Password Cracking Project

Viktor Potter

Password Cracking Results

The tool I chose to use for this project was Hashcat. I initially chose Hashcat because I was under the impression that it would have a GUI built into it. When I downloaded it, I realized that was not the case, but I chose to keep using it due to how helpful it was in explaining all the commands and functions it had. Another reason I chose Hashcat was due to the fact that it looked like it would be easier to install and get running in a Windows environment without me having to use any virtual machines in order to get it to work.

My average rate for checking passwords during the project was an average of 18 million passwords a second. The reason I believe I achieved this rate is due to the fact that I was using both a GPU and a CPU in order to crack passwords simultaneously. Although I was using a 3070 TI and the GPU above that, an RTX 3080, which can get around 2.5 billion a second, I believe my drastically lower cracking speed was because I was not fully utilizing my GPU while cracking passwords and was just using a part of it.

In order to crack the passwords, I employed 4 methods. The first method was setting up Hashcat to guess alphanumeric passwords starting from a length of 2 and then working its way up to 8. The second strategy I used was to guess only random lowercase passwords starting from a length of 2 up to a length of 8. The third strategy I employed was to use the given dictionary and then have Hascat use that to check if any of the passwords came from the dictionary. The fourth and final strategy I employed was to take the given dictionary and then use Python code to make a list of all possible l33t speak permutations of the words list given, and then use that generated l33t wordlist in order to try and crack all of the l33t speak passwords.

For the passwords I managed to crack, I cracked all of them. For users 2,5,9,10,11,12,15,17, and 20, the first method I used managed to crack all of them in around 4 hours. For user 18, the alphanumeric method did not manage to get it, so instead of letting it run longer, I switched to the lowercase letter strategy to get that one in around 5 minutes. Then, for users 6,7,14,16, and 19, I used the dictionary attack to crack those passwords. Finally, for users 1,3,4,8, and 13, I used the l33t speak file I made to crack those passwords. Below is a table to show in an easier-to-read format the users, their passwords, and the method used to crack them.

User	Password	Cracking Method
1	41t3rn4t3	L33T Dictionary
2	7OgTHT	Alphanumeric
3	14b0r4t0ri35	L33T Dictionary
4	int3rn4ti0n411y	L33T Dictionary
5	lkyru	Alphanumeric
6	forwarding	Dictionary
7	increased	Dictionary
8	31iz4b3th	L33T Dictionary
9	31w4	Alphanumeric
10	SJ	Alphanumeric
11	2c8zc	Alphanumeric
12	uddg	Alphanumeric
13	c0mp4ni35	L33T Dictionary
14	longitude	Dictionary
15	ta	Alphanumeric
16	revolutionary	Dictionary
17	lr4	Alphanumeric
18	ugrknq	Lowercase Letters
19	cincinnati	Dictionary
20	lsv	Alphanumeric

Project Questions Answers

1. Password Cracking Time Personal Setups
    a. For 6 alphanumeric characters, there are a total of 62^6 combinations possible, or a total of 56,800,235,584 (56.8 billion). Given my cracking speed of 18 million a second, it would take a maximum amount of time of 3156 seconds, or around 53 minutes, to crack a 6 alphanumeric password.
    b. For 8 alphanumeric characters, it would take 12,130,006 seconds or around 141 days, maximum, to crack any password.
    c. For 10 alphanumeric characters, it would take 46,627,742,548 seconds or around 1,478.5 years maximum, to crack any password
    d. For 12 alphanumeric characters, it would take 179,237,040,000,000 seconds or around 56,835.69 centuries, maximum, to crack a 12 alphanumeric character password.
2. Password Cracking Time With RTX 3080 at 2.5 Billion a second
    a. For 6 alphanumeric characters, it would take 22.7 seconds maximum.
    b. For 8 alphanumeric characters, it would take 87,336 seconds or around 1 Day maximum.
    c. For 10 alphanumeric characters, it would take 335,719,746 seconds or around 10.6 years maximum.
    d. For 12 alphanumeric characters, it would take 1,290,506,700,000 seconds or around 409.2 centuries maximum.
3. I believe that a password meter is not a good measure of password strength. The reason for this is that most of a password's strength comes from its length. If a password strength meter tells you your password is really strong, even though it is just 6 characters long, it is doing you a disservice. If someone were to try to crack that password, it would take them under a day in order to get your password and have access to your account. From the results of the experiment, I believe the best password length is 12 alphanumeric characters. The reason for this is due to the fact of how long it would take to crack a truly random 12-character password, even if the attacker has many GPUs running as the amount of time and power and time it would take to crack your password would not be worth the attacker's use of resources and they would most likely give up after trying for a long time with no results.
4. The use of SHA-512 for hashing passwords is much more secure than using MD5. The reason for this is that calculating a SHA-512 hash is significantly more computationally expensive, increasing the difficulty of any attack, as the time required per password rises drastically, making cracking any password take much longer and require more resources.
5. The use of a salt does increase password security. By using a salt, the use of things like rainbow tables is prevented, drastically increasing the amount of time it will take to crack

any leaked passwords, as, instead of just using the table, the hashes now have to be manually cracked due to the salt making each hash unique.

6. Although an attack like the one in the project would not work on an online system, that does not mean that the importance of offline password protection is any less. For example, if the sites' stored passwords are leaked but are not salted or hashed, the attacker will instantly have access to every account on the site. In the case of store passwords being leaked and they are salted and hashed, it makes the complexity of an offline attack much higher, and if the passwords that are stored are strong enough, it could mean that the attacker is unable to crack any passwords, whereas if they were just stored in plain text, the attacker would not need to do anything. Even in the case that the passwords are not strong, just the fact that they are salted and hashed can prevent at least some passwords from being leaked. This is why even if a site is online, it still needs to worry about offline attack protection.