Remote Exploitation
Unit_4

Pallavi B

Assistant Professor, AI&ML Dept,

BMSCE

Understanding Network Protocols

As a penetration tester, most of the times, you would come across only three protocols:

1. TCP (Transmission Control Protocol)
2. UDP (User Datagram Protocol)
3. ICMP (Internet Control Messaging Protocol)

Transmission Control Protocol

Most of the protocols that we encounter in our daily lives are based upon TCP.

- TCP is used whenever we need to perform a reliable communication between a client and a server.
- TCP performs a reliable communication via the three-way handshake,
- Common examples are FTP, SMTP, Telnet, and HTTP

User Datagram Protocol

UDP is the exact opposite of TCP.

It is used for faster communications. An example would be for video streaming, such as Skype (VOIP) communication.

The advantage of this protocol over TCP is that it's much faster and efficient.

The disadvantage of UDP is that it does not guarantee that the packet will reach the destination, since it does not perform the three-way handshake, thus causing reliability issues.

Some of the common UDP protocols that we will run into as a penetration tester are DNS and SQL Server

Internet Control Messaging Protocol

ICMP runs upon layer 3 (network layer) of the OSI model, unlike TCP and UDP, which runs upon layer 4.

The protocol was developed for troubleshooting error messages on a network. It is a connectionless protocol, which means that it gives us no guarantee that the packet will reach the destination.

Common applications that use ICMP are “Ping” and “Traceroute.”

Server Protocols

All server protocols are divided into two basic categories:

1. Text-based protocols
2. Binary protocols

Text-Based Protocols

Text-based protocols are human readable protocols

as a penetration tester, need to spend most of your time as they are very easy to understand.

Common examples of text based protocols are HTTP, FTP, and SM

Binary Protocols

Binary protocols are not human readable and are very difficult to understand; they are designed for efficiency across the wire.

The popular text-based protocols such as

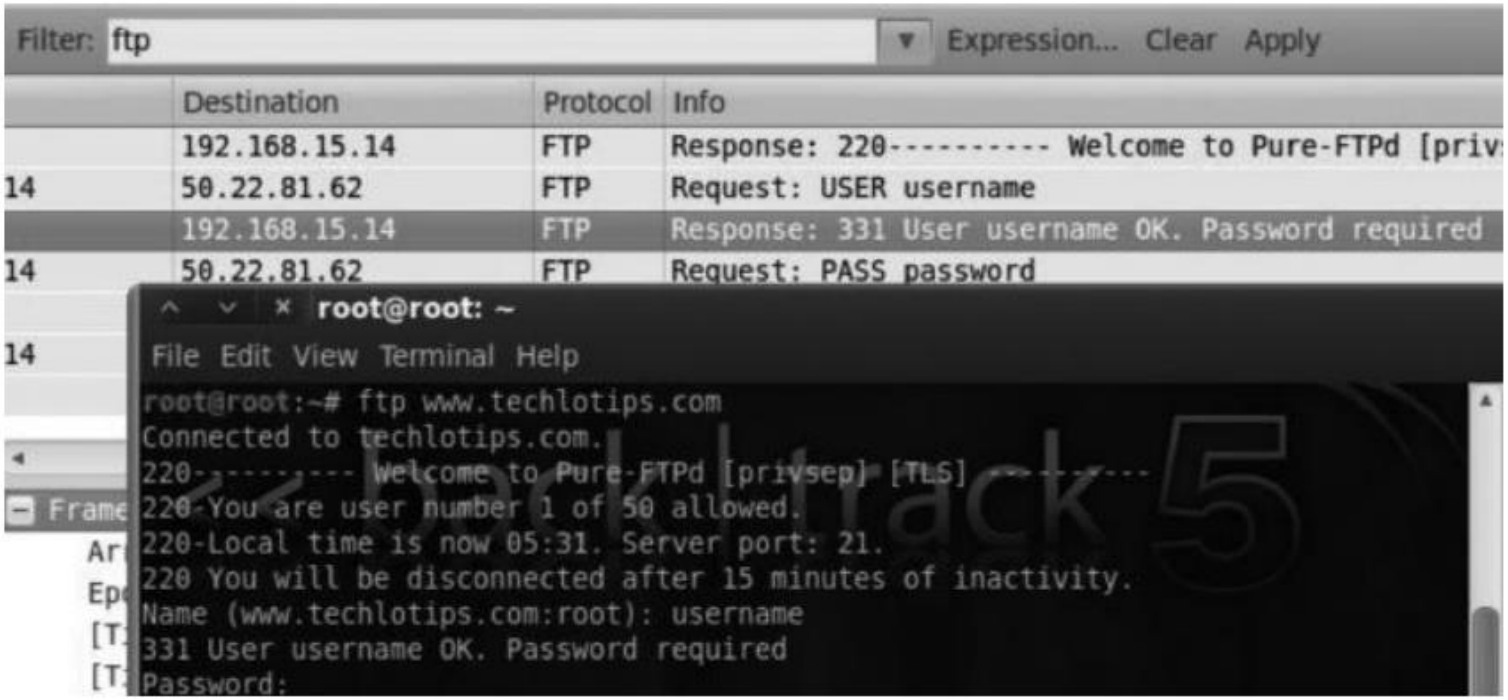
- FTP,
- HTTP,
- SMTP

FTP

FTP stands for File Transfer Protocol; it runs on port 21.

FTP is commonly used for uploading/ downloading files from a server. FTP, in my opinion, is the weakest link in a network because it’s unencrypted, meaning that anybody on a local network can use a network sniffer to capture all the communication.

The following image shows the Wireshark capture when I was trying to log in to an FTP server. The username was set to “username” and the password to “password”, as you can clearly see, the username and the password are unencrypted and sent in plain text



SMTP

SMTP stands for Simple Mail Transfer Protocol. It runs on port 25.

It is used in most of the mailing servers nowadays.

As a penetration tester, we will encounter SMTP a lot as it’s always exposed on the Internet and would mostly contain sensitive information.

HTTP

The protocol you are using to do this is HTTP It runs upon port 80. It’s a fundamental of the web.

Overview of Brute Force Attacks

Commonly, brute force attacks are divided into three categories:

Traditional Brute Force

Dictionary Attacks

Hybrid Attacks

In a traditional brute force attack, you will try all the possible combinations to guess the correct password. This process is very usually time consuming; if the password is long, it will take years to brute-force. But if the password is short, it can give quick results

It is much faster than traditional brute force attacks and is the recommended approach for penetration tests.

The only downside is that if the password is not available in the list, the attack won’t be successful.

Hybrid brute force attacks are a combination of both traditional brute force attack and dictionary based attack. The idea behind a hybrid attack is that it will apply a brute force attack on the dictionary list

Common Target Protocols

we will commonly come across only the following network protocols/services:

- FTP
- SSH
- SMB
- SMTP
- HTTP
- RDP
- VNC
- MySQL
- MS SQL

Tools of the Trade

THC Hydra

THC hydra is one of the oldest password cracking tools developed by “The Hackers Community.”

Hydra has the most protocol coverage than any other password cracking tool as per my knowledge, and it is available for almost all the modern operating systems.

Basic Syntax for Hydra

Here is the basic syntax for hydra to brute-force

Example with Username Set to "administrator"

Hydra –L administrator **–P** password.txt **<target ip > <service>**

Example with Username Set to username list

Hydra –L users.txt **–P** password.txt **<target ip > <service>**

Note: We need to define the location of the username/password list file for hydra to work.

Cracking Services with Hydra

Let’s start by cracking an ftp password with hydra, which is one of the most commonly found services.

For that, we need an ftp service to be running on the target. Consider the target machine having an IP address of 192.168.75.40.

Hydra GUI

All you need to do is to type “Xhydra” or “HydraGTK” from the command line to explore it.

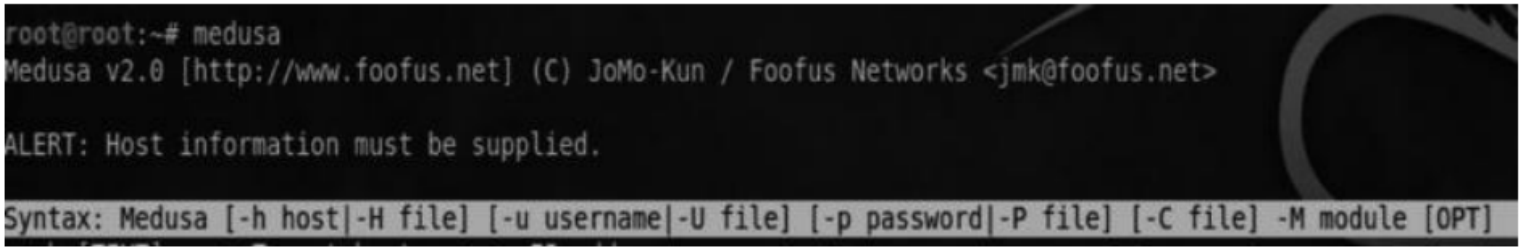
Medusa

Medusa is an alternative to Hydra and is a really fast password cracking tool.

