Using Randomly Generated Cryptographic Hash Functions to Enforce Passwords

Joshua L. Shunk

Abstract

While cyber security has dramatically improved over time, the abilities of hackers and computer programming has also increased. This calls for an increase in password security. This isn't just a problem for individuals, but it is also a significant, and costly, issue for small businesses to big corporations. The goal of this project was to provide people with an easy way to make their passwords more secure without the need for data or complicated websites.The original thought was to create an application that would increase the strength of the password, and this proved to be true. In this project the same passwords were tested raw, and re-tested using the created application. Once the procedures were set up, each trial was given an unique 1, 2, 3, and 4 character password. Once the data was collected from the two passwords that were too lengthy to be tested, they were extrapolated using the formula. Using the created application to elongate the original four character password it took 9.81E+24 seconds, or 211034000000000000 years, after being stratified to crack open. The password without the help of the application, when inputted as the original four characters, took 0.0002 seconds. This proves that the researcher's original thoughts were correct. In conclusion, the password, with the help of the application, took considerably longer to crack than the password without the help of the application. This could be used for both individuals and corporations to secure data.

Introduction

Each year, over 1.9 million passwords and usernames were confiscated by hackers. (University of California, Berkeley, 2017). That is approximately 17 usernames or passwords hacked a second. While many people are taught simple ways to make passwords more secure, it is often impractical to memorize long strings of numbers, letters, and special characters. A relationship of equations can be used to calculate the time it would take for a computer to crack a password. Represented by:

$$T = A^N$$

$$D = T/(109 \times 3{,}600)$$

$$X = 2\ log2[T/(109 \times 3{,}600)]$$

Where T = The possibility space based on the length. A = The list of valid characters. N = The number of characters in the password. D = the hours it would take for a computer to find out the password. Finally X = The number of years that will have to pass before the password can be checked in less than one hour.

Importance. The importance of adding special characters and numbers to a password is often undermined by people. To break a password, a computer will go through letters first, giving the computer a 1 in 26 chance of guessing the character letter. Once a number or special character is added, the probability turns into a 1 in 76. A strong password is "at least 12 characters, no repetition or monotone sequence," (Agence nationale de la sécurité des systèmes d'information, 2019).

Computer Limits. While all computers are different, according to Moore's law, a computer's capacity doubles every year. A computer in 2019 should be able to do one billion possibilities per hour. The recommended password length is 12 characters, but many people fail to follow this suggestion. Passwords that represent words, or phrases, are more vulnerable to dictionary attacks. A dictionary attack is a method used that utilizes a list of known passwords or words that a computer can go through. These passwords are far weaker than random values because less trails are required. When a password is leaked, it means a hacker has taken the passwords of thousands of accounts to a website, and created a list to brute-force into each account.

Current Methods. More popular websites, that have lots of users, often use what's called "fingerprints" of the passwords that a user puts in. This process uses algorithms known as cryptographic hash functions. These functions transform data files such as "F" into a sequence h(F). For example hash function SHA256 would turn "1111" into "0FFE1ABD1A08215353C233D6E009613E95EEC4253832A761AF28FF37AC5A150C"

Cryptographic Hash Function. Cryptographic hash functions use set values for strings of characters, for example, "1111" in hash function SHA256 will always be the same string of values. While sending unique passwords through a hash function can result in strong hash values, because of set values for characters, hash values can be reversed. To use a hash function on a website that doesn't naturally support it can be close to impossible. This paper looks at the results of using different length and combinations of passwords to better protect people online.

Methods

Most of this background information came from a study done by University of California Berkeley in 2017. Information was also found from various other internet sources.1 Information regarding the importance of passwords was colled from a government consumer report regarding the topic. The sources were also used to learn how to created a more secure password which influenced the design and algorithm of the application. The same sources were also used to review how to crack passwords and how to operate Kali Linux The application used to create that passwords was created using Xcode with prior knowledge of programing. Full app can be found at https://github.c om/HockeyFan9000/Science-Fair-Password-Protector/tree/master/ScienceFair2020.xcodeproj.Testing was conducted

using a Virtual Machine running Kali Linux. Passwords were set up then broken into using a Zenmap scan and crunch.

