remembered by humans [5]. Because of this, human-chosen passwords will always be more easily compromised via guessing attacks. We look to research in natural language processing (NLP) that uses probabilistic models to study the way humans generate sequences of characters and words [7]. The most common scheme employed in this research is the use of Markov-based models, which assign states and probabilities of each state changing into a different state. Consider two abstract states, A and E. The probability that A switches to B may be 0.4, while the probability it does not switch is 0.6. B may be assigned similar state changing probabilities, and in this way we can analyze the execution of a system based on probabilities of state changing. A diagram is presented in Figure 1. Extrapolation from this model is straight-forward, and it is Markov-based models such as these that are used in NLP, and thus password guessing programs. Markov-based models are discussed further in Section 6 with OMEN.

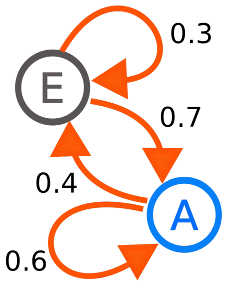

Figure 1: An example of a Markov chain, as described in Section 2.

Estimating password strength is an area of ongoing research in computer security. We generally define password strength as degrees of entropy, i.e. how resistant a password is to a sort of brute-force attack. The problem of estimating password strength is difficult because of the variety of attacks an attacker can perform, many stemming from the problem of human-chosen passwords as outlined above. The strongest theoretical password is one that takes an attacker a substantial amount of time to guess solely through iterative guessing, i.e. guessing every possible combination of characters (brute-force attack), and no dictionary/other attacks are possible. A strong password under this scheme is one that is longer than fifteen characters, and consists of completely random letters, numbers, and special characters [10]. We restrict our password analysis to ISO-8859-1 (Latin) characters, excluding for example, characters from the Cyrillic alphabet.

`zxcvbn` is a modern password strength estimation tool that determines the guessability of a password under sorts of dictionary attacks, such as string mangling (e.g. apple -> Apple -> ApPle -> aPpLE), regular expression matching, and a brute-force attack. Each of these concepts have been proposed separately [9] [8] [11], but zxcvbn combines these approaches in an easy-to-use and -understand tool. For an inputted password, zxcvbn will output the parts of the password it has identified to be vulnerable to certain attacks, as well as the overall password strength and even advice for strengthening a password if it is especially vulnerable. For example, for the string `p@ssW0rD2005qwerty`, zxcvbn will break this into distinct parts: `p@ssW0rD`, `2005`, and `qwerty`. It analyzes `p@ssW0rD` as being vulnerable to a dictionary attack that uses string mangling (l33t speak), `2005` as matching the regex for a recent year, and `qwerty` as being a common character combination due to the layout of English keyboards. It also outputs estimated guessing time under online/offline attacks, as well as different hashing schemes. We use `zxcvbn` in our research to model attacks on caches containing different strength passwords.

We further consider research [6] that discusses the regularity of password reuse. As discussed previously, with an average of dozens of online accounts and passwords, users often reuse passwords to decrease the amount of memorization needed to access services. The above cited research studies the rate at which users may reuse passwords, but also considers the case in which users may partially reuse passwords, i.e. changing certain characters, adding or removing characters, or even just adding an `!` to the end of their existing password. Pearman et al. found that users with multiple passwords reuse approximately 80% across all domains. We consider TypTop to be especially vulnerable in the case where its user has reused or partially reused a password, which we will analyze and incorporate into our evaluation of its overall security.

3 OVERVIEW OF THE TYPTOP SYSTEM

TypTop uses a cache of users' most common typos, and if a user enters a password that TypTop has learned is a common typo (i.e. is present in the cache), it will admit them. The user's true password is also stored in the cache. Before typos are allowed into cache, many typos are stored in an intermediate wait-list where they are only eventually cached if the user continues to make this exact typo. Typos are categorized deterministically via an algorithm: how can we determine the difference between a random string and a true password typo? For each string entered that is not the exact password, TypTop determines the Damerau-Levenshtein (DL) distance between these two strings. The DL distance between two strings is defined as the minimum number of substitutions (insert/delete characters, swap the case of characters, and other similar operations) needed to transform one string into the other [13]. This algorithm extends the existing "edit distance" algorithm, also known as the Levenshtein distance, by allowing transposition of characters [13]. An entered string is considered a legitimate typo if the distance between true password and entered string is at most some small , such as 2, as the authors' used in their original paper.

