South Asian Journal of Research in Engineering Science and Technology
Volume 4, Issue **3**, **March 2019**

IXplore SAJREST ⓒ2019 ISSN 2455-9261
pp. **29–35**

Graphical Password Authentication Using Various Click Point Techniques

A.Subashchandar[1], C.Revanth Kumar[2] and S.Sajid Ahamed[3]
[1,2,3]Department of Computer Science and Engineering,
Jeppiaar Engineering College, Chennai, Tamil Nadu
subashchandar@hotmail.com

ABSTRACT. *Authentication is one of the most important requirements for information security. There exist various methods for authentication based on what we know (e.g. passwords, PINs), what we have (e.g. security hardware tokens) and who we are (e.g. biometric fingerprints). Among the existing methods, password-based systems are easier to implement and therefore the most frequently used method for authentication. Being very critical for security, passwords are often targeted during cyber-attacks as well. An attacker that hacks a system and reveals user passwords stored within the database gets unauthorized access to accounts of all users. In the past many enterprise companies and organizations were victims of such attacks. Attackers use frequently SQL injection vulnerabilities that exist within applications in order to access database tables. They send arbitrary SQL queries to retrieve passwords and other sensitive data from tables and manipulate stored data, even by using automated tools such as sqlmap or Havij. Considering this fact, developers must never store passwords in plaintext within databases. Developers mostly know the fact that they should store hash values of passwords instead of plaintext. However, it is also a critical security weakness if the hash value of a password is calculated and stored without appending per-user unique salt value to the password before hashing. In a classical scenario, a user chooses a password by a registration process. The hash value (md5, sha1, sha256 etc.) of the password is calculated on the backend server and this calculated hash value is stored in the database.*

Keywords: Pass word, Authentication, Attack

1. **Introduction.** Hash functions are one-way functions, attackers can perform brute force, dictionary or rainbow-table attacks in order to reveal input values (i.e. plaintext password) from the given output values (i.e. hash value). By brute-force attacks, the hash value of each possible input value is calculated and compared with the given hash value to crack. By dictionary attacks, large dictionary files containing thousands or millions of possible passwords are utilized. Given a hash value to crack, an attacker calculates
the hash value of each plaintext word from the dictionary line by line and compares the

calculated hash values with the given hash value. If they are matched, the plaintext password is thus revealed. On the other hand, a very large set of pre-computed hash tables containing hash values and their corresponding plaintext values are used by rainbow-table attacks. Given a hash value to crack, an attacker checks if the given hash value exists within the pre-computed lookup table. If it exists within the table, the plaintext password is found out. If we compare brute-force, dictionary and rainbow-table attacks, they all have pros and cons. Brute-force attacks find out the plaintext definitely in the end but they are very time consuming. Dictionary attacks are fast but the success rate is not sufficient. Rainbow-table attacks are fast and successful at cracking but they require having a very big disk storage capacity. They are especially non-practical if a salt value is used for password hashes. In this paper, we propose a new method for increasing success rates of dictionary attacks. For our method we analyzed leaked real-life user passwords and identified several patterns which are commonly chosen by many users to create a complex and strong password from a dictionary word. For example, a dot ("."), an exclamation mark ("!") or "123" are often appended at the end of a dictionary word. Similarly, a dictionary word is repeated two times (e.g. kingking) or three times (e.g. kingkingking).

2. **Problem Statement and Preliminaries.** It is always suggested that a secure password must not consist of only lowercase letters. Instead, it must contain lowercase and uppercase letters, digits and special symbol characters. A password fulfilling these complexity requirements would provide high entropy [13] and therefore should be more resistant against password guessing attacks. Today, enterprise companies and organizations define such password rules within their security policies and try to enforce their employees and customers to choose complex passwords. On the other hand, it is questionable if a password fulfilling the complexity rules including minimum length can be considered as a strong password. Let's take the following password "P4s5w0rd1." into consideration. This password has the length of ten characters and contains five lowercase letters, one uppercase letter, four digits and one special symbol. This password is considered and accepted in general as a strong password according to many password policies of enterprise companies and organizations. But we believe, this is an insecure password and can be easily cracked by using our pattern-based attack. The password "P4s5w0rd1." contains three different common patterns. The first pattern is capitalization of the table with the top

ten list of regular expressions for passwords with the length between 2 and 5first letter. The second pattern is replacing certain letters with numbers (a→4, o→0, s→5) and the third pattern is appending "1." to the password. Since people are bad at remembering complicated passwords and have to use complex passwords due to password policies, they tend to create "strong" passwords by using such patterns. However, these common patterns jeopardize security of the passwords. If many passwords share the same patterns, they can be identified and then misused to guess passwords successfully with the help of automated tools.

