The Hong Kong Polytechnic University

COMP 3334 – Computer Systems Security (Semester 2, Winter 2024)

Lab 2 – Password cracking

Section 2 (updated after the lab)

In this lab, you will engage in hands-on exercises to enhance your understanding and proficiency in password cracking, and understand the value of a good password hash function.

Preliminaries

For this lab, you will need to download the hashcat tool and a dicitonary.

1. Hashcat can be downloaded directly at `https://hashcat.net/files/hashcat-6.2.6.7z`.
2. A common password wordlist can be downloaded at: `https://github.com/3ndG4me/KaliLists/raw/master/rockyou.txt.gz` (50.8MiB).

Extract all archives at a convenient location (e.g., Desktop). Next, launch a command prompt (Terminal but not PowerShell to avoid unfortunate circumstances if you don't know how to use it), and change the currently working directory to that location (e.g., `cd Desktop`). Check hashcat with "`hashcat-6.2.6\hashcat -V`". Add the extension `.exe` if working with PowerShell.

Exercises

Exercise 1: Password Cracking [60min]

Password hashing is the process of converting a plain password into a fixed-length string of characters, known as a hash, using a cryptographic hash function. This technique is used to securely store passwords in a system so that systems administrators as well as attackers cannot easily recover user passwords. For example, `42f749ade7f9e195bf475f37a44cafcb` is the MD5 hash for "Password123". While cryptographic hash functions are not reversible, it is possible to enumerate possible passwords, hash them and compare the hash value with the target password hashes. Such attacks are called brute-force or dictionary attacks.

Hashcat is a fast password recovery tool that helps break complex password hashes through brute-force and dictionary attacks. It is a flexible and feature-rich tool that offers many ways of finding passwords from hashes, and leverages the power of Graphical Processing Units (GPUs) to accelerate the task. Hashcat supports a wide range of hashing algorithms and attack modes, making it versatile for different password cracking scenarios.

Objective. In this exercise, we will learn how to crack password with Hashcat. Your task is to use Hashcat to crack some of the most commonly encountered hashes.

How hashcat works

Hashcat is used in the following way:

```
hashcat.exe [options]... hash|hashfile [dictionary|mask|directory]...
```

Options include the type of hash, attack mode, as well as other parameters. Hashcat can take a single hash or more commonly a hashfile as input, followed by dictionary files or a mask (discussed below).

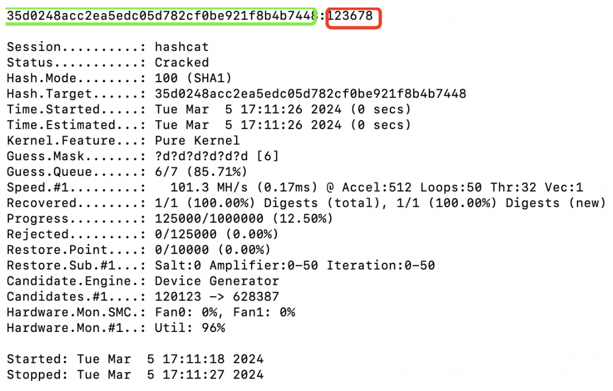
Hashcat's general syntax is as follows:

```
hashcat.exe -m type -a attack hashfile wordlist
```

Let's dissect this command.

- -m type: This is the option for the hash type. For instance, type 0 represents MD5, 100 represents SHA1. The value here would change depending on the hash type of the target hashes.
- -a attack: This is the attack mode. In this case, 0 stands for "straight" mode, a dictionary attack.
- hashfile: This file contains the hash or hashes you're trying to crack. It should be a text file with one hash per line.
- wordlist: This is your dictionary or list of potential passwords. Like the hash file, this should be a text file with one entry per line.

Below is hashcat's output for a mask attack against the MD5 hash of the password "123678". In particular, see the line "Recovered" to know how many hashes were recovered.



```
35d0248acc2ea5edc05d782cf0be921f8b4b7448:123678

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 100 (SHA1)
Hash.Target......: 35d0248acc2ea5edc05d782cf0be921f8b4b7448
Time.Started.....: Tue Mar  5 17:11:26 2024 (0 secs)
Time.Estimated...: Tue Mar  5 17:11:26 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?d?d?d?d?d?d [6]
Guess.Queue......: 6/7 (85.71%)
Speed.#1.........:   101.3 MH/s (0.17ms) @ Accel:512 Loops:50 Thr:32 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 125000/1000000 (12.50%)
Rejected.........: 0/125000 (0.00%)
Restore.Point....: 0/10000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-50 Iteration:0-50
Candidate.Engine.: Device Generator
Candidates.#1....: 120123 -> 628387
Hardware.Mon.SMC.: Fan0: 0%, Fan1: 0%
Hardware.Mon.#1..: Util: 96%

Started: Tue Mar  5 17:11:18 2024
Stopped: Tue Mar  5 17:11:27 2024
```