Once a string is entered and it is determined to be a valid typo (or, the true password), there are two options: 1) it is in cache, or 2) it is not in cache. The second case is simpler: here, the typo is simply encrypted using the public key and inserted into the wait-list. In this scenario, the user is not granted admission. If the typo is in the cache, a series of things happen: first, the secret key is decrypted (discussed below) and used to (secondly) decrypt the wait list. Both cache and wait-list are updated according to a given caching policy. In this step, typos in the wait-list may move to cache, or be discarded. Similarly, typos in the cache may either remain there, or be discarded. Once the cache and wait-list states are updated, the user is admitted. These cache and wait-list states will be used for subsequent logins.

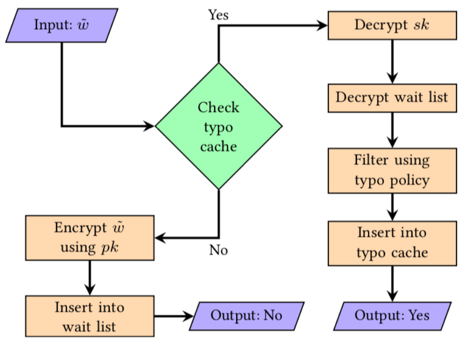

Figure 2: The execution of TypTop as a user enters a password, $\vec{w}$, as described in Section 3.

3.1 Security Overview

Cryptographic Security. It is important to consider the cryptographic security of TypTop. Both symmetric and public-key encryption are used across the system. Each wait-list and cache entry is encrypted via symmetric encryption. Each of these symmetrically encrypted entries is further encrypted as follows: the most recent entry in the wait-list is encrypted via the public key, and each subsequent wait-list entry is encrypted using the cipher text of the previous entry. A random salt is used for all encryption schemes. The secret key is also encrypted via symmetric encryption, with a new symmetric scheme based on the user's password. Chatterjee et al. have extensive proofs of their encryption schemes in their original paper(s), and we assume them to be correct [2].

Online and Offline Guessing Attacks. Cryptographic security aside, we turn now to the security of TypTop against guessing attacks. While guessing attacks come in two flavors, online and offline, we are principally concerned with offline guessing attacks. Online guessing attacks are interesting to us as well. However, online guessing attacks are more easily mitigated by use of login attempt throttling, maximum number of login attempts, and flagging logins from unknown IP addresses, for example. TypTop successfully employs all of these things, by only allowing 3 logins per minute, locking the system at 10 incorrect logins for an unspecified period of time, and treating logins from different IP addresses differently from those from a known IP address (typos from an unknown IP address are added to neither wait-list nor cache). Knowing this, we can assume that the number of guesses a blind/semi-blind attacker would need to make will be greater than ten, and thus can omit from our study. An interesting scenario to consider is the one in which an attacker "poisons" the cache, by interleaving their (typo-accepted) guesses with the true password, since any authentication by TypTop triggers a cache update. This sort of online attack may have a goal of denial of service, as opposed to simple authentication, and a successful completion of this attack may render TypTop unusable to the user, except by way of their true password. We consider this scenario outside the scope of this paper, and defer it to further research.

With this in mind, we consider the case in which the system state is somehow replicated and an oracle is obtained to make queries into the cache and wait-list, which we will define as an offline guessing attack. In this sort of attack, an attacker is limited only by the computational resources and time they have available to them. In the authors' original paper, they have a thorough mathematical proof about security in the face of an offline guessing attack, but as we know, proofs in the area of computer security are often incomplete, and thus have a limited scope of **meaningfulness** in the sense of the "real-world" [6]. The original authors limit their security proof to password distributions provided by the RockYou data set, as well as by length: they consider only passwords between 7 and 12 characters, and eliminate passwords that contain excessive special characters. In our analysis we do not consider such limitations, and focus on human-chosen passwords.