It is a parallel brute force tool just like Hydra. However, it is much more stable and faster than Hydra because it uses “Pthread,” meaning that it won’t necessarily duplicate the information, whereas Hydra uses “fork” for parallel processing.

Basic Syntax

To check for available options in Medusa, we will execute "Medusa" command without parameters.

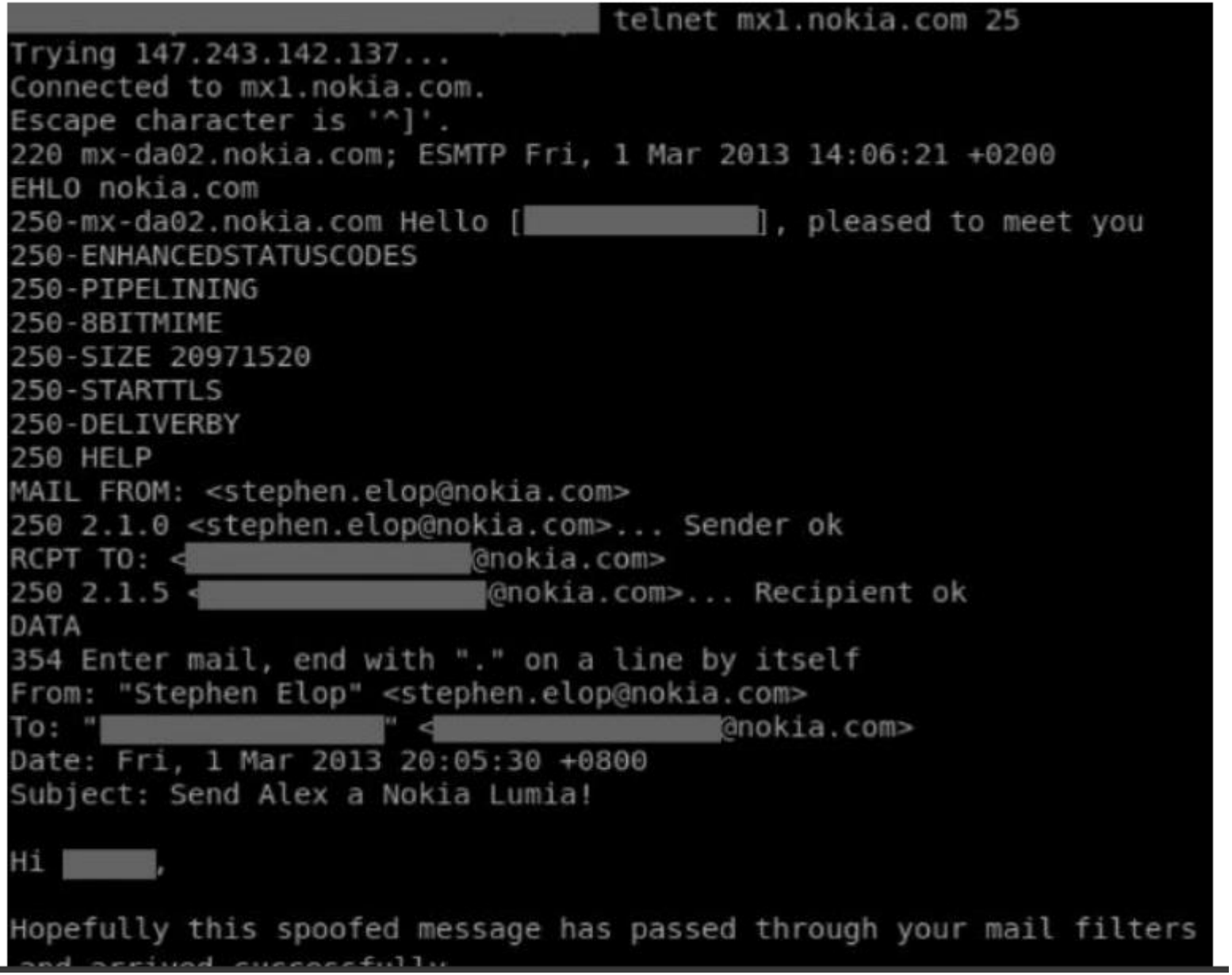


As you can see from the screenshot, we need four parameters in order to run Medusa.

–h = Hostname to attack
–u = Username to attack
–P = Password file
–M = Service to attack

Real-Life Example

screenshot explains the whole story.



Attacking SQL Servers

Attacking SQL Servers

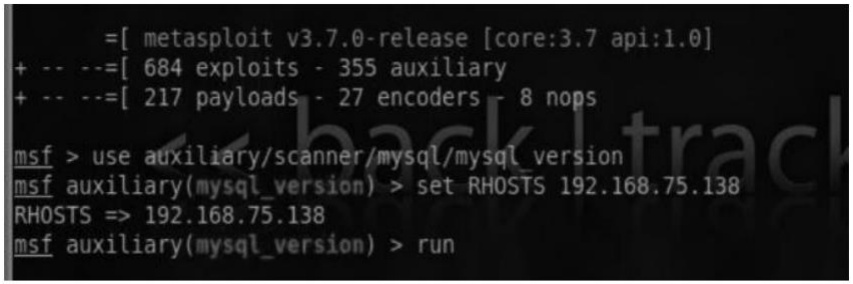
MySQL Servers

MySQL servers are the most widely used databases in modern web applications. You are likely to find them in 8 out of 10 web applications that you perform penetration test against. One of the first attacks is to, of course, test for weak credentials that can give us immediate access to the SQL database.

Fingerprinting MySQL Version

We have a built-in auxiliary module in Metasploit that could help us fingerprint the exact version of MySQL being used. The module is called mysql _ version. All we need to do is supply only one input: the target IP that is running the SQL server.

Commands:
`msfconsole` – To launch metasploit
`use auxiliary/scanner/mysql/mysql _ login` (Within Metasploit Console)
`set RHOSTS` <Target IP>
`Run`



MS SQL Servers

MS SQL is the Microsoft version of SQL server. Unlike in MySQL servers, there are various other attacks we can perform against some old versions of MS SQL server, for example, in SQL server 2000. The stored procedure XP _ CMDSHELL is enabled by default, so we can take advantage of it and execute some commands. We will discuss this when we get to exploiting SQL injection attacks with web applications

Introduction to Metasploit

Metasploit is a free open-source software that could be used to automate lots of complex tasks. Since Metasploit is a huge framework,

Metasploit Interfaces

There are several interfaces for Metasploit. It's available in all forms, that is, interactive, command line, and GUI. Let's take a look at some of its popular interfaces:

MSFConsole

In order to launch msfconsole, all we need to do is enter “msfconsole” command in the shell, and it will be launched.

MSFcli

Another interface in the Metasploit Framework is the "MSFcli" interface, though it's not interactive like msfconsole. An advantage in MSFcli is that we can redirect output from other tools as well as redirect MSFcli's output to other tools.

To launch MSFcli, we need to execute "msfcli" command in the shell followed by the options that we would like to use.

MSFGUI

MSFGUI was the first official GUI version for Metasploit, but it's not frequently updated any more. Therefore, we won't discuss it in this book. What we will discuss next is another GUI named "Armitage," which is updated frequently.

Armitage

Armitage is a powerful GUI interface for Metasploit; it's fully interactive and also comes preinstalled with BackTrack. Later in this section, we will look at how similar tasks can be accomplished faster with Armitage than with Metasploit.

Metasploit Utilities

Over the years, there have been a couple of utilities introduced with Metasploit. The main purpose of introducing these utilities was to use the components *outside* the Metasploit Framework *within* it.

The most popular ones are MSFpayload and MSFencode. Let's look at them in brief. We will learn how to use them in the "Client Side Exploitation" chapter (Chapter 8).

MSFPayload

MSFPayload is used for generating payloads, shell codes, and other executables. A payload is the code that you want to run on the victim's machine after the exploit is completed, whereas a shell code is usually part of the payload written in the assembly language.

MSFEncode

MSFEncode utilizes different methods to encode payloads so that they don't end up getting detected by antivirus engines. Almost all encoding techniques would fail to get past antiviruses, but with some tweaking, we can bypass most of them. Anyway, in the end our main goal is to just get past the particular antivirus that the victim is using.

