



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials Bootcamp Style (Security 401)"
at <http://www.giac.org/registration/gsec>

GIAC (GSEC) Gold Certification

GIAC (GSEC) Gold Certification

Author: Tam Weng Seng, wengsentam@gmail.com

Advisor: Lori Homsher

Accepted: May 06th 2009

Abstract

There are many "best practices" on password policies. For example, FIPS Publication 112 has a requirement that a password expire every 30 days, with a maximum life time of 1 year. This paper will propose that "Password Aging" is not a statistically effective preventive measure. To prove this argument, this paper will show that strong passwords do not benefit from password aging. On the other extreme, weak passwords would require an impractically short password aging duration to be effective. To conclude the paper, I would like to propose that further research should be done in the realm of detection and reaction.

1. Introduction

This paper is inspired by a ";login:" article by Mike Howard titled, "how often should you change your password?" (Howard) The purpose of this paper is to examine several scenarios, from a probabilistic approach, to show the effect of password aging on offline and online brute force password compromising attacks.

To achieve this objective, the paper is divided into three sections. The paper begins with an examination of some of the attacks that can be used against a password. These attacks are categorized into three groups – Social Engineering Attacks, Password Capture Attacks and Brute Force Attacks. Having done a quick review of the possible type of attacks, the next section describes rainbow tables. The focus on rainbow tables is to highlight the speed and reliability of rainbow tables in uncovering a password plaintext based on the password hash. A brief discussion of the formulas used in this paper concludes the first section.

The analysis section consists of two sets of data. The first set is to show the rate of change in the probability of a successful attack against a truly random password over time from 30 to 360 days. A second set of graphs will show the rate of change against the estimated entropy calculated by NIST (Burr, Dodson & Polk). This data will show that an online attack has such a low probability of success, which does not improve significantly over time.

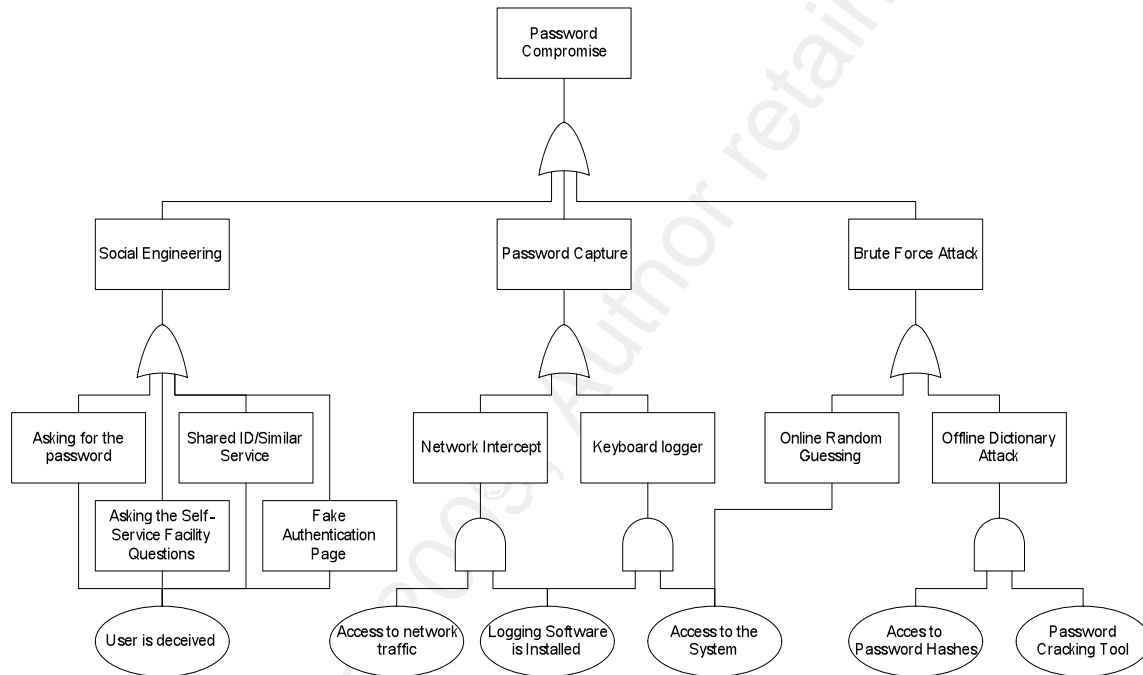
A second set of data is included with regards to building rainbow tables. It examines the power or limits of rainbow tables by examining the key ingredient to rainbow tables. This key ingredient, which is the generation of the rainbow table, is the most crucial step because only passwords that are in the table can be cracked. Therefore, the information from this set of data is to examine what types of passwords can be attacked with a significant probability of success, given different levels of computational power dedicated to the task for a year.

The last section of the paper is dedicated to the analysis of the numbers and to draw some conclusions. In replacement of static password aging, anomaly detection and

Author Name, email@address

tripwire accounts may be implemented, on top of strong passwords. These automated and manual processes can help detect when it might be prudent to force all the users to change their passwords on the system, rather than setting an arbitrary number of days for forced password resets.