4 OMEN OVERVIEW

The OMEN system is a Markov-based enumeration system that outputs password guesses based on their probability of being correct. Markov-based guessing systems generate passwords principally on the fact that human generated passwords are seldom random - humans use words, phrases, and predictable letter and number ordering in their passwords. Consider the 2-grams "th" and "tk." "Th" is a much more common 2-gram in English words than "tk," and thus if we are generating a password we are much more likely to generate a real password containing "th" than "tk." Markov models in password based guessing systems are not a new concept, and in fact have been present in password guessing systems and security research for quite some time [5]. Popular password guessing system John the Ripper includes a Markov mode, yet we consider the use of OMEN to be better because of its ordering.

The main difference between OMEN and other Markov-based systems is that it is able to not only determine likely passwords based on a training set, but it is able to compute the probability that a password it has generated will be correct in comparison to other passwords it has generated. By guessing passwords in order of decreasing likelihood of being correct, we are able to, in turn, significantly decrease the amount of time spent guessing [5]. A previously proposed Markov-based guessing algorithm sorts n-gram probabilities into "bins," and for each bin it will output associated passwords, i.e. it will output passwords grouped by probability, but groups in no particular order (source). OMEN extends this algorithm by ordering these password groups. We use OMEN because of its speed in password guessing, as well as its ease of training. For our use purposes, we use OMEN with 4-grams and a fixed alphabet size of the 72 most common characters, {`a...z, A...Z, 0..9`, `!`, `.`, `*`, `@`, `-`, `_`, `$`, `#`, `<`, `?`}.

5 GENERAL PERCEPTION OF A TYPO-TOLERANT SYSTEM

To supplement the testing of the TypTop system, we decided to poll a sample on their thoughts on a typo-tolerant password system. The survey consisted mainly of statements in which the participant would need to answer using the Likert Scale. The first part of the survey asked individuals on what their trust levels would be in a system that would accept their most commonly made password typos. The second part of the survey polled users habits and beliefs in passwords more generally. Finally, we asked participants to write down why or why not they would trust a typo-tolerant password system.

5.1 Perception of a Typo-Tolerant System

The first aspect of user perception we wanted to collect data on was the generalized perception of a typo-tolerant password system. To do this, we decided to survey users with two statements: I would trust a login system that adapts to my own common password typos and a login system that accepts the user's common password typos is more secure than an exact password matching system.

The first two statements shown to the participants were used to build a baseline understanding of how the general user population would feel about a typo-tolerant system. The results of the first statement, "I would trust a password system that would accept my common typos", revealed disagreement.

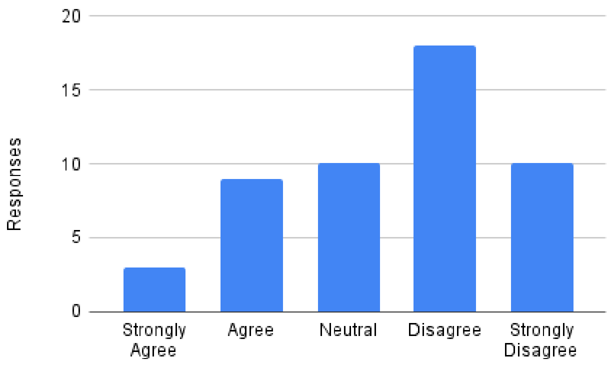

Figure 3: "I would trust a password system that would accept my common typos"

Of those surveyed, 56% either responded disagree or strongly disagree while only 24% saying they agree or strongly agree. The second prompt only emphasized the results of the first. When asked whether they believed if a password system was more secure than a traditional system, 74% of participants either disagreed or strongly disagreed.

The trends of Figures 3 and 4 emphasize that general users do not currently trust a typo-tolerant system or feel confident that such a system would protect them. Many of the open-ended responses highlighted the perception that a system that would allow certain common typos has much less entropy due to the multiple accepted password responses. However, many participants also emphasized that a system such as TypTop would be very convenient and that they would be excited to use such a system. This was shown through responses such as, "Seems cool. Would try," or "Honestly it definitely seems less secure, but it would be very convenient."

