Password Cracking

Conner Brinkley
11.04.2020

Applied Cryptography
The University of Tennessee

INTRODUCTION

The purpose of this experiment is to explore how secure passwords are when stacked up against popular cracking tools with modern computing power. In order to accomplish this, four different categories, each with five unique passwords, were created using an online password strength checker, and a popular command line tool, John the Ripper, was used to attempt to crack them.

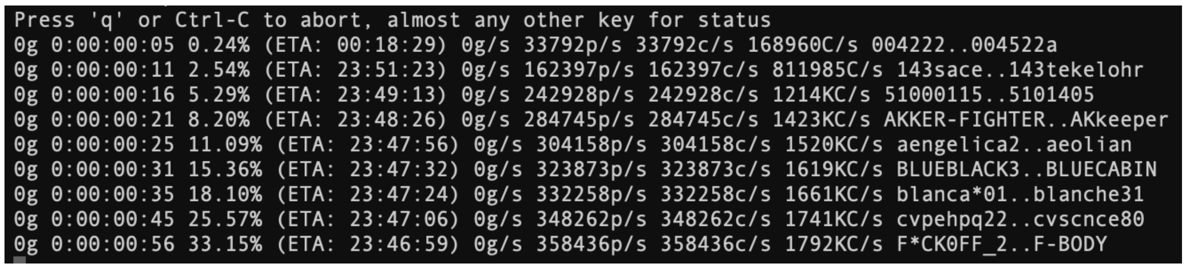
PROCESS

Starting out, a simple C++ program was written that generates Unix style passwd and shadow files, given a list of plaintext passwords on standard input. This program was then used to create suitable files to feed to John the Ripper for cracking after determining what passwords should be attempted. Table 1 shows the passwords that were used in this experiment.

Password Types			
Weak	**Good**	**Strong**	**Very Strong**
passwordone	pa$$word	Password123	Password!!!
username1	john1234	IHateYou1	Pa$$w0rd!
UserOne	Kitty13	HorseGirl10	!@#$%^&*(
User1	g00dpassword	MetsFan12	!mp055!bl3
A1!	H4ck3r	Princess!!	J2g%&#@

Table 1. Passwords used for cracking.

These passwords were placed under the four categories (weak, good, strong, and very strong) using the password meter provided in the project description. One thing that was clear right from the start was that some of the passwords the website deemed as “strong” or “very strong” did not seem all that secure, like “Password123” or “Password!!!”.



```
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:05 0.24% (ETA: 00:18:29) 0g/s 33792p/s 33792c/s 168960C/s 004222..004522a
0g 0:00:00:11 2.54% (ETA: 23:51:23) 0g/s 162397p/s 162397c/s 811985C/s 143sace..143tekelohr
0g 0:00:00:16 5.29% (ETA: 23:49:13) 0g/s 242928p/s 242928c/s 1214KC/s 51000115..5101405
0g 0:00:00:21 8.20% (ETA: 23:48:26) 0g/s 284745p/s 284745c/s 1423KC/s AKKER-FIGHTER..AKkeeper
0g 0:00:00:25 11.09% (ETA: 23:47:56) 0g/s 304158p/s 304158c/s 1520KC/s aengelica2..aeolian
0g 0:00:00:31 15.36% (ETA: 23:47:32) 0g/s 323873p/s 323873c/s 1619KC/s BLUEBLACK3..BLUECABIN
0g 0:00:00:35 18.10% (ETA: 23:47:24) 0g/s 332258p/s 332258c/s 1661KC/s blanca*01..blanche31
0g 0:00:00:45 25.57% (ETA: 23:47:06) 0g/s 348262p/s 348262c/s 1741KC/s cvpehpq22..cvscnce80
0g 0:00:00:56 33.15% (ETA: 23:46:59) 0g/s 358436p/s 358436c/s 1792KC/s F*CK0FF_2..F-BODY
```

Figure 1. Varying hash rates on Hydra 13 in Min Kao.

Figure 1 above shows some statistics after running John the Ripper on the Hydra 13 machine in the Min Kao building. The c/s measurement shows the number of hashes that the computer is generating each second, which will be useful later when determining how long it would take to crack varying types of passwords on that machine. As you can see from Figure 1, that number ranges from around 33,000 c/s to well over 350,000 c/s, and it seemed to average a little over 300,000 c/s during the entire experiment.

Before attempting to crack passwords, I downloaded a wordlist from Crackstation.net with about 64 million human passwords that have been leaked online and free to use instead of using John the Ripper's built in wordlist. I tried downloading a more comprehensive list with almost 1.5 billion words and passwords from the same website, but its massive 15 GB size prevented me from being able to use it on the Hydra computers.

RESULTS

For each category, I started out with a dictionary attack before moving on to John the Ripper's (JTR) default sequence, which starts with "single crack" mode, then a dictionary attack with the default wordlist, then incremental mode. Out of all 20 tested passwords, JTR was able to crack 11 of them. Out of those, only one was cracked with incremental mode, and one was cracked with the "single crack" mode. The 9 remaining were a result of dictionary attacks.

Weak
After leaving JTR running overnight, it was able to crack four out of five of the weak passwords. "User1" was cracked with the "single crack" mode, and "A1!" was the only one cracked with incremental mode. The only password that JTR was unable to crack was "UserOne," which surprisingly did not have any digits or special characters.

Good
JTR was able to crack three out of the five good passwords in about three minutes with a dictionary attack, but it was not able to find any more after running for over seven hours. The remaining uncracked passwords were "g00dpassword" and "H4ck3r."

