Reducing Decryption Time of NTLM Hash Using Rainbow Tables

Priya Sharma (Department of Computer Science &Engineering , JIET Jind)

Asst. Prof. Gurdev Singh (Department of Computer Science&Engineering ,JIETJind)

Kurukshetra University, Kurukshetra.

Abstract:*The majority of computer systems are still protected primarily with a user name and password, and many users employ the same password on multiple systems.Using passwords and password management routines for giving access rights is a technique that is as quite old. 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.Moreover, the recent use of Graphics Processing Unit (GPU) in high-performance servers is changing this trend.We want to devise an experiment where we will test the strength of a selection of passwords when converted to LM, NT and NTLMv2, respectively, using commonly available tools. This is to show that a large number of passwords can be cracked within working days, and that all LM hash passwords can be recovered*.

Keywords:RainbowTable ,Hash Generation ,Frequency Distribution Algorithm

1. INTRODUCTION

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 per-computation in a reasonable amount of space. When looking 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 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[1].

The majority of modern authentication systems use one-way hash functions to store passwords. This allows a hash representation of the password to be stored without storing the actual plain text password.. As hash functions are often used in cryptography and for verifying file integrity during and after transfers, hash functions have the additional property of being very fast[2].

1.1 How Rainbow Table Works

Rainbowtable[3] used for doing cryptanalysis very quickly and efficiently. Suppose if we are a hacker and we have get access to database of usernames and encrypted passwords. We can't log on to the user's account directly because to do that, we need to know his password in clear text.

Today web portals do not store passwords in their back end databases in clear text. Instead, they encode the password using a hash functions such as MD5[4], which is basically a way of condensing a given set of data into a condensed string. For example, the NTLM algorithm encrypts password "MyPassword" for the user terry@example.com

as 48503dfd58720bd5ff35c102065a52d7.

If a hacker has access to the password above, he wouldn't know what the password is just by looking at it. Instead, he'd refer to rainbow table. One *could* just take an MD5 or NTLM hash of every word in the English language, waiting until he has a hit. But that would take forever. Besides, he would also have to hash substitute letters like "MyP@ssword".

Or, one could do a little bit of work ahead of time and pre-compute every word in the English language and sort the results, then store them in a database. That way, the hash that we acquire we could simply look up in our table. Since you've already pre-sorted the database (DB), this lookup is pretty quick. But that also takes a lot of processing power and takes up a lot of disk space.Another optimization is to create a rainbow table. A rainbow table (RT) works by efficiently compressing a table of pre-computed hashes. One still have to do a lot of pre-computing of the hashes, but it doesn't take nearly as much disk space to store everything

1.2 Rainbow Table Analysis

Rainbow tables work by taking a hash of a string of text, and then "reducing" the hash to create a new string, and then reducing string of text again. There are various ways to reduce a string of text, but let's suppose we want the first 8 characters because that's what many passwords are.

A chain in a rainbow table starts with an random plain text, hashes it and reduces the hash to another plain text and hashes the new plain text, and so on. The rainbow table only stores the starting plaintext, and the final hash we choose to end with, and so a chain "containing" millions of hashes can be represented with only a single starting plain text and a finishing hash.

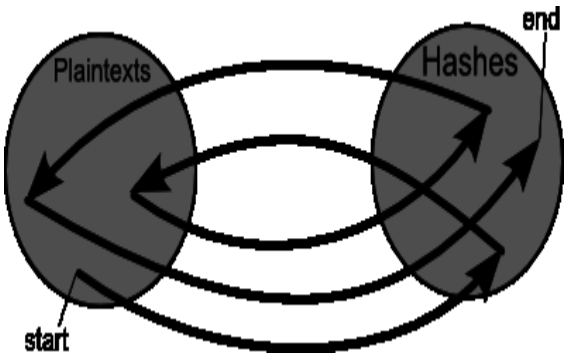

Fig 1 Rainbow Table stores starting Plaintext and Ending Hash

2. Present Work

The data structure with which it is possible to retrieve the key used to generate a cipher text with a fixed plaintext is called Rainbow table. One common application is to reverse (password) hashes; which can be seen as cipher functions who take the message as the key and the cipher text as the generated hash.