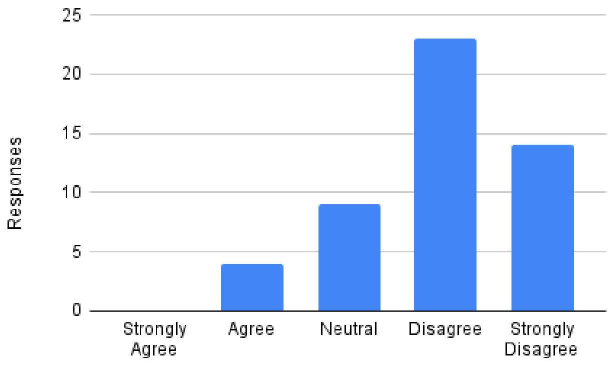

Figure 4: "A login system that accepts the user's common password typos is more secure than an exact password matching system."

5.2 Perceptions of Passwords Generally

Our next goal in understanding user perception on a typo-tolerant password system was to gauge their password habits and consistency. In order to gather this data, we first asked participants if they would use a more complex password for their logins if they knew that their own common typos would be learned and accepted.

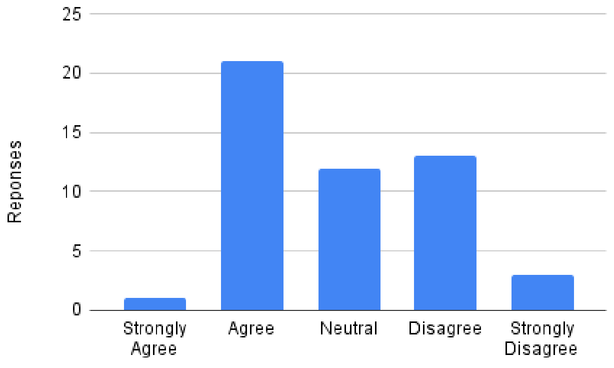

Figure 5: "I would use a more complex password if I knew that my common typos would be accepted."

According to Figure 5, there is not a clear consensus to this poll. It trends slightly towards agreement, with 44% either agreeing or strongly agreeing, while 32% of participants disagreed or strongly disagreed. We do not believe this gap alone is enough to definitively conclude that an adaptable typo-tolerant system would convince users to add more complexity in their password habits. To extend the results from Figure 3, we decided to poll users' password habits on a general scale. We provided the statement, "regardless of the password system, I would not change my password habits." The responses were clear cut from this statement as shown in Figure 6. 64% of those surveyed agreed or strongly agreed while only 12% disagreed. Clearly, users cling to their personal password habits and it would take a revolutionary new system to spark change.

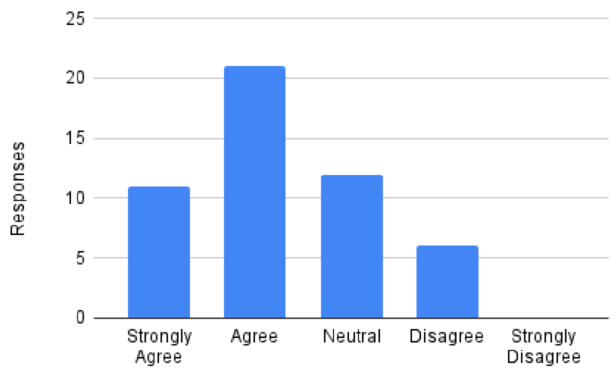

Figure 6: "Regardless of the password system, I would not change my password habits."

According to Figures 3 and 4, without explaining the intricacies of a typo-tolerant password system, users would not trust or properly utilize TypTop. They fear less entropy for their password and would not be comfortable using such a system; however, according to Figures 5 and 6, users would not change their passwords or password habits to possibly combat this fear. Initially, we believed that a system that would allow them easier access would incentivize users to embrace more complex passwords and safer habits, or if they were afraid of less entropy, would be more likely to choose a more complex password. Instead, those surveyed showed skepticism towards a typo-tolerant password system and a stubbornness to their own password habits.

5.3 Possible Skews in our Responses

While we did our best to ensure our survey would produce as accurate of results as possible, as with any survey, there are a number of factors that skewed our data. The first is that our data pool was relatively narrow. We sent the survey to as many people as possible, but not all of them responded. In addition, most of those who did were computer science majors at the University of Minnesota. This meant a large percentage of our responses came from people in a similar age group and geographical location, all of whom (presumably) with similar interests, skill sets, and experiences. To increase the accuracy of our results, ideally we would have collected data from more people from a larger and more diverse population.