Figure 1: Hahcat's output when successfully recovering a password, followed by a session status

Hashcat supports plenty of hashing algorithms with various attack modes. Let's look at a few attack modes and see how they work. For more information about hashcat's capabilities, see the documentation at https://hashcat.net/wiki/.

Tip 1. After hashcat recovers a hash, it will remember the corresponding password and it will not try to guess the password again. Recovered passwords are stored in the pot file, located in the file `hashcat.potfile`. If you would like hashcat to ignore the potfile, append `--potfile-disable` to the options.

Tip 2. You can show the status of a cracking session (e.g., see the progress) by hitting `S`. You can also interrupt a cracking session by pressing `Q`.

Tip 3. If hashcat already cracked a hash and you simply want to see the result, use the option `--show`.

Attack modes

Brute-force and mask attacks. Also called mask attack, hashcat will try all possible combinations of characters, hash each combination and compare it to the target hash(es) until a match is found. This attack can be time-consuming and resource-intensive, especially with long and complex passwords. Hashcat takes as input a *mask* that describes what possible values each character may take. The mask consists of placeholders and special characters that represent different types of characters.

Here is an example of a mask attack for passwords between 1–7 characters (note the `-i` to increment the length), each of them can be any (`?a`) printable character; essentially a simple brute-force attack over all 1 to 7-character passwords.

```
hashcat.exe -a 3 -m type hashfile -i ?a?a?a?a?a?a?a
```

Here is a mask attack with placeholders to describe a six-character password starting with an uppercase letter (`?u`), followed by lowercase letters (`?l`), and ending with a digit (`?d`).

```
hashcat.exe -a 3 -m type hashfile ?u?l?l?l?l?d
```

Dictionary attack. The dictionary attack is performed by using a wordlist. A dictionary attack is also the default option in Hashcat. The better the wordlist is, the greater the chances of cracking the password.

```
hashcat.exe -a 0 -m type hashfile wordlist
```

Dictionary + rules attack. Rules in hashcat allow you to modify (we say *mangle*) the words from your wordlist in certain ways, which can greatly expand the number of potential passwords you are testing and will reflect common variations of a given password. For example, a rule could be to try the word in lowercase and uppercase or to append certain digits at the end of the word. This option `-r` allows you to specify a rules file.

```
hashcat.exe -a 0 -m type -r rule hashfile wordlist
```

Hashcat comes with numerous rule files. A common rule file is `rules\best64.rule`.

Your job

Complete the following tasks.

Task 1. For each of the SHA1 hash below (`-m 100`), find the corresponding password based on the knowledge given about the password. Choose an appropriate mask and launch a mask attack against them (`-a 3`).

- a 8-digit password and its hash value is `8807496363c5346684cba0d9101be93744a27652`
- a 8-char lowercase and its hash value is `e4099db8f15134974e750c132ede21bb7dcf7052`
- a password of 8 characters starting with uppercase and ending with a digit and lowercase in the middle, its hash value is `20d75f2aa30ae7101504d95bdda40ce235bb258c`

Note: Do not run parallel cracking sessions in this lab (or ever).

Task 2. You are given the following SHA1 hash (-m 100). The password consists of one common word but it is mangled. First, launch a dictionary attack using the RockYou dictionary downloaded earlier. You will see that it will not find the password. Next, try the same dictionary attack but with the `best64` rules (`-r`) located in the `rules` directory.

- hash value is `b5a0a46442a34a244f6ad69dcaa7353c99623eee`

Note: So far, the password recoveries have been blazing fast. This is because the hashing algorithm is very fast and hashcat is taking advantage of the GPU of the computer you are using to speed up hashing.

