

available at www.sciencedirect.comjournal homepage: www.elsevier.com/locate/diinDigital
Investigation

A novel time-memory trade-off method for password recovery

Vrizlynn L.L. Thing*, Hwei-Ming Ying

Institute for Infocomm Research, Cryptography and Security Department, 1 Fusionopolis Way, # 21-01, Connexis (South Tower), Singapore 138632, Singapore

ABSTRACT

Keywords:

Password recovery
Time-memory trade-off
Cryptanalysis
Pre-computation
Rainbow Table

As users become increasingly aware of the need to adopt strong password, it hinders the digital forensics investigations due to the password protection of potential evidence data. In this paper, we analyse and discuss existing password recovery methods, and identify the need for a more efficient and effective method to aid the digital forensics investigation process. We show that our new time-memory trade-off method is able to achieve up to a 50% reduction in terms of the storage requirement in comparison to the well-known rainbow table method while maintaining the same success rate. Even when taking into consideration the effect of collisions, we are able to demonstrate a significant increase (e.g. 13.28% to 19.14%, or up to 100% based on considering total plaintext–hash pairs generation) in terms of the success rate of recovery if the storage requirement and the computational complexity are to remain the same.

© 2009 Digital Forensic Research workshop. Published by Elsevier Ltd. All rights reserved.

1. Introduction

In Digital Forensics, the use of password protection presents a challenge for investigators while conducting examinations. While there exist methods to decode hashes to reveal passwords used to protect potential evidence, lengthier passwords with larger character sets have been encouraged to thwart password recovery. Awareness of the need to use stronger passwords and active adoption has rendered many existing password recovery tools inefficient or even ineffective.

The more common methods of password recovery techniques are guessing, dictionary, brute force and more recently, using rainbow tables. The guessing method is attempting to crack passwords by trying “easy-to-remember” and common passwords like “password”, “1234”, etc. It can also be a password based on a user’s personal information or a fuzzy index of words on the user’s storage media. A statistical analysis of 28,000 passwords recently stolen from a popular U.S. website and posted on the Internet revealed that 16% took a first name as a password and 14% relied on “easy-to-remember” keyboard combinations (Yahoo News, 2009). Therefore, the guessing

method can be quite effective in some cases where users are willing to compromise security for the sake of convenience.

The dictionary attack method composes of loading a file of dictionary words into a password cracking tool to search for a match of their hash values with the stored one. Examples of some of these password cracking tools include Cain and Abel (Cain and Abel), John the Ripper (John The Ripper) and LCP (LCPSoft).

In the brute force cryptanalytic attack, every possible combination of the password characters is attempted to perform a match comparison. It is an extremely time consuming process but the password will be recovered eventually if a long enough time is given. Cain and Abel, John the Ripper as well as LCP are able to conduct brute force attacks.

In Hellman (1980), Hellman introduced a method which involves a trade-off between the computation time and storage space needed for performing recovery of a hash value to its plaintext form. It can be applied to retrieve Windows login passwords encrypted into LM or NTLM hashes (Todorov, 2007), as well as passwords in applications using the above-

* Corresponding author.

E-mail addresses: vriz@i2r.a-star.edu.sg (V.L.L. Thing), hmying@i2r.a-star.edu.sg (H.M. Ying).

1742-2876/\$ – see front matter © 2009 Digital Forensic Research workshop. Published by Elsevier Ltd. All rights reserved.

doi:10.1016/j.diin.2009.06.004

mentioned hash algorithms, such as the Adobe Acrobat. It can also be used to crack the encryption used in Microsoft Word and Excel documents. Passwords encrypted with hashing algorithms such as MD5 (Rivest, 1992), SHA-2 (National Institute of Standards and Technology (NIST), 2002) and RIPEMD-160 (Dobbertin et al., 1996) are also susceptible to this recovery method. In addition, this method is applicable to many searching tasks including the knapsack and discrete logarithm problems.

In Oechslin (2003), Oechslin proposed a faster cryptanalytical time-memory trade-off method, which is an improvement over Hellman's method. Since then, this method has been widely used and implemented in many popular password recovery tools. The pre-computed tables that are generated in this method are known as the rainbow tables. The details of this method will be covered further in Section 2.