Another aspect which may have skewed our results was the lack of specificity on the survey itself. In the interest of clarity and simplicity, we did not explain the ins and outs of how the TypTop system works; we only made reference to a "password system which accepts common password typos". While this is an accurate description of TypTop, a user who is unfamiliar with the system may misinterpret that statement or make assumptions which may not be entirely accurate. Specifically, the way we described TypTop may have contributed to the strongly negative opinions about password systems accepting typos. However, this does highlight the inherent problems the system faces in terms of perception–objectively, it is less secure than a typical password system, no matter how small the security loss is, and the vast majority of users of a system like TypTop should it become widely used would have little to no knowledge of how said system works. In fact, by over-explaining the system and emphasizing how minimal the loss of security is, it is possible the survey would have been skewed in the other direction.

A third source of skewed data in our survey was the format of the questions themselves. Most of the questions were statements that participants stated how much they agree or disagree with, and only two were open-ended. In addition, the questions were designed with the assumption that a negative evaluation of the security aspects of the system implied a negative view of the system as a whole. This assumption was exposed as faulty by the fact that a number of responses indicated interest in the system *despite* its shortcomings. On the other hand, the opposite assumption–that a positive view of the system's security implied a positive view of the system itself–was also proven false; there was a contingent of respondents who acknowledged the system's safety, but still would not use it, believing there was not a strong enough benefit to warrant such a change.

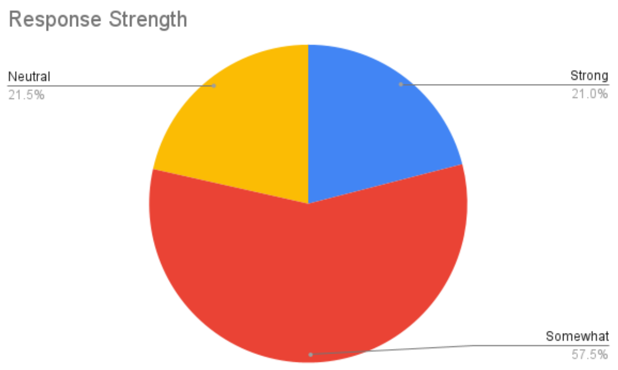

Figure 7: Strength of responses to all questions

Lastly, we discovered that those surveyed did not feel strongly about any of the topics asked to them. According to Figure 7, only 21% of respondents had a strong opinion on one or more of the statements while 21.5% answered neutral to one or more. One possible explanation to this could be a lack of understanding of the meaning of the statement. Typically, neutral is a response given when a respondent has no familiarity with the question or fails to understand what is being asked of them. In hindsight, we could have provided a detailed explanation of what we mean by a "typo-tolerant" password system before presenting any of the statements; however, we feared in doing this, we would have a lower participation rate. Instead, we incorporated a loose definition of a typo-tolerant password system in some of the statements, such as, "I would use a more complex password if I knew that my common typos, for example missed capitalization, would be accepted." This lack of familiarity could have skewed our data towards the center instead of having cleaner trend lines in our poll responses.

6 OFFLINE GUESSING ATTACKS AGAINST TYPTOP

6.1 Setting up our Testing Environment

It is important to evaluate the security of the typo-tolerant system under different scenarios, representing different password strength scenarios. In each scenario, we have access to the current cache/wait-list state (hereon referred to simply as cache state), stored as a plain text file. The scenarios we will consider are: weak passwords, medium-strength passwords, and strong passwords. We can measure password strength by using the tool `zxcvbn`. `zxcvbn` is a password strength estimation tool that uses regular expressions to search for patterns in a given string. It outputs data about the inputted string, including analysis of strength on a scale from 0 (weakest) to 4 (strongest) [10]. We consider 3 scenarios: a cache state that contains a password of `zxcvbn` strength 0, a cache state that contains a password of strength 2, and a cache state that contains a password of strength 4. We will refer to these scenarios as least-secure, medium-secure, and most-secure, respectively. The chosen passwords are: least-secure:`password`; medium-secure: `JohnDoe50`; most-secure:`1999b@tm@n&r0bin!`

