Trinity Bissahoyo

COSC 483

Dr. Ruoti

November 9, 2025

Password Cracking

For this project I used John the Rippers because I was using my personal computer (Mac) and did not test if hashcat was compatible with my device. My rate for password checks was about 26212 for multiple salts and 27516 c/s for a single salt

For guessing, I utilized different assortments of mask and a few modes to try and crack the passwords. I first made sure to configurations that matched the four rules. the min and max size to 2 – 8 in the command lines. I then wrote a leet-speak rule that used variations of combinations such as ‘sa4sb6sO0’ or ‘sa4sb6’. In many of the configurations I used the given wordlist as well. I used a mask of all lowercase, all uppercase, all digits and a combination of all three. I used other rule groups as well including hashcat which included best64, d3ad0ne, dive, InsidePro, T0XlC, rockyou-30000, and specific. Between guessing I would also change the max time of at least 4 hours

Passwords Cracked

Passords	Strategy
1bAgedIxw$qHvoO8m1ret74ZvdLaeGH.: **increased**	Set rules=hashcat
1qV12ONrm$pexwVxdboE2Ae44B3mka01: **cincinnati**	Set rules=hashcat
1Xv7JAxzV$3lhJir1rSrua2Ex4tiOj0/: **revolutionary**	Set rules=hashcat
1CRWV4i7d$1Rc99I52lsZSsLL6nF6NY/: **forwarding**	Set rules=hashcat

1feMN1Aci$py8cBbqYPNt0tPD7PvPc51: **longitude**	Set rules=hashcat
1FgvYqsDt$Ta/jEIgd/ms8SkKpa5Tnm0: **ta**	Used a mask that only looked for all lowercase characters
1.Zg6BkMx$vpWtITZWHtehSTVFi/3.i.: **lsv**	Used a mask that only looked for all lowercase characters
1s9guw.eA$6h7RjsJTEGnM6GjWAm6Ko/: **uddg**	Set rules=hashcat
1h.2qRj2I$e5U/HsFNkOe0.HNDfT8ps/: **SJ**	Used a mask that only looked for all uppercaes characters
1ZdudGygx$De9sJbWOtjgbcZur6wGXJ/: **lkyru**	Set rules=hashcat
1EUfPQVlE$XAVa9Ury1zpAMKAxbO1fP0: **lr4**	Set rules=hashcat

Additional Questions

1. Calculation Time: alphanumeric character set = 62 (26 lowercase, 26 uppercase, 10 digits) | MD5crypt = 26212 c/s
    a. 6-character: **25 days**
        i. (62^6 / 26212) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day = 25.1 days
    b. 8-character: **264 years**
        i. (62^8 / 26212) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day * 365 days / 1 year = 264.13 years
    c. 10-character: **1,015,336 years**
        i. (62^10 / 26212) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day * 365 days / 1 year = 1015336.804 years
    d. 12-character: **3.9 billion years**
        i. (62^12 / 26212) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day * 365 days / 1 year = 3.90295467 * 10^9 years

2. Calculation Time: alphanumeric character set = 62 (26 lowercase, 26 uppercase, 10 digits) | MD5crypt = 2.5 billion c/s
    a. 6-character: **23 sec**
        i. (62^6 / 2.5 * 10^9) = 22.72 sec
    b. 8-character: **24 hours**
        i. (62^8 / 2.5 * 10^9) * 60 secs / 1 min * 60 min / 1hr = 24.26 hours
    c. 10-character: **11 years**
        i. (62^10 / 2.5 * 10^9) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day * 365 days / 1 year = 10.65 years
    d. 12-character: **40921 years**
        i. (62^12 / 2.5 * 10^9) * 60 secs / 1 min * 60 min / 1hr * 24 hr/ 1 day * 365 days / 1 year = 40921.69 years

3. I think a password meter is a helpful tool when terming password strength but should be in combination with password suggestions. People do well with visual response and affirmation, but password meters have different metrics and cannot always determine a secure password. They are great at portraying how difficult a password looks to us but bit how difficult it is to crack. For example, a password of length 8 could be paasword$, which isn't particularly strong but would be considered ok by a password meter. An attacker could use an offline attack to easily guess this with a strong wordlist and their own GPUs. For me, a password length of 12 is good because even if a password isn't that strong, the longer it is that harder it is to crack.

4. Using SHA512 is an improvement because it has a larger size of 512 bits compared to 128 bits. It protects against preimage and collision attacks and has less known vulnerabilities compared to MD5 that has many rainbow tables against it.

5. Using a salt does increase password security because every password (user) has a unique salt that is hashed with the plaintext. This allows users to have the same password but different hashes and prevent attackers from using a rainbow table to try and match hashes.

6. It does not lessen it as offline attacks do not have the limitations that online attacks do and have time on their side. Even if a takes an indeterminate amount of time, offline attacks can wait it out until a password is cracked. The strength of the password is the only defense against offline attacks and that involves salting and hashing.