MSFVenom

MSFVenom is a newly introduced feature in the Metasploit Framework. It is a combination of both MSFpayload and MSFencode. With MSFvenom, we can perform both create/encode shell

Client Side Exploitation

Client Side Exploitation Methods

1. Attack Scenario 1: E-Mails Leading to Malicious Attachments
2. Attack Scenario 2: E-Mails Leading to Malicious Links
3. ***Attack Scenario 3: Compromising Client Side Update***
4. ***Attack Scenario 4: Malware Loaded on USB Sticks***

1. E-Mails with Malicious Attachments

1. Creating a Custom Executable

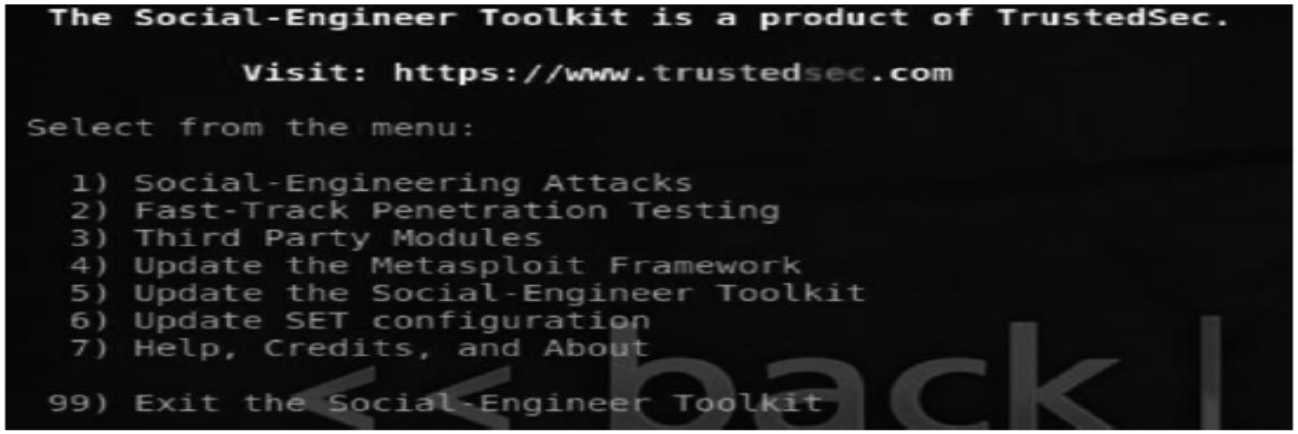
- This attack can be a bit difficult to accomplish, as you need to convince the victim to execute
- your .exe file. Another major hurdle would be the victim's antivirus, which you need to bypass.

2. Creating a Backdoor with SET

- SET, in my opinion, is one of the best tools to perform client side attacks. It harnesses the power
- of Metasploit to carry out a wide variety of client side attacks. In this chapter, we will use the SET
- to perform multiple client side attacks. So let us start by creating a backdoor from SET.

Step 1—Navigate to the /pentest/exploits/set directory in BackTrack and run the following command from the /set directory:

```
root@bt:~# cd/pentest/exploits/set
root@bt:~#./set
```

Step 2—Press "1" and it will display all the social engineering attack vectors and then press the fourth option that states "Create a payload and a listener."

Note: It is always good practice to update the SET before using it, which you can do by pressing "5" on your keyboard.

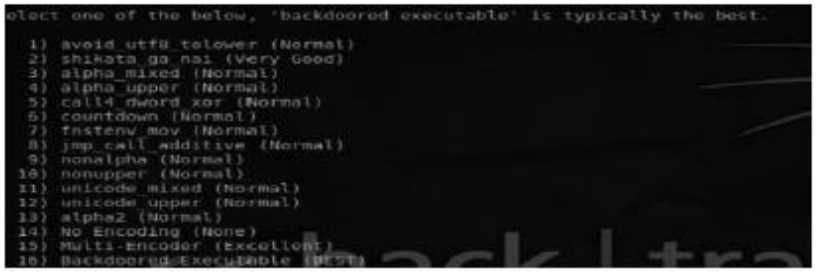
Step 3—Next, it will ask for your reverse IP, which in this case is my local IP address for my BackTrack box. If you are attacking over the Internet, you need to do port forwarding on your router, which we will discuss in Attack Scenario 2.

```
set> 4
set:payloads> Enter the IP address for the payload (reverse):192.168.75.144
```

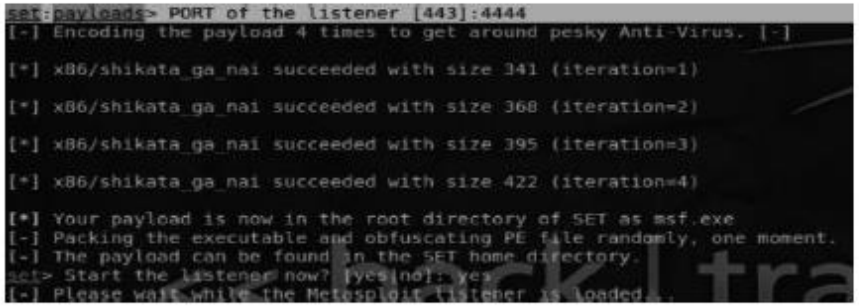
Step 4—Next, you need to choose the appropriate payload. You can choose any one of them based on your requirements. For the sake of simplicity, I would be choosing the first one, "Windows Shell Reverse_TCP", which will send a reverse shell back to my IP, which in this case is 192.168.75.144.

```
set:payloads> Enter the IP address for the payload (reverse):192.168.75.144

What payload do you want to generate:

  Name:                                        Description:

    1) Windows Shell Reverse_TCP               Spawn a command shell on victim an
d send back to attacker
```

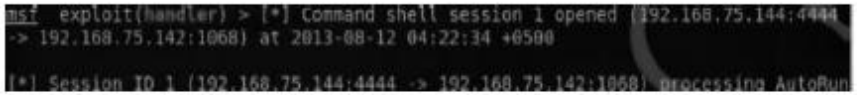
Step 5—Next, it will ask you what type of encoding you want. In this case, we will use shikata_ga_nai. Notice that the SET has suggested that "backdoored executable" is the best type of encoding. In real-world scenarios, you need to encode them multiple times before you get past multiple antiviruses.



Step 6—Next, it will ask you on what port to listen for connections. In my case, I would choose port "4444"; you can select any port you want. This might take some time, since it would start up Metasploit in the back end, which itself takes much time to launch.



Step 7—Now, our backdoor would be created on root directory `our/pentest/exploits/set` named msf.exe. Now you need to convince the victim to execute it inside his system; once he executes it, you will have a session opened.



You can now interact with the shell, by using the following command:

```
sessions -i 1
```

Using an executable may not be the best method, so we will talk about an approach that is more useful in real-world scenarios.

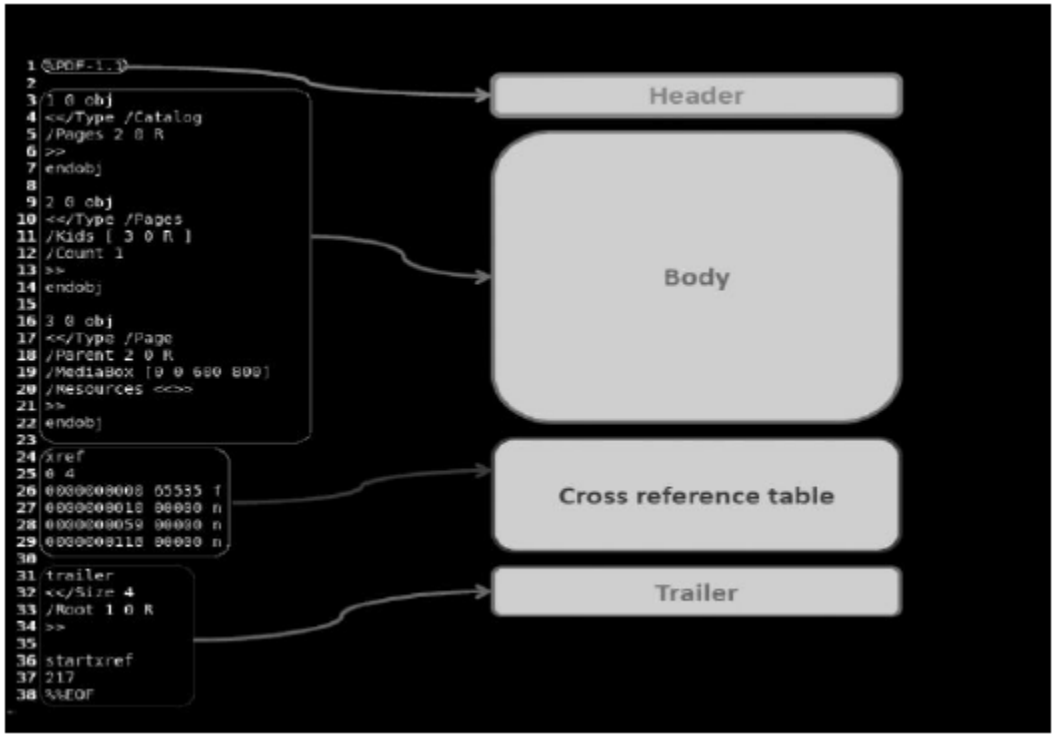
PDF Hacking

- PDF hacking is one of the topics on ethical hacking and penetration testing, PDF hacking became one of my favorite subjects in ethical hacking.
- Lots of penetration testers are unaware of the power of PDFs and their effectiveness in penetration tests.
- PDF hacking and PDF reconnaissance are most of the times ignored by penetration
- testers, even those at an advanced level.