In the author's original paper, they examine not only the security implications of cache size, but also of the implications of different caching schemes, such as **L**east **R**ecently **U**sed (LRU), **L**east **F**requently **U**sed (LFU), and **M**ost **F**requently **U**sed (MFU). Table 1 features the definitions the paper uses for each replacement policy. They simulate cache states and attacks based on different password distributions, passwords present in cache, and different caching schemes. They determine that the MFU caching policy is the most secure. We accept their analysis, and consider evaluation of different caching schemes outside the scope of this paper, since MFU is the caching policy used by TypTop when deployed to users, and this is the security we are principally concerned with.

Scheme	***Replacement Policy***
LRU	Replace least recently used typo with input
LFU	Replace least frequently used typo and its frequency with input and its frequency
MFU	Make any replacements to ensure n most frequent typos remain in cache
Best-t	Initialize cache with t most probable typos, never alter cache

Table 1: Table of Replacement Policies

We train OMEN on the RockYou dataset, and have it generate passwords, checking them against the plaintext cache passwords. For each evaluation, OMEN generates 50,000,000 password guesses in decreasing order of likelihood of being correct based on the training data. It further checks each generated password against those present in the plaintext cache state, marking matches, should they occur. It does not learn adaptively while it runs, so a password match during execution has no effect on further generated passwords. OMEN outputs the number of password matches it generates and the number of passwords of each length generated, which we use in our analysis, as well as other statistics from its execution.

6.2 Examining the Security of Different Sized Caches

To simulate an attack against different sized caches, we can create three cache states for each "scenario" (one scenario = one of least-secure, medium-secure, most-secure). For each scenario, we construct three caches: one of two typos, one of five typos, and one of ten. Each cache state also contains the true password. Typos and the true password are stored in plain text form. OMEN can be run to test its generated passwords against passwords in a given file, which is how we will run it, against our three cache state files for each scenario.

It may be no surprise that the password `password` is an insecure choice. For each cache state in the least-secure scenario, OMEN was able to determine approximately 25-30% of the passwords in the cache. Concretely, it could determine an average of 26.19% of the cached typos in the least-secure cache states. It should be kept in mind that OMEN generates passwords based on its input dictionary — while a typo such as `passworf` may seem obvious to us as humans, OMEN may not generate it because it favors true English words, especially when its training dictionary consists of mainly true English words.

The password `JohnDoe50` fared slightly better: on average OMEN was able to determine approximately 20.08% of the cached passwords. Finally, the password `1999b@tm@n&r0bin!` was able to be cracked by OMEN 0% of the time. Solely based on cache size, it would appear that the largest security risk is still based on password strength: stronger passwords are less likely to be cracked, since their typos are most likely also strong "passwords." While our password `password` was easily cracked, the stronger `1999b@tm@n&r0bin!` was unable to be cracked, even when part of a set of 10 related typos.

We can further support this from [7], which says that human generated passwords will always be more easily cracked then those generated completely randomly. While the most-secure password has a `zxcvbn` strength of 4, it still contains words (`batman` and `robin`) and a recent year (1999). However, our use of OMEN was unable to crack this password or any of its 2, 5, or 10 typos. This should give users some hope, that it is indeed possible to have a secure, unguessable password that is still human readable and rememberable.

6.3 Examining the Security of a Cache Leak for Different Strength Passwords

We can simulate a worst-case scenario in which one or more of the typos present in the cache or the wait-list has been accessed. In this scenario, OMEN can be trained using the leaked password, with weight over other strings in its training dictionary. From here it is able to generate similar passwords (i.e. similar typos) to try. As before, we can evaluate security using three different scenarios: a weak password, a medium-secure password, and a strong password. However, to evaluate the strength of TypTop in the face of a leaked cache style attack, we will need to retrain OMEN for each evaluation, appropriately modifying the training dictionary each time. For example, we may include `passworf` in OMEN's training dictionary for the least-secure scenario, given that `passworf` is one of the typos present in the least-secure cache state.

