Keep It Safe: Developing Strong Passwords to Protect Against Password Cracking

TABLE OF CONTENTS

Section A: Introduction

Cybercrime is one of the fastest-growing crimes in the world. The total value at risk globally over the next five years as a result of cybercrime is estimated at US$5.2 trillion or higher (Bissell and Cin). Although much of this is the result of attacks against corporations, there is a significant risk to individuals as well. I have seen this firsthand as my mom was the victim of identity theft and the payroll account at the company where my dad works was hacked. As a result, I recognize the importance of protecting online information and accounts through the use of strong passwords. But many people still use weak passwords that can be cracked in a matter of seconds. In this investigation, I will use mathematical calculations to determine the time it would take to guess passwords of various lengths and complexities. I will then evaluate the strength of these passwords based on the amount of time it would take a hacker to "guess" the password using available password cracking software. The aim of my investigation is to determine the length and composition of a password that would be sufficiently complex to provide reasonable protection without being so long and complicated that it cannot be easily remembered. Based on my mathematical exploration, I believe that 11 characters (9 letters, 1 symbol and 1 number) is an appropriate size and complexity for a password.

Section B: Investigation

In order to understand the need for a sufficiently complex password, it is important to first understand the current capabilities of and techniques used by password cracking software. At the moment, Hashcat is one of the most famous password hacking computer programs, because of its high tech abilities and cheap cost (Poston). Hackers will often use a method called

a rainbow table attack which is the fastest way of hacking a password. Rainbow table attacks are usually accomplished using Hashcat or similar computer programs. To understand a rainbow table attack you have to realize that every password is encrypted with a hash. To login, a user has to enter his or her password. If the hash encryption matches what the user enters as a password, then the user will be logged in. A rainbow table attack is when a hacker gets into the "rainbow table" which is the hash used to protect passwords. Once a hacker is into the rainbow table, they can crack every password stored in it (McKee). For example, if a hacker gets into the FNB bank rainbow table, then they can figure out everyone's account information. Although this is very unlikely due to FNB's high-level encryption protection, it is still possible.

Password cracking software uses three main methods to guess passwords. One is a dictionary attack, where the software goes through the words in a dictionary and then tries variations of these words, such as substituting ! for i. This is a very successful type of attack because so many people use simple passwords that are actual words, sometimes with minor modifications. According to SplashData, based on a review of over 5 million leaked passwords in 2017, the top 10 worst passwords were: (1) 123456; (2) password; (3) 12345678; (4) qwerty; (5) 12345; (6) 123456789; (7) letmein; (8) 1234567; (9) football; (10) iloveyou; (11) admin; (12) welcome (Securitymagazine.com). It is no wonder that so many accounts are hacked every year with passwords such as these.

Another common password cracking attack is a brute-force guessing attack. Since there are only so many combinations of characters of a given length, a brute-force guessing attack tries every possible combination (Poston). Although this can be time consuming, if given enough time the program will eventually guess the password. This is why it is important to select a password that is in a data set that is large enough that it will take the software a long time to try

every combination. Hopefully, this will provide security software adequate time to detect and disable the attack. Finally, a hybrid attack combines the two types of attacks above. It first checks to see if a password can be cracked using a dictionary attack and if that is unsuccessful it moves to a brute-force attack. (Poston)

This brings me to the analysis portion of my investigation. I will test 5 different password lengths and combinations to determine how long it would take for password cracking software to guess these passwords. My expectation is that as we begin to add more numbers and letters to the password sets, the possible combinations will increase and therefore make it harder and take more time to crack. I will be using the probability formula to figure out how many combinations are in the sample space and will then calculate the amount of time it would take for a computer program to guess every combination if the program guesses 100 thousand, 100 million and 100 billion password combinations every second. Based on available information there are password cracking programs that can process at these speeds (Haskell and O'shea).

The password combinations I have evaluated are as follows:

- 6 characters: 5 letters and 1 number.
- 7 characters: 5 letters, 1 number and 1 symbol.
- 10 characters: 9 letters and 1 number.
- 11 characters: 9 letters, 1 number and 1 symbol.
- 12 characters: 8 letters, 2 numbers and 2 symbols.

It is important to note that I used the following parameters in these calculations:

1. I treated upper case and lower case letters as separate characters, so where a password calls for a letter, there are 52 possibilities to choose from.

2. I limited special characters to the following 10 characters: !@#$%^&*().
3. I rounded numbers to the nearest hundredth.

Example 1:

I will begin my investigation with a password consisting of 6 characters: 5 letters and 1 number. This password is relatively weak, but it is still much stronger than the top 10 most used passwords (Dyrda). My prediction for this combination is that a hacker would be able to hack this password in under 1 hour. Since this example contains only letters and numbers, I will be multiplying 52 and 10. The reason there are 52 letters is because the alphabet is doubled to take into account upper and lower case. The 10 comes from the numbers on the keyboard, which range from 0 to 9. Since I am using characters, I need single-digit numbers, and this is the same for the letters.

6 characters: 5 letters and 1 number.

L (a → z + A → Z), # (0 → 9)

L x L x L x L x L x # = _____ combinations
52 x 52 x 52 x 52 x 52 x 10 = 3,802,040,320 possible combinations

Now, how long would this take with Hashcat programs that can guess the following number of passwords per second:

- 100 thousand
- 100 million
- 100 billion

100 thousand:	100 million	100 billion
3,802,040,320 / 100,000 =38,020.4 seconds Ans/60 =633.67 minutes Ans/60 =10.56 hours	3,802,040,320 / 100,000,000 =38.02 seconds	3,802,040,320/ 100 B =.038 seconds

This data shows that a Hashcat program running at 100 billion password combinations every second would only take .04 seconds or less to guess the top 10 most frequently used passwords. To put this in perspective the average blink of the eye is .1 seconds, which means password cracker software could figure out this type of password two and a half times faster than it takes to blink once.

Example 2:

In example two I am increasing the complexity by adding a symbol to the combination. The purpose of adding this one symbol is to show how much the password's strength increases by only adding 1 symbol. My expectation for this example is that when we add a symbol, the number of possible combinations will grow exponentially.

7 characters: 5 letters, 1 number and 1 symbol.

L (a → z + A → Z), # (0 → 9) and S ("!" → ")")

L x L x L x L x L x # x S = _____ combinations
52 x 52 x 52 x 52 x 52 x 10 x 10 = 38,020,403,200 possible combinations
Now, how long would this take with Hashcat programs that can guess the following number of passwords per second:

- 100 thousand
- 100 million
- 100 billion

100 thousand:
138,020,403,200 / 100,000
=380,204.03 seconds
Ans/60
=6,336.73 minutes
Ans/60
=105.61 hours
Ans/24
=4.40 days

100 million:
38,020,403,200 / 100,000,000
=380.20 seconds
Ans/60
6.34 minutes

100 billion:
38,020,403,200 / 100 B
=.38 seconds

We see from this set of calculations, that just by adding 1 symbol, the number of possible combinations goes from about 3.8 billion to about 38 billion. Although this password would be solved only slightly slower than the blink of an eye, the Hashcat program with 100,000 guesses per second increased from about ten and a half hours to almost four and a half days.

Example 3:

In this example, I am creating a more complex password than examples 1 and 2 by increasing the number of characters. My prediction for this example is that the password will take a much greater time to guess than the other two due to the increased amount of letters, which will create a much larger sample space. Similar to examples 1 and 2, I am using the numbers 52 and 10.

10 characters: 9 letters and 1 number.

L (a → z + A → Z), # (0 → 9)

L x L x L x L x L x L x L x L x L x # = _____ combinations
52 x 52 x 52 x 52 x 52 x 52 x 52 x 52 x 52 x 10 = 2.78×10^{16} possible combinations

Now, how long would this take with Hashcat programs that can guess the following number of passwords per second:

- 100 thousand
- 100 million
- 100 billion

100 thousand:	100 million	100 billion
$2.78x10^{16}$ / 100,000	$2.78x10^{16}$ / 100,000,000	$2.78x10^{16}$ / 100 B
=277,990,588,363.57 seconds	=278,000,000 seconds	=278,000 seconds
Ans/60	Ans/60	Ans/60
=4,633,176,472.73 minutes	=4,633,333.33 minutes	=4,633.33 minutes
Ans/60	Ans/60	Ans/60
=77,219,607.88 hours	=77,222.22 hours	77.22 hours
Ans/24	Ans/24	Ans/24
=3,217,483.66 days	=3,217.59 days	3.22 days
Ans/365	Ans/365	
=8,815.02 years	8.82 years	
Ans/100		
=88.15 centuries		

This appears to be a reasonably safe password, but I still have concerns that a password cracking software that can guess 100 billion passwords per second could crack this password before being detected, so I think it is necessary to test additional passwords. I did find it interesting that by just adding 4 letters, the time it takes to crack increases from about ten and a half hours in Example 1 to over 88 centuries for a web app guessing 100,000 password combinations a second. The strength and time to crack increase since you are multiplying the sequence by 52 whenever you add a letter, which is much higher than when you add a number or symbol since in those cases you would only be multiplying the sequence by 10. This means that if you are going to add more characters to your password, then it is best to add letters since this will strengthen it and take longer to crack.