2.1 Rockyou Pattern Analysis Based on Regular Expressions

Skull security [14] provides various leaked real-life password dictionaries to download. We utilized their special"rockyou" password list that includes additionally the total count for each unique password.

In the first step, we analyzed the rockyou passwords based on their regular expression representations. We created different Top 10 lists which consist of the most common regular expressions and their hit counts according to the different password lengths as shown in Table I, II and III

The Top 10 lists showed us some interesting facts. Most of the passwords are composed of appending numbers to letters. Therefore, we decided to continue with the analysis of dual and triple combinations of different character groups as explained in the following section. Another interesting fact is that the top one regular expression of passwords with the length of ten characters is ˆ[0-9]{10}$. This shows us that passwords belonging to this group consist of only numbers with the length of ten digits. We examined such passwords manually and concluded that these are mostly telephone numbers.

3. **Proposed System**

This paper explains tool, namely pbp-generator (pattern based password generator), that implements our identified patterns and creates a new pattern-based large dictionary file from a given dictionary file. We generated a pattern-based dictionary file with ca. 2.3 billion passwords to crack password hashes belonging to fifteen different datasets which consist of real-life leaked password hashes. Digital forensic investigators are involved with the analysis of crime cases. They often come across password protection during investigation. They need to crack passwords either in order to access a particular user account or to unlock encrypted or otherwise obfuscated digital evidence. Our pattern-based method would help forensic investigators for more efficient password cracking. It is important to note that security of hash functions is not within the scope of this paper. If a user chooses a weak password with a certain pattern, even a very secure hash function cannot prevent attackers from cracking password hashes. Patterns have no negative effect on computed hash values. In conclusion, the focus of this paper is the analysis of user-chosen plaintext passwords rather than the formal security model of hash functions.

In the proposed method increases the success rates of dictionary attacks. For our method we analyzed leaked real-life user passwords and identified several patterns which are commonly chosen by many users to create a complex and strong password from a dictionary word. We developed a software tool, namely pbp-generator (pattern based password generator), that implements our identified patterns and creates a new pattern-based large dictionary file from a given dictionary file. We generated a pattern-based dictionary file with 2.3 billion passwords to crack password hashes belonging to fifteen different datasets which consist of real-life leaked password hashes.

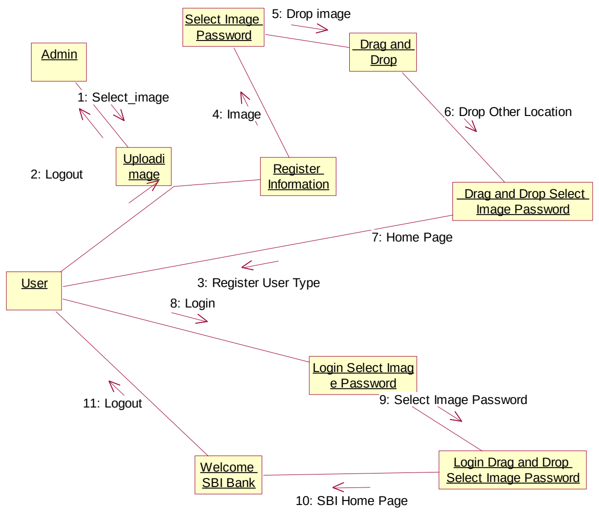

FIGURE 1. Collaboration between entities

4. Rock you Pattern Analysis Based on Dual and Triple Combinations

After analyzing the most common regular expressions representations, we analyzed the frequency of dual and triple combinations of different character groups (i.e. alpha, digit TABLE IVDUAL COMBINATION OF CHARACTER GROUPS WITH EXAMPLES and special symbol). In this analysis, [:alpha:] represents any alpha character between a to z and between A to Z. [:digit:]represents numbers between 0 and 9. [:symbol:] represents the following punctuation characters: . , " ' ? ! ; : # $ % &( ) * + - / <> = @ [ ] ˆ _ { } |.