Introduction

- Before we actually get into creating a malicious PDF document, we will learn about the basics , which include the structure of a PDF document, using it for performing reconnaissance.
- The language of PDF is very descriptive, which gives us a wide variety of attack surface, so before jumping into the reconnaissance, first, let's look at the basic structure of a PDF file.
- In-case if you open up a PDF document inside wordpad or a notepad editor, you would see the following sections:

  1. Header
  2. Body
  3. Cross reference table
  4. Trailer


%PDF-1.1
1 0 obj
<</Type /Catalog
/Pages 2 0 R
>>
endobj
2 0 obj
<</Type /Pages
/Kids [3 0 R]
/Count 1
>>
endobj
3 0 obj
<</Type /Page
/Parent 2 0 R
/MediaBox [0 0 600 800]
/Resources <<>>
>>
endobj
xref
0 4
0000000000 65535 f
0000000010 00000 n
0000000059 00000 n
0000000110 00000 n
trailer
<</Size 4
/Root 1 0 R
>>
startxref
217
%%EOF
Header
Body
Cross reference table
Trailer

1.Header

- The header, indicated in green, specifies the version of the PDF document, %PDF-1.1 in this case.
- The versions may vary from 1.0 to 1.7.

2.Body

- The body is the part of a PDF document where all the objects, names, etc., are located.

3.Cross Reference Table

- The cross reference table is indicated in purple. It has a highly defined structure and specifies
- where an object is located in a PDF document.

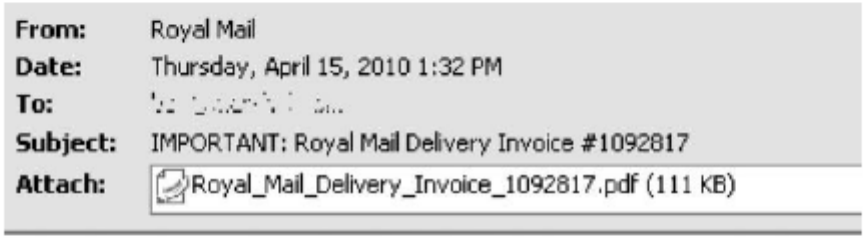
4. Trailer

- The trailer will always begin from %%EOF as PDFs are always rendered from bottom up, so
- whenever you open up, it will start reading it from %%EOF and then it will jump and start to locate the line “Start Xref”, which is always followed by a number.

PDF Launch Action

PDF launch action is one of the most useful features of a PDF document. With PDF launch action, you can actually launch other things along with PDF. PDF launch action was widely abused in the older version of Adobe Reader in which PDF launch action was used to spread malware and botnets such as Zeus.

This discovery was first made by M86 Security researchers. According to them, users would receive an e-mail with the subject "Royal mail delivery invoice."



Creating a PDF Document with a Launch Action

Let's see how we can use the launch action in the PDF document. Experimenting with PDF launch action will be more convenient if you have an empty PDF file or one with minimum text. Once you have created a blank PDF, open it in Notepad or WordPad. It will look something similar to the following:

Note: Before you perform the exercise, make sure you download Adobe Reader 9.3.2 as the launch action is not patched. You can get it from oldapps.com

Controlling the Dialog Boxes

From what we have done so far, it's quite clear what we are executing on the victim's machine, which will make the victim suspicious and will prevent him from launching it.

So in order to get things going, we need to control the dialog box. There are several methods to do that, but we will use the most effective one. You just need to add the following lines after /F (cmd.exe):

> /p (The file has too many errors in it, In order for windows to open your file properly, Click "Ok" or if you wish to terminate this program click "Cancel")

The `/P` command is used to pass an additional parameter along with /F. Now after adding this line, you can save your PDF and launch it again. You will see that the calc.exe executing command has moved upward.

You might still be wondering of what use is a PDF launch action, but you will soon find out how dangerous PDF attacks can be when we come to the exploitation part.

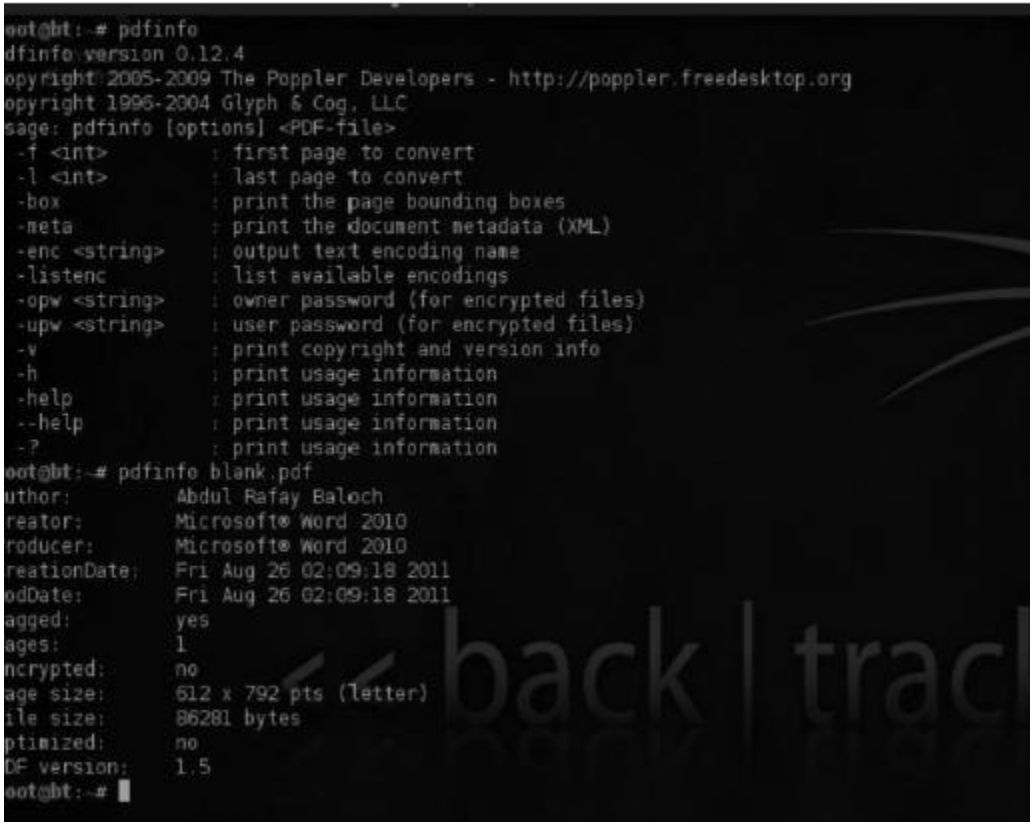
Tools of the Trade

- There are a couple of tools you can use to collect metadata from PDF, namely, metagoofil and PDFINFO
- ***PDFINFO***

PDFINFO is a command line Unix-based tool used to gather information about a particular PDF document. The information includes the operating system, PDF reader version, etc. Now, let's begin experimenting with PDFINFO.

We will use the blank.pdf we created in the launch action exercise. So let's say that we want to gather information about blank.pdf.

'DFINFO "Your PDF Document"



PDFTK

- PDFTK is another useful tool for generating PDF files, which has multiple functionalities like combining and compressing PDF files. It's not very efficient though when compared to Origami Framework, which could be used to generate PDF files more conveniently.


```
root@bt:~# pdftk
SYNOPSIS
      pdftk <input PDF files | - | PROMPT>
            [input_pw <input PDF owner passwords | PROMPT>]
            [<operation> <operation arguments>]
            [output <output filename | - | PROMPT>]
            [encrypt_40bit | encrypt_128bit]
            [allow <permissions>]
            [owner_pw <owner password | PROMPT>]
            [user_pw <user password | PROMPT>]
            [flatten] [compress | uncompress]
            [keep_first_id | keep_final_id] [drop_xfa]
            [verbose] [dont_ask | do_ask]
      Where:
            <operation> may be empty, or:
            [cat | attach_files | unpack_files | burst |
             fill_form | background | stamp | generate_fdf |
             multibackground | multistamp |
             dump_data | dump_data_fields | update_info]

      For Complete Help: pdftk --help
root@bt:~#
```

If you would like to know more about this tool, visit http://www.pdflabs.com/docs/pdftk-cli-examples/

Attacking with PDF

- The exploits may range from buffer overflows to misuse of the configurations, such as PDFlaunch action discussed earlier. As you can see from the following screenshot that PDF exploits are
- generally been broken down into two categories:
    1. Fileformat exploits
    2. Browser exploits

Fileformat Exploits

- File format exploits are one of the most efficient and most common PDF exploits used by penetration testers.
- File format exploits enable you to create a malicious PDF file, which once executed by the victim will give the shell to the attacker. Using exploits present in Metasploit, once you infect a single file on the victim's computer, it's possible for you to infect all other PDF files on that computer.