1.1. The Attacks



In this paper, the vulnerabilities targeted to compromise passwords are categorized into three types. The first category is the Social Engineering attack. For this attack to be successful, an attacker must obtain some information that will allow building a deception to trick the user into giving up their password.

A second category of attack is the password capture attack. This family of attacks can be carried out at the client or the server, or in the connecting medium in between. It requires access to either the network or one of the end hosts. Logging software has to be put in place and may require the use of a social engineering attack. If this attack is successful plaintext passwords can be captured as they traverse the network. On the end hosts, the passwords can be captured as they are entered.

A third category of attack is the “Brute Force Attack” which has two types of attacks. The online attack requires that the attack has access to the system to randomly

Author Name, email@address

enter a password into the system. This attack is very time consuming and, without additional knowledge beyond the user name, is unlikely to succeed against a complex password. This statement will be justified by statistical analysis later in this paper. The other attack in this category is the offline dictionary attack. For this attack to be done, the attacker must have access to the password hashes that represent a password. Without access to these hashes, which are often only readable by an administrator account, a password cracking tool has nothing to compare. However, assuming that the password hashes have been compromised and made available to an attacker, the attacker will require a password cracking tool.

In the beginning, password cracking tools, such as John the Ripper, Crack or LophCrack would often compute the password hashes on the fly, based on a dictionary or a set of rules, and compare them to a list of hashes that were to be attacked. A paper by Morris and Thompson describes this approach and makes some recommendation to counter this type of attack (Morris & Thompson). To speed up this process, well known passwords can be pre-computed. This leads to the concept of a space-time trade off. The more passwords that are pre-computed, the faster the entire dictionary of words can be checked at the time of the attack. However, more memory and disk space may be required to store the pre-computed hashes. A one off processing time is also required to pre-compute the passwords. This space-time trade off can be seen as a continuum where in one extreme case all the possible passwords are pre-computed and stored on disk. On the other extreme, all the possible plaintext passwords are hashed and compared to the list of password hashes at run time.

In theory, given a small enough password space and sufficient disk space, all the possible passwords can be pre-computed, making access to the password hashes equivalent to having access to the unencrypted password. Fortunately, in this case, the numbers favor the defender. Increasing the maximum and minimum password length increases the size of the search space exponentially. This forces attackers to use more intelligent heuristics to select the most likely candidates, as an exhaustive dictionary attack of all password hashes becomes less feasible (Narayanan & Shmatikov).

Author Name, email@address

1.2. The Search Space

As a foundation for the rest of this paper, this section will highlight the enormous search space that even the most trivial 8 character, lower case alphanumeric password can generate. The formula for calculating this search space is the number of unique valid characters in the password raised to the power of the password length. If the character set size is c and the password length is l , then the search space is c^l .

| | 36 characters | 62 characters | 95 characters |
|----|---------------|---------------|---------------|
| 8 | 2.82111E+12 | 2.1834E+14 | 6.6342E+15 |
| 12 | 4.73838E+18 | 3.22627E+21 | 5.4036E+23 |
| 16 | 7.95866E+24 | 4.76724E+28 | 4.40127E+31 |

With the formulas in place, we can see that even a small character set and password length generates a huge number of possible

passwords. On the other hand, weak passwords continue to be a problem. This is evident from the success of simple dictionary attacks seen in the recent family of conficker viruses (Porras, Saidi & Yegneswaran). The idea that user generated passwords tend to be weak is also reflected by research done by NIST (Burr, Dodson & Polk).

According to NIST, a user-chosen password can be estimated based on Claude Shannon concept of Entropy. In Table A.1 of the document by NIST, they estimate that an 8 character password, without checks, is estimated to have entropy of 18 bits. Dictionary rules increase it to 24 bits and additional composition rules could bring it up to 30 bits of entropy. With 16 character passwords, they estimate that a user would pick a password with about 30 bits of entropy. It is estimated to have 32 bits of entropy with dictionary rules and 36 bit of entropy with dictionary and composition rules (Burr, Dodson & Polk). This is a startling contrast and a drastic theoretical reduction to the search space, as a truly random 1 to 8 character long password, built on a 32 character set ($2^5 = 32$ or 5 bits of entropy per character), should have 40 bits of entropy (8 characters * 5 bit/character). With a 64 character set, there should be 48 bits of entropy. Meanwhile, a truly random 1-16 character long password, made up of 32 possible characters, should have 80 bits of entropy.

Having said that, these numbers are highly theoretical and inaccurate as estimating the choice of passwords is not an exact science. It is also unlikely that an

Author Name, email@address

attacker could generate a perfect dictionary containing only the passwords that a user might generate. Nevertheless, these ball park figures can be used to estimate the risk of password compromise, and specifically to estimate the potential impact of weak passwords on security.

1.3. The variables

Before any analysis can be done, we must first define the variables for the analysis. The next step is to specify the formulas that will be applied to the variables. By applying the formulas, objective comparisons and analysis can be made for our conclusions.

For the purpose of this paper, we examine passwords with a length of 8, 12, and 16 characters. We examine a character set consisting of a case insensitive alphanumeric character set, case sensitive alphanumeric character set and a case sensitive alphanumeric character set with 32 punctuation or special characters and the space character makes 33.

For the purpose of introducing time into the study, we will examine the probability of a successful attack against a password age after 30 days of nonstop attempts. Additional points of data will be obtained with increments of 30 days, up to 360 days. These numbers were chosen because FIPS Publication 112 starts with 30 days and recommends that it does not exceed 1 year. The size of the increment was arbitrarily chosen in order to create a suitable number of sample points to establish a trend over time.

In addition to the time frame, we will assume that an attack has the ability to maintain a nonstop attack starting with 2 passwords per second. The next sample points will be determined by exponentially multiplying the figure by 2^{10} . In this way, this variable will start with 1 and end at 2^{70} . While it might not be very practical, or even possible, to perform such a large sustained brute force attack, the number 2^{70} was chosen to show how much computing power is required to brute force a truly random 12 character password that is built from a set of 95 valid symbols.

For the second set of data on rainbow tables, the assumptions in this paper are based on numbers found on the websites offering rainbow tables (freerainbowtables,

Author Name, email@address

project-rainbowcrack, & ophcrack). As a simplification, we will assume that disk space is not an issue, as the chain length could theoretically be increased to improve on the savings in disk space at the expense of lengthening the time required to search the rainbow table.

To estimate the size of a rainbow table that can be built in 360 days, we will examine the results of building approximately 10,000 links/second and 1 billion links/second. This estimate, of 1.2 billion links/second, is based on the statistics section found the home page on freerainbowtables.com. This large amount of computation is achieved by distributing the task across approximately 1600 registered hosts, of which a significant number of the machines might not be active at any one time. Having a reasonable estimate on the number of chains that can be generated in a year, the probability of success for 1, 1 to 2, 1 to 3, and going up to 1 to 16 character passwords can be calculated.

1.4. Rainbow Tables

While the other forms of attacks should be reasonable well understood by a security audience, rainbow tables are a specific type of brute force attack that has interesting consequences to the discussion on password aging and the security of password hashes.

The theory behind rainbows tables is actually quite complex but we do not really require a detailed understanding of how it works or how to derive the formulas to calculate the effectiveness of the algorithm. The important thing to consider is that even small character sets and short passwords create a large search space that makes it impractical to store every password plaintext and password hash. Meanwhile, computing the hashes of a large dictionary of words and their variation, at runtime, takes a lot of computational power and time. This work has to be repeated for each new set of attacks.

In a paper titled, “Making a Faster Cryptanalytic Time-Memory Trade-Off” by Philippe Oechslin, a modification of a proposal by Hellman is proposed. This is further expanded in a later paper (Avoine, Pascal and Oechslin). Both the original approach and rainbow tables build chains of password/hash pairs. Starting the chain is a plaintext

Author Name, email@address

password. A reduction function takes the resulting password hash and generates another plaintext password. The method proposed by Hellman uses the same reduction function for each chain. By this method, the amount of space required to store the password plaintext and password hash pairs is greatly reduced, as an entire chain can be stored with only the starting plaintext password and the last password hash. In a chain, that is 10,000 links long, only the starting plaintext password and the last password hash value in the chain has to be stored. This means that 9,999 password plaintext/hash pairs did not have to be stored (Oechslin).

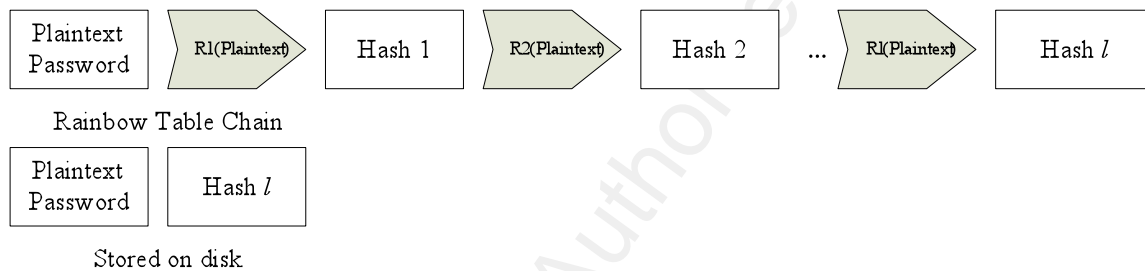


Figure 1 Rainbow tables using l reduction functions.

The modification by Oechslin is the use of multiple reduction functions in each table. Unlike the original algorithm which uses the same reduction function for each link in a table, a rainbow table uses a different function for each link in a chain. If a reduction function were a color, it would make each chain look like a rainbow. An important advantage with this modification is the reduction in probability of merges. When two chains merge, the number of unique passwords represented in the table decreases as the two chains join together. One possible analogy is to imagine two different lanes merging into one. This means that the number of unique passwords stored in the table is reduced. Therefore, the consequence of reducing mergers is that there are more unique passwords represented in the chains in the same amount of storage. This in turn improves the chance of a password being in the pre-computed table given an equal amount of storage. Meanwhile, as rainbow tables use fixed length chains, one can predict an accurate amount of storage required by knowing the size of the chain, number of tables and the number chains. One more advantage, noted by the author, is that rainbow tables can be searched completely faster than the original algorithm to determine if a password is in the table (Oechslin).

Author Name, email@address

To get an idea of how powerful this is, there are several sites which offer rainbow tables ranging from a few hundred MB to over a hundred GB in size. Many of the tables have a 99% to 99.9% probability of success against the targeted password length and character set (freerainbowtables, project-rainbowcrack, & ophcrack).

It should be noted that this attack assumes that the attacker has the password hash information. It should also be noted that a pre-computed attack will fail if the password is longer than the maximum length or includes characters not found in the character set of the pre-computed dictionary. As of May 2009, none of the following sites offer rainbow tables, with a full coverage of the 95 possible characters, for passwords greater than 8 characters in length. There exists limited characters sets for passwords of length 9 (freerainbowtables, project-rainbowcrack, & ophcrack).

2. The Analysis

The section examines two sets of data mentioned in the introduction. The first section will explore the probability of success against an online brute force attack. This analysis can also be used for brute force attacks that compute the password hashes at run time, instead of using pre-computed password hashes like rainbow tables.

The second set of data will examine the time required to generate a rainbow table. The reason for choosing to analyze the time required to generate the tables is that the bulk of the time required by a pre-computed attack is not the time required using the tables. The bulk of time is needed to generate the tables. In fact, Oechslin estimates in his paper that the pre-calculation of the tables could be 10 times higher than generating the hashes for the entire search space (Oechslin).

With this data we can establish a theoretical maximum number of chains that can be pre-computed over a period of time. With the maximum number of chains, we can estimate the probability of success for a given rainbow table using the formulas provided by Oechslin in his 2003 paper on Rainbow tables (Oechslin).

2.1. Estimating Success for Online Brute Force Attacks

The following graphs use a base 10 logarithmic scale for the x-axis and can be used to estimate the probability of a success online brute force attack against a password. Anything below 1×10^{-2} has a probability below 1%, and is really quite negligible. To understand how these graphs were generated, let us assume that N is the key space, or number of possible valid passwords. Let us assume that l is the number of characters in the password and that c is the number of valid characters in the password. Let us assume that r is the number of passwords tested per second and that A is the number attempts over a period of time. Let the y-axis, Y , represent the time period in days and the x-axis show the probability of successfully cracking a password. With these variables established, the following formula can be stated:

$$N = c^l$$

$$Y = y * 30 \text{ (for } y \text{ ranging from 1 to 12, with increments of 1)}$$

$$A = r * 86400 * Y$$

$$X = P(\text{Success}) = \text{minimum}(A, N) / N$$

If these graphs represent an automated online brute force attack, then it assumes that the attack rate is constant for the entire duration, and it targets the password for a single user. It also assumes the attack does not trigger any action by the system to lock the account or slow the attack. With that in mind, it is unlikely that an attacker can sustain more than a few thousand attacks per second due to the limits of resources available to connect to the system. In addition, the risk of detection, by any security monitoring, increases over time and as the rate of attempts increase.

Subsequently, these numbers can also be used to estimate the probability of success for an offline attack, which generates the password hashes at run time, against a particular password. In the case of an offline brute force attack, which does not use pre-computed hashes, the probability of success depends on the number rate at which the hashes can be computed at run time. Based on benchmarks captured, using the tool John the Ripper, it make sense to think about attack rates that can exceed 2^{24} and up to 2^{30} attempts per second for cracking Windows LanMan passwords. (Mator).

Author Name, email@address

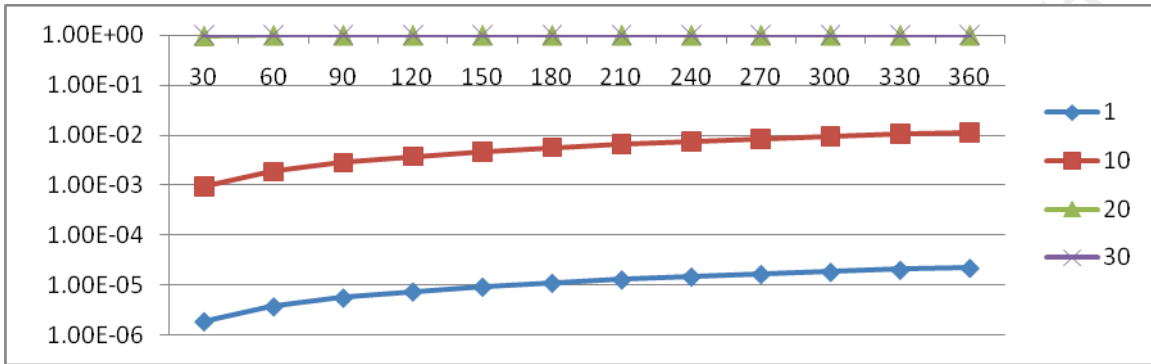


Figure 2 Probability of Success a password (8 Characaters; 36 character set)

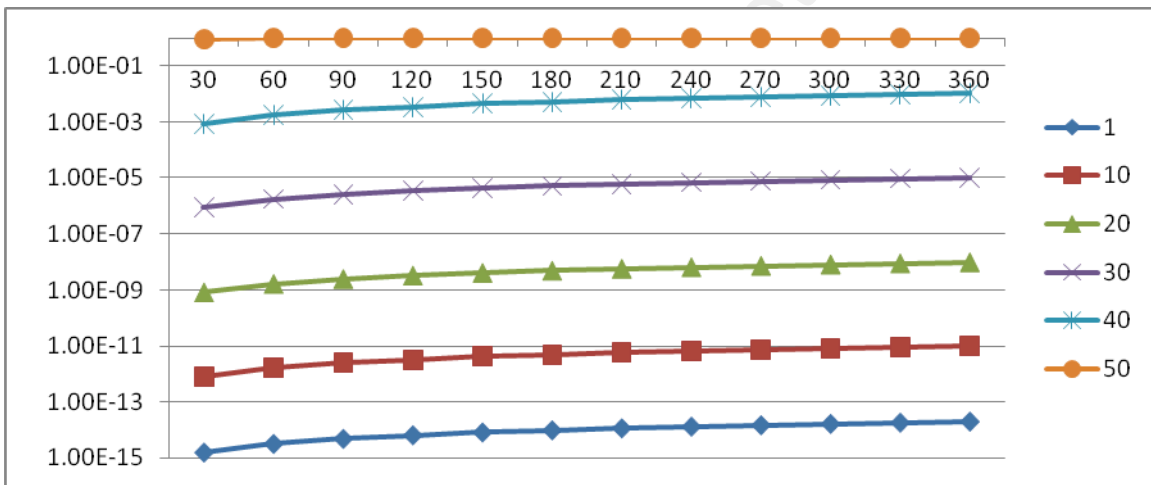


Figure 3 Probability of Success a password (12 Characaters; 52 character set)

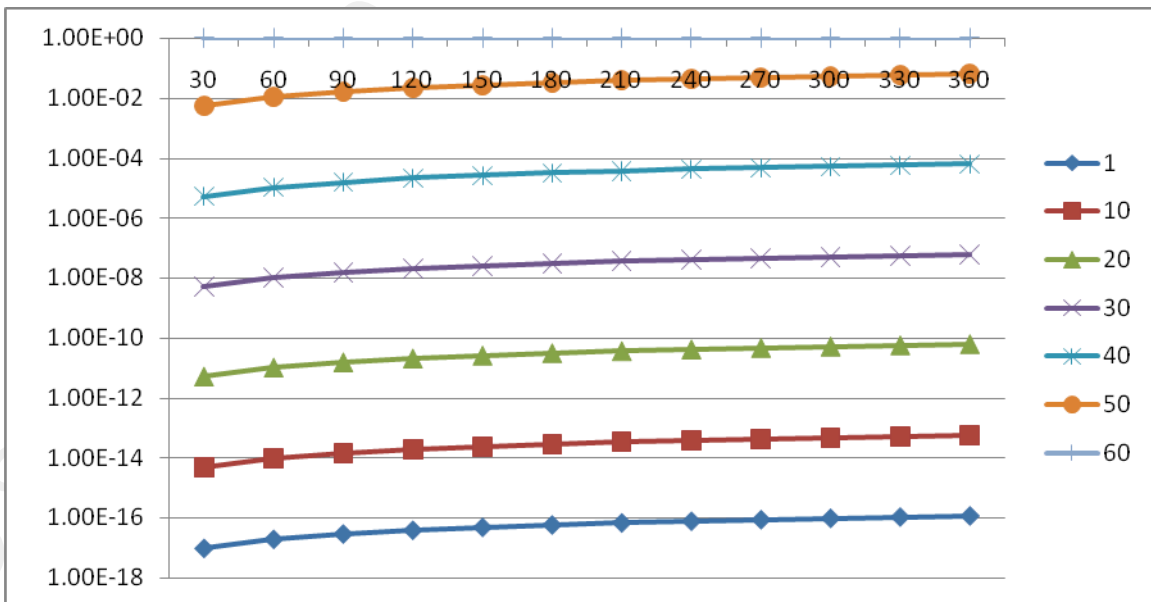


Figure 4 Probability of Success a password (12 Characaters; 95 character set)

Author Name, email@address

As you can see in **Figure 2**, the probability of success does not change significantly over time. This gradual increase means that if the initial chance of success is low, time alone does very little to improve the odds. Against a truly random alphanumeric password that is case insensitive, one would need to be able to sustain an attack of 2^{10} or 1,024 attacks a second to have a 1% chance of success in 30 days and a 2% chance after 360 days. This should not be possible for an online attack, but an attacker with a tool like John the Ripper could manage this rate of attack with help of the password hash.

With **Figure 3**, we once again see that the probabilities do not increase significantly over time. It also shows that an attacker would have to sustain an attack of at least 2^{40} to have slightly over 1% chance of success, after about 360 days, against a 12 character alphanumeric password that has a mix of upper and lower case alphabets. At this time, it should be safe to say that most attackers cannot obtain this kind of computing power. Even if it were possible, the cost of this computing power is unlikely to be available to the casual attacker. For example, a multi-million node zombie network might be able crack 2^{50} to 2^{60} passwords a second. While, a multi-million node zombie network might seem improbable, some analysis of the conficker virus suggests that there may be several million hosts infected by this worm (Porras, Saidi & Yegneswaran). In this way, a well organized group might be able to muster the required number zombie hosts.