Here , we generate hashes of text using Hash function then "reducing" the hash to create a new string, and then reducing string of text again.One important thing to note is that all keys and cipher text pairs stored in the table have been pre-computed during generation of the table.

After surveying the literature we have found that Rainbow tables are best suitable for decrypting passwords with minimum time and less storage. In our work we use an algorithm named Frequency Distribution algorithm.With the help of which we assign different frequencies to the characters. These characters may include alphabet, numeric, alphanumeric sets. According to the frequencies of occurrence of every character we eliminate least occurrence characters from our character set , which lead to reduction of time and space in rainbow table.

3. Frequency Distribution Algorithm

Frequency distribution is calculated for each character in the password. After applying frequency distribution we perform the character class reduction by subtracting the characters from the passwords list that are not used mostly. Based on reduced char set we generate hashes for NTLM. After generating NTLM hashes we create rainbow table for the reduced char set class.

Algorithm for generating Rainbow Tables:

Ds: =Load data _set "**The data set contains list of most commonly used user passwords**"

Cs: =Load character _classes "**The character classes contains list of characters like a……z, A……Z.**

For-each pass **in** data _set **do//calculate frequency of each character in the dataset w.r.t character class.**

For-eachchin pass **do**

Ifchexist incsthen

Update_character_frequency ()

Else

Change character class

End

End

End

Following is a flowchart for generation of NTLM Rainbow Tables using Rainbow tables, the complete process starts form password collection

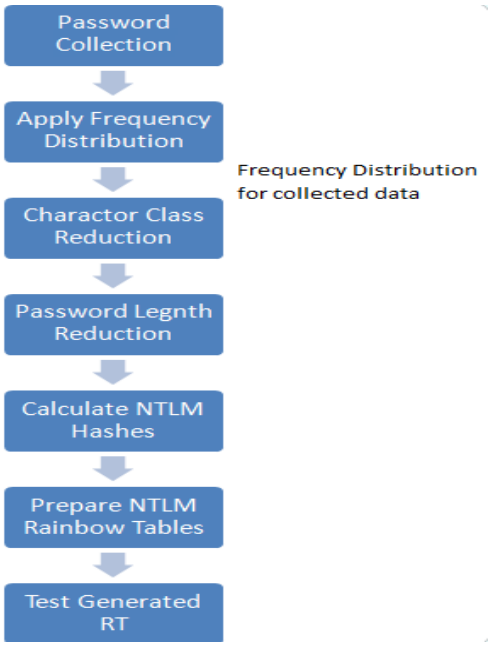

Fig 2 : Generation of NTLM Rainbow Tables using Frequency Distribution

After applying frequency distribution algorithm we check occurance of character in character set . the character with highest frequency of occurance will be given priority in the table. The figure below shows

