ISSN:0975-9646

A Cognitive Model for the Forensic Recovery of End-User Passwords

R.Arun kumar,
M.E., Computer Science &Engg.,
School of Engineering, Vels University, Chennai, India.

Dr.Kumar,
Asst. Prof., Dept. of Computer Science &Engg.,
School of Engineering, Vels University, Chennai, India.

Abstract: Most of the application developers did the mistaken to store user passwords within databases as text format or only as their seasoning hash values. More real-life successful hacking attempts that enabled attackers to get unauthorized access to reactive database entries including user passwords have been experienced in the past. Appropriate password hashes, attackers perform bruteforce, dictionary or rainbow-table attacks to expose text format passwords from their hashes. Dictionary attacks are very fast for cracking hashes but their victory velocity is not enough. In this paper, we propose a book method for improving dictionary attacks. Our method exploits several password patterns that are commonly preferred by users when trying to choose a difficult and strong password. In order to analyze and show victory velocity of our developed method, we performed cracking tests on real-life leaked password hashes by using both a conventional dictionary and our pattern-based dictionary. We observed that our pattern based method is superior for cracking password hashes.

Index Terms—password security, authentication, data security, dictionary attacks, hash cracking passwords.

INTRODUCTION

Validation is one of the most important requirements for information security. There exist various methods for validation based on passwords,PINs ,security hardware tokens and biometric fingerprints. Among the existing methods, password-based systems is easy to implement and consequently the most commonly used method for validation. 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 illegal 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 sql map . Considering this fact, developers must never store passwords in text format within databases. Developers mostly know the fact that they should store hash values of passwords instead of text format.However, it is also a crucial security weakness if the hash value of a password is calculated and stored without appending per-user uses hashes to unique salt value to the password. In a traditional situation, a user chooses a password by a registration process. The hash value. of the password is calculated on the backend-server and this calculated hash value is stored in the database. This execution is very unsure of yourself too. Even though hash functions are one-way functions, attackers can perform brute-force, dictionary or rainbow-table attacks in order to reveal input values (i.e.text format password) from the given output values (i.e. hash value).By brute-force attacks , the hash value of each feasible input value is calculated and compared with the given hash value to crack. By dictionary attacks , large dictionary files containing thousands or millions of feasible passwords are utilized. Given a hash value to crack, an attacker calculates the hash value of each text format from the dictionary line by line and compares the calculated hash values with the given hash value. If they are matched, the text format password is therefore exposed. On the other hand, a very large set of pre-computed hash tables containing hash values and their corresponding text format 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 text format password is establish out. If we compare brute-force, dictionary and rainbow-table attacks, they all have pros and cons. Brute-force attacks find out the text format absolutely in the end but they are very time overriding. Dictionary attacks are fast but the victory velocity 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.

RELATED WORK

We propose a new method for increasing victory velocity of dictionary attacks. For our method we have analyzed leaked real-life user passwords and identified several patterns which are commonly chosen by many users to create a difficult and tough password from a dictionary word.

1.PBP-generator (pattern based password generator):

We developed a software tool namely pbp-generator for benchmarking. pbp-generator gets a dictionary file as input, creates several variations of each dictionary word from the given input file based on Type 1 identified patterns and adds them to the output file which represents the generated pattern-based dictionary file. as well, pbp-generator adds many other passwords from Type 2 identified patterns into the output file. The Type 2 patterns are not applied on input file, but they are used to create certain passwords (e.g. Month name with year, special keyboard sequences etc.) to be added directly into the output dictionary file. The passwords from the given dictionary file are explicitly included within the output dictionary file, because this enables us to differentiate if a given hash can be cracked only with the pattern-based dictionary but not with the given input dictionary file.

2.The Cracking Tests:
We used pbp-generator to generate a pattern-based dictionary file from the original rockyou password list which contains 14,344,399 unique passwords. pbp-generator generated a pattern-based dictionary file that contains 2,247,786,433 (circa 2.3 billion) unique passwords. The new dictionary file contains 156 times more passwords compared with the rock you list.

3.Performance Analysis:
Since the pattern-based dictionary contains many more passwords than the rockyou list, it takes longer to perform hash cracking with the pattern-based dictionary. The hash cracking tests were performed on a 64-bit machine with an Intel i5 dual core 3.2 GHz processor and 12 GB RAM. Hashcat was executed with 32 parallel-running threads.