A penetration test (pen test)

A penetration test (pen test) is an authorized simulated attack performed on a computer system to evaluate its security. Penetration testers use the same tools, techniques, and processes as attackers to find and demonstrate the business impacts of weaknesses in a system.

Penetration tests usually simulate a variety of attacks that could threaten a business. They can examine whether a system is robust enough to withstand attacks from authenticated and unauthenticated positions, as well as a range of system roles. With the right scope, a pen test can dive into any aspect of a system.

Browser Exploits

- Browser exploits are not used much by pen testers. However, they can prove beneficial in some situations. Here is how PDF browser exploit works:

1. The attacker chooses a browser PDF exploit module.

2. The browser PDF exploits take advantage of the built-in webserver from Metasploit.

3. Once the webserver is set up and the PDF exploits are loaded onto it, the URL is sent to the victim via social engineering.

4. Once the victim clicks on the URL, the PDF exploit is injected and does the rest of the work for you.

Attack Scenario 2: E-Mails Leading to Malicious Links

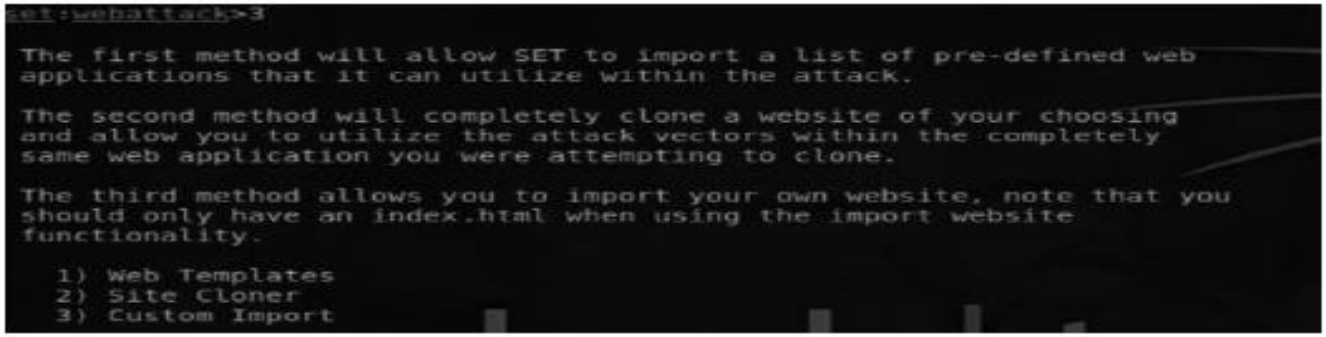
In this scenario, we will send the victim a malicious link, and when the victim clicks on it, we will be able to perform various attacks. Here are some examples:

1. We can set up a fake log-in page of any particular website, for example, facebook.com, and ask the victim to log in to the fake log-in page actually located at facebookfakepage.freehost.com.
2. If we are on the same network as the victim, we can launch a DNS spoofing attack, where we can replace the IP of facebook.com with that of our fake log-in page, and as soon as the victim visits facebook.com, he would log in to our fake page instead.
3. We can also perform DNS spoofing, where instead of the fake log-in page we can redirect the victim to our malicious webserver that would use relevant browser exploits to compromise the victim's browser.

Credential Harvester Attack

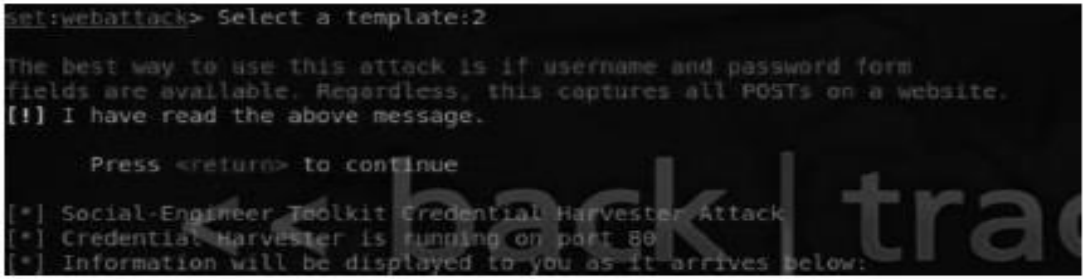
Credential harvester is a very popular attack; it can be used to perform a phishing attack. In a phishing attack, an attacker sets up a replica of a website, say, gmail.com, whenever the victim logs in to it, the credentials will be saved. This can be done with the "Credential Harvester Attack" in SET. Let's see how to do it.

Step 1—From the website attack vectors, select "Credential Harvester Attack." Now you will have three options: you can use predefined templates in SET, clone a site of your choice, or import your own template, in case option 2 does not work for you. For the sake of simplicity, I will choose the first option.



Step 2—It will now ask you the "IP address" to which you want the credentials posted, which in this case would be my local IP, since in this case I am attacking my LAN.

Step 3—It will not show you the list of built-in templates. In this case, I want to use gmail.com.

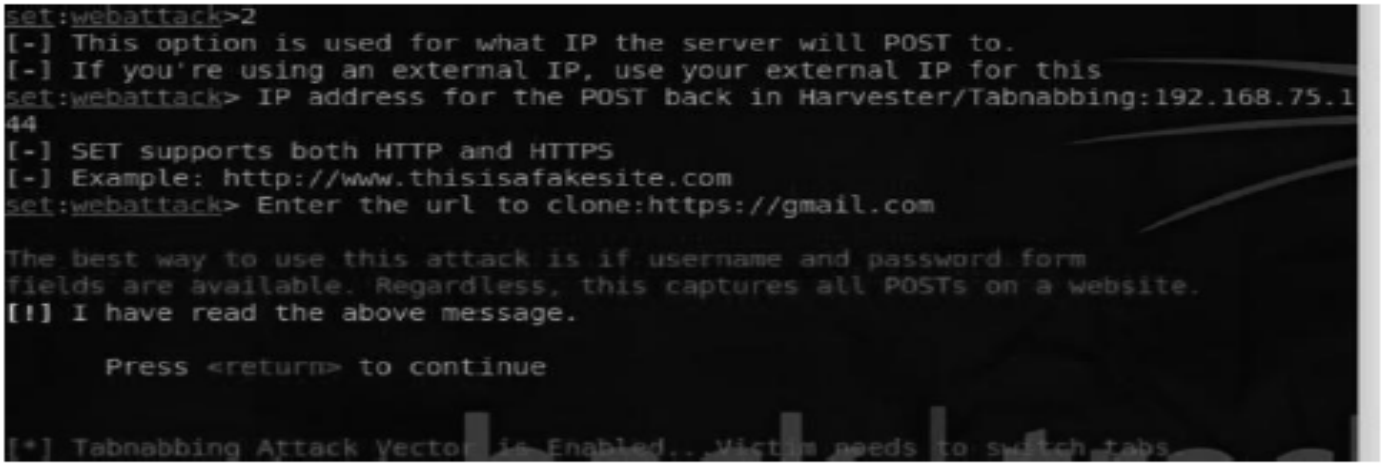


Tabnabbing Attack

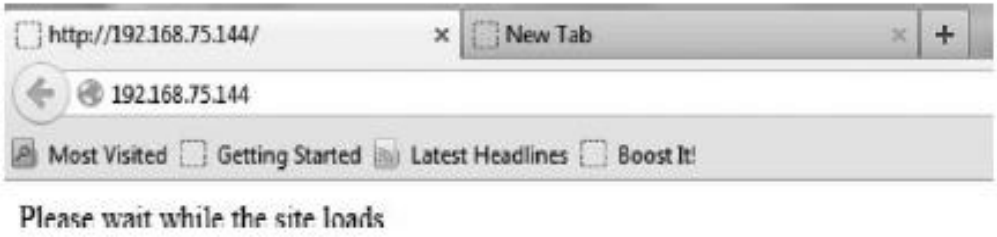
Tabnabbing is another form of phishing attack, where the attacker takes advantage of the fact that the victim doesn't normally think that tabs will change when he is not around. This type of attack would rewrite the existing tab with the attacker's website. Whenever the victim comes back to that tab, he will think that he has logged out of a particular website and would try to log in again, and as soon as the victim logs in to his account, the attacker will capture the credentials. The SET can be used to launch this attack. Let's see how it's done.

