User password repetitive patterns analysis and visualization

Xiaoying Yu and Qi Liao
Department of Computer Science, Central Michigan University, Mount Pleasant, Michigan, USA

Received 24 June 2015
Revised 24 June 2015
Accepted 7 August 2015

Abstract

Purpose – Passwords have been designed to protect individual privacy and security and widely used in almost every area of our life. The strength of passwords is therefore critical to the security of our systems. However, due to the explosion of user accounts and increasing complexity of password rules, users are struggling to find ways to make up sufficiently secure yet easy-to-remember passwords. This paper aims to investigate whether there are repetitive patterns when users choose passwords and how such behaviors may affect us to rethink password security policy.

Design/methodology/approach – The authors develop a model to formalize the password repetitive problem and design efficient algorithms to analyze the repeat patterns. To help security practitioners to analyze patterns, the authors design and implement a lightweight, Web-based visualization tool for interactive exploration of password data.

Findings – Through case studies on a real-world leaked password data set, the authors demonstrate how the tool can be used to identify various interesting patterns, e.g. shorter substrings of the same type used to make up longer strings, which are then repeated to make up the final passwords, suggesting that the length requirement of password policy does not necessarily increase security.

Originality/value – The contributions of this study are two-fold. First, the authors formalize the problem of password repetitive patterns by considering both short and long substrings and in both directions, which have not yet been considered in past. Efficient algorithms are developed and implemented that can analyze various repeat patterns quickly even in large data set. Second, the authors design and implement four novel visualization views that are particularly useful for exploration of password repeat patterns, i.e. the character frequency charts view, the short repeat heatmap view, the long repeat parallel coordinates view and the repeat word cloud view.

Keywords Computer security, Information visualization, Password analysis

Paper type Research paper

1. Introduction

We have long been using passwords, from online banking and ATMs to office computers, mobile devices, cloud data storage, etc. We rely on passwords to protect our information security and privacy. While the history has shown the weakness of password security (Morris and Thompson, 1979) in the past decades, and there have been proposals to use graphical passwords (Davis *et al.*, 2004), text passwords are still the dominant knowledge-based method of authentication nowadays, and possibly in the near future.

With the ever expansion of data, applications and services, it is not unusual for modern users to have dozens (or even hundreds) of accounts for their online shopping, financial needs, work-related tasks, etc. To make it even more challenging, each site and

Information & Computer Security
Vol. 24 No. 1, 2016
pp. 93-115

2056-4961
DOI 10.1108/ICS-06-2015-0026

organization has different password requirements. It has been a trend that those password rules become more complex and confusing. As a result, people have been struggling to find ways to use easy-to-remember passwords or choose some memorable patterns to help them make longer passwords that satisfy the password requirements. During a recent National Science Foundation (NSF) talk (Chisnell, 2014), most of the audience's concern was about the ability to remember multiple long passwords in multiple applications.

To make up long and sufficiently secure passwords that pass the length requirements, users have adopted various strategies, such as using digits that combine phone numbers and dates (birthdays, special dates, etc.) (Bonneau *et al.*, 2012; Veras *et al.*, 2012). Other users are keen on names of themselves, special places or special persons. Some use the keyboard combinations (Schweitzer *et al.*, 2009), which may look like random characters, but the sequence of characters actually follow a specific pattern (e.g. consecutive keys on the keyboard).

In this paper, we analyze the hypothesis that users may repeat words as part of their overall strategies to make up longer passwords. For example, users may repeatedly use "hihihi […]", "121212 […]" or "babybaby […]" in their passwords. The contributions of our study are two-fold. First, we formalize the password repetitive patterns by considering both short (1-2 characters) and long (3+ characters) substrings that are repeated in consecutive locations and in both unidirectional and bidirectional orders. Efficient algorithms are developed and implemented that can analyze various combinations of repeat patterns quickly even in very big password data. Second, we design and implement four visualization views for exploration of such password repeat patterns. The character frequency charts (CFC) provide a quick view of the character frequency distribution from a password file and can be ordered by American Standard Code for Information Interchange (ASCII) values or appearance times. The short repeat heatmap (SRH) is detailed representation of repeat frequencies of exactly one or two characters in a two-dimensional matrix. The long repeat parallel coordinates (LRPC) view is to display repeating substrings equal to or longer than three characters. Bidirectional repetitive patterns may be visualized from both ends. Finally, the repeat word cloud (RWC) is an alternative view for visualizing top consecutive repeating words of various length using different colors and sizes. The lightweight, Web-based visualization tool (Figure 1) is based on D3 technologies (Bostock *et al.*, 2011). The interactive nature of visualization reduces the analytic complexity of big data that is inherited from traditional statistical or data mining approaches.

Through case studies on real-world leaked password [RockYou, Weir *et al.*, 2010] data set, which contains over 14 million passwords, we demonstrate how researchers use the tool to find various repeat patterns in user passwords. First, we compare the password character distribution with modern English letter appearance and Benford's digits law. We find there is discrepancy among them. Second, we analyze both short and long repeat patterns and find repetition is indeed a common practice in user passwords. Even worse, the repetitive strings are almost exclusively chosen from the same group type (i.e. digits, lowercase or uppercase letters). Users seldom switch cases or switch between letters and digits. Third, often users use shorter repeating substrings to make up longer repeating strings which make up the final passwords. Finally, the reverse order repetitions are surprisingly more than the forward-order repetitions.

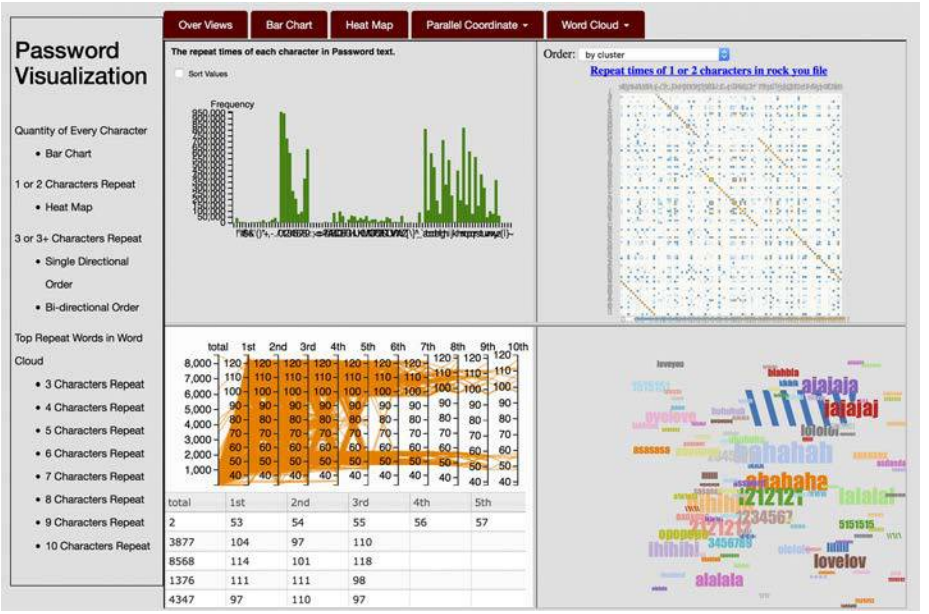


Figure 1.
Overview of the visualization tool for password repetitive patterns

Although security practitioners have been adapting security policy by making passwords longer and more heterogeneous with a combination of letters, digits and special characters, users' repetition habits bypass and weaken such defense strategy. Through the useful visualization tool, system administrators and managers may effectively explore, analyze and understand users' repetitive patterns when making up user passwords, and the findings have potential impact on improving the security of many systems that still involve passwords.

The remainder of this paper is organized as follows. Section 2 discusses the related work of password security and analytic methods. In Section 3, we define various repetitive patterns for short and long passwords and directions of such repetitions. More importantly, we present the algorithms for analyzing such patterns. Following that, in Section 4, we discuss the design and implementation of views to visualize such repeat patterns in user passwords. In Section 5, we present several case studies to show how the visualization tool may be used to find various user password repetitive patterns. Finally, we conclude our work in Section 6.

2. Related work

Since the expansion of the Internet and increased online activities, users have been challenged by both the explosion of numbers of passwords and the length requirement of passwords. A recent study by Microsoft Research (Florencio and Herley, 2007) shows that the average Web user has about 25 accounts that require passwords and has 6.5 passwords, each of which contains mostly lower case letters with an average bitstrength of 40.54 bits. In addition, passwords are re-used and forgotten very often, confirming the convention wisdom. An online experiment conducted by Carnegie Mellon University (Shay *et al.*, 2014) evaluates the password

policy for a security/usability tradeoff. The study reveals that adding requirements to policies on longer passwords can reduce the number of easily guessed passwords, and that certain combinations of requirements can increase both security and usability than the traditional complex policy.

Traditionally, metrics such as Shannon entropy and guessing entropy have been used to analyze the security of passwords. For example, entropy, as defined in NIST SP800-63 and as a measurement of the security provided by various password creation policies, has been studied for its effectiveness by modeling success rate of password cracking over a real-word password data set (Weir *et al.*, 2010). As measuring the strength of passwords is crucial for a secure system, adaptive password strength meter (Perito *et al.*, 2012) uses Markov model for better accuracy.

Through analyzing 70 million passwords of Yahoo! users (Bonneau, 2012), metrics are formalized for evaluating the guessing difficulty of skewed distribution of passwords. In addition, an empirical analysis of real-world passwords (Dell'Amico *et al.*, 2010) is performed and revealed that no single attack strategy prevails over the others, such as dictionary attacks, dictionary mangling and Markov chain techniques. All attack techniques are affected by diminishing returns: the probability that a guess will succeed decreases by orders of magnitude. An approximation algorithm (Zhang *et al.*, 2010) is developed to search out new passwords from old ones which is non-deterministic polynomial-time (NP)-hard. The linkage between the new and old passwords leads to the authors' question on the merit of continuing the password expiration practice. As the real-world passwords have been leaked out in the past few years, researchers have tried to find out the different patterns from the user passwords. Among the patterns, semantic patterns of passwords have been analyzed. One of these patterns is dates, such as birthdays, holidays or personal special dates. Other recent findings indicate that dates are common among digit sequences (Bonneau *et al.*, 2012). Bonneau *et al.* (2012) explain the patterns of four-digit PINs used in different fields and how the customers select their PINs and what kind of sequence they usually use (e.g. MMDD). Dates in passwords have been visualized (Veras *et al.*, 2012).

The keyboard patterns in user passwords have been analyzed (Chou *et al.*, 2012) and visualized (Schweitzer *et al.*, 2009). Even though some passwords seem strong and random, the passwords may be attacked by analyzing the orders of the characters on keyboard. For example, keyboard sequence of "7ygvbnmju" may be random and strong at the first look, but the characters follow a Christmas-tree-like shape on the physical keyboard.

As a complement to the above studies, in this paper, we study a different type of pattern, i.e. the repetitive patterns in user passwords. In particular, we study whether repetition is also often used by users to set a long password. In addition, whether such repetition, if existing, occurs either recursively (i.e. repetition in repetition) or reversely (i.e. in both directions).

3. Repetitive patterns in passwords

In this section, we begin by first discussing various repetitive patterns that we are interested to find in user passwords. We define the term "repeat pattern" (or used interchangeably "repetitive pattern") in the paper as any substrings that repeat in the same password. The "repeat times" are considered as accumulation of each instance of repeat counts for all passwords. The repeat patterns can be categorized into single

character, twin characters and three or more characters. In addition, there can also be directions of repeat (i.e. left-to-right or right-to-left).

3.1 Single character repeat

It is always important for system administrators to get an overview of the character distribution of user passwords. Therefore, as a starting point, the visualization tool needs to include a password distribution view: CFC (Section 4.1) to display how many times the 95 printable ASCII characters (values 32-126) are used to make up the actual passwords. After computing the overall single character distribution, we then count the single character repeat pattern in each password. We consider one *consecutive* character repeat situation, e.g. the strings "aaaaa", "1145111", "****" have repeat patterns of 4 "a", 3 "1" and 3 "*", respectively. This will be viewed in a diagonal line in the SRH (Section 4.2).

When analyzing the single character repeat patterns, we do not consider the non-consecutive repeat characters because there are too many single character repeats in non-successive positions. These single character repeats in non-consecutive positions cannot represent the users' deliberate usage of repeat patterns.

3.2 Two-character repeat

Two consecutive character repeat patterns are computed by counting the appearance of two consecutive character substrings in each password, and then computing the summation over all passwords. For example, if a password contains "A33A33A", then the two-character repeat patterns for this password are: "A3" repeats one time; "33" repeats one time; and "3A" repeats one time. Two-character repeat pattern is also shown in the SRH (Section 4.2).

3.3 Three or more character repeat

Three or more consecutive character repeat patterns are defined as a 3-10 character substring that repeats at least once in the same password. For example, the password "a563ba563bba5" has the following substrings that repeat exactly once: "a56", "563", "63b", "ba5", "a563", "563b" and "a563b".

3.4 Bidirections of repeat patterns

Referring to the order of substrings, we further divide the password repeat patterns of three or more characters into two directions, i.e. unidirectional (or single-directional) and bidirectional patterns. Single-directional pattern means that the words repeat in one direction, i.e. from left to right. Bidirectional pattern means that words repeat in both left-to-right and right-to-left orders. For example, Figure 2 illustrates the concept of bidirectional repeat patterns for one theoretical password "wer34534werwer3wer34353rew". The different length of repeat patterns is highlighted in colors. Under the context of bidirectional repeat pattern: "wer3" repeats once; "534" repeats once; and "wer" repeats four times.

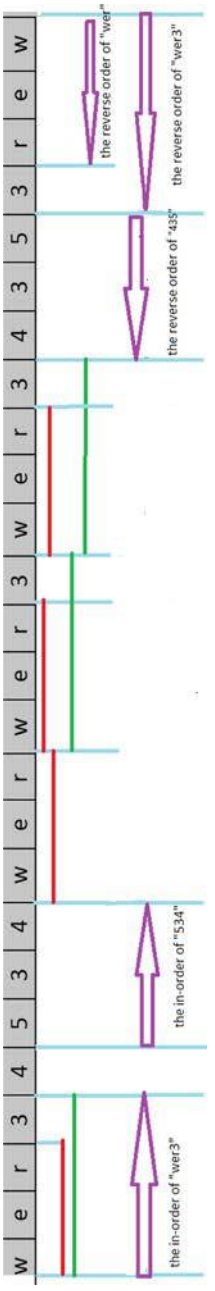


Figure 2.
Repetitive pattern analysis of a sample password "wer34534werwer3wer34353rew" with three or more characters in both unidirectional and bidirectional order

Algorithm 1. 1 or 2 character Repeat Algorithm

```
1:  procedure SHORTREPEAT(password)
2:    read password from file
3:    for every password do
4:      length ← stringLength
5:      for i ← 0 to length – 1 do
6:        if password[i] = password[i + 1] then
7:          sub str ← password[i:i + 2]
8:          OneRe = count(sub str)
9:      for i ← 0 to length – 1 do
10:       if password[i]notequaltopassword[i + 1]
    then
11:         sub str ← password[i.i + 2]
12:         TwoRe = count(sub–str) − 1
13:  end procedure
```

3.5 Algorithms

We formulate the above password repeat patterns as a dynamic programming problem and design algorithms to solve the problem. Algorithm 1 illustrates the idea of finding the short (one or two characters) repeat patterns. The SHORTREPEAT procedure takes the input of *password* and output *OneRe* and *TwoRe*: the repeat times of one character and two consecutive characters. For every password (line) in a file, we first find the repeat patterns of one character and then two consecutive characters. If two characters in adjacent positions are the same character, the occurrence times of such characters are counted as single character repeat. The repeat times of two consecutive characters are the total occupance of such strings minus one, as shown in examples in Section 3.2.

Algorithm 2. Detect long repeat patterns

```
1:  procedure COMPARE (password, status)
2:    LENGT H ← 10
3:    structList ← 0
4:    strLen = length of password
5:    for r ← 3 to LENGTH do
6:      for j ← 0 to (strLen - r) do
7:        subPass = password[j : j+r]
8:        SAMECHAR(subPass,r)
9:        if flag=1 then
10:         for k ← 0 to (strLen - r - j) do
11:           revPass = password[r + j + k]
12:           if status = bio then
13:             REVERSE(revPass)
14:           if (subPass = revPass) then
15:             COUNTED(subPass)
16:  end procedure
17:  procedure SAMECHAR(subPass, r)
18:    for i ← 0 to r do
19:      if all subPass[i] the same then
20:        f lag = 0
21:        break
22:      else f lag = 1
```

```
23:     return flag
24:   end procedure
25:   procedure REVERSE(revPass)
26:     len = length of revPass
27:     for i ← 0 to len do
28:       switch revPass[i] and revPass [len - i]
29:     return revPass
30:   end procedure
31:   procedure COUNTED(subPass)
32:     len = length of structList
33:     for i ← 0 to len do
34:       if subPass = structList[i] then
35:         repeat[subPass]++
36:       else
37:         subPass add to structList
38:     return revPass
39:   end procedure
```

The searching algorithm for longer (three or more characters) repeat patterns is shown in Algorithm 2. Among the variables defined, *strLen* is the password length; *r*, length of compared strings; *j*, the compared times; *subPass*, the compared string; *k*, the translational times; *revPass*, the reversed string; *structList*, list to temporarily store the repeating strings; *status*, unidirectional or bidirectional; and *repeat*, repeat times count. The algorithm takes *password* and *status* as the input and returns the repeat strings and their repeat times as the output.

There are several nested steps in the COMPARE procedure. First, when reading a line of password to the variables *password* and *status*, the length of the *password* is assigned to the variable *strLen*. Second, the string is analyzed to extract the repeat substrings that have three to ten characters. Third, the compared substring *password*$[j + r]$ is defined from left to right, and calls the function SAMECHAR, which checks whether the compared substrings are composed of the same characters. If so, we do not count these substrings because these characters have been counted in one character repeat pattern in Algorithm 1. Otherwise, if the characters in the compared string are not the same and the status is equal to *bi*, which means the algorithm is for searching bidirectional repeat patterns, then perform the reverse of the rest of password (REVERSE procedure). The analysis of unidirectional and bidirectional repeat strings is almost identical. Furthermore, COUNTED function is implemented to avoid recounting the repeat strings that have been counted before.

3.6 Complexity and implementation

The complexity of the algorithm runs in polynomial time, i.e. $O(N * n^7)$, where n is the number of characters for each password ($n < 26$), and N is the number of passwords. We implement the algorithm using both Python and Linux C by taking their advantages of simplification of string manipulation (Python) and efficiency (C). To be scalable, we divide the larger data sets into several parts for better concurrency. To prepare the data, we sort the relative information of the repeat strings from the password file from the quantity of characters, the position of characters and the type of characters, and we then format the results of the analysis for visualization. The system spends about 40 min to

analyze a large password file that has 13,183,056 lines on a workstation running Debian Linux Sparc64 built with UltraSparc T2 (Niagara2) multi-core CPU and 16 GB memory.

4. Password repeat pattern visualization

In this section, we discuss the design and implementation of four different views for analyzing the password repeat patterns. In particular, the CFC provides a quick view of the distribution of character appearance frequency in all user passwords. The SRH uses two dimensions to represent the repeat of single or two consecutive characters, and allows various ordering on the dimension axis. The LRPC expands the password repeat patterns into multidimensions, i.e. three or more consecutive characters repeat, and also allows analysis in both single-directional and bidirectional order. The RWC gives visual representation of top repeated substrings in passwords.

4.1 Character frequency chart

The CFC provides an overview of the frequency distribution of each character of user passwords, as shown in Figure 3. This is usually the first thing an administrator will examine when given a new password file. In CFC, the *x*-axis is a list of 95 printable ASCII characters. The *y*-axis is appearance frequency of each character used in the passwords. For easy examination, when a mouse is over the bars, tool tips will pop to show which character and exact frequency count, e.g. "Letter:0, Frequency:958168" in Figure 3. The bars can be sorted by selecting the "Sort Values" checkbox. Figure 3(a) illustrates characters sorted by natural ASCII values, and Figure 3(b) illustrates sorting the characters in *x*-axis in decreasing order according to frequency count in *y*-axis. The CFC gives an investigator the tips on what characters are most often used (and least used) in user passwords. The view gives an overview of character distribution and helps users narrow down the investigation scale and the comparison of interesting characters.

4.2 Short repeat heatmap

As CFC is limited to explore the quantity of each element in the password file, when considering the repeat patterns of one or more characters, SRH is more helpful. SRH is a graphical representation of repeat frequencies of single or two characters in a 2D matrix. The rows and columns represent the ASCII characters used in user passwords. Each grid or cell represents the repeat frequency of the corresponding two characters in the order of row–column.

Colors are encoded to represent the magnitude of appearance frequency, as shown in the color spectrum legend at the bottom of SRH (Figure 4). Gradient colors mapping from light blue to dark orange in the legend bar are defined in the following function [Equation (1)]:

$$Color_i = Color_s + \frac{(Color_e - Color_s) * i}{n} \tag{1}$$

where the starting color $Color_s$ is set as light blue; the ending color $Color_e$ is set as dark orange; n is the range of colors. Fractions are rounded into integers, which are used to calculate the discrete colors for the intervals. The color spectrum ranges from white, dark blue, light blue, light orange, to dark orange colors. Each interval is on behalf of a range of values. For example, white represents to the characters that do not have 60 repeat letters or strings in password file; dark blue stands for repeat times of letters or strings in the interval