4.New Password Schemes:
Proposes a password creation scheme based on effective Technology. This scheme inserts or replaces randomly fixed number of characters in a user chosen password. As explained in this paper, inserting or replacing characters are typical patterns which can be misused to guess passwords successfully.and also propose some password mechanisms in which a user can choose a virtual password scheme ranging from weak security to strong security.

RESULT

One possible solution can be that users exploit secure password managers (SPM) to store their passwords. SPMs generate unique, random and complex passwords without any pattern, store them within a database and store the database in an encrypted form on file systems. In order to decrypt the database and retrieve the passwords, a master secure password must be provided by users. In addition, some SPMs ask users to provide a physical file which is generated randomly during the setup phase of the password database creation. Providing this, users generate secure passwords for each service they use with the help of their SPM and do not need to memorize them. They just need to memorize the master password and protect the physical file against unauthorized access. It is in this case important that the master password is difficult, accidentally generated and contains no pattern. But it is not a problem for users to memorize a single difficult password and remember it later. Furthermore, some SPMs offer smart-card validation.Another solution can be two-factor validation. Today validation systems should not depend only on knowledge of username-password pairs, especially for critical applications like email, online banking or e-commerce. A new validation factor based hardware token, smart-card or fingerprint should be additionally checked during validation process. As examples, online banking applications benefit today alter opposing hardware tokens and similarly some online services like Google Mail, Twitter, Wordpress etc. support software tokens that are sent over SMS or generated by a native mobile app Google Authenticator. Considering the pattern risks, it is vital to revise current password validation systems as well. They normally check if a user-given password is a dictionary word or not. If it is a dictionary word, it is black-listed and rejected. The user is asked to choose a non-dictionary password. This existing feature should be unlimited to cover passwords with patterns. They can let pbp-generator create a pattern-based dictionary file from their current dictionary file and afterward check if users enter passwords which exist within the pattern-based dictionary file. Academic researchers focusing on password security and validation systems should take patterns into concern and propose solutions consequently. The related academic works from the past should be re-evaluated by considering the risks caused by patterns. Security attentiveness trainings held especially for non-security experts should take patterns into consideration as well. Attendees should be informed about the patterns and warned not to use pattern-based passwords.

V. CONCLUSION

Weak passwords are critical threats for validation systems. Seizing password hashes, especially unsalted hashes, attackers can use different attack techniques brute-force, dictionary, rainbow-tables to crack hashes and expose text format passwords. Security experts try to found security awareness for tough passwords. In accumulation, validation systems implement password policies to execute complication rules. Being forced to use tough passwords, people lean to use like patterns when choosing their "tough" passwords. But such patterns imperil security of passwords. In this paper we explain how commonly used patterns can be identified and altered to cause pattern-based password dictionaries. These common patterns can be afterward browbeaten to split more password hashes compared with conventional dictionary attacks. In order to identify common password patterns, we performed both physical and robotic analysis on a large set of leaked real-life. 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 alongside 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 password list. From this perspective, our proposed pattern-based attacks improve dictionary attacks and can be considered as the new generation of dictionary attacks. It can especially help forensic investigators for more efficient password cracking compared with the existing techniques.

REFERENCES

[1] L. O'Gorman, "Comparing passwords, tokens, and biometrics for userauthentication," Proceedings of the IEEE, vol. 91, no. 12, pp. 2021–2040, Dec 2003.

[2] "Playstation Network Hack," http://www.theguardian.com/ technology/gamesblog/2011/apr/27/playstation-network-hack-sony, 2011.

[3] "RockYou hack compromises 32 million passwords," http://www.scmagazine.com/rockyou-hack-compromises-32-millionpasswords/article/159676/, 2009.

[4] "#OpWorldCup: Anonymous Hacks Brazilian Government, Police, Court, Globo TV and Cemig Telecom," http://hackread.com/anonymoushackers-brazil-worldcup-hacks/, 2014.

[5] "Software Company Tom Sawyer Hacked, 61,000 Vendors Accounts Leaked,"http://www.databreaches.net/software-company-tomsawyer-hacked-61000-vendors-accounts-leaked/, 2013.