Step 1—Just beneath the "Credential Harvester" option, you will see "Tabnabbing attack." Inside it, you will see the options for "Web templates." Click on the "Site Cloner," since the tabnabbing attack method does not support the first one.

Step 2—Next, it will ask for the IP address where the attack is to be hosted followed by the website to clone, which in our case is gmail.com. Once you are done providing this information, the attack will be launched automatically.



Step 3—Now, let's see the attack on the victim's website. As soon as the victim loads the site, he will see the following screen:



As soon as he switches the tab, the website will be redirected to the fake gmail log-in page.

As soon as our victim enters the credentials, his credentials will be saved.

Attack Scenario 3: Compromising Client Side Update

- Evilgrade takes advantage of insecure update processes as the user normally does not double-check before an update because they trust that the application is being downloaded from the right place.
- **How Evilgrade Works**
  Evilgrade is an open-source modular framework developed in Perl. It is capable of injecting its own fake updates.
  Evilgrade comes with built-in modules of different applications such as Notepad, iTunes, Safari, Windows Upgrade, and many other applications.
- **Prerequisites**
  In order for Evilgrade to work, you need to be able to manipulate the victim's DNS traffic, which can be achieved in many ways. We will talk about this later.
- ***Attack Vectors***
  Let's talk about some of the possible attack vectors for Evilgrade, for both internal and external networks. Basically, any attack that can be used to manipulate the victim's DNS traffic could be performed via evilgrade.

Internal Network Attack Vectors

Here are some of the attack vectors to use when you are on the same network as the target is:

Exploiting DNS Servers—This is the easiest way by which you would compromise the DNS servers and manipulate DNS records.

ARP Spoofing—This can be used to manipulate DNS records. We learned about it in the "Network Sniffing" chapter (Chapter 6).

DNS Spoofing—Discussed in the "Network Sniffing" chapter (Chapter 6).

Faking an Access Point—You can set up a fake wireless access point, as you are able to control the DNS; the client would trust all your settings. We will see all about this attack in the "Wireless Hacking" chapter (Chapter 11).

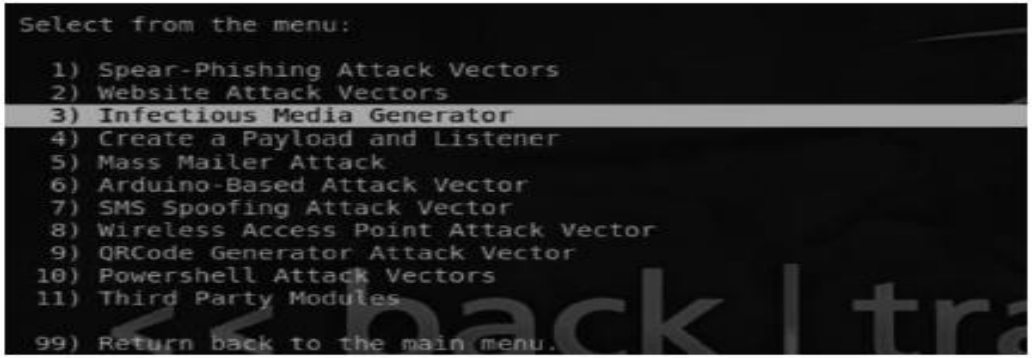
External Network Attack Vectors

Exploiting DNS Servers—Again, you manage to compromise the DNS server externally, so you can easily manipulate the records.

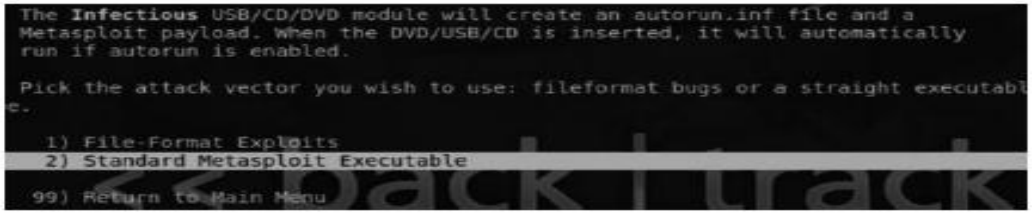
DNS Cache Poisoning—DNS cache poisoning can be launched externally to manipulate DNS records. However, this attack is not that common nowadays and is a bit harder to pull off, since most of the DNS servers are patched against it.

Attack Scenario 4: Malware Loaded on USB Sticks

Step 1—From the SET's main menu, select the third option "Infectious Media Generator."

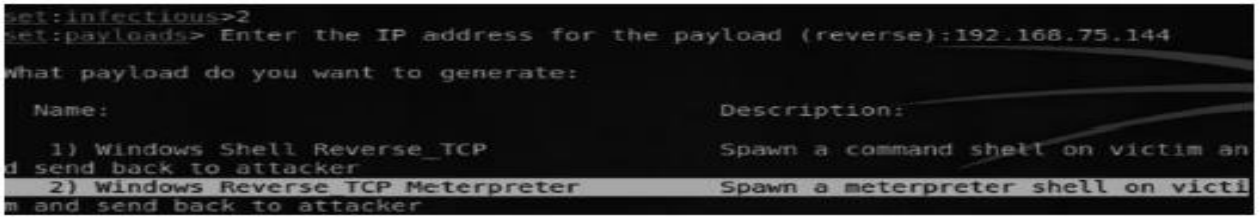


Step 2—From there, select the second option "Standard Metasploit Executable," which will enable you to generate an executable with an autorun.inf file.



Step 3—It will now ask for our reverse IP that is going to be our LHOST. Enter your LHOST and press "Enter."

Step 4—Next, it will ask for the type of the payload we want to use; we will use our favorite Meterpreter reverse TCP payload.



Postexploitation

Privilege Escalation:

- Once we have gained situation awareness, our next goal would be to escalate our privileges to the NT Authority SYSTEM, which has the highest privileges on a Windows machine, or at least we should try to get administrator-level privileges.

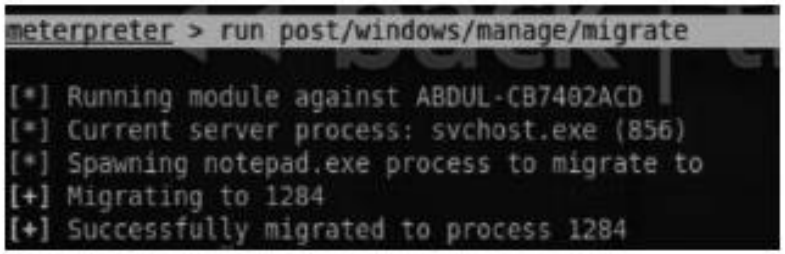
Privilege Escalation:

Maintaining Stability

The Meterpreter session often dies or gets killed, because the process that the meterpreter is running on closes. For example, let's say we used the aurora exploit to compromise a victim running Internet Explorer 6. Whenever the victim closes his browser, our meterpreter session will die.

To mitigate this issue we would need to migrate to another stable process such as explorer.exe or svchost.exe. Luckily, we have a built-in script inside of Metasploit that can help us migrate to another process. For this, we can use a post module called migrate, which is located in the `post/windows/manage/migrate` directory. The command is as follows:

```
meterpreter> run post/windows/manage/migrate
```

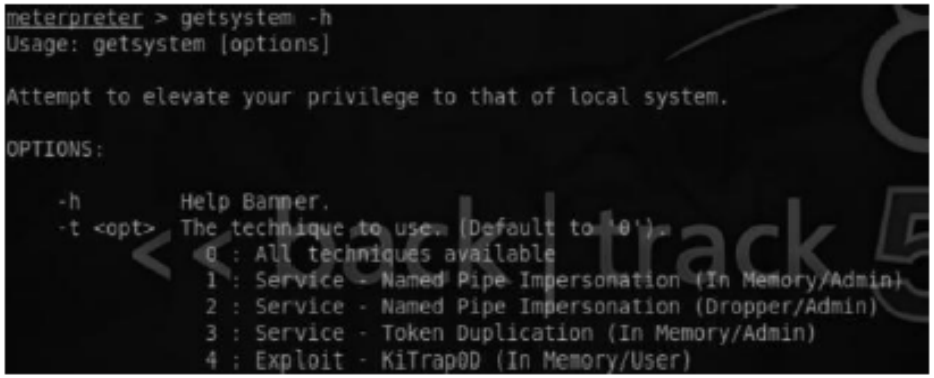


If you would like to migrate to a specific process, first issue the "`ps`" command to check for PIDs.

