The Tangled Web of Password Reuse

Anupam Das*, Joseph Bonneau†, Matthew Caesar*, Nikita Borisov* and XiaoFeng Wang‡
*University of Illinois at Urbana-Champaign
{das17, caesar, nikita}@illinois.edu
†Princeton University
jbonneau@princeton.edu
‡Indiana University at Bloomington
xw7@indiana.edu

***Abstract*—Today's Internet services rely heavily on text-based passwords for user authentication. The pervasiveness of these services coupled with the difficulty of remembering large numbers of secure passwords tempts users to reuse passwords at multiple sites. In this paper, we investigate for the first time how an attacker can leverage a known password from one site to more easily guess that user's password at other sites. We study several hundred thousand leaked passwords from eleven web sites and conduct a user survey on password reuse; we estimate that 43-51% of users reuse the same password across multiple sites. We further identify a few simple tricks users often employ to transform a basic password between sites which can be used by an attacker to make password guessing vastly easier. We develop the first cross-site password-guessing algorithm, which is able to guess 30% of transformed passwords within 100 attempts compared to just 14% for a standard password-guessing algorithm without cross-site password knowledge.**

I. Introduction

Text passwords are the most common mechanism for authenticating human users of computing systems, particularly on the Internet. Password security has become a key research interest due to the pervasiveness of modern web services and their increasingly critical nature. Passwords form the foundation of security policy for a broad spectrum of online services, protecting users' financial transactions, health records and personal communications, as well as blocking intrusions into corporate, power grid, and military networks.

Security can be undermined if passwords are easy to guess and research has consistently shown that users tend to choose simple passwords that are easy to remember [20], [33]. To counter this, online services often make use of password composition policies (e.g., "the password must contain a mix of upper- and lower-case letters and at least one number"), or password meters to help users understand the strength of their passwords. Studies have shown that password composition policies along with password meters (or verbal notifications) do help users to choose stronger passwords [35], [44], [46]. However, they also increase user fatigue.

Unfortunately, the number of passwords a user must remember continues to increase, with typical Internet user estimated to have 25 distinct online accounts [10], [11], [32]. Because of this, users often *reuse* passwords across accounts on different online services. Password reuse introduces a security vulnerability as an attacker who is able to compromise one service can compromise other services protected by the same password, reducing overall security to that of the weakest site. Password reuse cannot be prevented by traditional composition policies or meters, as these tools only see passwords at a single site. Recent high-profile leaks of large numbers of passwords (including 55 000 accounts from Twitter [1], [12], [16], 450 000 accounts from Yahoo [17]–[19], and 6.5 million accounts from LinkedIn [2], [6], [8]) demonstrate that attackers can potentially gain huge lists of valid credentials for use in cross-site password attacks.

Beyond attacks exploiting exact password reuse, it is an open question if an attacker can use knowledge of a user's password at one site to more easily guess a *different* password chosen by the same user at another site. In this work, we study this question. We examine several leaked password data sets to measure password reuse across Internet sites and find that exact reuse of passwords is often mitigated by the fact that different sites have different complexity policies. However, we also find that users often use simple tricks to work around these different policies, for example making small edits to a common passphrase (e.g., adding a number *1* to the end of a password used at another site). We find that users typically use a very small set of simple rules to make these edits which can vastly improve an attacker's ability to guess passwords at other sites. For example, we were able to guess 30% of non-identical leaked password pairs within 100 attempts (10% password pairs required less than 10 attempts) while existing password cracking libraries (like John the Ripper [4]) were able to crack 14% of the passwords.

In this work, we make the following key contributions:

NDSS '14, 23-26 February 2014, San Diego, CA, USA

http://dx.doi.org/doi-info-to-be-provided-later

- An empirical estimate of the rate of direct password reuse for accounts controlled by the same user at different websites, 43%, based on the largest data set yet collected for this purpose.
- Extensive analysis of the similarity of non-identical passwords from the same users across different online accounts. This sheds light on how users modify their password for reuse across different sites, as well as the specific transformation patterns users use when faced with various password composition policies.
- We conduct a survey to understand users' behavior in password construction across different online accounts. This survey sheds light on how people use transformation rules to modify their existing passwords for different online accounts.
- A cross-site guessing algorithm which uses a leaked password at one site to produce guesses for passwords potentially used by the same user at other sites. We evaluate the feasibility of our guessing algorithm under both offline and online attack scenarios.

The rest of the paper is organized as follows: Section II gives an overview of some password composition policies and related work. Section III introduces our measurement studies and results. We summarize our survey results in Section IV. We describe the basic design principle of our guessing algorithm in Section V. Finally we discuss some implications of our research in Section VI and conclude in Section VII.

II. Background and Related Work

To better understand how passwords are composed we first look at some of the most frequently used password composition policies. We will then summarize recent academic literature in the field of password analysis.

A. Password Composition Policies

The hardest passwords to crack are random character strings. However, such passwords are considered too difficult to remember in many applications. Given the choice, human users typically produce a highly-skewed distribution of password choices reflecting common semantics that ease recall (e.g., incorporating a word or number sequence with special meaning). To prevent users from selecting passwords that are too simple (and therefore potentially common), designers of password authentication mechanisms often impose a *password composition policy*, comprising a set of rules that constrain the length and formation of passwords. Here is an example policy:

I. Passwords must not contain the user's entire name/user ID.

II. At least n characters (usually $n \geq 6$).

III. Passwords must contain characters from two or more of the following four categories:

1. Uppercase characters (A through Z)
2. Lowercase characters (a through z)
3. Base 10 digits (0 through 9)
4. Non-alphanumeric ASCII characters: ˜!@%ˆ&*_-+=—(){}[]:;”‘’<>,.?$\

More advanced policies can be used (e.g., NIST guidelines [25]) which may further increase guessing difficulty. However, password policies usually instill a tradeoff between usability and security. Complex policies lead to *password fatigue*, complicating usability of the service and potentially decreasing security by encouraging users to write down or electronically store lists of passwords. Because of this, the majority of online services today use relatively simple password composition policies, leaving password choice up to users.

To help users understand the security of their passwords a *password meter* can be used. However, password meters have been shown experimentally [46] and in practice [21] to make a relatively modest impact on password choices. Furthermore, designers of password meters must make assumptions about the way attackers will guess passwords (e.g., brute force). An attacker with better knowledge of how the user constructed the password may be able to easily guess a password designated as strong by a password meter.

B. Related Work

There are alternatives to passwords, such as biometrics, public-key certificates, and hardware tokens. However, authentication using text-based passwords remains the de facto standard for authentication in today's Internet. Passwords have several advantages over alternative schemes, including the lack of need for custom hardware, ease of deployment and incorporation with existing services, their simplicity, and lack of need to memorize or store complex cryptographic keys [22].

A large body of research has been conducted in understanding the security of user-created passwords. For example, there has been studies that look into understanding users' reaction to visual and textual based feedback provided by password meters [46]. Others have looked at the strength of different password composition policies [21], [35], [41], [44]. These studies have shown that users do tend to create stronger passwords in the presence of password composition policies accompanied with password meters. Interestingly, Bonneau et al. [24] showed that most websites do have some password policy, but provide no strength meter.

There have also been studies that analyze password attack strategies beyond brute force and dictionary attacks. The most well-known password guessing algorithms are based on Markov models and probabilistic context free grammars. Markov chain-based modeling of substrings with parametric lengths was first proposed by Narayanan and Shmatikov [39]. In their approach they use a training set to obtain the probabilities of generating candidate substrings with certain length. Their new dictionary, which comprises a list of all possible candidate passwords is then checked across a test set in

decreasing probability. Weir et al. [48] construct a probabilistic context free grammar (PCFG) for generating new mangling rules to increase the set of trial passwords to guess. Their PCFG is generated from a training set and then evaluated against a distinct test set. In both works all training and testing sets are obtained from publicly leaked passwords. There have also been studies that have done extensive comparisons among these guessing algorithms [29], [34]. All of these works focus on cracking passwords in an offline scenario where the attacker is not bounded by the number of guesses he/she can make. However, online services often rate-limit password attempts. Hence in contrast to these prior works, we evaluate an attacker with a limited number of guesses but leveraging leaked password information and knowledge of how users modify passwords across sites.

Florêncio et al. [32] monitored password habits of half a million users over a three months period. Their study revealed that the average user has 6.5 passwords, each of which is shared across 3.9 different sites. However, their study only considered identical passwords between sites, and not related ones. Thus, while we can consider their estimates of exact password reuse to be the best available, they did not study transformation rules or similarity of passwords between different sites, which is the main focus of our paper.