Task 3. You are given below 10 SHA1 hashes. Create a hash file with these hashes. Use a dictionary attack (with the RockYou wordlist), without rules to recover passwords.

```
b89eaac7e61417341b710b727768294d0e6a277b
23869b733fcd6665832f65258ac650e6ec89a4a7
d874a8f7ac3bcbf39c66863d459439c333f330a9
d60e716a021a387f29172b296a6eea0375c9d2a5
5d8d14b66bce7781b4c79d31ed209124d6ef34cc
56f29f94829b52a485efd7cfb7d1a5977e06d257
043a558250409758b64f73d07d7f06b3df654bc0
e7c87fcd8d574c31872d6a661f0a1cbb2fd68252
4233137d1c510f2e55ba5cb220b864b11033f156
6d1ca6e8ed5e3b25d5450d9942dd641b8a331bda
```

1. How many hashes are you able to recover?
2. What is the number of hashes that Hashcat needed to compute? (See the Progress line)
3. Knowing that the RockYou wordlist contains 14,344,385 passwords, did hashcat need to compute more hashes than the number of passwords in the list?

Task 4. Next, launch a dictionary attack again against the above SHA1 hashes, this time using the `best64` rules together with the RockYou wordlist, in an attempt to recover more hashes. Use the `--potfile-disable` option to let hashcat attempt to crack all hashes again. Answer the questions below.

1. How many hashes are you able to recover?
2. What is the number of hashes that Hashcat needs to compute in the worst case?

Task 5. You are given below 10 SHA1 *salted* hashes with their corresponding salt. Launch a dictionary attack using the RockYou wordlist without rules. Note that you will need to adjust the hash type to 110 (i.e., `sha1($pass.$salt)`).

```
2fc5a684737ce1bf7b3b239df432416e0dd07357:2014
df3ffbe04a16fcb290273f5f609cbb7aa30030c0:be526380efbc88f175b1856f32841
cd7025b34675b90d62ced746f1a3c433e09f798b:2c6734059663b0f4c570f4df0
956aae52c6734059663b0f4c570f4df0aa5e2e59:u6fcb290273f5f609c
ad155a7396fde10097372d24208022702a6c6d0b:502cf58613cf3dd108
5b9ecfabbe526380efbc88f175b1856f32841052:4792fbcb486ba7ddc5
d3bdde431cd740502cf58613cf3dd108d4b4a260:ee7dbebf493
63f756f3d1e4792fbcb486ba7ddc56c1894123b4:de10097372d
3178070d7bc61af3eb3cf9e4ebeacee7dbebf493:fuesgsdfg
7dfa83b1cb9e5753a39791e8f0f67714214edaa7:34675b90d62ced746f1a3c433e09f
```

Answer the questions below:

1. How many hashes are you able to recover?
2. What is the number of hashes that Hashcat needs to compute? (See the Progress line)
3. Knowing that the RockYou wordlist contains 14,344,385 passwords, did hashcat need to compute more hashes than the number of passwords in the list? Why?

Task 6. Similar to Task 4, use the `best64` rules together with the RockYou wordlist to recover all hashes, and answer the questions below. Use the `--potfile-disable` option to let hashcat attempt to crack all hashes again.

1. How many hashes are you able to recover? Notice they are the same passwords as in Task 4.
2. What is the number of hashes that Hashcat needs to compute in the worst case? Compare this number to your answer for Task 4. What do you notice?

Task 7. You are given a bcrypt hash of a simple password. Recover the corresponding password using a dictionary attack (no rules). Note: The hash type 3200 corresponds to bcrypt.

- If your lab computer has a NVIDIA GeForce GTX 1080: hash value is
  `$2y$10$axwReONDWPw5CEDOUWOd7e9Ck.Hzq4xHxDNArn91q3lcoojrkwUTa`
- If your lab computer has a NVIDIA GeForce GTX 3070: hash value is
  `$2y$10$3UrtlQG6nEwHyPqqXYgQj.DPWKfUuRW6.U8NHgL7D3OZG/LEcUWdy`

Answer the questions below:

1. What is the hashing speed?
2. How long did it take to crack this password? (be a little bit patient)
3. How long would it take hashcat to try all dictionary words? Get a session status (press `S`) while cracking the hash to see the expected time. Compare this result with your results in Task 6.

Need help?

Teaching Assistants you can contact about this lab:

1. Xuyuan CAI, xuyuan.cai@connect.polyu.hk, exercise 1

Bibliography

1. Hashcat's documentation. https://hashcat.net/wiki/
2. Picolet, J. (2017). Hash Crack: Password Cracking Manual v3.0. ISBN-10: 1793458618