There are currently numerous products of pre-computed rainbow tables in the market. The more prominent ones are RainbowCrack (Shuanglei) and Ophcrack (Objectif Sécurité). RainbowCrack is developed by S. Zhu. It is a direct implementation of Oechslin's method. Several patches have since been released to support more hash algorithms including MD5 and SHA-1 (National Institute of Standards and Technology (NIST), 1995), as well as to increase the character set size. Ophcrack is owned by Objectif Sécurité, a leading Swiss consulting company in the field of information systems. Ophcrack is also a direct implementation of Oechslin's method developed by Oechslin himself and C. Tissieres. However, Ophcrack can only process LM and NTLM hashes. AccessData (AccessData) is another company adopting rainbow tables in their password recovery tools.

In this paper, we present a new design of a time-memory trade-off pre-computed table structure. Comparison of our method is made against the rainbow table method (Oechslin, 2003), as it is the most efficient and effective password recovery method to-date. We demonstrate the improvement in effectiveness and performance, in terms of increased success rate in password recovery while maintaining the same amount of storage space and computational complexity.

Our new method is shown to significantly increase the success rate by up to 100% for total plaintext-hash pairs comparison and 13.28% to 19.14% in a few example scenarios for distinct pairs comparison. Up to a 50% reduction in terms of storage requirement is achieved. This translates to about 1.35 TB storage conservation compared to the AccessData rainbow tables, which has an average size of 2.7 TB each (AccessData), while maintaining the same success rate.

The rest of the paper is organised as follow. In Section 2, we present a detailed analysis and discussion on existing time-memory trade-off password recovery methods. We describe the rainbow table method in more details in Section 3. We present the design of our new method in Section 4. Analysis and performance evaluations are carried out in Section 5 to demonstrate the performance enhancement. Future work is described in Section 6. Conclusions follow in Section 7.

2. Analysis of existing work

The idea of a general time-memory trade-off was first proposed by Hellman in 1980 (Hellman, 1980). In the context of password recovery, we describe the Hellman algorithm as follows:

We let X be the plaintext password and Y be the corresponding stored hash value of X . Given Y , we need to find X which satisfies $h(X) = Y$, where h is a known hash function. However, finding $X = h^{-1}(Y)$ is feasibly impossible since hashes are computed using one-way functions, where the reversal function, h^{-1} , is unknown. Hellman suggested taking plaintext values and applying alternate hashing and reducing, to generate a pre-computed table. For example, for a 7-character possible password (composed from a character set of English alphabets), a corresponding 128-bit hash value is obtained by performing the password hashing function on the password. With a reduction function such as $H \bmod 26^7$, where H is the hash value converted to decimal digits, the resulting values are distributed in a best-effort uniform manner. For example, if we start with the initial plaintext value of "abcdefg" and upon hashing, we may get a binary output of 0000000...00001000000...01, which is 64 0's and a 1 followed by 62 0's and a 1. In this case, $H = 2^{63} + 1 = 922\,337\,203\,685\,477\,5809$. The reduction function will then convert this value to "3665127553" which will correspond to a plaintext representation "lwmkgij", computed from $(11(26^6) + 22(26^5) + 12(26^4) + 10(26^3) + 6(26^2) + 8(26^1) + 9(26^0))$. After a pre-defined number of rounds of hashing and reducing (making up a chain), only the initial and final plaintext values are stored. Therefore, only the "head" and "tail" of a chain are stored in the table. Using different initial plaintexts, the hashing and reducing operations are repeated, to generate a larger table (of increasing rows or chains). A larger table will theoretically contain more pre-computed values (i.e. disregarding collisions), thereby increasing the success rate of password recovery, while taking up more storage space. The pre-defined number of rounds of hashing and reducing, on the other hand, increases the success rate by increasing the length of the "virtual" chain, while bringing about a higher computation overhead. To recover a plaintext from a given hash, a reduction operation is performed on the hash and a search for a match of the computed plaintext with the final value in the table is conducted. If a match is not found, the hashing and reducing operations are performed on the computed plaintext to arrive at a new plaintext for another round of search to be made. The maximum number of rounds of hashing, reducing and searching operations is determined by the chain length, before giving up. If the hash value is contained in one of these chains in the first place, a match would be found in a particular chain. The values in the chain is then worked out by performing the hashing and reducing functions to arrive at the plaintext giving the specific hash value. Unfortunately, there is a likelihood that chains with different initial values may merge due to collisions. These merges will reduce the number of distinct hash values contained in the chains and therefore, diminish the rate of successful recovery. The success rate can be increased by using multiple tables with each table using a different reduction function. If we let $P(t)$ be the success rate of using t tables, then $P(t) = 1 - (1 - P(1))^t$, which is an increasing function of t since $P(1)$ is between 0 and 1. Hence, introducing more tables increase the

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات