FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

1. Course, Subject & Experiment Details

Practical No:	
Title:	**Hashcat**
Name of the Student:	**Warren Fernandes**
Roll No:	**8940**
Date of Performance:	**21-03-2022**
Date of Submission:	**04-04-2022**

Evaluation:

Sr. No.	Rubric	Grade
1	On time submission/completion (2)	
2	Preparedness (2)	
3	Skill (4)	
4	Output (2)	

Signature of the Teacher

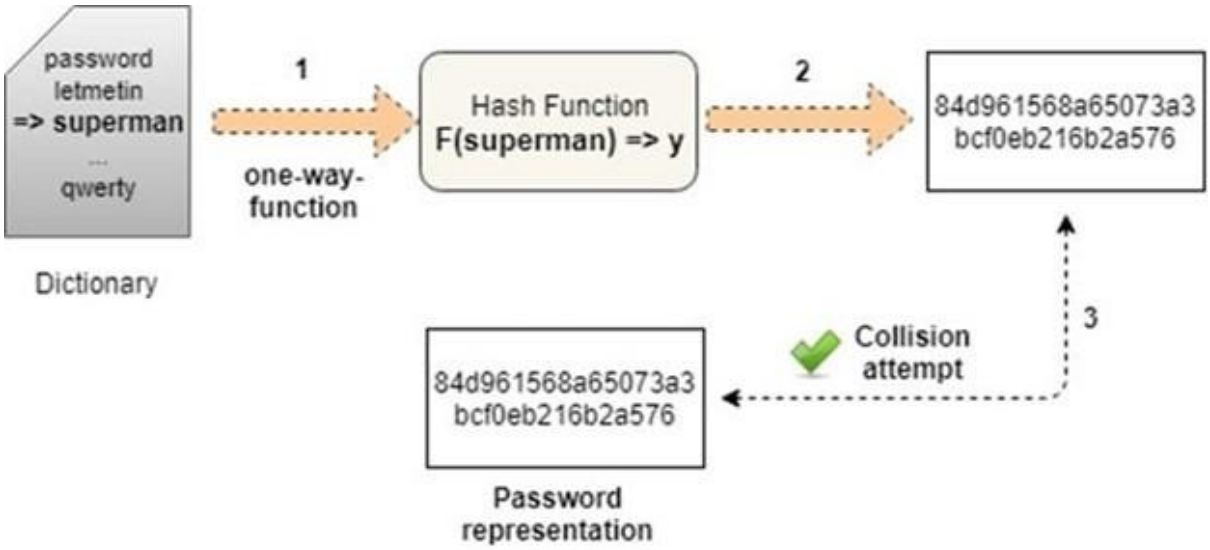
Hashcat:

Hashcat is a popular password cracker and designed to break even the most complex passwords representation. To do this, it enables the cracking of a specific password in multiple ways, combined with versatility and speed. Hashcat turns readable data into a garbled state (this is a random string of fixed-length size). Hashes do not allow someone to decrypt data with a specific key, as standard encryption protocols allow. Hashcat uses precomputed dictionaries, rainbow tables and even brute-force approaches to find an effective and efficient way to crack passwords.

How to crack hashes:

The simplest way to crack a hash is to try first to guess the password. Each attempt is hashed and then is compared to the actual hashed value to see if they are the same, but the process can take a long time

Dictionary and brute-force attacks are the most common ways of guessing passwords. These techniques make use of a file that contains words, phrases, common passwords and other strings that are likely to be used as a viable password.



It should be noted that there is no guaranteed way to prevent dictionary attacks or brute-force attacks.

Other approaches used to crack passwords:

1. Lookup tables: Hashes are pre-computed from a dictionary and then stored with their corresponding password into a lookup table structure.
2. Reverse lookup tables: This attack allows for a cyber attacker to apply a dictionary or brute-force attack to many hashes at the same time without having to pre-compute a lookup table.

3. Rainbow tables: Rainbow tables are a time-memory technique. They are similar to lookup tables, except that they sacrifice hash cracking speed to make the lookup tables smaller.
4. Hashing with salt: With this technique, the hashes are randomized by appending or prepending a random string, called a "salt." This is applied to the password before hashing.

Implementation:

1. Create a dictionary with MBD5 hashes

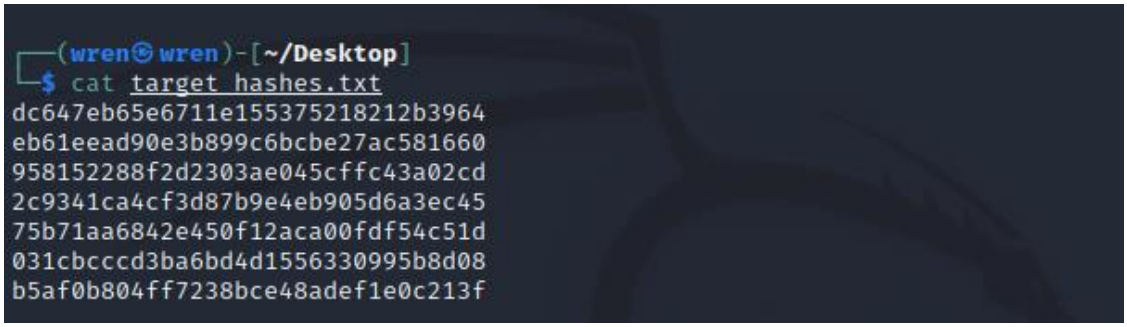
To start this demonstration, we will create multiple hash entries containing several passwords.