The work of Zhang et al. [49] is perhaps the most closely related to ours. Their paper looked at how users modified their passwords when they were forced to change them due to a password expiration policy. They created a generic algorithm that could guess future passwords based on a user's previous passwords. However, their analysis is based on passwords from a single source (staff accounts at the University of North Carolina Chapel Hill) and thus all passwords fulfill the same password composition policy. In our case, we model a more general attacker who is able to leverage the leaked password of a given user to guess that user's password on other Internet sites. Because of this, we not only have less training data per user but also have to deal with different composition policies for different web sites.

There has also been research on defending against cross-site password attacks by deploying password management tools like PwdHash [43]. However, even if PwdHash were universally deployed, password reuse would still be a problem. An attacker could still take a list of hashed passwords from one site and brute-force them (with the extra PwdHash round of hashing), then attempt to reuse recovered credentials at other sites (applying the PwdHash algorithm specific to the other site). Furthermore, Chiasson et al. [26] analyze the usability of different password management tools and conclude that password managers are not sufficiently usable for most web users. This has been borne out in practice as update remains low, even though PwdHash users are currently well-protected because there are so few that attackers will ignore them. Password reuse remains a real problem that is worth quantifying because most improvements have proved far too expensive to deploy [31].

TABLE I. SOURCES OF DATA

Site	#	Year	Hashed?
csdn.net	6428630	2011	no
gawker.com	748559	2010	yes
voices.yahoo.com	442837	2012	no
militarysingles.com	163482	2012	yes
rootkit.com	81450	2011	yes
myspace.com	49711	2006	no
porn.com	25934	2011	no
hotmail.com	8504	2009	no
facebook.com	8183	2011	no
youporn.com	5388	2012	no

III. MEASUREMENT STUDY

We would first like to understand how often users reuse passwords across sites and the specific approaches they use to vary their password at different sites (e.g., to work around different composition policies). To explore these questions, we conduct a measurement study.

A. Collection of Data Set

Our goal is to correlate user passwords across Internet sites. To do this, we collected publicly available leaked passwords over a period of several years from pastebin.com[1] and other online sources. The provenance of these data sources varies but most are the result of either SQL injection attacks or phishing campaigns.[2] It would be unethical to verify the validity of the data directly by attempting to use the credentials, and in some cases many of the accounts have long since expired or changed passwords. However, all of the data sources we used were publicly announced and none had the veracity questioned by the affected web site.

For our analysis we were only able to use leaked data sets with both user identifiers (in the form of email addresses) and passwords. As a result some prominent password leaks from sites like RockYou, Twitter, or LinkedIn could not be used. For a summary of our data sets, see Table I.

We then analyzed these data sets to find passwords belonging to users for whom we have at least one other password, matching up users by their hashed email addresses. We ended up with 6077 unique users for which we had a minimum of two leaked passwords from different websites. Table II shows the number of passwords per user. As we can see from Table II, for most users we had only 2 leaked passwords (the maximum we observed was 6). Some of the passwords were in plaintext while others were hashed (unsalted) with MD5. To better facilitate string comparison we inverted as many of the hashes as possible using the *John the Ripper* toolkit [4]. For the remaining hashes we searched one of the largest online database of 8.7 billion MD5 hash preimages [7] and we were able to decrypt all but 12 of them, which we discarded (less than 0.2% of our data).

[1] Some of these leaked passwords are no longer available online.

[2] All of the data sets were leaked as MD5 hashes or as plain texts, and in no case are we confining our analysis to "only what an attacker could crack". We confirmed this by examining the complexity of the leaked passwords.

TABLE II. PASSWORDS PER USER

No. of passwords	Percentage
2	97.75%
3	1.82%
4	0.26%
5	0.15%
6	0.02%

B. Password Composition Policies at Different Sites

To put our results in context, we manually investigated the password composition policies of the websites from which user passwords were leaked. All the sites[3] (except for the pornographic web sites and hotmail) had a password composition policy requiring use of at least six characters. Pornographic web sites either had no restriction or a restriction of using at minimum four characters in creating a password. Hotmail had the most restrictive policy where a password had to have at least eight characters. To put our results in a larger context, we describe here password composition policies across a broader spectrum of popular Internet sites obtained from Alexa [3] under different categories. In particular, we randomly selected a set of popular sites from the top 25 sites under several different categories (Table III) and manually determined their composition policies. We found the following constraints across different web sites (we assign each a short code name to ease reference to them later in the paper):

- **Char5**: Minimum 5 characters.[4]
- **Char6**: Minimum 6 characters.
- **Char8**: Minimum 8 characters.
- **Char6LU**: Minimum 6 characters containing at least one lowercase letter and one uppercase letter.
- **Char8DSU**: Minimum 8 characters containing at least one number, symbol, or uppercase letter.
- **Char8LDS**: Minimum 8 characters, with at least one letter (either uppercase and/or lowercase) and at least one number and/or symbol.
- **Char6D**: Minimum 6 characters containing at least one letter and one number.
- **Char6-12**: Contains 6-12 characters. Characters can be letters, digits or even symbols.
- **Strong1**: Contains 7-32 characters with at least one letter and one number. Cannot include special characters (&, %, *, etc.). Cannot be the same as user ID and cannot be the same as any of the last five passwords used.
- **Strong2**: Contains 8-20 characters with at least one letter and one number. Cannot include any spaces or the following special characters $, <, >, &, ^, !, [,]. Cannot be the same as user ID. Password is case-sensitive.
- **Strong3**: Contains 8-20 characters with at least one letter and one number. May include the following characters: %,&,_, ?, #, =, -. Cannot have any spaces and will not be case sensitive. Must be different from user ID.

[3]We couldn't check *www.rootkit.com* because it is currently down

[4]Characters can be letters, digits or symbols.

TABLE III. PASSWORD COMPLEXITY POLICIES FOR POPULAR WEB SITES

Category	Website	Constraints
Social	Facebook	Char6
	Twitter	Char6
	LinkedIn	Char6
	Google+	Char8
	Pinterest	Char6
Blogging	Blogger	Char8
	Tumblr	Char5
	WordPress	Char6
Email	Gmail	Char8
	Yahoo	Char6
	Hotmail	Char8
	Outlook	Char8
	AIM/AOL	Char6
Shopping	Amazon	Char6
	Ebay	Char6LU
	Target	Char8DSU
	BestBuy	Char6D
	Walmart	Char6-12
Financial	Chase	Strong1
	Bank of America	Strong2
	American Express	Strong3
	Paypal	Char8LDS

Table III summarizes the different password composition policies for different popular web sites. As we can see, most of the non-financial web sites have similar password composition policies, which could possibly encourage users to use identical or similar passwords across different accounts.

C. Password Similarity

Next, we analyze our password data set to measure the similarity between different passwords used by a given user. For this purpose we looked at 9 string similarity metrics:

- **Distance-like functions**: *Manhattan* [36], *Cosine* [27]. These functions compute the distance between two strings by first mapping each string into a point in a multidimensional space (each character can be thought of as a dimension with their frequency being the dimension value) and then computing the distance between those points.
- **Edit-distance like functions**: *Levenshtein* [38], *Damerau-Levenshtein* [28]. These functions determine the number of edit operations (insertion, deletion, replacement, transposition) required to transform one string into another.
- **Token-based distance functions**: *Dice* [30], *Overlap* [37]. These functions first split the strings in

smaller tokens (i.e., bigrams) and then compute the similarity between them.

- **Alignment-like functions**: *Smith-Waterman* [45], *Neddleman Wunsch* [40], *Largest Common Subsequence (LCS)*. This set of functions provide similarity scores that reflect the largest alignment or subsequence between a pair of strings.

For the following set of similarity experiments we randomly sample 1000 password pairs to ensure the total sample size in each the experiment is equal. We first investigated the similarity score among passwords from different users for the same website. We took 1000 password pairs randomly where each password belonged to different users and then computed the similarity among them. Figure 1(a) shows the CDF of the similarity score under different metrics. We can see that the CDF quickly rises for *smaller similarity scores* which signifies that users within the same website use significantly different passwords by almost any measure.

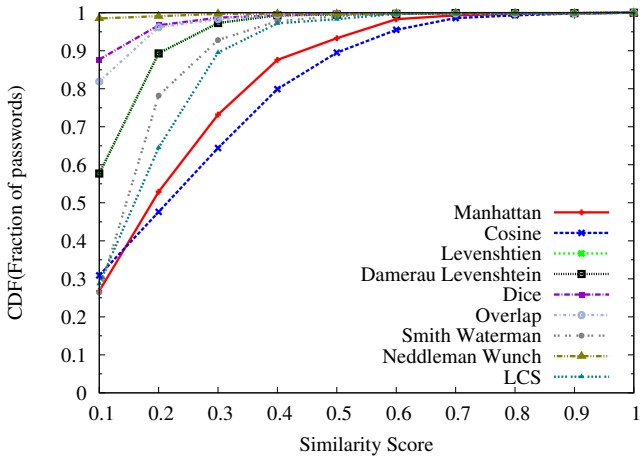
Next we randomly sampled 1000 password pairs from randomly selected users at (Figure 1(b)). Again we see that the CDF quickly climbs towards its maximum value for smaller similarity scores. Thus we conclude that passwords selected by different people are quite different.

