Surveying Window NT Security and Cryptographic Hash Generation Technologies

[1]Priya Sharma, [2]Gurdev Singh
[1,2]Dept. of Computer Science & Applications, Jind Institute of Engg. and Tech., KU, Haryana, India

Abstract
Breaking encrypted passwords has been of interest to hackers for a long time, and protecting them has always been one of the biggest security problems operating systems have faced, with Microsoft's Windows being no exception. Due to errors in the design of the password encryption scheme, especially in the Lan Man(LM) scheme, Windows has a bad track in this field of information security. Especially in the last couple of years, where the outdated DES encryption algorithm that LanMan is based on faced more and more processing power in the average household, combined with ever increasing harddisk size, made it crystal clear that LanMan nowadays is not just outdated, but even antiquated. This paper surveys various flaws in window NT environment security.

Keywords
Cryptography, Window Operating Systems, Rainbow Tables, Hash generation, Algorithms.

I. Introduction

LAN Manager [1], or LM, is an authentication protocol designed (at its time) to maximize password security in a Windows-based environment. The LM protocol was first used in Microsoft's LAN Manager Product a very long time ago and is still the authentication protocol of choice for older operating systems, such as Windows 95 and Windows NT 3.51 and earlier. Later, when Windows NT was introduced, LM was enhanced and renamed the NTLM [2] authentication protocol. Although NTLM has been around for a long time, it's still a basically good authentication protocol, and it is the native network authentication protocol of Windows NT 4.0 and earlier operating systems.

NTLM MAJOR Weaknesses[3]

1. SAM has several vulnerabilities, which allowed attackers to access the hashed passwords.
2. NTLM can use a maximum of 14 characters to create its stored hash. These 14 characters are split into two seven-character strings. Crypto-graphically, it is reasonably easy to brute force attack[8] two seven-character strings with modern computers.
3. NTLM cannot use lowercase letters. It converts all lowercase letters to uppercase before creating the hash. This reduces the character set for the password, making brute force attacks far more likely to succeed.
4. The hash algorithm used to store passwords became well known. That allowed attackers to guess users' passwords by running password guesses through the hash until the result matched the result stored in the SAM. Because the algorithm remained constant, large libraries of hashed passwords could be stored and used to quickly attack a SAM.
5. NTLM used a mechanism known as pass-through authentication to distribute the authentication task. The way pass-through authentication was designed created a bottleneck at the primary domain controller (PDC) of each domain. Some of the tasks done by the PDC, such as password changes, could not be offloaded to any other server.
6. Attackers began accessing passwords by pretending to be trusted servers. Users' client computers would transmit logon information to the attackers, thinking that they were domain controllers or file servers. NTLM provided no way for users to verify that the server they were connecting to be the one they intended to connect to.
7. NTLM was largely limited to interoperability with Microsoft products. As computer networks became more heterogeneous, NTLM didn't provide a way to interoperate with non-Microsoft operating systems.
8. NTLM provided no way for a middle-tier application to access resources on a user's behalf. When a user's client application accessed a middle-tier application, the middle-tier application usually used a generic administrator credential to access backend resources. This technique works, but presents a security threat, because the middle-tier application is running under powerful security credentials.

II. Rainbow Tables

A rainbow table [5] is a way of doing cryptanalysis very quickly and efficiently. Suppose that you are a hacker and you have acquired a database of usernames and encrypted passwords. The System encodes the password using a hash function, which is basically a way of condensing a given set of data into a condensed string. For example, the MD5 algorithm encrypts password "MyPassword" as 48503dfd58720bd5f-f35c102065a52d7 if one, as a hacker, have the password described above, one wouldn't know what the password is just by looking.

Instead, one would refer to the rainbow table for the password. Rainbow tables are a pre-computation based approach to reversing hashes. They require a large amount of pre-computation, but can store the results of this in a reasonable amount of space. When searching for a hash, additional computation is required, but the computation required for searching is significantly less than the amount required for the pre-computation, and significantly less than the amount required to brute force [3] a password.

By generating long chains of passwords and hashes, tied together by the hash function and a reduction function, rainbow tables store a compressed representation of a password search space. By performing similar computations on a provided hash, they are able to dramatically reduce the amount of computation required to find the original password. As with many algorithms, there are limitations with rainbow tables. Unlike a brute force algorithm, they are not guaranteed to find a password within the search space, as the algorithm is probabilistic in the coverage of the password space, and a password will only be found if it is represented in the generated tables. However, very high success probabilities can be achieved, and the search time is significantly less than with a brute force algorithm.

The crack time/storage space tradeoff of rainbow tables is adjusted by changing the chain length. Longer chains require less storage space, but require more computation (and more time) to crack passwords.

III. Related Works

MARTIN E. HELLMAN, [4], in "A Cryptanalytic Time - Memory Trade-Off", describes that A probabilistic method is presented which crypt analyzes any N key cryptosystem in $N^{2/3}$ operations with $N^{2/3}$ words of memory (average values) after a precomputation which requires N operations. If the precomputation can be performed in a reasonable time period (e.g. several years), the additional computation required recovering each key compares very favorably with the N operations required by an exhaustive search and the N words of memory required by table lookup. When applied to the Data Encryption Standard (DES) used in block mode. It Indicate that solutions should cost between $1 and $100 each. The method Works in a chosen plain text attack and, if cipher block chaining is not used, can also be used in a cipher text-only attack.

The time-memory trade-off was described for use with a block cipher, but the same approach works with a synchronous stream cipher. The first k bits of key stream are taken as the f (K) function, where k is the number of bits of key. This can be done under a known plaintext attack. The method works on all systems in a chosen plaintext attack but does not work with a known plaintext attack on a cipher feedback system if the initial load of the shift register is random and varies between conversations.

Proposed Federal standards suggest this precaution. Even a block cipher can foil the time-memory trade-off in a known plaintext attack through cipher block chaining or other techniques which introduce memory into the encipherment. Then, even when eight blanks occur in the plaintext, their encipherment depends on the preceding text. Even if the first block of text is fairly standard (e.g., "Login: "), this technique can be foiled by the transmission of a random "indicator" which is used to affect the encipherment (e.g., it is taken as the 0th plaintext block).

Again, proposed standards include provision for cipher block chaining with a random indicator. While this time-memory trade-off cryptanalytic technique can be easily foiled, it does work on the DES in basic block mode, more importantly; it indicates that even when cipher block chaining or other techniques are added, a larger key size is needed to have a reasonable assurance of security.

While table lookup and exhaustive search are currently infeasible on systems with 64-bit or larger key sizes, an $N^{1/2}$ time-memory trade-off would push the minimum usable key size up to 128 bits. The $N^{2/3}$ technique described here, coupled with the large number of $N^{1/2}$ time-memory tradeoffs known for other searching problems, indicates that valuable data should not be entrusted to a device with smaller key size.

Philippe Oechslin, [5], in "Making a Faster Cryptanalytic Time-Memory Trade Off", describes that In 1980 Martin Hellman described a cryptanalytic time-memory trade-off which reduces the time of cryptanalysis by using precalculated data stored in memory. This technique was improved by Rivest before 1982 with the introduction of distinguished points which drastically reduces the number of memory lookups during cryptanalysis. This improved technique has been studied extensively but no new optimizations have been published ever since. The Authors proposed a new way of precalculating the data which reduces by two the number of calculations needed during cryptanalysis. Moreover, since the method does not make use of distinguished points, it reduces the overhead due to the variable chain length, which again significantly reduces the number of calculations. As an example, the authors have implemented an attack on MS-Windows password hashes. Using 1.4GB of data Attacker can crack 99.9% of all alphanumerical passwords hashes (237) in 13.6 seconds whereas it takes 101 seconds with the current approach using distinguished points.

The Authors showed that the gain could be even much higher depending on the parameters used and they have introduced a new way of generating precomputed data in Hellman's original cryptanalytic time-memory trade-off. Our optimization has the same property as the use of distinguished points, namely that it reduces the number of table look-ups by a factor which is equal to the length of the chains. For an equivalent success rate our method reduces the number of calculations needed for cryptanalysis by a factor of two against the original method and by an even more important factor (12 in our experiment) against distinguished points.

The Authors showed that the reason for this extra gain is the variable length of chains that are delimited by distinguished points which results in more false alarms and more overhead per false alarm.

They conjecture that with different parameters the gain could be even much larger than the factor of 12 found in our experiment. These facts make our method a very attractive replacement for the original method improved with distinguished points.

The fact that their method yields chains that have a constant length also greatly simplifies the analysis of the method as compared to variable length chains using distinguished points. It also avoids the extra precalculation effort which occurs when variable length chains have to be discarded because they have an inappropriate length or contain a loop. Constant length could even prove to be advantageous for hardware implementations.

Finally their experiment has demonstrated that the time-memory trade-off allows anybody owning a modern personal computer to break cryptographic systems which were believed to be secure when implemented years ago and which are still in use today. This goes to demonstrate the importance of phasing out old cryptographic systems when better systems exist to replace them. In particular, since memory has the same importance as processing speed for this type of attack, typical workstations benefit doubly from the progress of technology.

Zhenqi Li et al [5] In this paper, They present a rigorous evaluation of Thing and Ying's attack (TY attack) along with practical implementations. They find that the cryptanalysis time of their attack is too high to be practical. They also propose a more general time memory trade-off by combining the distinguished points strategy with TY attack. Both theoretical analysis and experimental results show that our new design can save about 53.7% cryptanalysis time compared to TY attack and can reduce about 35.2% storage requirement compared to the original rainbow attack.

Hans Hedbom, et al [6], in "A Comparison of the Security of Windows NT and UNIX", describes that This paper presents a brief comparison of two operating systems, Windows NT and UNIX. The comparison covers two different aspects. First, we compare the main security features of the two operating systems and then we make a comparison of a selection of vulnerabilities most of which we know have been used for making real intrusions.

The Authors found that Windows NT has slightly more rigorous security features than "standard" UNIX but the two systems display similar vulnerabilities. The conclusion is that there are no significant differences in the "real" level of security between these systems.

This paper demonstrates that the security mechanisms of Windows NT are slightly better than those of UNIX. Despite this fact the

two systems display a similar set of vulnerabilities. This implies that Windows NT has the theoretical capacity of being more secure than "standard" UNIX. However, with the present way of installing and using the system there seems to be no significant difference between their security levels. It is true that there are presently more intrusions in UNIX systems, but the authors believed that this is due to the aging factor, i.e. the statement above should hold when comparing the systems at the same state of development and market penetration.

Thus, the only reason for more UNIX penetrations is that the system is older and more well-known and we should anticipate an increasing number of intrusions into Windows NT, a tendency that has already started. It is clear that the Achilles heel of both systems is networking. Since both systems utilize the same low level protocols, i.e. IP, TCP and UDP, and comparable high level protocols. This could to some extent explain that the security behavior of both systems is similar, but it does not provide the full explanation. However, as long as the networking is such a weak point, the usefulness of other security mechanisms is diminished.

Jorgen Blakstad, et al [7], in "All in a day's work: Password cracking for the rest of us", Describes that the majority of computer systems are still protected primarily with a

Username and password, and many users employ the same password on multiple systems. Additionally, some of the most popular operating systems such as Windows XP, Windows Vista and the upcoming Windows 8, still use ad-hoc constructed hash functions such as LM, while many Linux variants use the hash function MD5.

This paper describes an experiment where we have tested the strength of a selection of passwords when converted to LM, NT and MD5 hashes, respectively, using commonly available tools. Our conclusion is that a large number of passwords can be cracked within a normal working day, and that all LM hash passwords can be recovered easily. The use of such weak hash functions in the process of user authentication in these operating systems poses a significant threat to an organization's security.

IV. Conclusion

The main benefit of Rainbow Tables is that while the actual creation of the rainbow tables takes much more time than cracking a single hash, after they are generated you can use the tables over and over again. once you have generated the Rainbow Tables, Attacks using is faster than brute force attacks and needs less memory than full dictionary attacks.

In this paper we have reviewed some of the most important works in rainbow table generation and using rainbow tables in window NT environment, i.e. against NTLM. We have discuss how NTLM is weak against rainbow table attacks.

Future Scope

Rainbow Tables are popular with a particularly weak password algorithms such as Microsoft LM and NTLM hash, these password algorithm was used in earlier days of Windows and still lives on only for compatibility reasons.

In the future We want to devise an experiment where we will test the strength of a selection of passwords using commonly available tools. This is to show that a large number of passwords can be cracked within working days and Majority of passwords used commonly have very skewed frequency distributions. We want to device methodologies based upon calculation of frequency distribution algorithm.

References

[1] Wood, Peter. "In search of the uncrackable Windows password." Network Security 2006, no. 11 (2006): 12-13.

[2] Eric Glass, "The NTLM Authentication Protocol and Security Support Provider", 2006.

[3] Mike Danseglio, "Securing Windows Server 2003",O'Reilly Media, Vol-3, p. 171-180, November 2004

[4] Hellman martin e, fellow, IEEE, "a cryptanalytic time memory trade-off". IEEE, transactions on information theory, vol. It-26, no. 4

[5] Li, Zhenqi, Yao Lu, Wenhao Wang, Bin Zhang, and Dongdai Lin. "A New Variant of Time Memory Trade-Off on the Improvement of Thing and Ying's Attack." In Information and Communications Security, pp. 311-320. Springer Berlin Heidelberg, 2012.

[6] Hans Hedbom, Stefan Lindskog, Stefan Axelsson1 and Erland Jonsson1, "A Comparison of the Security of Windows NT and UNIX", 1. Dept of Computer Engineering Chalmers University of Technology S-412 96 Goteborg, SWEDEN, 2. Dept of Computer Science University of Karlstad S-651 88 Karlstad, SWEDEN, Third Nordic Workshop on Secure IT Systems, NORDSEC'98, Trondheim, Norway,5-6 November, 1998,.

[7] Jorgen Blakstad, Rune Walsø Nergård, Martin Gilje Jaatun, Danilo Gligoroski,"All in a day's work: Password cracking for the rest of us". ITEM, NTNU. SINTEF ICT.

[8] William Stallings. 2010. Cryptography and Network Security: Principles and Practice (5th ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.