By the dual combination analysis, the total numbers of [:alpha:]+[:digit:], [:alpha:]+[:symbol:] and[:digit:]+[:symbol:] combinations and their reverse order combinations were analyzed. This analysis showed us that circa 10 million rockyou passwords (30%) are in the form of [:alpha:] + [:digit:] combination, which means users mostly prefer appending a number to a dictionary word to create their passwords. Based on these results, we decided to examine [:alpha:]+[:digit:] combinations further to find more specific patterns. In the Table IV, the total counts of all dual combinations and their examples from the rockyou list are shown.

By the triple combination analysis, the total numbers of[:alpha:]+[:digit:]+[:symbol:], [:alpha:]+[:symbol:]+ [:digit:]and [:digit:]+[:symbol:]+[:alpha:] combinations and their reverse order combinations were analyzed. Compared with the dual combinations, the triple combinations are not very much preferred by the rock you users. The most frequently used triple combinations are [:alpha:]+[:symbol:]+[:digit:] with 0.57% and [:alpha:]+[:digit:]+[:symbol:] with 0.25%.Analyzing these combinations further we identified that digits and special symbols are together (e.g. “#1”, “123.”, “*1” etc.) appended to dictionary words to create passwords. The total counts of all triple combinations and their examples from the rockyou list are shown.

In addition to dual and triple combination analysis, we checked the frequencies of the punctuation characters. This analysis showed that certain symbols are more frequent than the others. The most frequently used punctuation character is point (.) with 0.7%. Underscore (_) has the second place with 0.58% and exclamation mark (!) has the third place with 0.55%. The total counts of each punctuation character in the password list are given. These frequencies were taken into consideration in further analysis.

5. **Conclusions.** This proposed method is based on frequently used patterns can be identified and misused to generate pattern-based password dictionaries. These common patterns can be afterward exploited to crack more password hashes compared with traditional dictionary attacks. In order to identify common password patterns, we performed both manual and automated analysis on a large set of leaked real-life passwords of rockyou.com gaming portal. After identifying the patterns, we developed a software tool, namely the pbp-generator, which creates many pattern-based passwords from a given traditional dictionary. We utilized the generated pattern-based dictionary to perform cracking tests against real-life leaked password hashes from 15 different datasets. According to the test results, we could crack with pattern-dictionaries many more password hashes, which cannot be cracked by using the rock you password list.

REFERENCES

[1] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proc. IEEE*, vol. 91, no. 12, pp. 2021–2040, Dec. 2003.

[2] (2011). *PlayStation Network Hack: Why it Took Sony Seven Days to Tell the World*. [Online]. Available: http://www.theguardian.com/technology/gamesblog/2011/apr/27/playstationnetwork-hack-sony

[3] (2009). *RockYou Hack Compromises 32 Million passwords*. [Online]. Available: http://www.scmagazine.com/rockyou-hack-compromises-32-million-passwords/article/159676/

[4] (2013). *Software Company Tom Sawyer Hacked, 61,000 Vendors Accounts Leaked*. [Online]. Available: http://www.databreaches.net/software- company-tom- sawyer-hacked-61000-vendors-accounts-leaked/

[5] (2013). *Hackers Leak Data Allegedly Stolen from Chinese Chamber of Commerce Website*. [Online]. Available: http://news.softpedia.com/news/Hackers-Leak-Data-Allegedly-Stolen-from-Chinese-Chamber-of-Commerce-Website-396936.shtml

[6] *LinkedIn Hack*. [Online]. Available: http://en.wikipedia.org/wiki/2012_LinkedIn_hack, accessed Apr. 22, 2015.

[7] *SQL Injection*. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection, accessed Apr. 22, 2015.

[8] *password Storage Cheat Sheet*. [Online]. Available:https://www.owasp.org/index.php/password_Storage_Cheat_Sheet,accessed Apr. 22, 2015.

[9] *Brute-Force Attacks*. [Online]. Available: https://www.owasp.org/index.php/Brute_force_attack, accessed Apr. 22, 2015.

[10] V. Goyal, V. Kumar, M. Singh, A. Abraham, and S. Sanyal, “CompChall:Addressing password guessing attacks,” in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC),* Apr. 2005, pp. 739–744.