Thomas Hart

CS 465

Project 7 – Password Cracking

1. My computer was able to run approximately 375,000 comparisons per second. Given that a password is in the word list, they are cracked almost instantly, regardless of their size. If they are not in the wordlist, John the Ripper attempts a brute force approach called “incremental”. This checks all the different combinations. A 6-character password with only lower-case letters would be over 300 million different combinations, and would take around 800 seconds, or 13 minutes in the worst case on my machine. With upper-case included, it jumps to around 2.25 billion combinations. This would take 100 minutes for my computer to compute in the worst case. An 8-character password with upper- and lower-case letters only would have over 200 billion combinations and take my computer almost 150 hours to compute. Any further than this and the number of possible combinations would make my computer useless when it comes to computing them all. If we add in special characters and numbers and make it a 12-character password, it would take millions of years to solve with 94^12 combinations (according to https://www.password-depot.de/en/know-how/brute-force-attacks.htm).
2. I’ll assume an attacker would have a similar machine that I do for the sake of simplicity. My machine is a decently expensive programming computer, so maybe this hypothetical is possible. I would not say that the password meter is accurate since I can type a 15-character lower-case password and it will still show up as “very weak”. An attacker could never break the password even if it is only lower-case. The length alone puts this into an entirely different category compared to shorter passwords. The strength meter is accurate in that upper-case, special characters, and digits together skyrocket the amount of combinations and would make a brute force attack impossible. I would say that with healthy paranoia, a 10-character password would be great, even if we are only using lower-case letters. It would likely be great at 8-characters, but only if we were using more than just lower-case letters.
3. 33.1 billion hashes per second can be done using 4 Radeon 5970s. Plugging in the same numbers, what would take my computer 150 hours to complete for 200 billion combinations would take this other device less than 7 seconds. The ones before that would be solved almost instantly. The 94^12 combinations that are included in a 12-character password using lower- and upper-case, digits, and special symbols would take approximately 3,993,960,345.87 hours, or over 455,931 years to solve. This is still infeasible, but much faster than my computer by a long shot. I would say a 12-character password with any possible characters would be secure even with this system. Only lower-case letters in the 12-character password would be dangerous, though. This system could solve it in about 800 hours, which is feasible. If we had a 14-character password with only lower-case, it would take this system over 61 years to break, so if we are only relying on length and using only lower-case, a 14-character password should be safe against a brute force attack.
4. I would think that SHA512 would be much better at securing passwords that MD5. SHA512 is unimaginably less likely to have a collision than MD5 because the number of possible combinations of the hash is far above the hash of MD5. If a collision counts as a successful

break, then SHA512 would certainly be more secure since a collision is more likely in MD5. If we are trying to find the exact password, however, a collision might show a password that produces the same hash but is not the actual password. In this case, the number of combinations an attacker would have to check is the same in SHA512 as it is MD5. Even with the same number of checks in the worst case, though, comparing the larger SHA512 hash each time would take 4 times as long since the length is 512 bits as opposed to the MD5 128-bit length. So SHA512 would still be more secure than MD5 with that increased computation time.

5. A salt makes a password much more secure. By adding random bits before passwords, two users who use the same password will produce a completely different hash. An attacker would have to know the salt and perform their time-consuming attack just to break a single password. If an attacker doesn't know the salt, it adds a bunch of new characters onto the front of the string and makes the task very infeasible as we saw before with just 12 random characters, which would take way too long to break. A salt adds a ton of security to passwords.
6. Knowing that an online attack of this nature would be infeasible does not lessen the importance of offline protection. If someone got a hold of your device and wanted your passwords, they could still perform an offline attack. The online safety against this type of attack is a perk, but it doesn't eliminate the possibility of offline attacks.
7. In the future, I see people continuing to use smaller passwords for accounts that don't particularly matter, and I don't see hackers breaking into them either, even if they can. Social media websites, for example, don't seem to matter enough for a hacker to steal, especially if you're just a normal person with nothing to offer an attacker. Dual authentication could also stop those types of attacks. Attackers would need to eavesdrop and steal passwords that way if I understand this correctly, that online attacks against an email account would not work. If an eavesdropper saw you type out a password, they'd have it even if it was 100 characters long. If people start requiring long passwords, I could see people repeating things to make the length requirement and reusing that one password for everything, so they don't have to memorize anything else. This may be a security issue, but unless someone creates a quantum computer, I don't think passwords will become that much longer since a supercomputer is already stumped at 12 random characters, or 94^12 combinations to try.