LAB 30: Hashcat Password Cracking Lab Report

Lab Objective

The objective of this lab was to learn how to use Hashcat to crack password hashes. Hashcat is a password recovery tool that uses different attack modes to guess passwords and match them against hashed values. This helps security professionals identify weak passwords in a system.

Lab Purpose

A *hash* is a one-way cryptographic function that turns any text (like a password) into a fixed-length string of seemingly random characters. Hashes cannot be reversed directly, but they can be cracked by repeatedly hashing password guesses and checking if any match the target hash.

Hashcat automates this process and is extremely efficient at cracking hashes with large dictionaries of common passwords.

Lab Environment

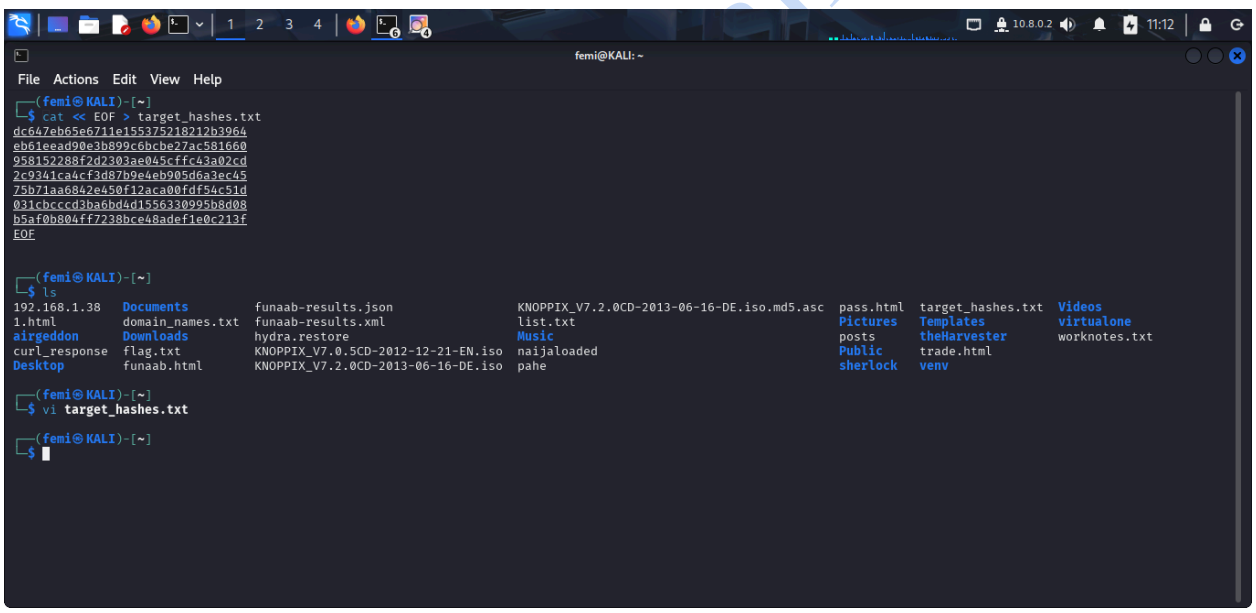
- Tool: Kali Linux (VM environment)
- Tool: Hashcat

Lab Steps

Task 1: Create the Hash File

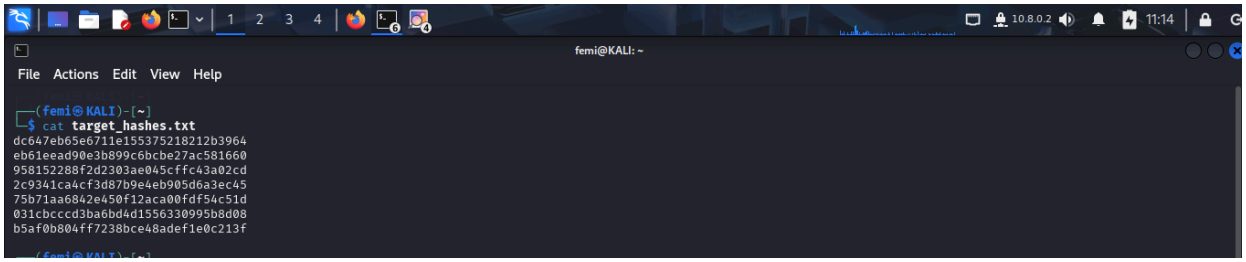
I created a text file named `target_hashes.txt` containing seven different MD5 hashes that need to be cracked. The command used was:

```
cat << EOF > target_hashes.txt
dc647eb65e6711e155375218212b3964
eb61eead90e3b899c6bcbe27ac581660
958152288f2d2303ae045cffc43a02cd
2c9341ca4cf3d87b9e4eb905d6a3ec45
75b71aa6842e450f12aca00fdf54c51d
031cbcccd3ba6bd4d1556330995b8d08
b5af0b804ff7238bce48adef1e0c213f
EOF
```



I confirmed the contents using:

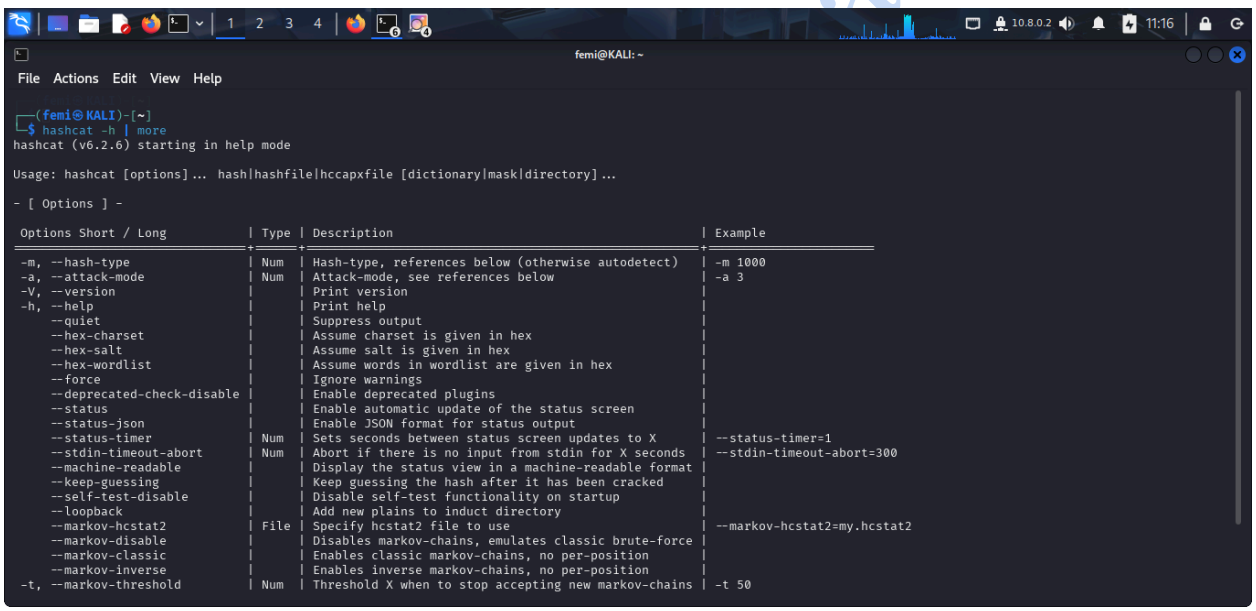
```
cat target_hashes.txt
```



Task 2: Explore Hashcat Options

I explored Hashcat's help options to learn about available attack modes and hash types. This was done with:

```
hashcat -h | more
```



I scrolled through using the **Space** key and exited with **CTRL+C**. This showed me that:

- `-m` sets the hash type (e.g. MD5 is `-m 0`)
- `-a` sets the attack mode (dictionary attack is `-a 0`)

Task 3: Prepare the Wordlist

I searched for the `rockyou.txt` password wordlist using:

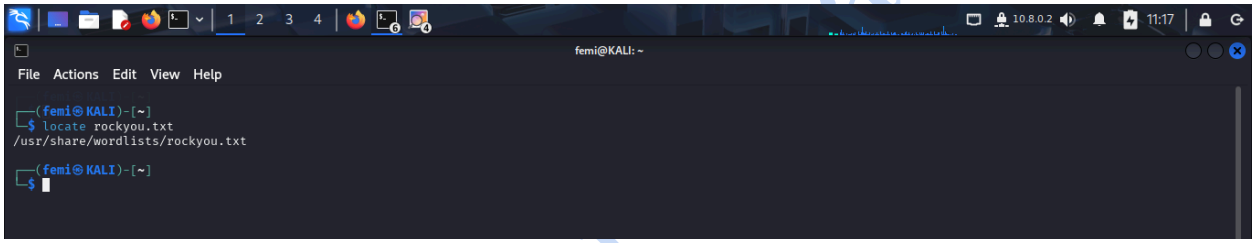
```
locate rockyou.txt
```

If it was zipped as `rockyou.txt.gz`, I unzipped it:

```
sudo gunzip /usr/share/wordlists/rockyou.txt.gz
```

After unzipping, I confirmed:

```
/usr/share/wordlists/rockyou.txt
```



was available for my dictionary attack.

Task 4: Run Hashcat

I navigated to my home directory and launched Hashcat with the following command:

```
hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt
/usr/share/wordlists/rockyou.txt
```

```
┌──(femi㉿KALI)-[~]
└─$ hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================
* Device #1: cpu-penryn-Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz, 1438/2941 MB (512 MB allocatable), 2MCU

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

Hashes: 7 digests; 7 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache built:
```

Explanation of options:

- `-m 0`: MD5 hash type
- `-a 0`: dictionary attack
- `-o cracked.txt`: output file for cracked passwords
- `target_hashes.txt`: file with the target hashes
- `/usr/share/wordlists/rockyou.txt`: dictionary wordlist

Hashcat then tried each password in the wordlist, hashed it, and compared it to the target hashes.

Task 5: View Results

When the attack finished, I viewed the cracked passwords:

```
cat cracked.txt
```

This displayed any successful password matches, showing which plaintext passwords corresponded to the given hashes.

Lab Conclusion

In this lab, I successfully used Hashcat to perform a dictionary-based attack against MD5 password hashes. This demonstrated how attackers (or penetration testers) can recover weak passwords using wordlists and hash comparison.

This lab reinforced why:
✅ strong passwords
✅ salted hashes
✅ and good password policies
are critical to protect user credentials.

End of Submission