Hash Cracking

Summary

Hashcat

Hashcat Install

```
apt install cmake build-essential -y
apt install checkinstall git -y
git clone https://github.com/hashcat/hashcat.git && cd hashcat && make -j 8 && make
install
```

1. Extract the hash
2. Get the hash format: https://hashcat.net/wiki/doku.php?id=example_hashes
3. Establish a cracking stratgy based on hash format (ex: wordlist -> wordlist + rules -> mask -> combinator mode -> prince attack -> ...)
4. Enjoy plains
5. Review strategy
6. Start over

Dictionary

> Every word of a given list (a.k.a. dictionary) is hashed and compared against the target hash.

```
hashcat --attack-mode 0 --hash-type $number $hashes_file $wordlist_file -r $my_rules
```

- Wordlists

  - packetstorm
  - weakpass_3a
  - weakpass_3
  - Hashes.org
  - kerberoast_pws
  - hashmob.net
  - clem9669/wordlists

- Rules
    - One Rule to Rule Them All
    - nsa-rules
    - hob064
    - d3adhob0
    - clem9669/hashcat-rule

Mask attack

Mask attack is an attack mode which optimize brute-force.

> Every possibility for a given character set and a given length (i.e. aaa, aab, aac, ...) is hashed and compared against the target hash.

```
# Mask: upper*1+lower*5+digit*2 and upper*1+lower*6+digit*2
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?l?l?d?d
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?l?l?l?d?d
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?u?l?l?l?l?l?d?d?1
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?u?l?l?l?l?l?l?d?d?1

# Mask: upper*1+lower*3+digit*4 and upper*1+lower*3+digit*4
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?d?d?d?d
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?l?d?d?d?d
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?l?l?d?d?d?d
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?u?l?l?l?d?d?d?d?1
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?u?l?l?l?l?d?d?d?d?1

# Mask: lower*6 + digit*2 + special digit(+!?*)
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?l?l?l?l?l?l?d?d?1
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 "*+!??" ?l?l?l?l?l?l?d?d?1?1

# Mask: lower*6 + digit*2
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 /content/hashcat/masks/8char-1l-1u-1d-1s-compliant.hcmask
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 -1 ?l?d?u ?1?1?1?1?1?1?1?1

# Other examples
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?a?a?a?a?a?a?a?a
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?a?a?a?a?a?a?a
hashcat -m 1000 --status --status-timer 300 -w 4 -O /content/*.ntds -a 3 ?u?l?l?l?l?l?d?d?d?d
hashcat --attack-mode 3 --increment --increment-min 4 --increment-max 8 --hash-type
```

```
$number $hashes_file "?a?a?a?a?a?a?a?a?a?a?a?a"
hashcat --attack-mode 3 --hash-type $number $hashes_file "?u?l?l?l?d?d?d?d?s"
hashcat --attack-mode 3 --hash-type $number $hashes_file "?a?a?a?a?a?a?a?a"
hashcat --attack-mode 3 --custom-charset1 "?u" --custom-charset2 "?l?u?d" --custom-
charset3 "?d" --hash-type $number $hashes_file "?1?2?2?2?2?3"
```

Shortcut	Characters
?l	abcdefghijklmnopqrstuvwxyz
?u	ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d	0123456789
?s	!"#$%&'()*+,-./:;<=>?@[]^_`{}~
?a	?l?u?d?s
?b	0x00 - 0xff

John

John Usage

```
# Run on password file containing hashes to be cracked
john passwd

# Use a specific wordlist
john --wordlist=<wordlist> passwd

# Use a specific wordlist with rules
john --wordlist=<wordlist> passwd --rules=Jumbo

# Show cracked passwords
john --show passwd

# Restore interrupted sessions
john --restore
```

Rainbow tables

> The hash is looked for in a pre-computed table. It is a time-memory trade-off that allows cracking hashes faster, but costing a greater amount of memory than traditional brute-force of dictionary attacks. This attack cannot work if the hashed value is salted (i.e. hashed with an additional random value as prefix/suffix, making the pre-computed table irrelevant)

Tips and Tricks

- Cloud GPU
  - penglab - Abuse of Google Colab for cracking hashes. 🐧
  - google-colab-hashcat - Google colab hash cracking
  - Cloudtopolis - Zero Infrastructure Password Cracking
  - Nephelees - also a NTDS cracking tool abusing Google Colab
- Build a rig on premise

    - Pentester's Portable Cracking Rig - $1000
    - How To Build A Password Cracking Rig - 5000$
- Online cracking
    - Hashes.com
    - hashmob.net: great community with Discord
- Use the `loopback` in combination with rules and dictionary to keep cracking until you don't find new passsword:
  ```
  hashcat --loopback --attack-mode 0 --rules-file $rules_file --hash-type $number
  $hashes_file $wordlist_file
  ```

Online Cracking Resources

- ~~hashes.com~~
- crackstation
- Hashmob

References

- Cracking - The Hacker Recipes
- Using Hashcat to Crack Hashes on Azure
- miloserdov.org hashcat
- miloserdov.org john