Next we wanted to see how similar passwords chosen by a given user across different web sites are. For this purpose we calculated the similarity score of passwords used by the same user across different sites. We again sampled 1000 random password pairs. Figure 2(a) shows the CDF of the similarity scores. In this case we see that almost 40% of the passwords have similarity score in the range of $[0.9, 1.0]$. Next we wanted to investigate the similarity score of passwords that were not identical, so we randomly sampled 1000 users whose passwords across websites were not identical. Figure 2(b) shows the similarity of nonidentical passwords. Again we see that almost 30% of the nonidentical passwords have similarity scores in the range of $[0.8, 1.0]$, thus indicating that there is non-trivial similarity among the passwords that users use across different web sites.

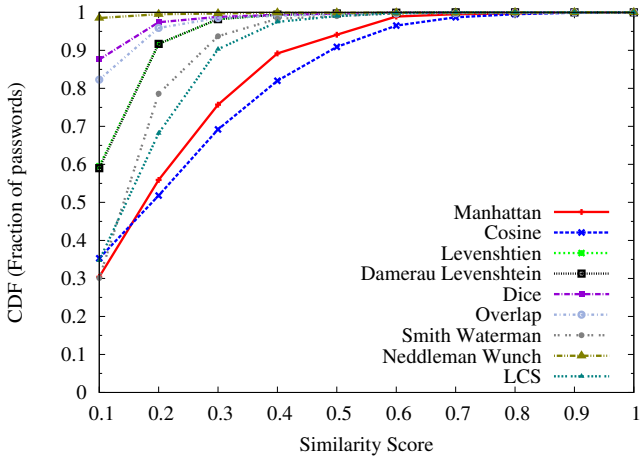
Upon looking more closely into the leaked passwords, we found that a significant fraction of the users used passwords that were substrings of one another. To investigate this further, we clustered the passwords into the following three groups:

1. Identical: Same password across different web accounts.
2. Substrings: Passwords from different web account are substrings of one another.
3. Other: Passwords are either completely different or follow more complex transformations.

This grouping also helps us in identifying the probable transformation rules (something we investigate in more detail later). The fraction of passwords belonging to each group is provided in Table IV.



(a)



(b)

Fig. 1. Password similarity for (a) passwords chosen by random users of the same web site (b) passwords chosen by random users of different sites.

TABLE IV. PASSWORD RELATIONSHIPS

Type	Percentage
Identical	43%
Substring	19%
Others	38%

From Table IV we see that 19% of passwords are substrings of one another, or one-third of all non-identical passwords. In these cases, simple insertion or deletion operations are sufficient to transform one password into another. We also identified what fraction of these insertion/deletion operations were taking place at the beginning or end or at both ends of a string. Table V summarizes these fractions. As we can see, the most frequent insertion/deletion operation(s) occur at the end of the string. Next, we determined the average length of insertions/deletions at these locations. Table VI shows that the overall average length of each insertion/deletion is about 2-3 characters. We also looked at what unigram, bigram and trigram character sequences occur more frequently than others. Appendix B shows the top 20 such n-grams used for our insertion operations.

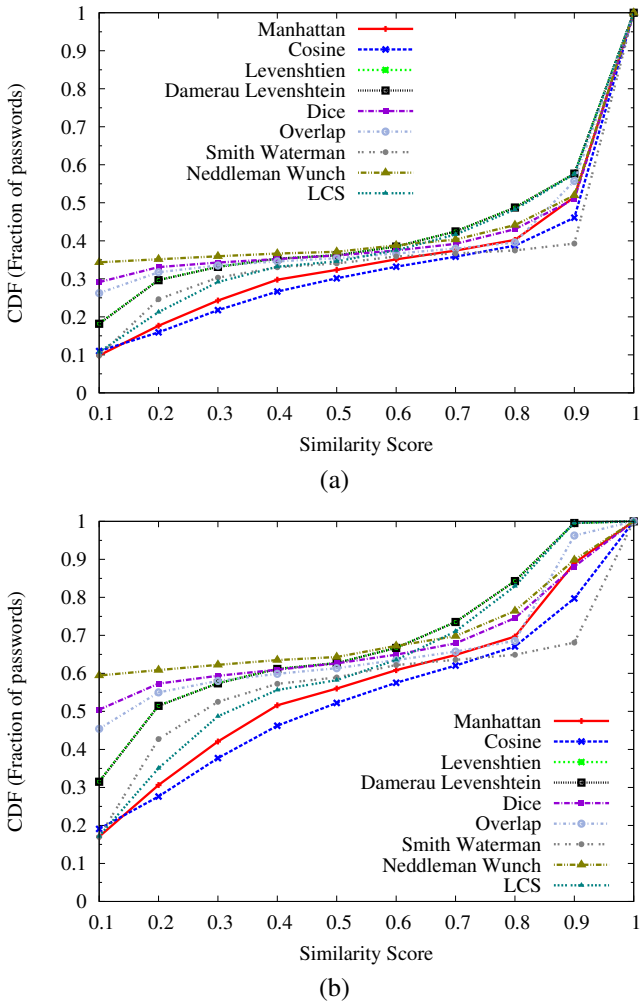


Fig. 2. Password similarity for (a) all pairs of passwords chosen by the same user (b) non-identical pairs of passwords chosen by the same user. We find that users inherently tend to reuse/slightly modify their passwords.

TABLE V. LOCATION OF INSERTION/DELETION

Location	Percentage
Start	10%
End	88%
Both Ends	2%

IV. SURVEY

To gain insight into users' behavior and thought processes when creating passwords for different websites, we conducted an anonymous survey of users at different universities, including students and professional staff across several academic departments. It is worth noting that through university initiatives, these participants have been exposed to fairly substantial documentation and training regarding the sensitivity of password information and the importance of choosing secure passwords. We received a total of 224 responses. A copy of our survey questions are available at [15]. In the survey participants were asked several questions about how they constructed their passwords for different online accounts. The goal of the survey was to better understand why and to what extent passwords used by these users differed across different online accounts. The results of the survey and our findings are given below.

TABLE VI. AVERAGE LENGTH OF INSERTION/DELETION

Location	Average	Standard Deviation
Start	3.37	2.01
End	2.25	1.42
Both Ends	3.77	1.83

A. Survey Responses

We started our survey by giving the participants a collection of scenarios in which they would create a new email account and asked them to think about the password that they would use for that account. Next, we asked them a series of questions regarding their choice of password. The first question determined how they chose their passwords and how similar the resulting password selection was to any of their existing passwords. The response to this question is summarized in Figure 3. We can see that 77% of participants would either modify or reuse existing passwords. We also found from Table IV that 43% of password pairs were identical, which is comparable to our survey result of 51%.

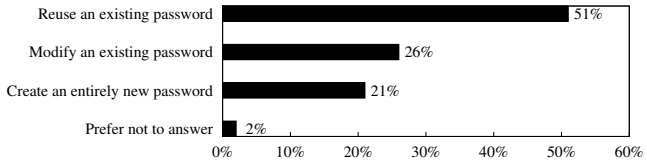


Fig. 3. Survey responses to the question: "How do you choose your password for a new email account?" We find that the majority of participants either modified or reused an existing password.

Next, we investigated what forms of transformation rules people use when they modify existing passwords for the purposes of reuse. From Figure 4(a), we can see that certain transformation rules are substantially more common than others. Rules like *adding a number at the end or front*, *capitalization of letters* and *adding a symbol at the end or front* were the most popular choices. These results match what we observed in Section III-C, where 98% of insertions/deletions occurred at either the front or end of the password. To investigate this further, we investigated why people were modifying or reusing existing passwords. Figure 4(b) summarizes our findings. We find that users modify their passwords to fulfill password constraints enforced by the different websites. When users come across a website with different or more restrictive constraints, they tend to introduce small modifications to existing passwords to work around those constraints.