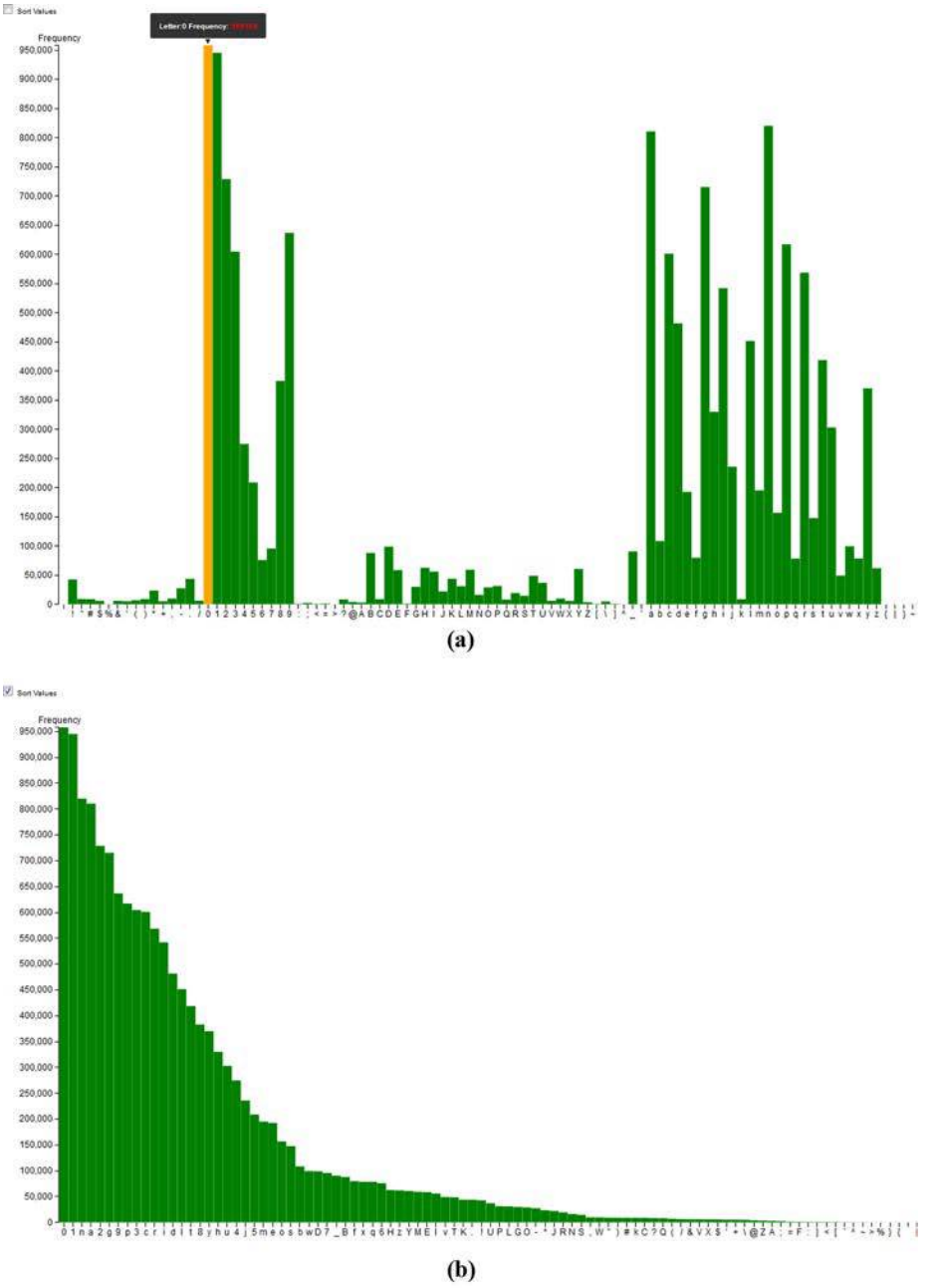
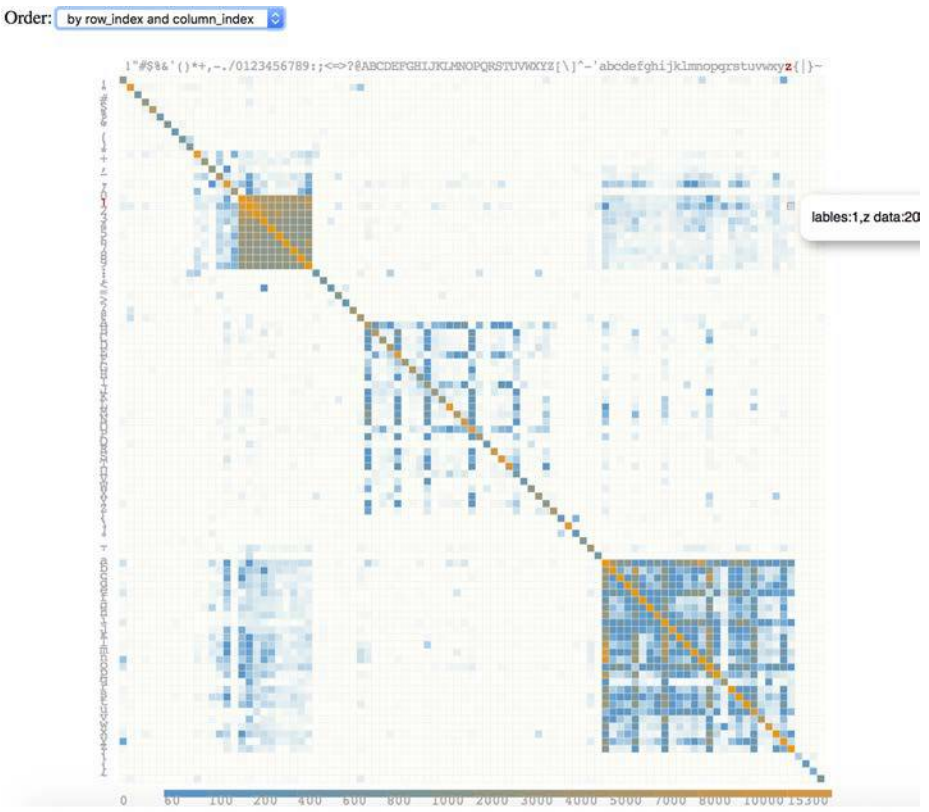


Figure 3.
CFC

Notes: (a) Sorted by ASCII values; (b) sorted by appearance frequency



Note: Clicking characters on the axes allows sorting by repeat frequency

Figure 4.
SRH view with alphabetical sorting on both rows and columns shows high frequency repeat patterns in grids and matrices along the diagonal line.

of 60 to 100; orange color represents the interval of 10,000 to 160,000. A legend bar is automatically generated so that the repeat times can be easily analyzed.

While colors are helpful to identify the patterns, colors only give a range of values for quick overviews. To get the exact number of repeat times, the tool tip of mouse events is implemented to offer an interactive visual way to users for quick access to the original data. As displayed in Figure 4, the row elements and the column elements are 95 printable ASCII characters, respectively. The characters on the rows are ordered from top to bottom, and the characters on the columns are ordered from left to right. If both row and column dimensional axes are sorted in the same order, the grids on the diagonal line represent the repeat patterns of the same character in a password.

For better user interaction, SRH is equipped with five sorting functions: sort by rows only, sort by columns only, sort by both rows and columns, sort by repeat time and no sorting. Sorting by rows only means that only the characters on the vertical axis are sorted in ASCII order. These characters are the first character in repeated substrings. Similarly, sorting by columns only means that only the characters on the horizontal axis are sorted in ASCII order. In other words, the second characters of the repeated substrings are sorted.

Sorting by both rows and columns means that both the first and the second character are sorted in the same order alphanumerically, as shown in Figure 4. In the view, it is easy to find the pattern, i.e. the grids on the diagonal line are in orange color meaning the high repeat frequency. Therefore, repetition mostly likely occurs in the same (one) consecutive character when users make their passwords. There are apparently a few rectangles of darker colors, notably along the diagonal line, i.e. numerical matrix ($0 \sim 9$), capital letter matrix ($A \sim Z$) and lower case letter matrix ($a \sim z$). Investigators can easily reach the conclusion that characters of the same type (inside each of these matrices) are more likely to repeat when users make up their passwords.

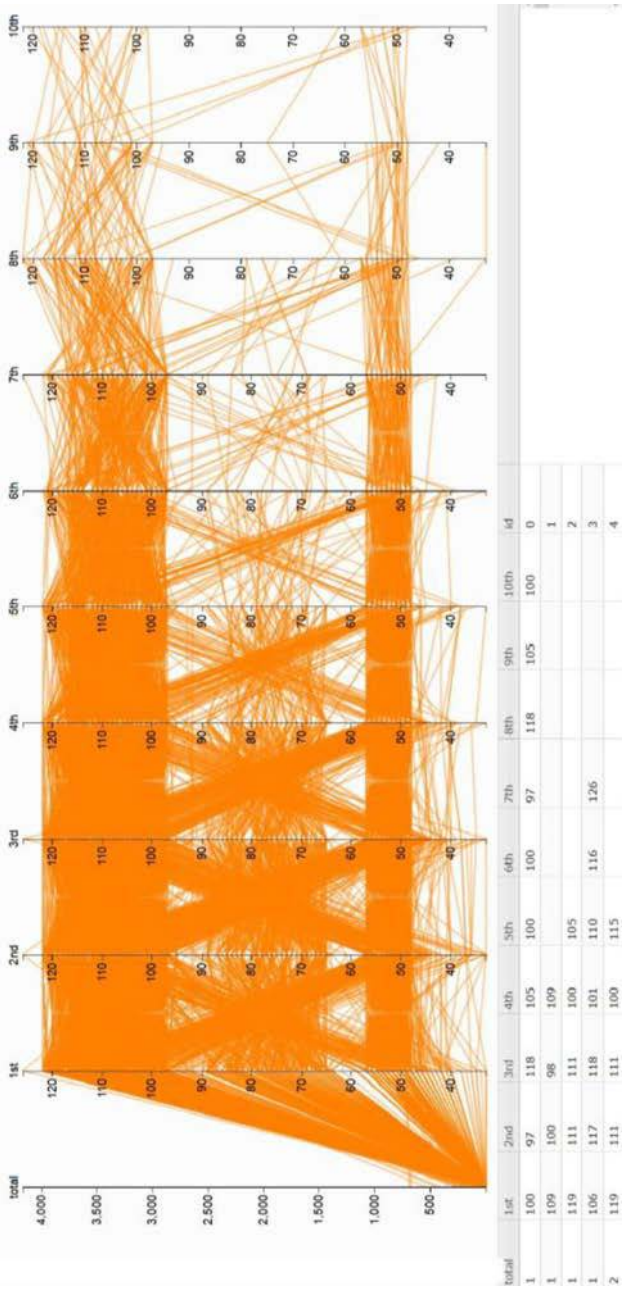
SRH also allows sorting by repeat time on a single character. By clicking any individual character, the grids on that row (or column) are arranged by an increasing order of repeat frequency of that character. If the user clicks the character again, those grids will be rearranged in decreasing order. The interactive view is ideal to explore the repeat patterns of a single character and helps us to explore the connection of characters. In summary, the SRH view provides researchers or investigators not only a quick overview of repeat patterns of all characters but also the details of repeat count for each individual character.

4.3 Long repeat parallel coordinates

While the above SRH view is ideal for identifying repeat patterns of one or two characters, sometimes we need to identify patterns of longer substrings. To that end, we use a LRPC plot for visualization purpose. Parallel coordinates is a geometric device for displaying points in high-dimensional spaces, usually larger than three (Opitz, 1997). We apply the LRPC view to display the longer substrings, which have three or more consecutive repeating characters in the same password. Examples of LRPC views are shown in Figures 5 and 8. In each example, there are 11 coordinates, which are "total" and "1st" to "10th". The "1st" to "10th" axes represent repeating substrings up to ten characters in that order. Each of such axes has a value range of 32 to 127, representing 95 displayable index value of ASCII characters. The "total" axis represents the total repeat counts of such repeating substrings.

An additional table containing the underlying data is attached at the bottom of LPRC. Users may interact with the view by selecting one or multiple rows in the table and, subsequently, highlighting only the interesting substrings under investigation. There are three functions that can perform the filtering and reorganizing of the data. The first one is to just move the cursor to the position on the table. The second way is that if users would like to focus on a specific range, they can use the mouse to brush a rectangle range on an axis to narrow down the scope to track the useful information. Finally, users can also switch the order of coordinates (by dragging and dropping the axes) to find other possibly interesting patterns. Figure 8 illustrates the single-directional repeating substring patterns in each password, and Figure 5 explains the same repeating substrings but in bidirectional order.

In LRPC, we specifically avoid the situations in which every character in the substring is the same, as the situation is already counted in the SRH view. Therefore, we only consider the third to tenth characters. LRPC is useful to analyze longer substring (three or more characters) repeat patterns in the same password.



total	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	id
1	100	97	118	105	100	100	97	118	105	100	0
1	109	100	98	109							1
1	119	111	111	100	105						2
1	106	117	118	101	110	116	126				3
2	119	111	111	100	115						4

Figure 5.
LRPC view for analyzing *bidirectional* repeat patterns of longer substrings (three or more consecutive characters) in the same password

4.4 Repeat word cloud

RWC is a visual representation of repetitive substrings in each password. As sizes and colors are two inherent attributes to the RWC, colors may be designed to separate the different categories of words (verb vs noun, etc.), and sizes are commonly used to define the quantity of the words.

In the tool, RWC is applied to display top n repeated words of length l, where $n \in 100$, 300, 500 and l is an integer that falls into the interval of [3, 10]. It is necessary to clarify that these words are *not* the top appeared passwords. Rather, these words are the top *repeated substrings* in each password. Figure 6 shows an example of top 500 repeated substrings, which include "ana", "123", "ing", "ama", "ang", "and", etc. It suggests that these words have high probabilities to be chosen by users who repeat these words to make up their passwords. RWC is useful because it provides a quick and intuitive view of the top repeated words in passwords.

Figure 6.
Top 500 repeated substrings containing three characters in the RWC

4.5 Implementation

We implement the password repeat pattern visualizations based on D3 (Bostock *et al.*, 2011) technologies. The advantages of using D3 and JavaScript are that the Web-based approach is lightweight and easily accessible from anywhere on the Internet with a browser with no software installation on investigator's machine. Despite the lightweight and convenient nature of the tool, the big data era has made it hard for investigators to compare the analysis results visually with these views. Notably, how to represent large data is a thorny problem. To address the scalability issue and to visualize repetitive patterns of dozens of millions of passwords, we must design the analytic algorithm efficiently (as described in early sections) and use filtering, brushing and linking effectively to convey largest entropy of data. Various filtering and interactive options are included in the tool such as check boxes, drop-down menus/lists, tables, mouse drag/drop on axis, zoom and pan, etc.

5. Case study

In this section, we illustrate several repeat patterns extracted by the visual analytic tool using a publicly available, real-world password file.

5.1 Data set and processing

The data set we use in this case study is RockYou (Weir *et al.*, 2010) passwords which were leaked in early 2010. The data set represents user passwords in real world. RockYou used to have poor password policies so that users were free to set up their passwords. Therefore, the data set has a variety of passwords of different strength levels. There are 14,442,061 passwords that equal to 139,921,466 characters. Due to the data size, the data set represents the real skills of customers to set their passwords. The passwords contain digits, lowercase and uppercase letters and special characters. There are 95 displayable characters of 126 ASCII characters that can be used to set passwords. Data cleaning is also conducted to purge data noise from the raw data set, e.g. URLs of at least 26 characters in a line and obvious programming codes.

5.2 Password character distributions

As suggested by the CFC shown in Figure 3, the higher bars concentrate on the digits' area and lowercase letters' area. That is, digits and lowercase letters are used much often to set passwords; uppercase letters are less used; and special characters are the least used. The top five most appeared characters are "0", "1", "n", "a" and "2". The distribution is consistent with RWC in Figure 6, i.e. the top repeating substrings consist of mainly digits and lowercase letters, most of which are "a", "1" and "2".

Referring to the naturalistic cognition (Storkerson, 2010), people usually have naturalistic cognition when a conscious body connects with the environment. Similarly, when people mention passwords, they usually recall the "0", "1", "a", "n" and so on. Therefore, these characters appear most frequently when people set passwords.

Interestingly, according to the letter frequencies in modern English (Lewand, 2000), "e", "t", "a", "o" and "i" are the top five used characters. However, according to our analysis, "e" is not the most commonly used character in user passwords. Instead, the letter "n" is the top one, as suggested by the comparison of the letter frequency distributions of passwords and natural English language in Figure 7.

Another discrepancy is that Benford's law describes that smaller digits have higher frequency to appear in many real-life data. For instance, "1" occurs as the leading digit

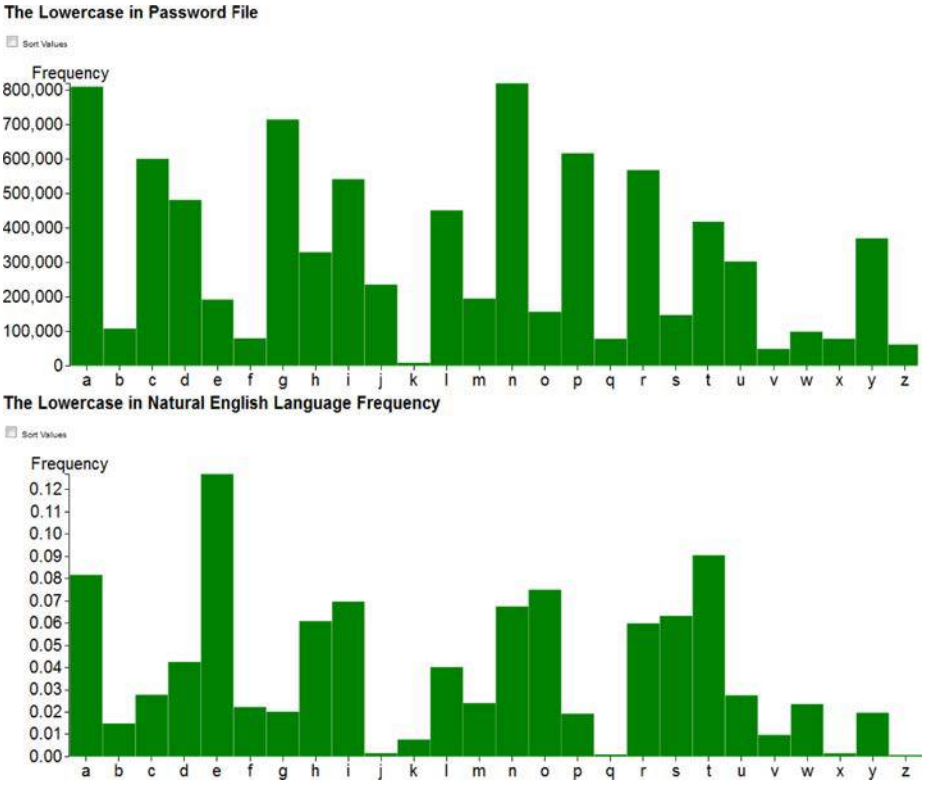


Figure 7.
The comparison of letter frequency in user passwords and natural letter frequency in English language

about 30 per cent of the time, while "9" is the least used. In Figure 3, "1", "2", "3" and "9" are common digits chosen by users. The result is close to Benford's law in terms of "1", "2" and "3". However, "9" does not follow the law. One possible explanation is that the digits in passwords are usually bound with user's birthdays, the holidays or their special dates. In addition, many users simply increment and append numbers to old passwords when it comes to the time their organization IT department forces them to change passwords according to the organization's security policy.

5.3 Short repeat patterns and magnitude

In SRH (Figure 4), both rows and columns are sorted alphabetically according to their ASCII index values. The colors of most cells on the diagonal line of the matrix are orange or close to orange. With comparison to the color legend at the bottom, it is not hard to find that these cells map to the largest numbers, which means that the same character is predominantly used to repeat itself when users choose passwords.

As to repeat patterns of two consecutive characters, we can clearly see a few squares colored in the range of light blue (800) and orange (7000). Three of the squares are significant along the diagonal line of the matrix. This pattern suggests that two characters are mostly likely to repeat if both characters belong to the same group, i.e. both are digits, lowercase letters or uppercase letters.

When we take into account the types of repeat strings, in SRH (Figure 4), there are 9,025 (95×95) cells, representing different repeat times. According to the *id* column in LPRC, there are 184,443 types of repeat strings in single direction, and there are 67,690 types of repeat strings in bidirections. That means every seven passwords have a new kind of repeat string. The repeat time for each string ranges from 1 to 15,300.

There is no doubt that the repeat patterns are widely applied in password setting. Repetitious principle is a very common habit for humans to behave in some fields, such as memory, music, understanding and so on. To memorize the passwords, users usually opt for familiarity to set the passwords. Every noun can be expressed in different formats. Therefore, there are a vast amount of types of repeating strings.

5.4 Long repeat patterns

For three or more consecutive characters' repeat patterns, the lower left view of Figure 1 is the LRPC. The printable ASCII characters are divided into four groups: digits, lowercase letters, uppercase letters and special characters. The dense lines suggest that the quantity of the repetitive strings that have the length between 3 and 10 is high.

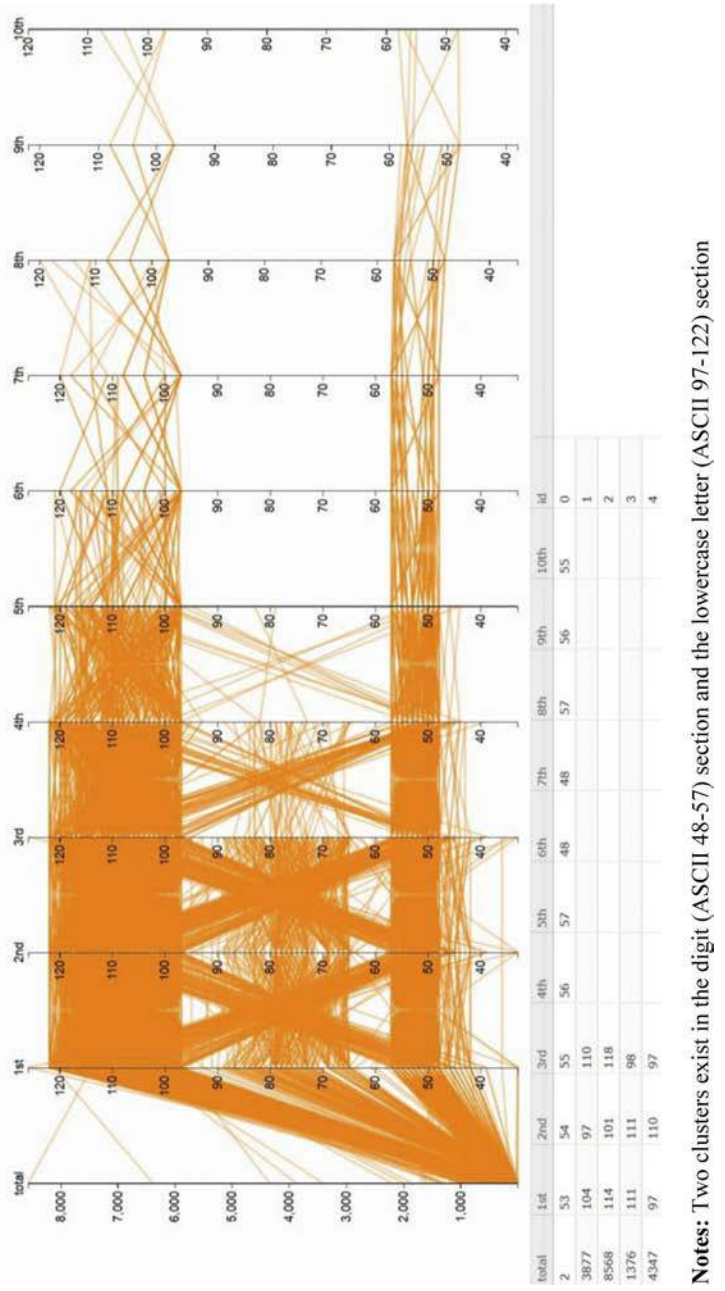
It is interesting to identify that the highest repeat frequency belongs to the strings containing the same type (one of the four groups) of characters. Long repeat patterns are consistent with short repeat patterns (Figure 4), in that characters of the same group are most likely to repeat consecutively in user passwords.

In Figures 5 and 8, the high frequency parts are from the "1st" to "5th" coordinates, suggesting that repeat substrings of length of five or less are most common. The views draw out two clusters in the digit (ASCII 48-57) section and the lowercase letter (ASCII 97-122) section. When we look for the longer repeating substrings from the "6th" to "10th" coordinates, the patterns remain.

We hypothesize that people usually struggle to remember their passwords, especially the longer passwords. To satisfy the password length requirement and to be able to remember passwords later on, users may adopt the strategy to repeat short words to make longer passwords. From the alternative RWC view in Figure 9, the same types of characters, specially some meaningful words, such as "love", "kiss", "ever", "mama", "haha", etc., are repeated many times (among the top 300 substrings) to make the actual passwords. The above observation leads to our conclusion that when users make their long passwords, they will repeat shorter words. In addition, for those repeating words, there is low possibility for users to shift the characters between uppercase and lowercase, and there is high probability that users will choose the same type (ASCII groups) of characters.

We expand our study to analyze repeat patterns of even longer repeat words than the four-character example in Figure 9. Figure 10 shows top 100 consecutive repeating words that have ten characters in length that are used to make the actual final passwords. It is interesting to note that most of the longer (ten-character) repeating strings are composed by the same shorter (two-character) substrings. For example, "hihihihihi", "hahahahaha" and "2121212121" are composed by "hi", "ha" and "21", respectively. When we investigate cases in other sizes (e.g. six to nine characters), the RWCs suggest the same patterns, i.e. shorter substrings are repeated to make longer strings which are repeated again to make longer passwords.

As users would like to use the same character group as the repeating strings in the same password, if the found repeating pattern is widely adopted by users, the



total	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	id
2	53	54	55	56	57	48	48	57	56	55	0
3877	104	97	110								1
8568	114	101	118								2
1376	111	111	98								3
4347	97	110	97								4

Notes: Two clusters exist in the digit (ASCII 48-57) section and the lowercase letter (ASCII 97-122) section

Figure 8.
LPRC view of *single-directional* repeat patterns of longer substrings (three or more consecutive characters) in the same password

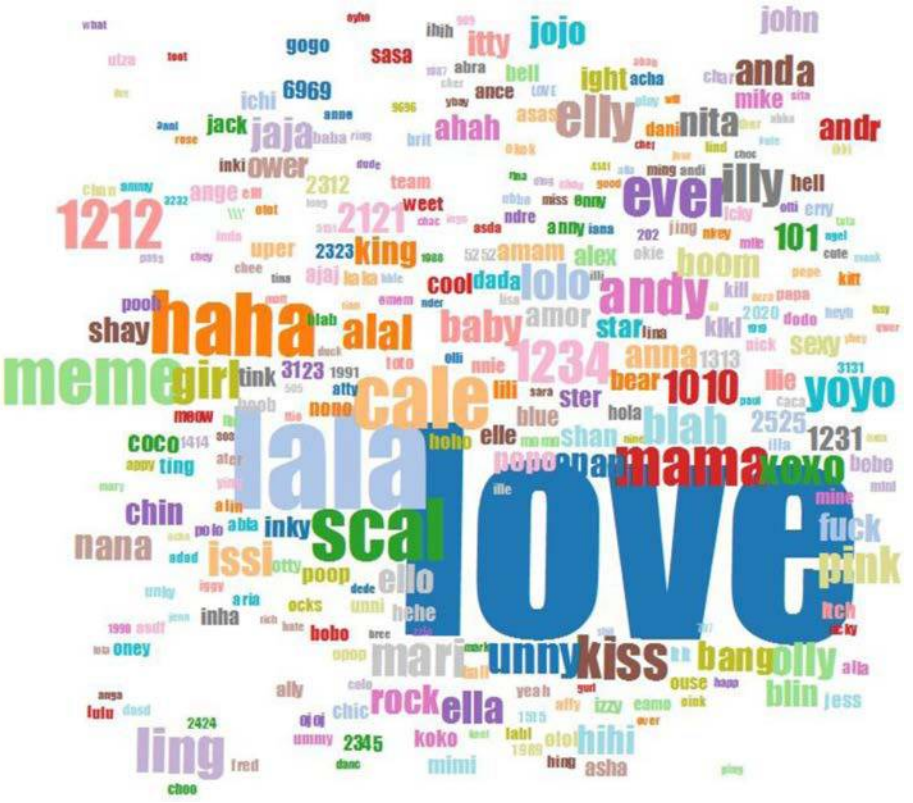


Figure 9.
Top 300 four-character consecutive repeat words in RWC suggesting users make long passwords by repeating shorter words of the same type only (digit, lowercase, uppercase, etc.)

policy that requires long passwords does not necessarily strengthen the security, even though longer passwords are required by many applications.

5.5 Bidirectional repeat asymmetric patterns

The largest number of repeat string in bidirectional LRPC view is 8,568, about twice of the largest number of repeating string in single direction (4171). Specifically, in Figure 11, the strings "rev" and "yba" repeat 8,568 and 6,389 times in bidirectional LRPC (i.e. repeat both left-to-right and right-to-left, respectively). However, the strings "rev" and "yba" repeat only one time in single-direction order (left-to-right).

While strings such as "rev" and "yba" have very high bidirectional repetitious frequency, their unidirectional repetition is very low. If we think about these bidirectional repeat strings, it is hard to match them into any popular words. Therefore, users may think that using unpopular strings and repeating them many times in bidirectional order instead of unidirectional order may solve their problems of making up long passwords while maintaining satisfying security.

6. Conclusion

Passwords have been and will remain the most important factor in user authentication and security in foreseeable future. We develop efficient algorithms to analyze the repetitive

Figure 10.
Top 100 ten-character consecutive repeat words in RWC view. Shorter (mostly two-character) substrings are repeated to make longer strings, which are used to make up the final actual passwords

patterns in passwords and implement an interactive visualization tool to help security practitioners to analyze passwords and detect their patterns. In particular, we formalize both short (one or two characters) and long (three or more characters) repeat patterns. We also design and implement four views, i.e. CFC, SRH, LRPC and RWC, which when linked together, provide insightful views of the repeat patterns of user passwords.