To drive home the point, **Figure 4** shows that there is no significant change in the probability of success over 30 day increments. In addition, an attacker now requires about 2^{50} attacks per second to achieve a significant probability of success. This increase in difficulty was achieved by simply requiring users to add a few punctuation characters, symbols or space key, where permitted, to the password.

2.2. Estimating the Power of Rainbow Tables

In the paper by Oechslin, a formula is provided to calculate the approximate probability of success given l number of table with M chains of T length to find a password located in a key space of size N . For the purpose of this paper, the targeted rainbow tables will be built with 10,000 links per chain and the chains 1 table. To

Author Name, email@address

calculate the number probability of success for one table we use the formula given in his paper as:

$$P(1 \text{ table}) = 1 - e^{-T^*(M/N)}$$

For the purpose of simplification, to calculate the number of chains that can be computed in 360 days, a simple formula will be to consider the most significant activity of creating R chains/second. Thus the formula for C chains/year is:

$$\text{Number of Chains} = (R * 3600 * 360) / (10000 * 5)$$

| | | | |
|---|------------|------------|---------------|
| R | 1000 | 1,000,000 | 1,000,000,000 |
| C | 3.1104E+11 | 3.1104E+13 | 3.1104E+16 |

Table 1 Number of chains generated in a year.

When the numbers of Chains are put into a spreadsheet, we can compute results to find patterns. These results can be found in the tables in Appendix B. From these results, it can be argued that a large network of computers could pre-compute rainbow tables up to a password length of 10 characters. However, a defender only has to set the minimum password length to 12 to 16 characters to negate the use of these rainbow tables.

3. Conclusions

One of the arguments used to rationalize the need for a maximum password age is that a password should be required to be changed before there is a significant probability that it can be compromised or cracked (Summers & Bosworth).

By examining the graphs from the first set of data, we can see that a strong password is highly resistant to online guessing. Even an 8 character alphanumeric passwords would be extremely difficult to discover by brute force guessing, if the attacker only knows the login identity. This does not change significantly over time. Therefore a password that has a low probability of being discovered initially is unlikely to be discovered over time. In fact, a strong password could tolerate having the password aging value set to several years without a statistically significant increased chance of being compromised.

However, on the other extreme, weak passwords might be found in a small dictionary of commonly used passwords. In this situation, an attacker may have a high

Author Name, email@address

probability of success by going through a small dictionary every few days. Although the probability of success will not increase significantly, a weak password which happens to be in that dictionary is highly likely to be found quickly. If a computer system relies solely on password aging to force regular password changes, it would not prevent a weak password from being compromised in a day. This is because a dictionary of a few hundred passwords, like the one in the conflicker worm, could be tested in moments. If the password lies in this dictionary, the password could potentially be compromised very quickly. This means that the password aging for this system would need to be set to a very low value, under a day, to be effective in protecting a weak dictionary derived password.

Subsequently, when the attacker has managed to obtain the password hashes, the passwords become vulnerable to offline brute force attacks. Password hashes that are stored without a salt are vulnerable to offline brute force attacks using rainbow tables, unless the required password length exceeds the maximum length of passwords that have been pre-computed in readily available rainbow tables for that given hash algorithm. In fact, the use of rainbow tables make it possible to search large pre-computed dictionaries within minutes. As indicated earlier in the paper, there are pre-computed rainbow tables for many popular hashes that have a 99.9% chance of success against passwords with 1 to 7 characters, and tables with a smaller subset of the possible characters, for a password length of 8 characters (freerainbowtables, project-rainbowcrack, & ophcrack). In this event, password aging would have to be set to few minutes.

In conclusion, it should be safe to argue that password aging does not offer more protection than strong password policies and longer minimum password lengths. In fact, due to the costs and computation time required, it is reasonable to state that a strong 12 to 16 character password, reinforced by strong composition rules, can make brute force attacks prohibitively expensive, if not impossible, for the attacker. In addition, short or weak passwords are so vulnerable that password aging would have to be set as low as a few minutes to be effective. This is not practical without using an automated one-time password system.

Author Name, email@address

3.1. Further Research

It can be argued that a password is like the combination to a safe. A bank seldom entrusts their vault with only the combination; they also use automated fail safes to detect tampering and hire security to respond to the alarms and attempts to breach the vault. Using a bank as an analogy, it is reasonable to claim that a preventive control in the form of passwords is not sufficient by itself.

In fact, Summers and Bosworth recommends that passwords should be changed if there is a reason to suspect it has been compromised (Summers and Bosworth). Moreover, although brute force attacks on strong passwords are statistically unlikely to succeed, there are other types of attacks, as mentioned in this paper. Sadly, this need can only be detected if there are detective and reactive controls in place.

In the mining industry, a canary might be used to detect the buildup of dangerous gases in the mine. In computer security, a honey pot or honey net can be used to decoy attackers away from production systems and to alert defenders that an attack is in progress. In this way, it might be interesting to use models and statistics to estimate the effectiveness of decoy accounts, acting as tripwires to alert a system administrator to a possible attack.