Results

The data from the password strength with, and without, the help of the application was collected at intervals of 1 characters, starting at one and going up to four. It was thought that the passwords, with the help of the application, would be stronger than the passwords that were tested that didn't use the application. The data collected shows that, on average, the password with the help of the application took longer to break into than the password without the help of the application. This is because the application takes a standard password and changes its values to four times the number of characters. In addition to being a stronger password, there is no longer a need to memorize different passwords because they can now plug any combination of the correct characters and get the correct password. The strength of the bagnets had an exponential curve. The password, with the help of the application, when inputted one character, took an average of 3.40E-04 seconds to crack, which in perspective is about 7 hundred picoseconds. The password, without the help of the application, when inputted one character, took an average of 6.10E-10 seconds. The password, with the help of the application, when inputted two characters took an average of 2.06E+06 seconds to crack. The password, without the help of the application, took an average of 6.72E-08 seconds to crack. The password, with the help of application, when inputted 3 characters, took on average 5.33E+15 seconds. The password, without the help of the application, when inputted 3 characters, took on average 3.37E-06 seconds to crack. The password, with the help of application, when inputted 4 characters, took about 1.40E+25 seconds to crack. The password, without the help of the application, when inputted 4 characters, took about 0.000266.

Discussion

Summary of results: Passwords have become a very important part of everyone's life. Using the results, the length and complexity of the password is shown to have an exponential effect on the effectiveness of the password. The passwords that were changed using the application performed exponentially better than those that were not changed by the application as expected. This is because the application increased the complexity of the password therefore making in stronger and harder to break. The study did have to extrapolate some results because there was not enough time to fully crack the password.

Future Research: Additionally if this test was to be done again more passwords would have been tested to get more accurate results and data. If there was more time different algorithm would have been produced as well as the equation mentioned in the introduction would have been used to compare to the results given by the computer.

References

Configuration recommendations of a GNU/LINUX system. (2019, February 22). Retrieved October 16, 2019, from https://www.ssi.gouv.fr/uploads/2019/03/linux_config uratio n-en-v1.2.pdf.

Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. (2017, November 3). Retrieved from https://static.googleusercontent.com/media/re search.google.com/en//pubs/archive/46437.pdf.

Merrill, W. (2018, March 13). Advanced password tips and tricks. Retrieved July 16, 2019, from https://www.consumer.ftc.gov/blog /2015 /07/ advanced-password-tips-and-tricks.

Sharan, A. (2017, July 5). Passwords and Cryptographic hash function. Retrieved October 15, 2019, from https://www.geeksforgeeks.org/passwords-and-cryptographic-hash-function/.

Smith, A. (2017, April 27). Americans and Cybersecurity. Retrieved October 14, 2019, from https://www.pewinternet.org/2017/01/26/a meri cans-and-cybersecurity/.

Thomas, K., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., … Bursztein, E. (2017, November 3). Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. Retrieved October 13, 2019, from https://www.pewinternet.org/201 7/01/26/americans-and-cybersecurity/.

Using Passwords. (2017, March 13). Retrieved October 16, 2019, from https://www.pewinternet.org/2017/01/26/americans-and-cybersecurity

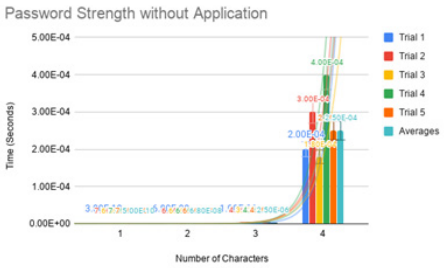

Fig. 1. Password Strength without Application

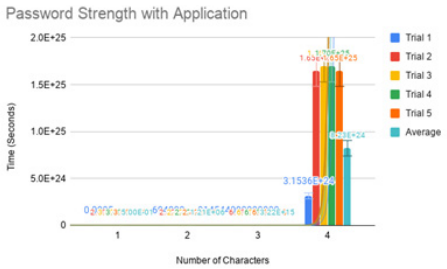

Fig. 2. Password Strength with Application

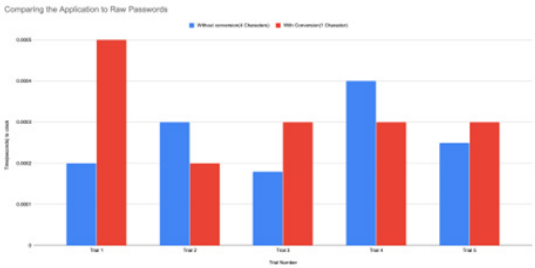

Fig. 3. Comparing the Application to Raw Passwords