Escalating Privileges

Now that we have moved to a secure process and we are pretty much sure that our session won't close during our privilege escalation process, we should attempt to escalate the privileges. The fastest way of escalating privileges with meterpreter is by using the "`getsystem`" command, which consists of many techniques. If one technique fails it will try another one and will report what technique succeeded in escalating the privileges.

We can type the command `getsystem -h` to see what type of techniques meterpreter uses to escalate the privileges.



Maintaining Access

- So now we have managed to escalate our privileges to either administrator level or SYSTEM level.
- Our next step would be to make it easier for us to access the system any time we want. So far, we have managed to maintain stability, but we haven't managed to establish persistency.
- Whenever the target computer reboots, the process on which we have attached our meterpreter session will be closed and we would lose access

Backdoors

- There are several backdoors that we would manually upload to our target machine and then make changes to the registry so that we can access it even when the computer reboots. But before installing a backdoor, we should make sure that we have turned simply encode our backdoor so that it evades the antivirus. Let’s see how to go about with these approaches.

Wireless Hacking

- **Requirements**

- Wireless access point
- Wireless adapter supporting packet injection

An Alfa network adapter that supports packet injection and has all drivers installed, can connect the adapter to your computer, and since we are running BackTrack from our virtual machine, we need to attach the network adapter to our BackTrack machine.

This can be done by going into Vm → Removable Devices → Realtek RTL8187_Wireless and clicking

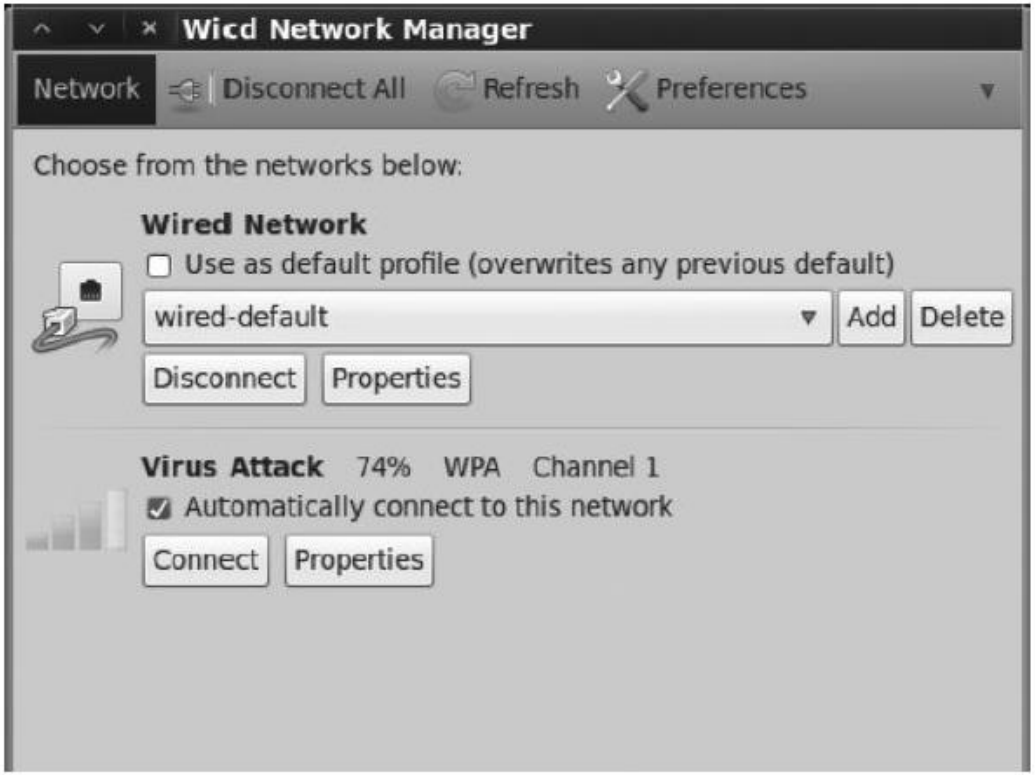
- the “Connect(Disconnect from HOST)” option

Next, we will execute “`iwconfig`” command to confirm that our BackTrack machine has been able to detect our network adapter.

Wireless adapter supporting packet injection

- Our BackTrack machine has managed to detect our wireless network adapter; however, as we can see, it is not associated with any access point. We could use WICD network manager from
- Application → Internet → Wicd Network Manager to check available wireless networks.

Wireless adapter supporting packet injection



Once we have connected to the appropriate access point and executed "iwconfig", we will see that the wlan0 interface contains information regarding ESSID, MAC address, etc.

Introducing Aircrack-ng

- it is a set of tools widely used to crack/recover WEP/WPA/WPA2-PSK. It supports various attacks such as PTW, which can be used to decrypt WEP key with a less number of initialization vectors, and dictionary/brute force attacks, which can be used against WPA/WPA2-PSK.
- It includes a wide variety of tools such as packet sniffer and packet injector.
- The most common ones are airodump-ng, aireply-ng, and airmon-ng.

Uncovering Hidden SSIDs

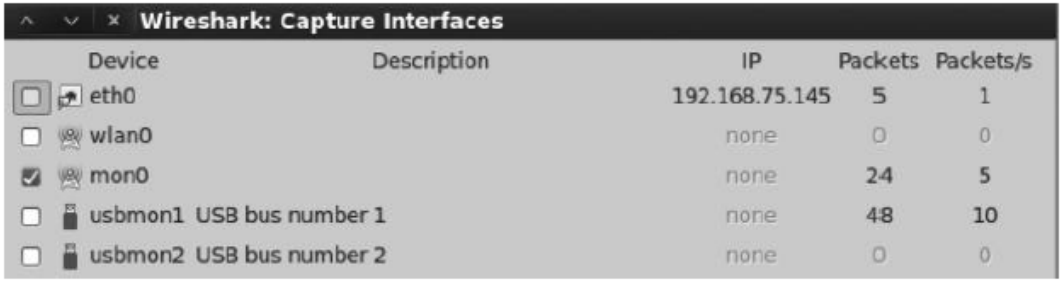
It’s common practice for network administrators to disable broadcasting SSID. Normally,

- the SSIDs are sent in the form of beacon frames, but this does not happen when a network administrator disables an SSID
- This is said to be a good security practice according to many network administrators;
- The reason being that anytime a client reassociates with the access point, it will send the SSID parameter in plain text, which will reveal the real SSID.

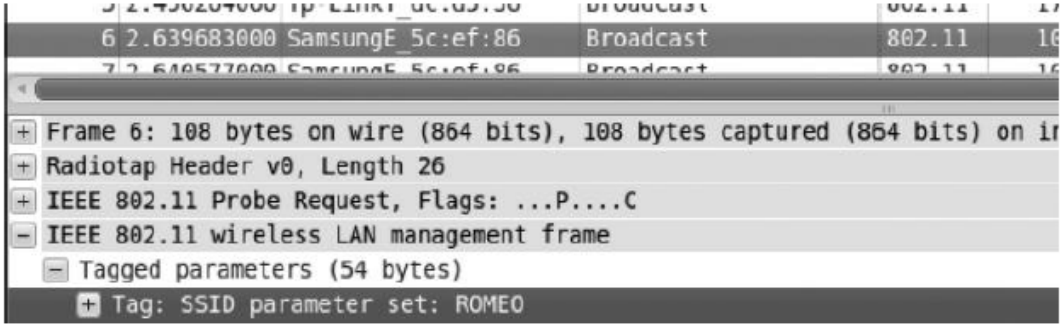
Turning on the Monitor Mode

- The next thing we want to do is switch our network card into monitor mode.
- the “Network Sniffing” to sniff on wired networks, we need to switch our network card into promiscuous mode.
- However, to sniff on wireless networks, we need to make sure that our network card is in the monitor mode.
- One of the advantages of the Alpha card is that it allows us to sniff in the monitor mode, so you need to make sure that your network card is allowed to sniff in the monitor mode for this work.

Monitoring Beacon Frames on Wireshark



We selected the appropriate interface to sniff on, and we are now able to see beacon frames from other access points, which we are not associated with. Whenever the client authenticates against the access point with the hidden SSID, it will send an SSID parameter; therefore, we can easily figure out what the real SSID is.