4. Appendix

4.1. Appendix A

| | |
|--|---|
| 36 Character Set | A-Z (Case insensitive) 0-9 |
| 62 Character Set | A-Z; a-z (Mixed Case) 0-9 |
| 95 Character Set (Alphanumeric + 32 Symbols + Space) | A-Z; a-z (Mixed Case) 0-9 !@#\$%^&*()- _+=~`{[]}\ :;'"<,>./ (space) |

4.2. Appendix B

These tables contain the probability of success for 1 to l length passwords with c valid characters in the password. The number of table in the rainbow is 1 and the chain length is 10,000.

| l/c | 26 | 36 | 52 | 62 | 95 |
|-----|----------|----------|----------|----------|----------|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 0.990333 |
| 9 | 1 | 1 | 0.999983 | 0.895715 | 0.047659 |
| 10 | 1 | 0.999744 | 0.190252 | 0.035805 | 0.000514 |
| 11 | 0.999711 | 0.205267 | 0.00405 | 0.000588 | 5.41E-06 |
| 12 | 0.269044 | 0.006362 | 7.8E-05 | 9.49E-06 | 5.7E-08 |
| 13 | 0.011982 | 0.000177 | 1.5E-06 | 1.53E-07 | 6E-10 |
| 14 | 0.000464 | 4.92E-06 | 2.89E-08 | 2.47E-09 | 6.31E-12 |
| 15 | 1.78E-05 | 1.37E-07 | 5.55E-10 | 3.98E-11 | 6.64E-14 |
| 16 | 6.86E-07 | 3.8E-09 | 1.07E-11 | 6.42E-13 | 0 |

Table 2 Rainbow table generated for 360 days at 1 000 000 000 links/second

Author Name, email@address

| l/c | 26 | 36 | 52 | 62 | 95 |
|-----|------------|-------------|----------|----------|----------|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 0.999832 | 0.356422 |
| 8 | 1 | 0.999977883 | 0.434832 | 0.13078 | 0.004628 |
| 9 | 0.99594728 | 0.257516885 | 0.010914 | 0.002258 | 4.88E-05 |
| 10 | 0.19092222 | 0.008236866 | 0.000211 | 3.65E-05 | 5.14E-07 |
| 11 | 0.00811536 | 0.000229723 | 4.06E-06 | 5.88E-07 | 5.41E-09 |
| 12 | 0.00031335 | 6.38191E-06 | 7.8E-08 | 9.49E-09 | 5.7E-11 |
| 13 | 1.2054E-05 | 1.77276E-07 | 1.5E-09 | 1.53E-10 | 6E-13 |
| 14 | 4.6361E-07 | 4.92433E-09 | 2.89E-11 | 2.47E-12 | 6.33E-15 |
| 15 | 1.7831E-08 | 1.36787E-10 | 5.55E-13 | 3.97E-14 | 0 |
| 16 | 6.8582E-10 | 3.79963E-12 | 1.07E-14 | 0 | 0 |

Table 3 Rainbow table generated for 360 days at 1 000 000 links/second

| l/c | 26 | 36 | 52 | 62 | 95 |
|-----|----------|----------|----------|----------|----------|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 0.995428 | 0.342083 |
| 7 | 1 | 0.978909 | 0.256754 | 0.08323 | 0.004397 |
| 8 | 0.761211 | 0.101647 | 0.00569 | 0.001401 | 4.64E-05 |
| 9 | 0.053594 | 0.002973 | 0.00011 | 2.26E-05 | 4.88E-07 |
| 10 | 0.002116 | 8.27E-05 | 2.11E-06 | 3.65E-07 | 5.14E-09 |
| 11 | 8.15E-05 | 2.3E-06 | 4.06E-08 | 5.88E-09 | 5.41E-11 |
| 12 | 3.13E-06 | 6.38E-08 | 7.8E-10 | 9.49E-11 | 5.7E-13 |
| 13 | 1.21E-07 | 1.77E-09 | 1.5E-11 | 1.53E-12 | 6E-15 |
| 14 | 4.64E-09 | 4.92E-11 | 2.89E-13 | 2.46E-14 | 0 |
| 15 | 1.78E-10 | 1.37E-12 | 5.55E-15 | 0 | 0 |
| 16 | 6.86E-12 | 3.8E-14 | 0 | 0 | 0 |