Through case studies on a real-world data set that contains nearly 14 million user passwords, our findings suggest that longer passwords do not necessarily equal to stronger passwords because people usually repeat short substrings to compose final passwords. When users make their passwords by repeating words, they tend to repeat only the same type of characters (i.e. digits, lowercase or uppercase letters). The top single-directional repeating strings usually are meaningful words, while the top bidirectional repeating strings are less meaningful. Furthermore, long repeating strings usually consist of shorter repeating strings to make up the final passwords. Overall, people focus on the ability to remember long passwords more than the security of the passwords. The repeating strings are the product of the requirement of long passwords. Although security experts have been making the wide appeal of making passwords more variable and of bigger size, people often set their long passwords by repeating the small substrings in various ways, providing a false illusion of security. To deal with the increasing complex policy of password rules mandated by many

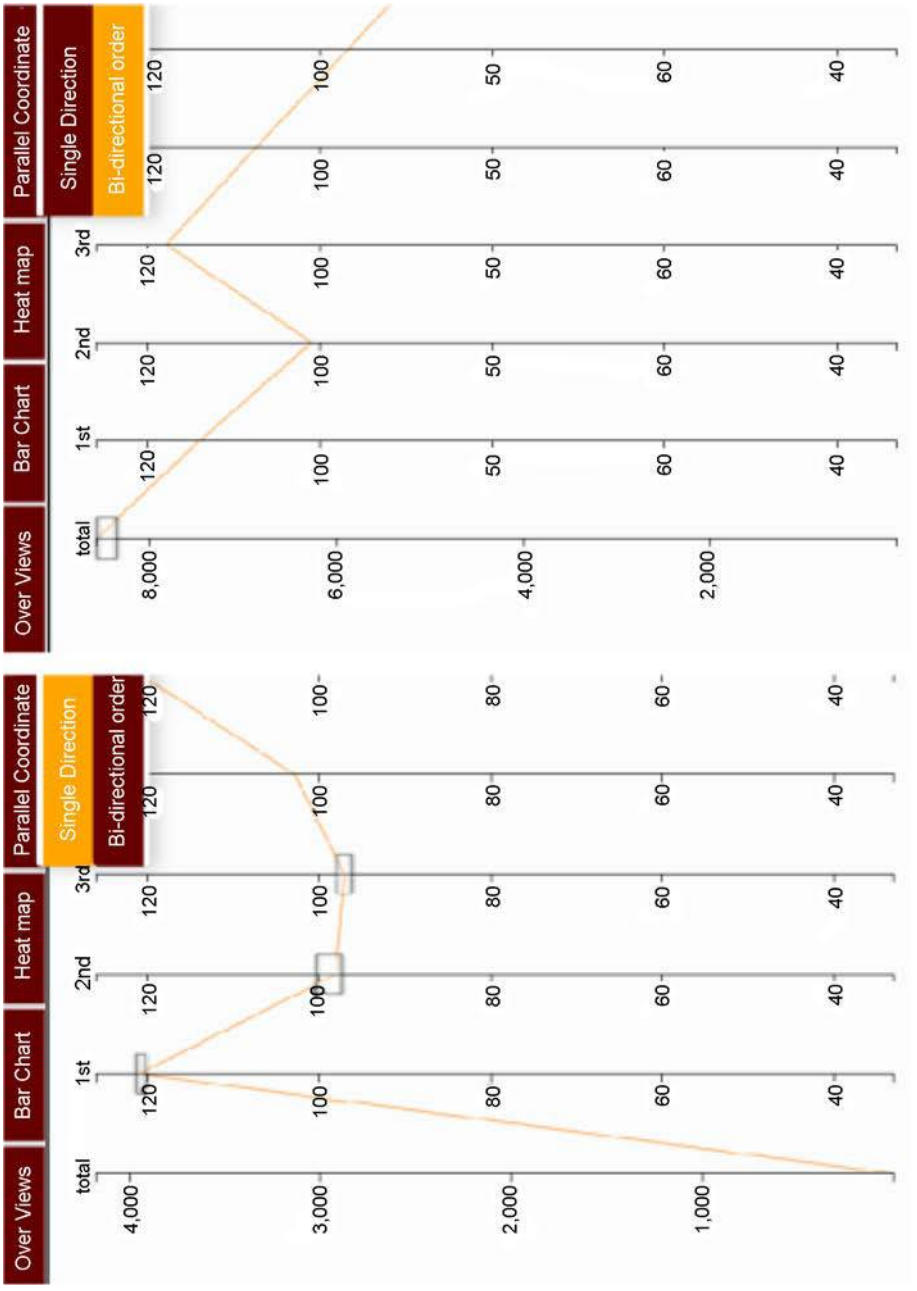


Figure 11. Comparing the repeating string "yba" in unidirectional LRPC (left) with the repeating string "rev" in bidirectional LRPC (right) reveals asymmetric patterns

financial, educational and governmental organizations, users may counter-fight with many unique yet interesting strategies, which will be our future study.

References

Bonneau, J. (2012), "The science of guessing: analyzing an anonymized corpus of 70 million passwords", *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, pp. 538-552.

Bonneau, J., Preibusch, S. and Anderson, R. (2012), "A birthday present every eleven wallets? The security of customer-chosen banking pins", *Financial Cryptography and Data Security*, Springer, pp. 25-40.

Bostock, M., Ogievetsky, V. and Heer, J. (2011), "d^3 data- driven documents", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17 No. 12, pp. 2301-2309.

Chisnell, D. (2014), *The Burden of Authentication: What Friction Points Reveal*, *WATCH Series*, National Science Foundation.

Chou, H.-C., Lee, H.-C., Hsueh, C.-W. and Lai, F.-P. (2012), "Password cracking based on special keyboard patterns", *International Journal of Innovative Computing, Information and Control*, Vol. 8 No. 1, pp. 387-402.

Davis, D., Monrose, F. and Reiter, M.K. (2004), "On user choice in graphical password schemes", *Proceedings of the 13th Conference on USENIX Security Symposium (SSYM'04)*, *San Diego, CA*, pp. 151-164.

Dell'Amico, M., Michiardi, P. and Roudier, Y. (2010), "Password strength: an empirical analysis", *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, *San Diego, CA*, pp. 1-9.

Florencio, D. and Herley, C. (2007), "A large-scale study of web password habits", *Proceedings of the 16th International Conference on World Wide Web*, *Banff*, pp. 657-666.

Klar, R. and Opitz, O. (1997), *Classification and Knowledge Organization*, Springer.

Lewand, R.E. (2000), *Cryptological Mathematics*, The Mathematical Association of America, p. 211, ISBN:978-0883857199.

Morris, R. and Thompson, K. (1979), "Password security: a case history", *Communications of the ACM*, Vol. 22 No. 11, pp. 594-597.

Perito, D., Castelluccia, C. and Duermuth, M. (2012), "Adaptive password strength meters from Markov models", *19th Network and Distributed Systems Security Symposium (NDSS'12)*, San Diego, CA.

Schweitzer, D., Boleng, J., Hughes, C. and Murphy, L. (2009), "Visualizing keyboard pattern passwords", *6th International Workshop on Visualization for Cyber Security (VizSec'09)*, Atlantic City, NJ, pp. 69-73.

Shay, R., Komanduri, S., Durity, A.L., Huh, P.S., Mazurek, M.L., Segreti, S.M., Ur, B., Bauer, L., Christin, N. and Cranor, L.F. (2014), "Can long passwords be secure and usable?", *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, *Toronto*, pp. 2927-2936.

Storkerson, P. (2010), "Naturalistic cognition: a research paradigm for human-centered design", *Journal of Re-search Practice*, Vol. 6 No. 2.

Veras, R., Thorpe, J. and Collins, C. (2012), "Visualizing semantics in passwords: the role of dates", *Proceedings of the Ninth International Symposium on Visualization for Cyber Security (VizSec'12)*, *Seattle, WA*, pp. 88-95.

Weir, M., Aggawal, S., Collins, M. and Stern, H. (2010), "Testing metrics for password creation policies by attacking large sets of revealed passwords", *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, *Chicago, IL*, pp. 162-175.

Zhang, Y., Monrose, F. and Reiter, M.K. (2010), "The security of modern password expiration: an algorithmic framework and empirical analysis", *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, *Chicago, IL*, pp. 176-186.

Corresponding author
Qi Liao can be contacted at: liao1q@cmich.edu