In this style of attack, OMEN fared much better than in Section 6.2. Against the least-secure caches OMEN was able to crack an average of 45.67% of the passwords, and specifically in the case of a cache size of two, OMEN was able to crack all of the passwords present. In the medium-secure case, OMEN was able to crack on average 34.12% of the passwords. Sadly (in terms of security), OMEN was indeed able to crack passwords in the most-secure cache set, guessing on average 15.29% of the passwords in cache. All of these values do not consider the password that was leaked to OMEN in its training set.

While the results of OMEN in this sort of attack seem dismal (even one cracked password will allow admission to the system, if that is an attacker's goal), this sort of attack is niche, impractical, and does not represent the majority of guessing attacks against user accounts. For a cache entry to become "leaked," an attacker most likely will have compromised one of the user's passwords from a different service, and is using this compromised password to attempt to crack a cache entry used by TypTop. This comes back to research about password reuse [6], and the degraded security of using the same or similar passwords for different services. Markov-based password guessing systems are only as good as their training data, so as long as a password that has been (partially) reused is not compromised, this sort of attack is not representative of a legitimate threat to a random user account.

7 SECURITY IMPLICATIONS

We concur with the TypTop paper authors in that their system provides a more usable model of password authentication with minimal security degradation. While we may assume that an increase in password possibilities is synonymous with a decrease in system security, we have seen in our analysis that this is not true, at least in the use case of a true user. We contrast our findings with the findings of the usability survey, where many responses say that they would be uncomfortable with a password authentication system that allows typos. To them we offer the formal security proof presented in [2] as well as our real-life modeled security analysis. However, it is unlikely that users beyond those in the technology community will find use in this evidence. We thus propose a password authentication system equivalent to TypTop, but potentially with some rebranding or new marketing, one that pulls human-friendly data about security of a system from studies done about it.

We also examine the possibility of different forms of offline guessing attacks, beyond those that we have performed. The use of the OMEN system in the second case (a leaked cache entry) seems to be the most realistic choice, as it generates strings similar to those it has already seen, and strings in its training dictionary can be assigned various weights to increase the likelihood of generating strings similar to those with higher weights. This is our goal in the second case: we have a leaked typo and want to find other similar typos, with an overarching goal of discovering the true password, allowing us to always access the TypTop system as cache entries are constantly being updated. Additionally, the recovery of a user's true password may have further implications, allowing an attacker to falsely authenticate themselves in a different system where the user has reused their password, such as in [6].

It is also necessary to reflect on the conditions under which OMEN was used. As previously stated, OMEN generates 50,000,000 password guesses per execution, and is run to predict the probability of 4-grams. A real world attacker may run OMEN multiple times against an obtained cache state, testing different n-grams, different password outputs, and different training sets — all of which were held constant in our evaluation. Additionally, we include only the top ten most used special characters in both our example passwords as well as in OMEN's alphabet. Many systems encourage, if not require, users to use a variety of special characters, which extend beyond the top 10 we used in our evaluation [].

In the first case in which we are guessing cached typos blindly, OMEN may have been a good starting point, but we need to consider the use of other modern password guessing programs, such as the popular John the Ripper, hashcat, and others. These programs are robust, and contain various forms of password guessing attacks according to rules specified at run time. These programs also check various hashing algorithms, and are able to learn from past executions to further increase the likelihood of a cached and hashed password being guessed on subsequent executions. We conclude that TypTop is safe in the face of a blind offline guessing attack, with OMEN generating password guesses, but must highlight the fact that the use of other modern password guessing programs must be evaluated against TypTop, and leave this to further research.

All of the guessing attacks we performed were against a cache state containing plain text passwords. This simplifies our research, but is not realistic in an actual use case. TypTop's typo cache is encrypted according to the scheme described in Section 3.1, where each cache entry is encrypted based on the cipher text of the previous cache entry, and the very first entry is encrypted via public-key encryption, with the secret key itself being encrypted based on the user's password. In a real case where the cache state is obtained, cache entries must be decrypted one-by-one in the order that they appear in cache, since each encrypted value depends on the previous value. The additional computational power required may be marginal, as we assume that if an attacker is willing to perform an offline guessing attack they are willing to expend a significant amount of time and resources. However, this may be good for security in that our security degradation analysis is too liberal, and that true security degradation in the face of an offline guessing attack is less than we have found.