Table 4 Rainbow table generated for 360 days at 10 000 links/second

Author Name, email@address

5. References

- Avoine, Gildas, Junod, Pascal and Oechslin, P (2008). Characterization and Improvement of Time-Memory Trade-Off Based on Perfect Tables, *ACM Transactions on Information and Systems Security*, vol 11, No. 4, Article 17,
- Burr ,William E., Dodson, Donna F., , & Polk, W. Timothy. Electronic Authentication Guideline. *NIST Special Publication 800-63*, Retrieved April 1, 2009, from http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.
- Howard, Mike (December, 2006). How Often Should You Change Password, ;*Login*., vol 31, No. 6, 48-51.
- Mator (2009, April 08). [[John the Ripper benchmarks]]. Retrieved April 14, 2009 from John the Ripper benchmarks site: <http://openwall.info/wiki/john/benchmarks>
- Morris, Robert, & Ken, Thompson (1979). Password Security: A Case History. *Communication of the ACM*. 11, 594-597.
- Narayanan, Arvind & Shmatikov, Vitaly (2005). Fast dictionary attacks on passwords using time-space tradeoff. *Proceedings of the 12th ACM Conference on Computer and Communications Security* (Alexandria, VA, USA, November 07 - 11, 2005). CCS '05. ACM, New York, NY, 364-372.
- Oechslin, P. 2003. Making a faster cryptanalytic time-memory trade-off. In *Proceedings of Advances in Cryptology – 23th Annual International Cryptology Conference (CRYPTO'03)*, D. Boneh, Ed. Lecture Notes in Computer Science, vol. 2729. IACR, Springer-Verlag, Santa Barbara, CA. 617–630.
- Porras, Phillip, Saidi, Hassen, and Yegneswaran, Vinod. An Analysis of Conficker's Logic and Rendezvous Points. Retrieved April 14th, 2009, from An Analysis of Conficker site: <http://mtc.sri.com/Conficker/>
- Summers, Wayne C. and Bosworth , Edward (Jan 2004). Password Policy: The Good, The Bad, and The Ugly. *ACM International Conference Proceeding Series*; Vol. 58: Proceedings of the winter international symposium on Information and communication technologies. Cancun, Mexico; 05-08 Jan. 2004, 1-6.

Author Name, email@address

Unknown (1985, May 30). FIPS 112 - Password Usage. Retrieved April 1, 2009, from Federal Information Processing Standards Publication Web site:
<http://www.itl.nist.gov/fipspubs/fip112.htm>

Unknown (2009, March 11). Ophcrack. Retrieved April 14, 2009, from Ophcrack Web site: <http://ophcrack.sourceforge.net/tables.php>

Unknown (2009, March 24). Free Rainbow Tables: Distributed Rainbow Table Project. Retrieved April 14, 2009, from Free Rainbow Tables » Distributed Rainbow Cracking » LM, NTLM, MD5, SHA1, HALFLMCHALL, MSCACHE site:
<http://www.freerainbowtables.com/>

Unknown (2009, February 12). Project RainbowCrack. Retrieved April 14, 2009, from RainbowCrack - The Time-Memory Tradeoff Hash Cracker Web site:
<http://project-rainbowcrack.com/>

Author Name, email@address

Upcoming Training

Click Here to
{Get CERTIFIED!}



| | | | |
|--|------------------------|-----------------------------|----------------|
| SANS Prague 2017 | Prague, Czech Republic | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| SANS Boston 2017 | Boston, MA | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| Community SANS Omaha SEC401* | Omaha, NE | Aug 14, 2017 - Aug 19, 2017 | Community SANS |
| SANS New York City 2017 | New York City, NY | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| SANS Salt Lake City 2017 | Salt Lake City, UT | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| SANS Virginia Beach 2017 | Virginia Beach, VA | Aug 21, 2017 - Sep 01, 2017 | Live Event |
| SANS Adelaide 2017 | Adelaide, Australia | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style | Virginia Beach, VA | Aug 21, 2017 - Aug 26, 2017 | vLive |
| SANS Chicago 2017 | Chicago, IL | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Community SANS Pasadena SEC401 @ NASA | Pasadena, CA | Aug 23, 2017 - Aug 30, 2017 | Community SANS |
| Mentor Session - SEC401 | Minneapolis, MN | Aug 29, 2017 - Oct 10, 2017 | Mentor |
| SANS San Francisco Fall 2017 | San Francisco, CA | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| SANS Tampa - Clearwater 2017 | Clearwater, FL | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| Mentor Session - SEC401 | Edmonton, AB | Sep 06, 2017 - Oct 18, 2017 | Mentor |
| SANS Network Security 2017 | Las Vegas, NV | Sep 10, 2017 - Sep 17, 2017 | Live Event |
| Community SANS Albany SEC401 | Albany, NY | Sep 11, 2017 - Sep 16, 2017 | Community SANS |
| Mentor Session - SEC401 | Ventura, CA | Sep 11, 2017 - Oct 12, 2017 | Mentor |
| Community SANS Columbia SEC401 | Columbia, MD | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| Community SANS Dallas SEC401 | Dallas, TX | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| Community SANS Boise SEC401 | Boise, ID | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | vLive |
| Community SANS New York SEC401 | New York, NY | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Rocky Mountain Fall 2017 | Denver, CO | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS London September 2017 | London, United Kingdom | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS Baltimore Fall 2017 | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS Copenhagen 2017 | Copenhagen, Denmark | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| Community SANS Sacramento SEC401 | Sacramento, CA | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| SANS DFIR Prague 2017 | Prague, Czech Republic | Oct 02, 2017 - Oct 08, 2017 | Live Event |
| Community SANS Charleston SEC401 | Charleston, SC | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| Mentor Session - SEC401 | Arlington, VA | Oct 04, 2017 - Nov 15, 2017 | Mentor |
| Community SANS Indianapolis SEC401 | Indianapolis, IN | Oct 09, 2017 - Oct 14, 2017 | Community SANS |