Strong
JTR cracked two out of the five strong passwords in about 2.5 minutes with a dictionary attack, but it was not able to crack any more after running for over 14 hours. The remaining uncracked passwords were "IHateYou1," "HorseGirl10," and "MetsFan12."

Very strong
Similar to the strong passwords, JTR only cracked two of the five very strong passwords in the same amount of time with a dictionary attack, but it could not find any more after running for

over 20 hours. The remaining uncracked passwords were “Password!!!,” “!mp055!bl3,” and “J2g%&#@.”

CONCLUSION

After running these series of experiments, it is hard to trust online password strength meters, especially with how quickly a simple dictionary attack cracked over half of them. In most cases, the passwords I chose were probably too long for any reasonable incremental attack with John the Ripper, which is why almost all were cracked with a wordlist.

It was shocking how many “strong” passwords JTR was able to effortlessly crack. If I had been able to use the 15 GB wordlist on the Hydra machines, then it likely would have been able to crack a few more in a reasonable time. Although the password meter would likely disagree, in practice, it seems like a long, lowercase-lettered password is more secure than a shorter, alphanumeric password with special characters because as the requirements become more complicated, users will likely pick predictable variations of common words. As backwards as it sounds, by enforcing more complicated passwords on users, password security goes down.

SUPPORTING QUESTIONS

1.) *Assuming that you used your setup for this project alone, how long do you calculate that it would take to crack a 6-character alphanumeric password? 8-characters? 10-characters? 12-characters? (use the c/s measurement from your experiments).*

$$HashRate = 300{,}000\ per\ second$$

$$AlphanumericSetSize = 26 + 26 + 10 = 62$$

$$Seconds\ to\ calculate\ = \frac{AlphanumericSetSize^{\#\ of\ characters}}{HashRate}$$

$$Days\ to\ calculate\ = \frac{Seconds\ to\ calculate}{60\ s\ \cdot 60\,\mathrm{min}\ \cdot\ 24\ hours}$$

$$Avg\ days\ to\ crack\ = \frac{Days\ to\ calculate}{2}$$

6-characters = **1.09 days**
8-characters = **4,211.8 days** <u>or</u> **11.5 years**
10-characters = **16,190,188 days** <u>or</u> **44,356 years**
12-characters = **62,235,084,151 days** <u>or</u> **170,507,079 years**

2.) *Do you think that the password meter is a good indication of actual password security? From the results of your experiment, what is your recommendation for minimum password length? Be creative in your response. Imagine what hardware and resources a potential attacker might have, and briefly justify your assessment of the attacker's capabilities.*

No, the password meter is not a good indication of password strength. In the experiment, a Hydra machine in Min Kao with a hash rate of 300,000 c/s was able to crack two of the five "very strong" passwords in under three minutes using a common wordlist. No matter what random combination I used or how long it was, the password meter always classified lowercase password strings as "weak." As seen in this experiment, some longer "weak" passwords held up better than shorter "strong" ones. Based on the calculations from question #1, it seems like a safe minimum password length would be 8-characters, and that is even without special characters. However, assuming that an attacker could calculate a whopping 1 billion hashes per second, that would only take an attacker a little over a day to crack. But if that number were raised to a 10-character minimum, the amount of time it would take to crack jumps up to over 13 years. Therefore, 10-characters at a minimum seems like a secure length.

3.) *Recently, high-end GPUs have revolutionized password cracking. One tool, ighashgpu, is able to perform 1.3 billion MD5 hashes per second on an AMD Radeon 5850 (a 2-year-old, mid-to-high range video card). Whitepixel, another tool, claims that it can perform 33.1 billion hashes per second using 4 Radeon 5970s. Consider your calculations in question #1, and redo them assuming you had access to a system with 4 Radeon 5970s. Do your answers for question #2 change?*

$$HashRate = 33{,}100{,}000{,}000 \cdot 4 = 132{,}400{,}000{,}000 \; per \; second$$

6-characters = **0.214 seconds**
8-characters = **13.74 minutes**
10-characters = **36.68 days**
12-characters = **386.34 years**

My answer from #2 doesn't change much because 36 days is still a long time for an attacker to wait to crack a single password. Unless the account in question is highly confidential, like a banking account, 10-characters is still probably a safe number. However, on more sensitive accounts, 12-characters should be the absolute minimum.

4.) *Fedora 14 and other modern Linux distributions use a SHA-512 (rather than MD5) for hashing passwords. Does the use of this hashing algorithm improve password security in some way? Why or why not?*

Yes, because SHA-512 has better collision resistance than MD5, so there is less of a chance of an adversary being able to find another key that hashes to the same value. It's much more likely that an attacker would have to find the exact value that was used to calculate a given hash, making the odds of cracking much harder with SHA-512.

5.) *Does the use of a salt increase password security? Why or why not?*

Yes, salts increase password security by preventing the use of pre-calculated rainbow tables in offline attacks. If a salt is not used, an attacker could easily look up common password hashes in these tables that are readily available online, making it much faster for them to crack more passwords. Even if a different salt is not used for every user, and instead, a constant salt was used on the machine, this would still provide more security than not using them at all. Although, it is still a good idea to use different salts for each account/user because it eliminates the risk of an attacker quickly cracking multiple passwords if different users use the same password.

6.) *Against any competent system, an online attack of this nature would not be possible due to network lag, timeouts, and throttling by the system administrator. Does this knowledge lessen the importance of offline password attack protection?*

No, in fact, it increases the importance of offline password attack protection. Because of all the obstacles with online attacks, it would make more sense for a hacker to try to steal the passwords to perform offline attacks in order to make it worth their time. Therefore, offline attack prevention is just as important as online attack defenses, and everyone should salt their passwords and use other techniques such as honeywords to know if a breach occurred.