Example 4:

Similar to examples 1, 2 and 3, my hypothesis is that as we add more characters and symbols, the time it takes to crack the passwords, as well as the complexity of the password increases. Since this example is the same thing as example 3, but with a symbol, I will just be multiplying example 3 by 10. This will obviously increase the time it would take a hacker to perform a brute-force attack.

11 characters: 9 letters, 1 number and 1 symbol.

L (a → z + A → Z), # (0 → 9) and S (! → -)

L x L x L x L x L x L x L x L x L x # x S = _____ combinations
52 x 52 x 52 x 52 x 52 x 52 x 52 x 52 x 52 x 10 x 10 = 2.77×10^{17} possible combinations

Now, how long would this take with Hashcat programs that can guess the following number of passwords per second:

- 100 thousand
- 100 million
- 100 billion

100 thousand:	100 million	100 billion
2.77×10^{17} / 100,000	2.77×10^{17} / 100,000,000	2.77×10^{17}/ 100 B
$=2.77 \times 10^{12}$ seconds	=2,770,000,000 seconds	=2,770,000 seconds
Ans/60	Ans/60	Ans/60
$=4.63 \times 10^{10}$ minutes	=46,166,666.67 minutes	=46,166.67 minutes
Ans/60	Ans/60	Ans/60
=772,196,079 hours	=769,444.44 hours	=769.44 hours
Ans/24	Ans/24	Ans/24
=32,174,836.6 days	=32,060.19 days	32.06 days
Ans/365	Ans/365	
=88,150.24 years	87.84 years	
Ans/100	Ans/10	
=881.5 centuries	8.78 decades	

As we add more characters, we can really see the role of exponential growth in password cracking. By adding 1 symbol, the time it would take to crack example 4 went from 88 centuries

(in example 3) to just over 881 centuries. Based on my calculations and research about password cracking software I think the combination in Example 4 is a reasonably safe password and this is what I recommend. It is possible that this could change over time as the hacking software becomes more advanced, but for now, I believe it is sufficient.

Example 5:

Although I believe the password in Example 4 is sufficient, I think it is worthwhile to test one more combination to see how much additional protection it would offer. This example is very similar to Example 4, except instead of 9 letters, I will remove one letter and add an extra symbol and number. My initial expectation was that this would reduce the number of possible combinations and time it takes to crack the password as compared to Example 4 because we are removing one letter, which removes a 52.

12 characters: 8 letters, 2 numbers and 2 symbols.

L (a → z + A → Z), # (0 → 9) and S ("!" → ")")

L x L x L x L x L x L x L x L x # x # x S = _____ combinations
52 x 52 x 52 x 52 x 52 x 52 x 52 x 52 x 10 x 10 x 10 x 10 = 5.35×10^{17} possible combinations

Now, how long would this take with Hashcat programs that can guess the following number of passwords per second:

- 100 thousand
- 100 million
- 100 billion

100 thousand:
$5.35x10^{17}$ / 100,000
=5,350,000,000,000 seconds
Ans/60
=89,166,666,666.67 minutes
Ans/60
=1,486,111,111.11 hours
Ans/24
=61,921,296.30 days
Ans/365
=169,647.39 years
Ans/100
=1,696.47 centuries
Ans/1,000
=1.70 millennium

100 million:
$5.35x10^{17}$ / 100,000,000
=5,350,000,000 seconds
Ans/60
=89,166,666.67 minutes
Ans/60
=1,486,111.11 hours
Ans/24
=61,921.30 days
Ans/365
=169.65 years
Ans/100
=1.70 centuries

100 billion:
$5.35x10^{17}$ / 100,000,000,000
=5,350,000 seconds
Ans/60
=89,166.67 minutes
Ans/60
=1,486.11 hours
Ans/24
=61.92 days

As we can see, my prediction was incorrect. Now that I have figured out the problem, I see that there are an extra 2 tens rather than just 1. This means that instead of multiplying the sequence by 10, we are multiplying by 100 (10 x 10 = 100), which is greater than 52 so the number of possible combinations and time to crack this password increased.