We then wanted to determine which locations in passwords were subject to the majority of modifications. Hence, we asked participants where they would place symbols, digits, and uppercase letters when they modify an existing password for reuse (Figure 5). Users tended to prefer placing symbols at either the middle or the end. However, we see a more significant bias for placement of digits and letters. Most

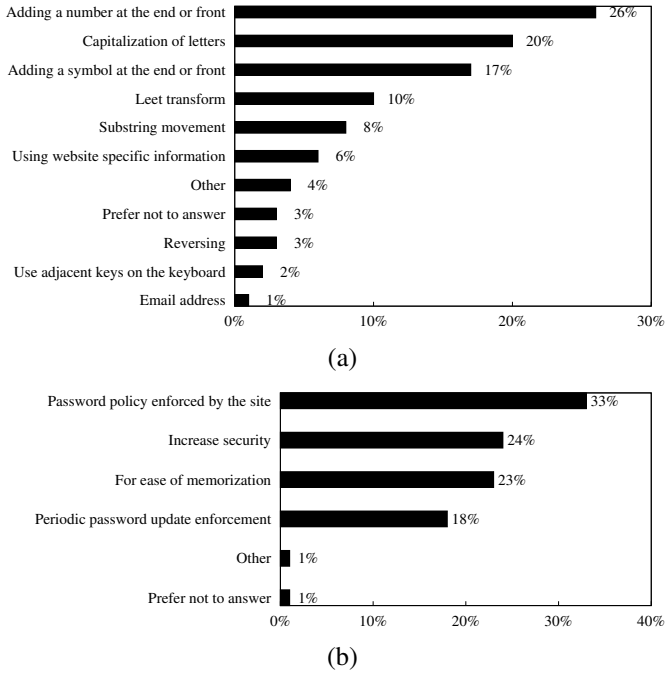


Fig. 4. Survey responses to the questions: (a) What transformation rules do you use? (b) Why do you modify existing passwords? We find that adding digits or symbols either at the front or end is a popular transformation rule used by users. Furthermore, we find that people tend to modify existing passwords to fulfill password constraints enforced by the different websites.

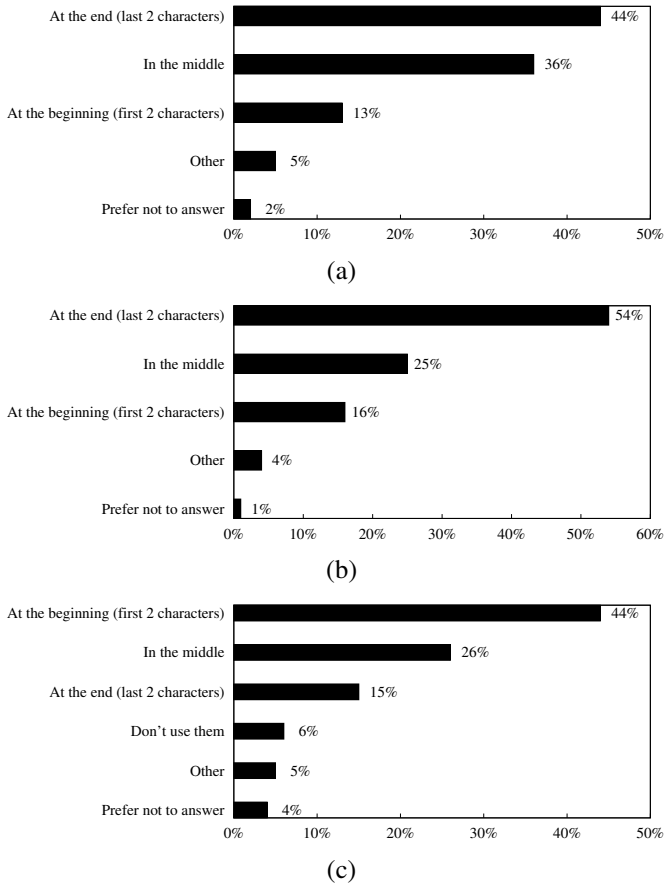


Fig. 5. Survey responses to questions: Where do you usually place (a) symbols? (b) digits? (c) uppercase letters? It seems the majority of the participants preferred placing symbols either at the end or middle of the password string. For digits they preferred placing them at the end and for uppercase letters they preferred placing them at the beginning.

participants insert digits at the end of password strings, but tend to insert uppercase letters at the beginning. These results match our observations in leaked data sets (see Table V). We also asked a question where we wanted to know how the digits used in passwords were related to the user, interestingly we did not find any dominant relationship (see Appendix C).

Next, we asked a set of questions to find how many different types of passwords users typically use, as well as how they go about remembering their passwords. From table III we saw that financial websites imposed stronger constraints in constructing passwords. Assuming people generally tend to value financial accounts more than other types of accounts, it is possible that people may maintain different types of passwords for different types of online account. Figure 6 summarizes the responses from our participants. We find that 65% of the participants maintained no more than four different types of passwords. A majority (61%) of the participants memorized their passwords.

We collected some limited demographic information about our participants. The overwhelming majority (93%) of our participants are aged 18–34. We also found that most of the participants (92%) have at least a bachelor's degree. This is expected as we sent out our survey to different universities. We conjecture that our survey population of young, highly educated users is more concerned about password security than the general population, suggesting we may be underestimating the problem.

V. Guessing Algorithm

The results of our measurements and survey lead naturally to the question: can an attacker, equipped with knowledge of how users typically modify passwords across sites, more easily guess user passwords? In this section, we will leverage results from our studies to construct a cross-site password guessing algorithm. Our guessing algorithm works as follows: given one leaked password, it tries to guess passwords that the same user might use for their other online accounts.[5] First, we will discuss the different transformation rules that are commonly used by users in modifying their passwords (Section V-A). We then describe the details of our guessing algorithm (Section V-B).

A. Prominent Transformation Rules

We start by investigating the most prominent transformation rules that we identified from our measurement studies (we looked at 40% of the leaked passwords for transformation rules

[5]In practice matching accounts is usually trivial as the vast majority of sites ($> 90\%$) use email address as user IDs [24] so we do not consider this problem further.

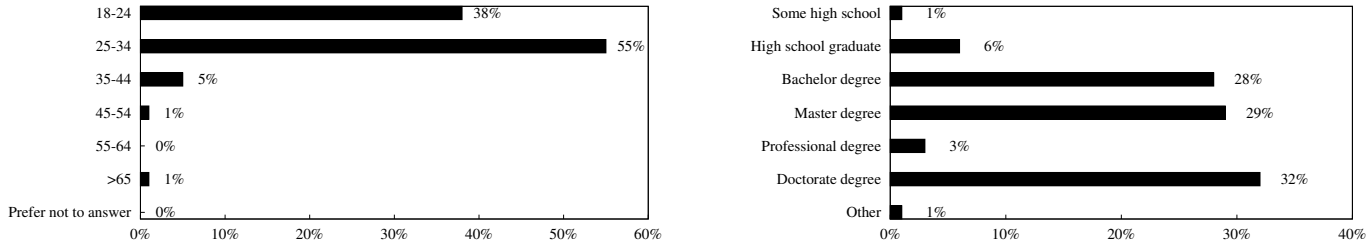


Fig. 7. Demographic information about our 224 survey participants age (a) and educational background (c).

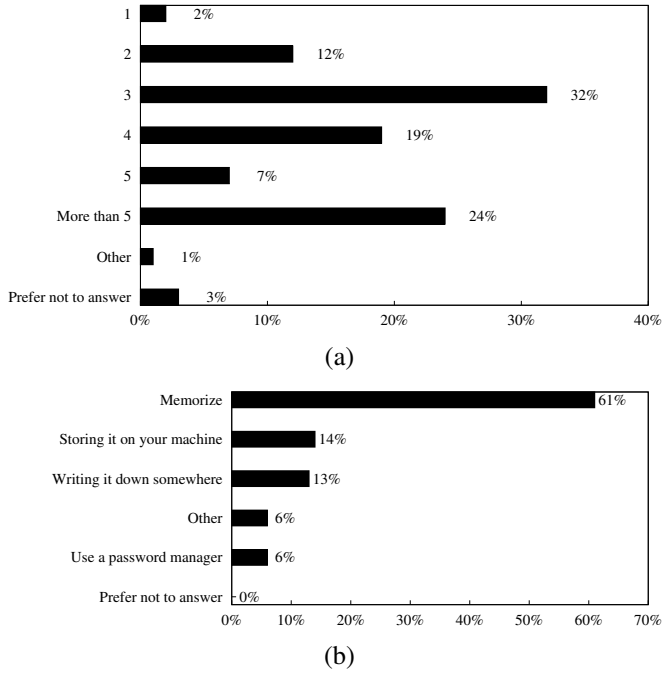


Fig. 6. Survey responses to questions: (a) How many different passwords do you maintain? (b) How do you remember your passwords? We find that the majority of participants maintain less than five different types of passwords and most people memorize their passwords.

and tested our guessing algorithm on the remaining 60% data). Table VII shows the most commonly used transformation rules from our leaked password set with examples (we also saw similar feedback from our survey) in decreasing order of probability. We also list less common transformations in Table VIII (also in decreasing order of probability). In addition, we found several interesting transformation rules used by our survey participants, such as adding a few random extra characters (e.g. `abcd` ⟶ `abyzcd`) or adding emoticons at the end (e.g `abd`⟶`abd:))`. It may be possible to improve our results by incorporating these transformations into our guessing algorithm, but to preserve simplicity of our algorithm we do not consider these additional transformations in this paper. We will show that simple transformations are sufficient to crack a moderate fraction of the unknown passwords.