8 USABILITY CONCLUSION

All of the evidence we gathered during our testing and research has led us to conclude that the system does not lose enough security to warrant not using it on those grounds. However, the fact remains that it does objectively lower security by a nonzero amount, and a significant portion of users will not want to take this risk, no matter how insignificant. Our survey results proved this; 56% of participants indicated that they would not trust a system like TypTop, although they were likely not as familiar with the implementation of the system to the same level we are. What stood out to us most about the results, though, was not their negativity in regards to the system, which we had predicted. The most surprising aspect was the level of apathy present in the responses, as evidenced by the fact that only 21% of responses had a strong opinion about one or more of the questions. While the aforementioned lack of knowledge about typo tolerant password systems was almost certainly a contributing factor, many of the responses to our open-ended questions showed a willingness to at least try the system, but did not believe it to be useful enough to provide a definitive judgement either way. To quote one user: "if I misspelled my password that much, I'd just change it".

In a world where the prevalence of touchscreen and voice recognition technology has enabled various methods of biometric authentication methods to become more and more commonplace, the traditional password is becoming increasingly archaic, often relegated to the role of backup authentication method when your phone's fingerprint scanner is being weird. For situations in which biometric authentication hasn't become integrated, password manager systems let users take advantage of more secure passwords without needing to memorize them. It is within this space that typo tolerant password systems like TypTop operate– by giving users some leeway in their login attempts, the system acts almost as a halfway point between the complete memorization required of traditional passwords and the complete hands-off approach of password managers. But despite its novel approach, it seems that for most users, the gained functionality just isn't worth the loss of security, no matter how negligible.

REFERENCES

[1] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. 2016. pASSWORD tYPOS and How to Correct Them Securely. In *2016 IEEE Symposium on Security and Privacy (SP)*. 799–818. https://doi.org/10.1109/SP.2016.53

[2] Rahul Chatterjee, Joanne Woodage, Yuval Pnueli, Anusha Chowdhury, and Thomas Ristenpart. 2017. The TypTop System: Personalized Typo-Tolerant Password Checking. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 329–346. https://doi.org/10.1145/3133956.3134000

[3] Markus Duermuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Abdelberi Chaabane. 2015. OMEN: Faster Password Guessing Using an Ordered Markov Enumerator. https://doi.org/10.1007/978-3-319-15618-7_10

[4] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. 2019. PassGAN: A Deep Learning Approach for Password Guessing. arXiv:1709.00440 [cs.CR]

[5] Arvind Narayanan and Vitaly Shmatikov. 2005. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security* (Alexandria, VA, USA) *(CCS '05)*. Association for Computing Machinery, New York, NY, USA, 364–372. https://doi.org/10.1145/1102120.1102168

[6] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. 2017. Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 295–310. https://doi.org/10.1145/3133956.3133973

[7] Nihar M. Ranjan, Kaushal Mundada, Kunal Phaltane, and Saim Ahmad. 2016. A Survey on Techniques in NLP. *International Journal of Computer Applications* 134 (2016), 6–9.

[8] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. 2016. Targeted Online Password Guessing: An Underestimated Threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1242–1254. https://doi.org/10.1145/2976749.2978339

[9] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *2009 30th IEEE Symposium on Security and Privacy*. 391–405. https://doi.org/10.1109/SP.2009.8

[10] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 157–173. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler

[11] Zhijie Xie, Min Zhang, Anqi Yin, and Zhenhan Li. 2020. A New Targeted Password Guessing Model. In *Information Security and Privacy*, Joseph K. Liu and Hui Cui (Eds.). Springer International Publishing, Cham, 350–368.

[12] M. Yildirim and I. Mackie. 2019. Encouraging users to improve password security and memorability. *International Journal of Information Security* 18, 6 (01 Dec 2019), 741–759. https://doi.org/10.1007/s10207-019-00429-y

[13] Chunchun Zhao and Sartaj Sahni. 2019. String correction using the Damerau-Levenshtein distance. *BMC Bioinformatics* 20, 11 (06 Jun 2019), 277. https://doi.org/10.1186/s12859-019-2819-0