In detail, they will then be outputted to a file called "target_hashes." Each command should be executed in the terminal, as demonstrated below:

The -n option removes the new line added to the end of "Password." This is important as we don't want the new line characters to be hashed with our password. The part "tr –d ' -' " removes any characters that are a space or hyphen from the output.

2. Check password hashes

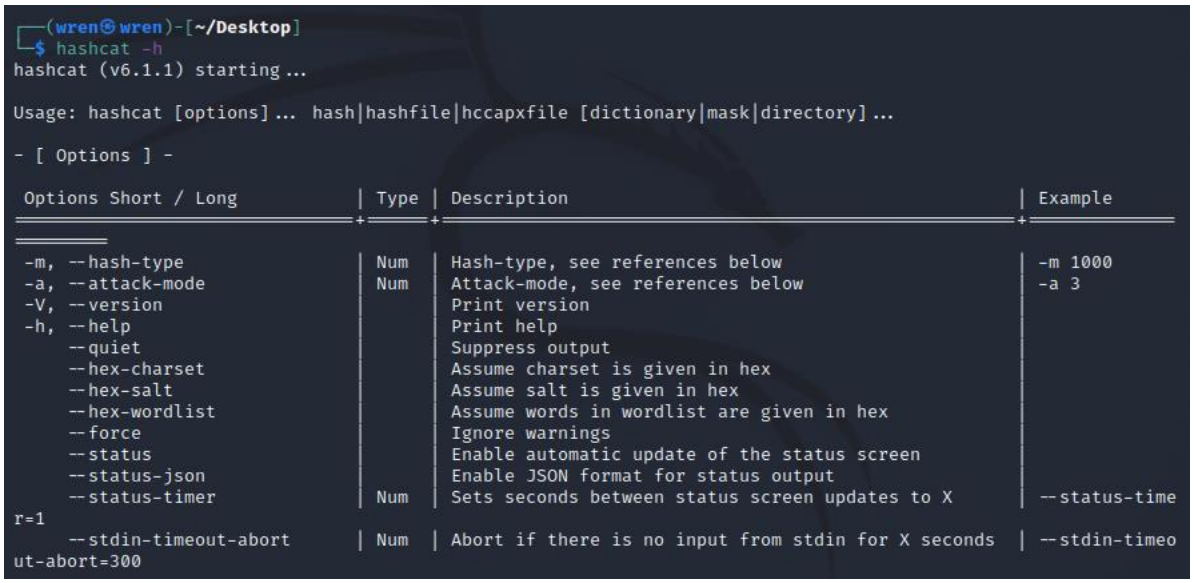
To do this, we need to type the following command line in the terminal:



3. Start Hashcat in Kali Linux

Hashcat can be started on the Kali console with the following command line: hashcat -h.

This is illustrated in the screenshot below:

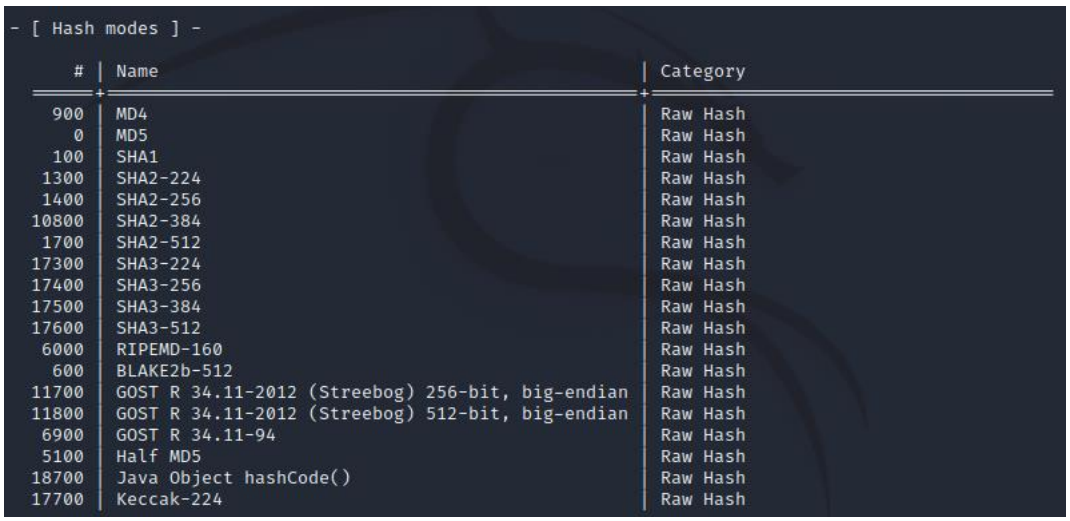


```
┌──(wren㉿wren)-[~/Desktop]
└─$ hashcat -h
hashcat (v6.1.1) starting ...

Usage: hashcat [options] ... hash|hashfile|hccapxfile [dictionary|mask|directory] ...

- [ Options ] -

 Options Short / Long          | Type | Description                                          | Example
================================+======+======================================================+=========
 -m, --hash-type               | Num  | Hash-type, see references below                      | -m 1000
 -a, --attack-mode             | Num  | Attack-mode, see references below                    | -a 3
 -V, --version                 |      | Print version                                        |
 -h, --help                    |      | Print help                                           |
     --quiet                   |      | Suppress output                                      |
     --hex-charset             |      | Assume charset is given in hex                       |
     --hex-salt                |      | Assume salt is given in hex                          |
     --hex-wordlist            |      | Assume words in wordlist are given in hex            |
     --force                   |      | Ignore warnings                                      |
     --status                  |      | Enable automatic update of the status screen         |
     --status-json             |      | Enable JSON format for status output                 |
     --status-timer            | Num  | Sets seconds between status screen updates to X      | --status-time
r=1
     --stdin-timeout-abort     | Num  | Abort if there is no input from stdin for X seconds  | --stdin-timeo
ut-abort=300
```