TABLE VII. PROMINENT TRANSFORMATION RULES

Transformation Rule	Example
sequential key	`qwerasdf` ⟶ `1234qwer`
sequential alternative key	`123456` ⟶ `!@#$%^`
sequential alphabet	`abcde` ⟶ `uvxyz`
capitalization	`naughty` ⟶ `NAUGHTY`
reverse	`123456` ⟶ `654321`
leet	`password` ⟶ `pa$$w0rd`
substring movement	`gzwz0204` ⟶ `0204gzwz`

TABLE VIII. LESS PROMINENT TRANSFORMATION RULES

Transformation rule	Example
character swap	`1abcd2` ⟶`2abcd1`
subword capitalization	`redcode` ⟶ `RedCode`
subword movement	`redcode` ⟶ `codered`
word repetition	`chat`⟶ `chatchat`
email address	`xiaona` ⟶ `xiaona@gmail`
website	`forget@csdn` ⟶ `forget@rootkit`

B. *Designing Our Guessing Algorithm*

Our goal is to design a simple guessing algorithm that, given a user's password for a particular site, can determine the user's password for other sites with a low enough number of guesses to make online attacks feasible. In particular, given an *input password* from one site, our algorithm generates an ordered set of *candidate passwords*, with the goal of the *target password* in use at another site being ordered early in the list (to minimize the number of guesses required).

Our guessing algorithm is shown in Algorithm 1. Its operation consists of several phases executed in the order given below. After each phase, we check if we have guessed the desired target password. We revert back to the original password after each step.

Character sequence: We attempt to look for known pattern sequences, such as adjacent keys on the keyboard, alphabetical ordering patterns, and sequential alternate key (e.g., holding down "alternate keys" such as Shift or Control while typing a sequence) inside the input password. Once we find these patterns, we apply the corresponding transformations sequentially. The intuition here is that users who use such sequential patterns are likely to use similar patterns in their passwords for other sites.

In more detail, we first split the password into tokens,

Algorithm 1 Guess new passwords from leaked passwords

Input: Input password α and target password β
Intermediate result: Candidate password α'
Output: Cracked or Not cracked
Check (α', β): if $\alpha' = \beta$ **return** Cracked
if α contains any sequential pattern **then**
  Sequential Transformation$(\alpha) \rightarrow \alpha'$
  Check (α', β)
end if
if len(α)>6 **then**
  $\alpha' \leftarrow$ Deletion(α)
  Check (α', β)
end if
if len(α)<10 **then**
  $\alpha' \leftarrow$ Insertion(α)
  Check (α', β)
end if
Capitalization$(\alpha) \rightarrow \alpha'$
Check (α', β)
Reverse$(\alpha) \rightarrow \alpha'$
Check (α', β)
Leet$(\alpha) \rightarrow \alpha'$
Check (α', β)
Substring Movement$(\alpha) \rightarrow \alpha'$
Check (α', β)
Subword transformation$(\alpha) \rightarrow \alpha'$
Check (α', β)
return Not Cracked

where each token represents a sequence of characters that fall under some pattern (e.g., 'qwerasdf' would be split into two parts 'qwer' and 'asdf', as each part represents an independent set of sequential characters from the keyboard). We then try out different permutations of tokens as candidate passwords. If we do not achieve the target password, we then modify each token by either extending the token to include the next character(s) in the sequence, or by replacing the token with a similar size token belonging to the same category. For example, 'qwer' could be extended to 'qwert', or could be replaced by '1234'. To increase the likelihood of choosing the right replacement, we analyze the RockYou password list [13], one of the most well-known and commonly analyzed password leaks with about 32 million leaked passwords, to find the most common sequences (we list the most common replacements used in our guessing algorithm in Appendix D). If this does not succeed, finally, we try all possible permutations of the modified tokens. We also use *reverse* transformation (described later on) as an additional transformation to each candidate password to take into account reverse sequential characters.

Deletions: Next our guesser tries deletion transformations. In the deletion transformation, we first try iteratively deleting characters belonging to the following set {*Digit, Symbol, Uppercase letter, Lowercase letter*}. The intuition behind this is that people usually add characters belonging to these different categories to fulfill different password composition policy. So we initially try deleting digits, followed by symbols, and so on. After each character deletion we check if we have acquired the target password. If the previous transformation does not produce the desired target password we revert back to the original password and try sequentially deleting characters from the front of the string, then the back, then combinations of both. As a heuristic, we only applied deletion transformations if the the password length is greater than six (in section III-B we saw that most websites have a password policy of containing at least six characters).

Insertions: Our guesser next tries insertion transformations. From our measurement studies (on the leaked data and from our survey) we saw that inserting numbers or symbols at the front or end was the most popular choice. We first attempt inserting numbers and symbols at the front and end. We limit ourselves to up to two insertions (insertions of length three would mean an exponential increase in the number of transformations). Next we concentrate on different password composition policies requiring passwords to contain characters from the following four groups: {*Digit, Symbol, Uppercase letter, Lowercase letter*}. We first determine what types of characters the input password contains, then we insert individual characters from the missing groups. The probability of inserting characters from these groups are calculated from the RockYou list. Finally to address longer insertions we compute the top 20 probable bigrams and trigrams (we consider only the top 20 to keep the number of guesses minimal; distribution of the bigrams and trigrams are shown in Appendix B) from the RockYou list and apply them to our input password to generate candidate passwords. As a heuristic, we only apply insertion transformations if the length of the resulting password remains less than ten characters. We choose 10 as the upper limit of a password length because 90% of passwords in RockYou data set had a length of less than or equal to 10 [14] (see Appendix A for more information).

Capitalizations: The capitalization transformation first attempts to capitalize all letters in the password at once. If the target password is not obtained, it then tries capitalizing the letters from the front of the string, then the back, and then combinations of both.

Reversals: The reverse transformation simply reverses the input password.

Leet-speak: For the *leet* transformation we first try the popular leet transforms ($o \leftrightarrow 0, a \leftrightarrow @, s \leftrightarrow \$, i \leftrightarrow 1, e \leftrightarrow 3, t \leftrightarrow 7$) and then try all possible leet transforms [5].

Substring movement: This transformation first splits the input password into substrings where the delimiting character belongs to the set {*Digit, Symbol, Uppercase letter*}. For example 'xyz@123' would be split into three substrings: 'xyz', '@', and '123'; using '@' as the delimiting character. Next, we sequentially try all possible permutations of the substrings. So '123@xyz' would produced as a candidate password.

Subword modification: Here, we first split the input password based on common English words, and then capitalize the first

letter of each word. We also try rearranging the words in different orders. For example 'darkknight' can to split into 'dark' and 'knight', so a possible candidate password would be 'DarkKnight'.

We apply transformation rules in the above-mentioned order to generate candidate passwords. We found this ordering to be most efficient after sampling 1000 random nonidentical password pairs (i.e., we analyzed 1000 random nonidentical password pairs and found the above ordering to generate the least number of guesses compared to other orderings). We use the following minimization formula: $\mathcal{O} = \arg\min_{i \in |\mathcal{T}|!} Guesses(\mathcal{O}_i)$, where $\mathcal{T}$ represents the set of transformations, $\mathcal{O}$ represents an ordering of the transformations and $Guesses$ returns the number of guesses needed using a particular ordering. Recall that we revert back to the main password before applying the next transformation. Hence the success rate does not change with the order, but the number of guesses required may vary based on the order selected. One point to note is that our guessing algorithm essentially applies a fixed order of transformations to the input password as opposed trying all possible transformations. This bounds the number of possible guesses that we can make.