the frequency of characters under Alpha character set

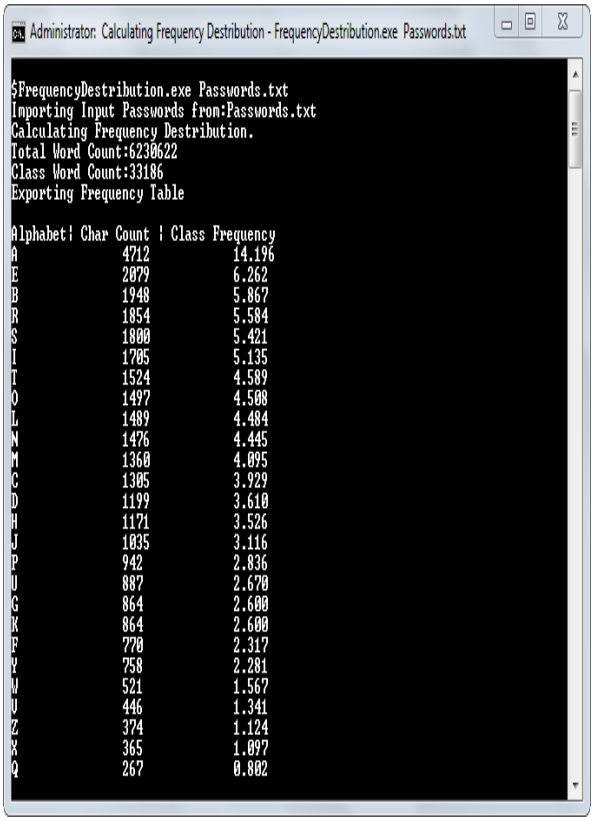

Fig 3: Calculating Frequency Distribution ofPasswords under Alpha Class

After calculating frequencies of characters in specified character set we plot graphs which shows their exact probability of occurance .plotting of graphs is in such a way that we get the exact output of characters chances of occurance.

The figure below , shows the frequency of occurance of characters in a character set Alpha for creating passwords.

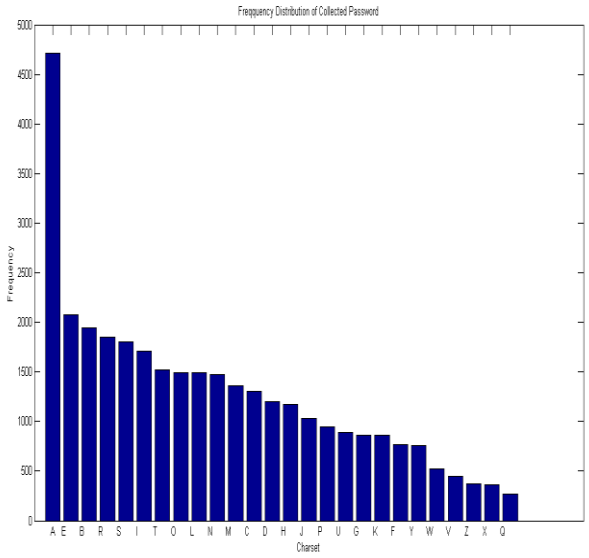

Fig 4: Frequency Distribution of Most Common Passwords in Alpha Range

When we get the frequencies of characters on behalf of bar graph , we eliminate the least frequently occurred characters from the character set in rainbow table by creating a

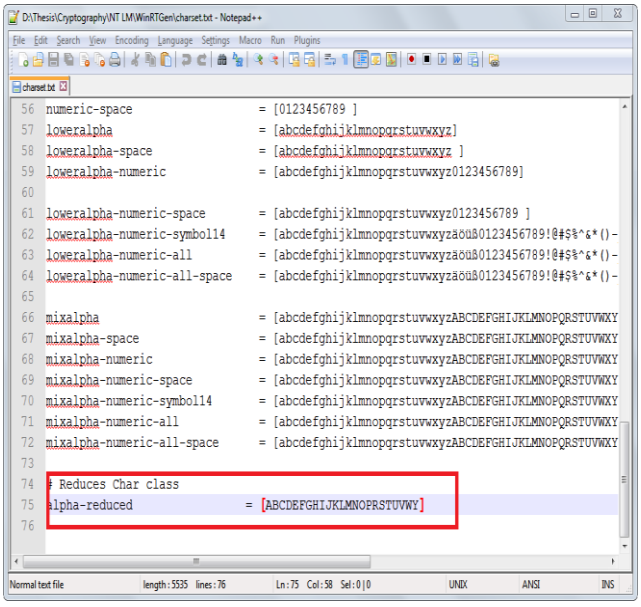

Fig 5: Adding a New in char(Reduced) in WinRT Gen Tables.

anew character set named as reduced character class of old class .fig 5 shows the addition of reduced class named alpha reduced class which doesn't contain the characters with least frequency. Which cause decrease in cryptanalysis time called decryption time of passwords and disk space. The fig 6 below shows decreased cryptanalysis time of alpha reduced class.

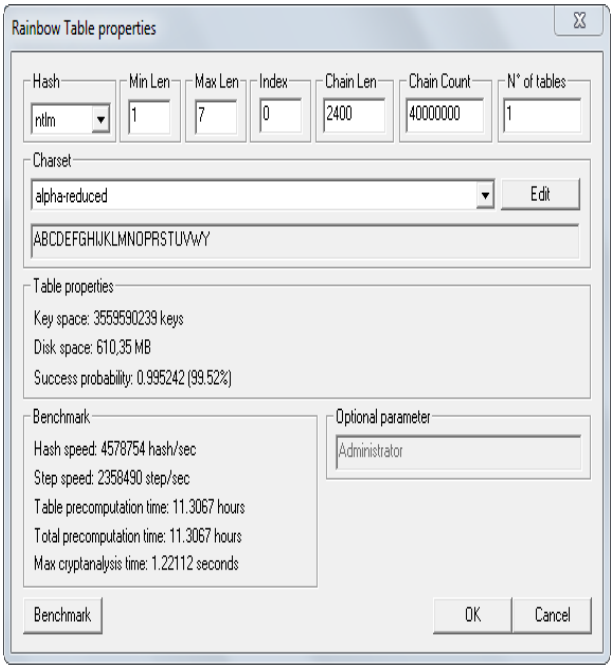

Fig 6: Cryptanalysis Time for NTLM using Reduced Char set

4. Conclusion

Many cryptanalytic problems can be solved in theory using an exhaustive search in the key space, but are still hard to solve in practice because each new instance of the problem requires restarting the process from scratch. The basic idea of a time memory trade offs is to carry out an exhaustive search once for all such that following instances of the problem become easier to solve.

We have shown that majority of password hashes can be decrypted in real time usingRainbow tables and commonly available tools such as "Rainbow table Generator" namely WinRTGen.

The Calculation of Frequency distribution led to increased time memory trade-offs as used in this report and faster decryption of NTLM passwords. Finally, we introduced a Frequency Distribution based reduction on rainbow tables that aims at reducing the time in calculation. We have shown that this technique has a real impact in practice.

5. Future Scope

Here we would like to describe in what direction our future work will be headed:

- Appling these techniques for generating other much more secure Hash Algorithms such as MD5, RSA, and SHA (SHA-1 or SHA-2).
- The lack of salt usage, however, brought a bigger vulnerability for NTLM Hashes. Rainbow Tables based crackers can find simple passwords in less than one hour with a successful rate of more than 99%, Hence adding Hash salts to NTLM Protocol could drastically increase its security.
- Using our techniques in Network domain with multiple users online to access security[5].
- Creation of new hashing algorithm for windows based upon much more secure hashing algorithms such as SHA[6,7].

6. References

[1] Oechslin, Philippe. "Making a faster cryptanalytic time-memory trade-off." In*Advances in Cryptology-CRYPTO 2003*, pp. 617-630. Springer Berlin Heidelberg, 2003.

[2] Avoine, Gildas, Pascal Junod, and Philippe Oechslin. "Characterization and improvement of time-memory trade-off based on perfect tables." *ACM Transactions on Information and System Security (TISSEC)* 11, no. 4 (2008): 17.

[3] 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.

[4] Ah Kioon, Mary Cindy, Zhao Shun Wang, and Shubra Deb Das. "Security Analysis of MD5 algorithm in Password Storage." *Applied Mechanics and Materials* 347 (2013): 2706-2711

[5] Gold, Steve. "Cracking wireless networks." *Network Security* 2011, no. 11 (2011): 14-18.

[6] Wood, Peter. "The hacker's top five routes into the network (and how to block them)." *Network Security* 2006, no. 2 (2006): 5-9.

[7] Theocharoulis, Kostas, IoannisssPapaefstathiou, and CharalamposManifavas. "Implementing rainbow tables in high-end fpgas for super-fast password cracking." In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pp. 145-150. IEEE, 2010.