Section C: Data Analysis:

My mathematical investigation shows us that by adding more characters to a password, the sample space increases making the password more secure. My examples have also shown that adding letters adds the most security since there are 52 possible options. This leads me to ask, why do websites typically require passwords that consist of more than just letters? Although I am not certain of the answer, I believe the reason may be because if people could use only letters they would be more likely to choose a word rather than a random combination of letters. A password that is made of an actual word is very vulnerable to a dictionary attack, regardless of how long the word is. My investigation also shows that if we continue adding letters, numbers and other characters the number of combinations and security of the password will increase. However, at some point it becomes impractical to have such a long and complex password and it is necessary to balance the need for security with usability. I believe that a password with 11 characters (9 letters, 1 symbol and 1 number) achieves the correct balance.

Section D: Conclusion

People put a lot of trust in the belief that passwords will protect their personal information, such as bank accounts, health records, credit history and financial information. If this information falls into the wrong hands it can have devastating consequences. I know because it has happened to my family. However, a lot of people still use passwords that are very vulnerable to being discovered. I think the reason for this is because they do not understand the risks and the sophistication of the cybercriminals who are trying to access personal information. I did not understand these risks either until I completed this investigation. I also didn't realize the role that mathematics plays in something like selecting a password. I suspect there are mathematicians working on both sides of this - some are creating formulas and equations for web apps such as Hashcat, while others are figuring out how to design passwords that will protect against attacks. This battle will likely continue so it is necessary to stay current on what it takes to create a strong password. For now, I believe that the password which I have recommended above is a reasonably safe password and that is what I will use and suggest to my friends and family.

Work Cited

1. Bissell, K., Lasalle, R. and Cin, P., 2021. *2019 Cost of Cybercrime Study | 9th Annual | Accenture*. [online] Accenture.com. Available at: <https://www.accenture.com/us-en/insights/security/cost-cybercrime-study#:~:text=Organizations%20spend%20more%20than%20ever,million%20to%20US%2413.0%20million.> [Accessed 11 February 2021].
2. Dyrda, L., 2021. *25 most common passwords*. [online] Beckershospitalreview.com. Available at: <https://www.beckershospitalreview.com/cybersecurity/25-most-common-passwords.html> [Accessed 11 February 2021].
3. Goodin, D., 2021. *25-GPU cluster cracks every standard Windows password in <6 hours*. [online] Ars Technica. Available at: <https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/> [Accessed 11 February 2021].
4. Haskell, P. and O'shea, B., 2021. *A computer can guess more than 100,000,000,000 passwords per second. Still think yours is secure?*. [online] The Conversation. Available at: <https://theconversation.com/a-computer-can-guess-more-than-100-000-000-000-passwords-per-second-still-think-yours-is-secure-144418> [Accessed 11 February 2021].
5. McKee, C., 2021. *Rainbow tables explained: Password hacking and how to prevent against it*. [online] Minim.com. Available at: <https://www.minim.com/blog/rainbow-tables-explained-password-hacking-and-how-to-prevent-against-it> [Accessed 11 February 2021].

6. Morgan, S., 2021. *Cybercrime To Cost The World $10.5 Trillion Annually By 2025*. [online] Cybercrime Magazine. Available at: <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/> [Accessed 11 February 2021].
7. Poston, H., 2021. *10 Most Popular Password Cracking Tools [Updated 2020] - Infosec Resources*. [online] Infosec Resources. Available at: <https://resources.infosecinstitute.com/topic/10-popular-password-cracking-tools/#:~:text=1.,300%20different%20types%20of%20hashes.> [Accessed 11 February 2021].
8. Securitymagazine.com. 2021. *The Worst Passwords of 2017 Revealed*. [online] Available at: <https://www.securitymagazine.com/articles/88626-the-worst-passwords-of-2017-revealed> [Accessed 11 February 2021].
9. Techslang — Today's most spoken tech explained. 2021. *What is a Rainbow Table Attack? — Definition by Techslang*. [online] Available at: <https://www.techslang.com/definition/what-is-a-rainbow-table-attack/#:~:text=A%20short%20definition%20of%20Rainbow,of%20a%20rainbow%20hash%20table.&text=Cybercriminals%20favor%20rainbow%20table%20attacks,them%20to%20crack%20passwords%20faster.> [Accessed 11 February 2021].