C. Evaluating Our Guessing Algorithm

In this section we evaluate the performance of our guessing algorithm in terms of the number of guesses required to crack the target password. As a baseline, we compare our approach with the following three plausible approaches:

- **RockYou guesser**: This guesser tries to guess passwords based on the 32 million leaked passwords from the RockYou social application site. Under this strategy we first build a list of words along with possible prefix-suffix insertions. To determine the possible prefix-suffix insertions we adopt two strategies. For a given leaked password, we first look for a word inside the leaked password that appears in an English dictionary [9] (we use Moby Words II, one of the largest wordlists in existence) and determine the possible prefix-suffix insertions. If we do not find any matching English words, we then look for other leaked RockYou passwords that have the leaked password as a substring, to determine the set of candidate prefix-suffix insertions. We do this for all the passwords in the RockYou data set. Once complete, we have a mapping of possible prefix-suffix insertions for a large set of words (either English words or other leaked passwords).
- **Edit Distance (ED) guesser**: This guesser tries to make edit transformations such as insertions, deletions, replacements or substring movements, to turn a leaked password into a probable candidate password. We used the ED based guesser implemented by Zhang et al. [49]. However, we did not train our model as they did. Instead we inserted characters in decreasing order of probability calculated from the RockYou list so our results are not directly comparable to theirs. The ED guesser applies edit transformations at specific locations. Therefore, the total number of possible transformations depends on the length of the password. We apply edit distance of up to depth two, as beyond this depth the number of transformations becomes infeasibly large [49].
- **John the Ripper (JTR)**: This guesser uses the John the Ripper toolkit [4] in wordlist mode with word-mangling rules enabled. JTR has 57 word-mangling rules in its configuration file, along with a default password list (*password.lst*). We append one of the passwords from each pair from our data set into this *password.lst* file. Next, we run JTR on the MD5 hashes of the remaining password of each pair (i.e., we first generated MD5 hashes of the remaining passwords and then ran JTR on them). We also compared the performance of JTR without cross-site password knowledge and it was able to crack 14% of the non-identical passwords using the default 57 word-mangling rules.

TABLE IX. GUESSING SUCCESS

Category from Table IV	Approaches			
	RockYou List(%)	ED Guesser(%)	Our(%)	JTR(%)
Substring	4	66	85	15
Others	0	6	4	15

From Table IV we know that approximately 43% of passwords in our data set (6077 unique users) were identical. Hence, we concentrate on evaluating our guessing algorithm only on the nonidentical passwords (the remaining 57% of our data set). Table IX highlights the fraction of different non-identical passwords correctly guessed by the three approaches. We see that the RockYou-list based guesser does not provide substantial benefit in guessing passwords across websites (we therefore ignore this approach in later comparisons). John the Ripper can guess approximately 15% for each category of passwords. Both ED-guesser and our approach successfully guessed a significant portion of the password pairs that were substrings of one another. From Table IX it is evident that password pairs that are neither identical nor substrings are difficult to guess. For such password pairs JTR [4] performs better as it has a large number of mangling rules to apply. Leveraging such mangling rules could potentially improve our results further, but it would also increase the number of guesses significantly (JTR generated 159 907 and 162 742 candidates for the 'substring' and 'other' categories respectively). We do not focus on using complex mangling rules because we want to show that simple transformations can successfully guess a significant fraction of the passwords with a small number of attempts.

We start by analyzing the number of guesses required by each of the approaches. Figure 8 shows the cumulative distribution of the number of guesses required for nonidentical

passwords. We see that our approach can guess approximately 30% of the nonidentical password pairs successfully (10% within 10 attempts). Compared to the ED-guesser our approach requires significantly fewer guesses, thus making our approach suitable for online attacks where the number of guesses is limited to a fixed number. Most sites (>80%) do not effectively rate-limit incorrect guesses at all [24], and those that do often use complex algorithms to determine when to lock an account. Both Facebook and Google will allow more than 10 guesses in some circumstances. It is an arms race for attackers to avoid getting accounts locked out, and success with fewer guesses is better, but there is rarely a firm cutoff number.

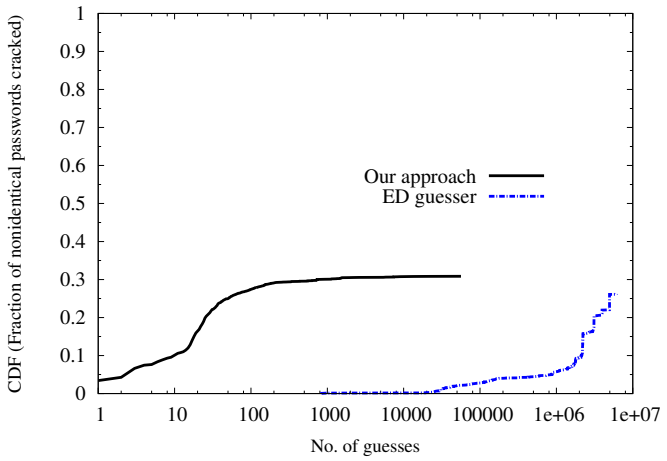


Fig. 8. Number of guesses needed to guess all nonidentical passwords. We see that we need approximately 10 attempts to crack 10% of the nonidentical password pairs.

Now let us look more closely into the number of guesses required for each category of password other than the identical category. Figure 9 shows the cumulative distribution of the number of guesses required for each category of nonidentical password. We can see that if the number of retries is limited to 10 then we can crack almost 30% of the password pairs that are substrings of each other. We can crack 80% of the password pairs that are substrings of each other within 100 attempts. ED-guesser on the other hand requires several million attempts to crack at most 65% of the substring password pairs. We can also see that even though the ED-guesser cracks more passwords in the "other" category the number of guesses required is significantly higher compared to our approach. Because of this, our approach is more suitable for online attack scenarios, where rate limiting of password checking is common. If we revisit figure 2(b) we see that approximately 30% of the nonidentical password pairs have a similarity score in the range of $[0.8, 1]$. We believe our guessing algorithm was successful in predicting those similar passwords. To verify this we computed the similarity score of all the password pairs that our guessing algothim successfully cracked. Figure 10 shows the similarity score of the cracked password pairs. We can see that majority of the similarity scores lie in the range of $[0.8, 1]$. We also looked into the similarity score of the password pairs that we could not crack. Figure 11 highlights the similarity score of the password pairs that we could not crack. As expected the similarity score for non-cracked password pairs are low; most of them have a similarity score in the range of $[0.1, 0.5]$. Thus we can say that our guessing algorithm can successfully predict similar passwords and has quickly diminishing returns afterwards.This is not a limitation, as a real-attacker could try an approach like ours using knowledge of a leaked password and then transition to a generic attack to generate a large number of other guesses.

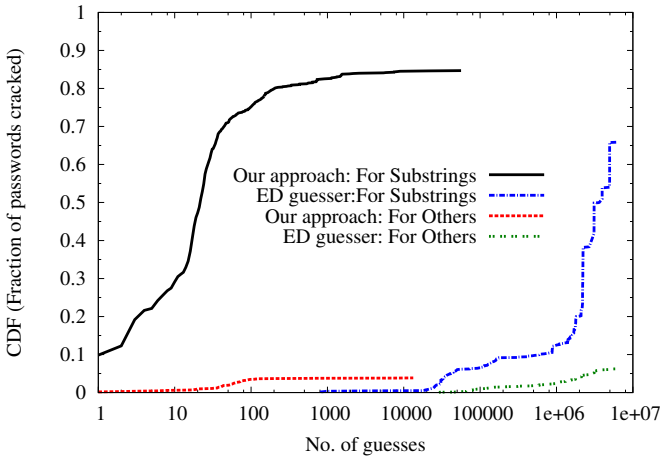


Fig. 9. Number of guesses needed to guess passwords, broken down by substring pairs and non-substring pairs. Our approach can guess 30% of the substring pairs within 10 attempts.

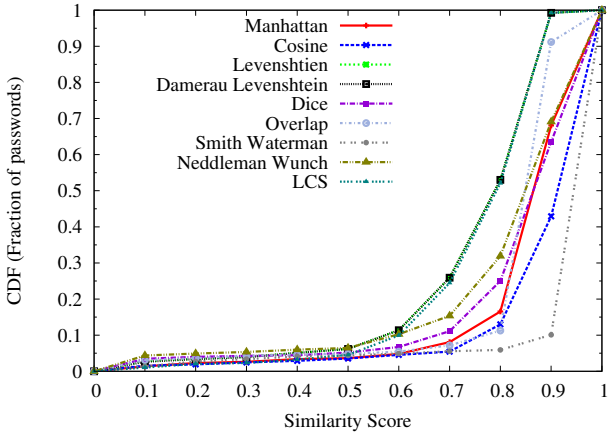


Fig. 10. Similarity score of cracked password pairs. Most of the similarity scores lie within the range of $[0.8, 1]$.

VI. Discussion

In this section we discuss issues regarding ethics, ecological validity, and the limitations of our methodology.