Some of the most important hashcat options are -m (the hashtype) and -a (attack mode). In general, we need to use both options in most password-cracking attempts when using Hashcat.

Hashcat also has specifically designed rules to use on a wordlist file. The character list can be customized to crack the password(s).

Finally, Hashcat provides numerous options for password hashes that can be cracked. This can be seen in the screenshot below:



```
- [ Hash modes ] -

      # | Name                                               | Category
  ======+====================================================+======================================
    900 | MD4                                                | Raw Hash
      0 | MD5                                                | Raw Hash
    100 | SHA1                                               | Raw Hash
   1300 | SHA2-224                                           | Raw Hash
   1400 | SHA2-256                                           | Raw Hash
  10800 | SHA2-384                                           | Raw Hash
   1700 | SHA2-512                                           | Raw Hash
  17300 | SHA3-224                                           | Raw Hash
  17400 | SHA3-256                                           | Raw Hash
  17500 | SHA3-384                                           | Raw Hash
  17600 | SHA3-512                                           | Raw Hash
   6000 | RIPEMD-160                                         | Raw Hash
    600 | BLAKE2b-512                                        | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian   | Raw Hash
  11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian   | Raw Hash
   6900 | GOST R 34.11-94                                    | Raw Hash
   5100 | Half MD5                                           | Raw Hash
  18700 | Java Object hashCode()                             | Raw Hash
  17700 | Keccak-224                                         | Raw Hash
```

4. Cracking the hashes

In the final step, we can now start cracking the hashes contained in the target_hashes.txt file. We will use the following command line, as illustrated below:

```
kali@kali:~$ cat hash.txt
8743b52063cd84097a65d1633f5c74f5
kali@kali:~$ hashcat -m 0 -a 0 hash.txt passwordlist.txt
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.5, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=============================================================================================================================
* Device #1: pthread-Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 1424/1488 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
```

-m 0 designates the type of hash we are cracking (MD5)

-a 0 designates a dictionary attack

-o cracked.txt is the output file for the cracked passwords

target_hashes.txt is our input file of hashes

/usr/share/wordlists/rockyou.txt is the absolute path to the wordlist file for this dictionary attack

6. Results

Finally, we have cracked five out of seven target hashes that were initially proposed. These can be seen below:

```
┌──(wren㉿wren)-[~/Desktop]
└─$ cat cracked.txt
dc647eb65e6711e155375218212b3964:Password
eb61eead90e3b899c6bcbe27ac581660:HELLO
958152288f2d2303ae045cffc43a02cd:MYSECRET
2c9341ca4cf3d87b9e4eb905d6a3ec45:Test1234
75b71aa6842e450f12aca00fdf54c51d:P455w0rd
031cbcccd3ba6bd4d1556330995b8d08:
b5af0b804ff7238bce48adef1e0c213f:
```

Postlab:

1. Describe about Hashcat and John the ripper?

Hashat is a particularly fast, efficient, and versatile hacking tool that assists brute-force attacks by conducting them with hash values of passwords that the tool is guessing or applying. When used for benign purposes, such as in penetration testing one's own infrastructure, it can reveal compromised or easy to guess credentials

Hashcat is, however, better known for being used for nefarious purposes. Hackers use Hashcat, readily available for download on all major operating systems, to automate attacks against passwords and other shared secrets. It gives the user the ability to brute-force credential stores using known hashes, to conduct dictionary attacks and rainbow tables, and to reverse engineer readable information on user behavior into hashed-password combination attacks.

John the Ripper is a free password cracking software tool. It was designed to test password strength, brute-force encrypted (hashed) passwords, and crack passwords via dictionary attacks. John the Ripper is a part of the Rapid7 family of penetration testing/ hacking tools. Also, John is already installed on Kali Linux.