A. Ethical Considerations

Our results rely on passwords that are publicly available, even though they were originally gathered illegally. The question of whether data acquired illegally should used by

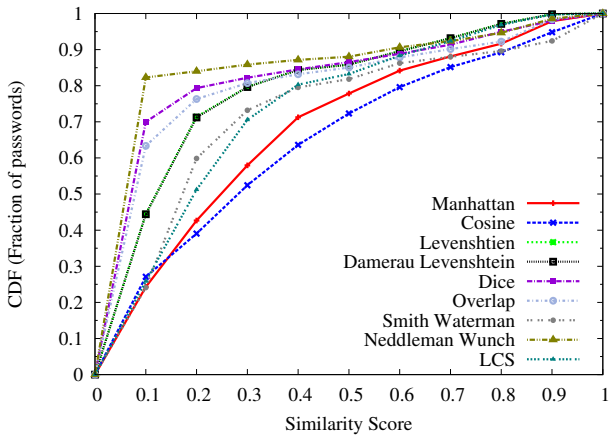


Fig. 11. Similarity score of non-cracked password pairs. Most of the similarity scores lie within the range of $[0.1, 0.5]$.

researchers is an active ethical debate (more on the ethics of using public data of illicit origin for research is discussed in [31]). That said, many of our data sets have been used in prior scientific studies [23], [29], [34], [47], [48]. When conducting our work, we use passwords and hashed email addresses. We obtained approval from our institution's IRB for our survey related to password construction behavior. The survey responses were recorded anonymously without any personally identifying information on participants. We hope that our work brings benefits to researchers and system administrators by improving the understanding of cross-site password attacks, which may lead to stronger measures to mitigate them.

B. Ecological Validity

To attain real-world measures of password-related behavior, we surveyed participants anonymously. Most of our survey participants are somewhat younger and more educated than the general population, but more diverse than typical small-sample password studies. Moreover, since users were submitting responses anonymously, we believe it would be non-beneficial for them to provide false information. In addition, all our password data sets represent real password data. We compared results of our survey with our leaked password data sets, and found similar observations, for example pertaining to the location of insertion/deletion operations, and the fraction of users using identical passwords across different websites. Thus, we believe our findings are likely to hold true in practice, at least for some classes of passwords and users.

C. Limitations

There are a number of limitations of our work. For example, our study only pertains to text-based passwords, as opposed to other authentication schemes like biometrics or graphical passwords. That said, we feel the extreme pervasiveness of text based passwords makes them the most useful mechanism to study. In future work it may be interesting to see if our approach could be generalized to non-text user-constructed passwords such as graphical passwords. In addition, our guessing algorithm is relatively simple, not leveraging more advanced models which may possibly improve performance such as Markov Chain models [41] and other algorithmic frameworks [49]. This was an intentional choice as more complicated approaches are highly dependent on training data, which we had a limited supply of. Our main goal was to show that even using a fairly simple set of rules can significantly improve attack performance.

Considering the category of a website is another feature we wanted to study. But 97.75% of the users in our data set had only two passwords (see Table II) and in most of the cases the two passwords did not belong to websites of similar category. So we did not have sufficient data to analyze if similar website category influenced the choice of password reuse. However, from our survey we observed that 65% of the participants maintained no more than four different types of passwords across different categories of websites (see section IV-A).

D. Countermeasures

Developing countermeasures to improve cross-site password security is challenging. Direct coordination across different sites is hard, as user account information is often viewed as private, which may preclude direct sharing or comparison of entered passwords. Single sign-on technologies may help, but face challenges in deployment due to competing provider interests and the difficulty of forming strong shared security policies across administrative boundaries. Single sign-on technologies may also worsen security: if the password is compromised (e.g., via a phishing attack), security of all participating sites is compromised. Two-factor authentication is another viable option. In this authentication process a user first provides his/her username and password; next the user is asked to enter a verification code that is sent directly to the user's phone. Google and Twitter, among others have started using such authentication scheme.

A first step to address this problem may be educating users of the importance of using substantially different passwords across sites. Our password guessing algorithm could be a useful tool for users, as a user could input their passwords at different sites and locally evaluate the difficulty of cross-site guessing. A more ambitious approach would be to develop a cross-site password security metric, deployed at online services, based on our design. A password construction page could compare the user's entered password against that stored at other participating sites to see if it can be easily guessed using our techniques. To avoid sharing user data, sites could employ privacy-preserving variants of our string comparison operations. Several privacy-preserving operations for string comparison (e.g., edit distance [42]) exist and could be leveraged for use here.

VII. Conclusion

To the best of our knowledge, our work comprises the first analytical study of cross-site password security. To study

this problem, we first conduct an analysis of real-world user behavior, including a measurement study of real leaked passwords combined with a user survey. We leverage the results of this study to evaluate strategies that attackers can use to infer passwords used at other sites, and develop a cross-site password guessing algorithm. We observe that a substantial portion, 43%, of users directly re-use passwords between sites, confirming suspicion that this is a significant security vulnerability. We further demonstrate that many users introduce small modifications to their passwords across sites, and many users share the same procedures for introducing these modifications. However, these modifications are simple enough that an attacker who is aware of typical user behavior can significantly improve guessing efficiency. Our protoype guessing algorithm is able to crack approximately 10% of the nonidentical password pairs in less than 10 attempts and approximately 30% such pairs in less than 100 attempts. This makes a real security impact as an attacker with a leaked, non-identical password can mount an online guessing attack with orders of magnitude higher success than an attacker without a leaked password.

ACKNOWLEDGEMENTS

We thank our shepherd, Nicolas Christin, and the anonymous reviewers for their invaluable comments. We are grateful to Yinqian Zhang who helped with the design of Edit Distance (ED) based guesser. We give special thanks to all the participants who took the time to answer our survey. This material is based upon work supported in part by the National Science Foundation under Grant Nos. CNS 0953655 and 1223477, as well as an International Fulbright S&T Fellowship.

REFERENCES

[1] 55K Twitter Passwords Leaked. http://www.newser.com/story/145750/55k-twitter-passwords-leaked.html.

[2] 6.46 million LinkedIn passwords leaked online. http://www.zdnet.com/blog/btl/6-46-million-linkedin-passwords-leaked-online/79290.

[3] Alexa. http://www.alexa.com.

[4] John the ripper password cracking toolkit. http://www.openwall.com/john/.

[5] Leet Transformations. http://qntm.org/l33t.

[6] LinkedIn confirms password leak, encourages users to update passwords. http://www.cbsnews.com/8301-501465_162-57448443-501465/linkedin-confirms-password-leak-encourages-users-to-update-passwords/.

[7] MD5 Decryptor. http://www.md5decrypter.co.uk/.

[8] Millions of LinkedIn passwords reportedly leaked online. http://news.cnet.com/8301-1009_3-57448079-83/millions-of-linkedin-passwords-reportedly-leaked-online/.

[9] Moby Words II. http://icon.shef.ac.uk/Moby/.

[10] No wonder hackers have it easy: Most of us now have 26 different online accounts - but only five passwords. http://www.dailymail.co.uk/sciencetech/article-2174274/No-wonder-hackers-easy-Most-26-different-online-accounts--passwords.html.

[11] Online Passwords: Keep it Complicated. http://www.guardian.co.uk/technology/2012/oct/05/online-security-passwords-tricks-hacking.

[12] Over 55,000 Twitter passwords exposed, posted online. http://news.yahoo.com/blogs/technology-blog/over-55-000-twitter-passwords-exposed-posted-online-165625320.html.

[13] RockYou leak. http://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/.

[14] RockYou Password Analysis. http://thepasswordproject.com/rockyou_passpal_0.4_dump.

[15] Survey Questions. http://web.engr.illinois.edu/~das17/passwordreuse.html.

[16] Thousands of Twitter passwords exposed. http://news.cnet.com/8301-1009_3-57430475-83/thousands-of-twitter-passwords-exposed/.

[17] Yahoo hacked, 450,000 passwords posted online. http://www.cnn.com/2012/07/12/tech/web/yahoo-users-hacked.

[18] Yahoo Voice hack leaks 450,000 passwords. http://www.guardian.co.uk/technology/2012/jul/12/yahoo-voice-hack-attack-passwords-stolen.

[19] Yahoo's password leak. http://news.cnet.com/8301-1009_3-57471178-83/yahoos-password-leak-what-you-need-to-know-faq/.

[20] A. Adams and M. A. Sasse, "Users Are Not The Enemy," *Commun. ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[21] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, ser. SP '12, May 2012.

[22] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, ser. SP '12, 2012, pp. 553–567.

[23] J. Bonneau, M. Just, and G. Matthews, "What's in a Name? Evaluating Statistical Attacks on Personal Knowledge Questions," in *Proceedings of the 14th International conference on Financial Cryptography and Data Security*, ser. FC'10, 2010, pp. 98–113.

[24] J. Bonneau and S. Preibusch, "The password thicket: technical and market failures in human authentication on the web," in *Proceedings of the 9th Workshop on the Economics of Information Security*, June 2010. [Online]. Available: http://www.jbonneau.com/doc/BP10-WEIS-password_thicket.pdf

[25] W. E. Burr, D. F. Dodson, and W. T. Polk. (2006) Electronic Authentication Guideline. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

[26] S. Chiasson, P. C. van Oorschot, and R. Biddle, "A Usability Study and Critique of Two Password Managers," in *Proceedings of the 15th conference on USENIX Security Symposium*, ser. USENIX-SS'06, 2006.

[27] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks," in *Proceedings of IJCAI-03 Workshop on Information Integration*, 2003, pp. 73–78.

[28] F. J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Commun. ACM*, vol. 7, no. 3, pp. 171–176, Mar. 1964.

[29] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password Strength: An Empirical Analysis," in *Proceedings of the 29th conference on*

Information Communications, ser. INFOCOM'10, 2010, pp. 983–991.

[30] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, vol. 26, no. 3, pp. 297–302, Jul. 1945.

[31] S. Egelman, J. Bonneau, S. Chiasson, D. Dittrich, and S. Schechter, "It's Not Stealing If You Need It: A Panel on the Ethics of Performing Research Using Public Data of Illicit origin," in *Proceedings of the 3rd Workshop on Ethics in Computer Security Research*, ser. WECSR '12, 2012, pp. 124–132.

[32] D. Florêncio and C. Herley, "A Large-Scale Study of Web Password Habits," in *Proceedings of the 16th International Conference on the World Wide Web*, ser. WWW '07, 2007, pp. 657–666.

[33] S. Gaw and E. W. Felten, "Password Management Strategies for Online Accounts," in *Proceedings of the 2nd Symposium on Usable Privacy and Security*, ser. SOUPS '06, 2006, pp. 44–55.

[34] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, ser. SP '12, 2012, pp. 523–537.

[35] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of Passwords and People: Measuring the Effect of Password-Composition Policies," in *Proceedings of the International Conference on Human Factors in Computing Systems*, ser. CHI '11, 2011, pp. 2595–2604.

[36] E. F. Krause, *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Dover Publications, 1987.

[37] M. Levandowsky and D. Winter, "Distance between Sets," *Nature*, vol. 234, no. 5, pp. 34–34, 1971.

[38] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[39] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. CCS '05, 2005, pp. 364–372.

[40] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[41] D. Perito, C. Castelluccia, and M. Duermuth, "Adaptive Password-Strength Meters from Markov Models," in *Proceedings of the 19th Network and Distributed Systems Security Symposium*, ser. NDSS '12, 2012.

[42] S. Rane and W. Sun, "Privacy Preserving String Comparisons Based on Levenshtein Distance," in *Proceedings of IEEE International Workshop on Information Forensics and Security*, ser. WIFS '10, 2010.

[43] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger Password Authentication Using Browser Extensions," in *Proceedings of the 14th conference on USENIX Security Symposium*, ser. USENIX-SS '05, 2005.

[44] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering Stronger Password Requirements: User Attitudes and Behaviors," in *Proceedings of the 6th Symposium on Usable Privacy and Security*, ser. SOUPS '10, 2010, pp. 1–20.

[45] T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[46] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "How does your password measure up? The effect of strength meters on password creation," in *Proceedings of the 21st USENIX conference on Security symposium*, ser. USENIX-SS '12. USENIX Association, 2012, pp. 65–80.

[47] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords," in *Proceedings of the 17th ACM conference on Computer and Communications Security*, ser. CCS '10, 2010, pp. 162–175.

[48] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars," in *Proceedings of the 30th IEEE Symposium on Security and Privacy*, ser. SP '09, 2009, pp. 391–405.

[49] Y. Zhang, F. Monrose, and M. K. Reiter, "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis," in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10, 2010, pp. 176–186.

APPENDIX A
AVERAGE PASSWORD LENGTH

We analyze the average password length of the passwords in the RockYou data set. Figure 12 shows the CDF of the password length. As we can see from the figure 90% of the passwords have a length of less than or equal to 10 characters and 5% of the passwords have a length less than or equal to 5 characters. We therefore considered insertion and deletion operation only if the given password's length is in the range of $[6, 10]$ respectively.

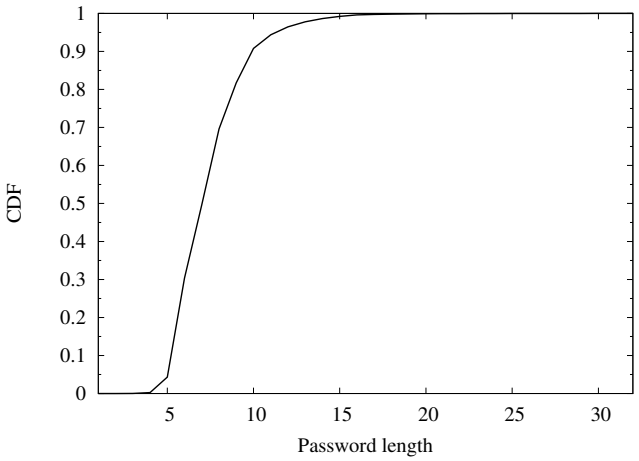


Fig. 12. CDF of password length

APPENDIX B
ANALYSIS OF ROCKYOU LIST

First, we look at the distribution of unigram insertions. This is different from the character frequency count because we are looking at insertions of such characters at the beginning or end of existing words. Figure 13 shows the distribution.

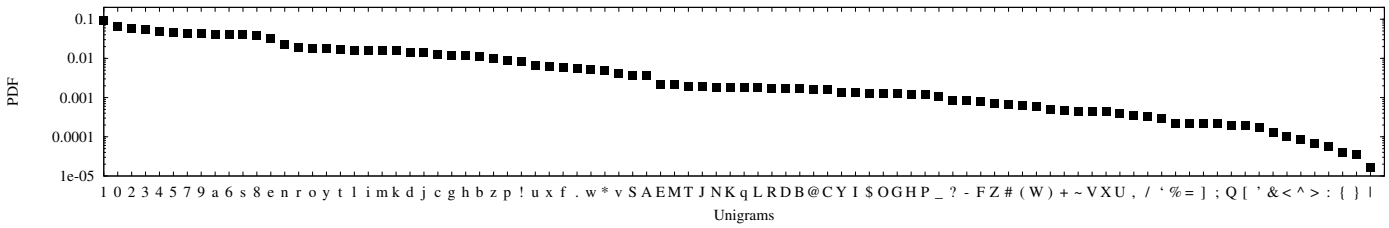


Fig. 13. PDF of unigram insertions

Next we calculate the insertion probabilities of some of the top bigrams and trigrams. Table X shows the top 20 insertion probabilities. As we can see from the table inserting number is a popular trend among people. It would be interesting to know how people chose these numbers. To figure this out we asked a related question in our survey. The details of our findings are available at Appendix C.

TABLE X. INSERTION PROBABILITY OF BIGRAMS AND TRIGRAMS

bigram	Pr.	trigram	Pr.
08	0.027	123	0.015
01	0.016	087	0.0047
07	0.015	007	0.0047
23	0.014	083	0.0040
06	0.013	084	0.0040
09	0.013	089	0.0037
12	0.012	086	0.0036
05	0.011	666	0.0036
21	0.0097	085	0.0034
04	0.0096	man	0.0030
11	0.0093	143	0.0028
22	0.0096	boy	0.0026
02	0.0092	321	0.0022
13	0.0090	101	0.0022
03	0.0090	420	0.0021
69	0.0087	456	0.0020
00	0.0083	000	0.0019
10	0.0081	001	0.0019
88	0.0076	777	0.0018
20	0.0074	ita	0.0018

APPENDIX C
SIGNIFICANCE OF NUMBERS IN PASSWORDS

We asked our participants the following question - *If you use digits in your password then how are the digits in your password related to you?*. Figure 14 summarizes the responses that we got from our survey. As we can see from figure 14, 43% of the people used numbers that are significant to them. However, a significant fraction (42%) of the people chose the **'Other'** option. We looked through their written responses and found that most of them just used random numbers that they could remember.

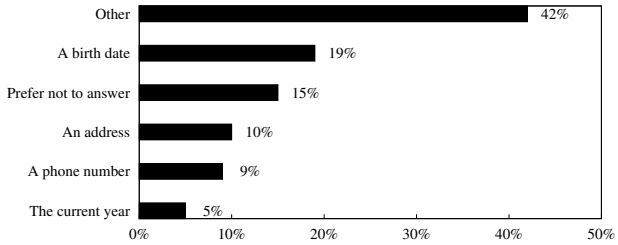


Fig. 14. Survey responses to the question: If you use digits in your password then how are the digits in your password related to you?

APPENDIX D
SEQUENTIAL SUBSTRING REPLACEMENT

We found the following sequential substring replacements most popular - $qwer \rightarrow [1234, 1qaz]$, $qwe \rightarrow [qaz, qwer]$, $asd \rightarrow [asdf, wsx]$, $wsx \rightarrow [2wsx, wer]$, $asdf \rightarrow [1234, zxcv]$, $5678 \rightarrow [qwer, 1234]$, $qa \rightarrow [qwe, qaz]$.