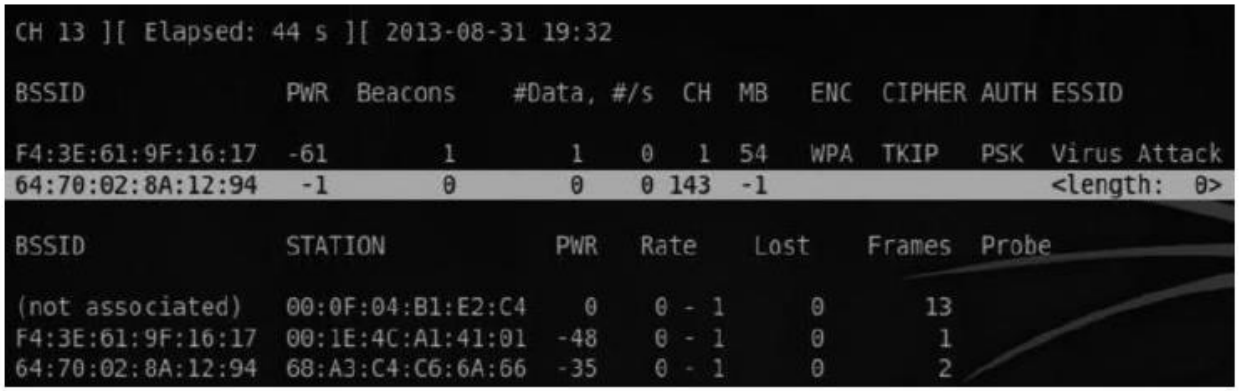


Monitoring with Airodump-ng

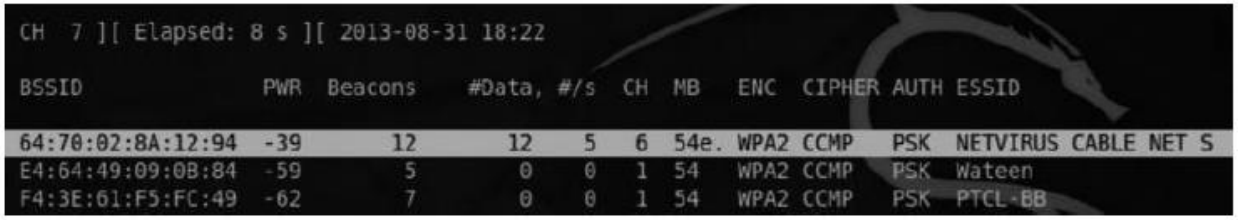
The easy way around is to use airodump-ng to start monitoring the traffic; as soon as the client authenticates, the SSID will be revealed.

Command:

```
airodump-ng mon0
```



The access point that is not broadcasting it's ESSID would appear with the names such as "<length: 0>", as soon as the client would re-authenticate the hidden SSID would appear.

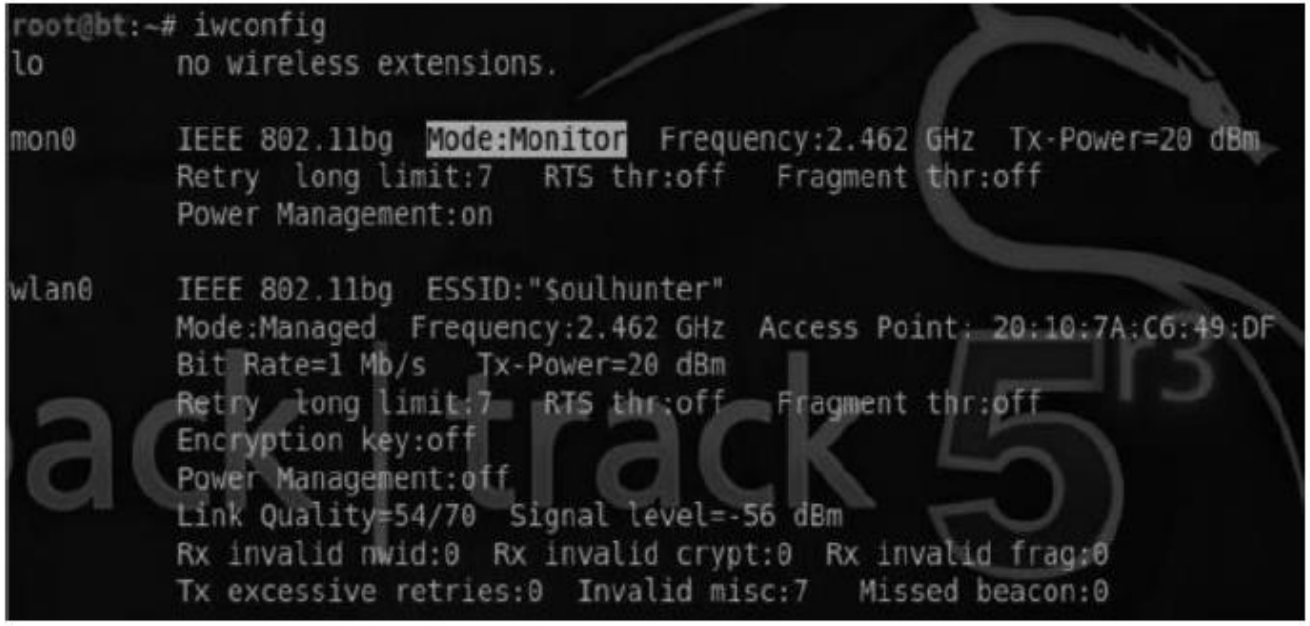


Cracking a WEP Wireless Network with Aircrack-ng

- WEP (Wired Equivalent Privacy) was one of the first authentication and encryption used for wireless networks;
- it's been known to be insecure for a decade due to some cryptographic weaknesses related to initialization vectors, key management, etc., which we won't discuss in this book

Placing Your Wireless Adapter in Monitor Mode

Step 1—First things first: we need to make sure that our network card is placed into monitor mode, we have already learnt that we can use the "airmon-ng start wlan0" command to accomplish this task. We can use "iwconfig" to verify that our wireless adapter is now able to sniff in monitor mode.

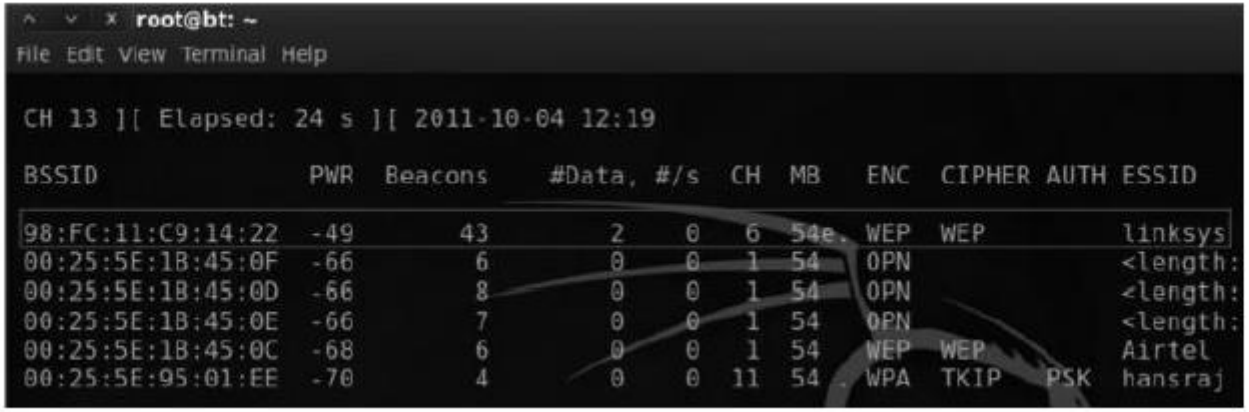


Determining the Target with Airodump-ng

Step 2—Next, we will use airodump-ng to discover our neighbor networks with WEP encryption enabled. We can see our target with an essid (same as ssid) of "Linksys" and with BSSID of 98:FC:11:C9:14:22 and it's on the channel 6. We should make a note of the essid, bssid, and channel because we will need them in future.

Command:

```
airodump-ng mon0
```

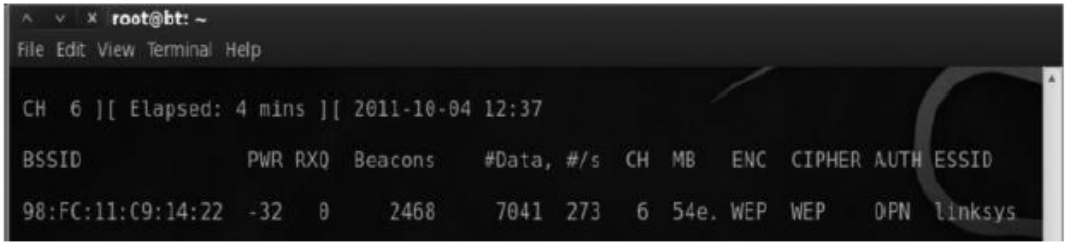


Attacking the Target

Step 3—In order to crack the WEP key, we would need to capture of the contents of the data file and write it to a file which we can analyze later. To accomplish this task, we would use airodump and restrict our monitoring only to the access point (ap) we are targeting.

Structure
airodump-ng mon0 --bssid –c (channel) –w (file name to save)

Command:
```
airodump-ng mon0 --bssid 98:fc:11:c9:14:22 --channel 6 --write RHAWEP
```



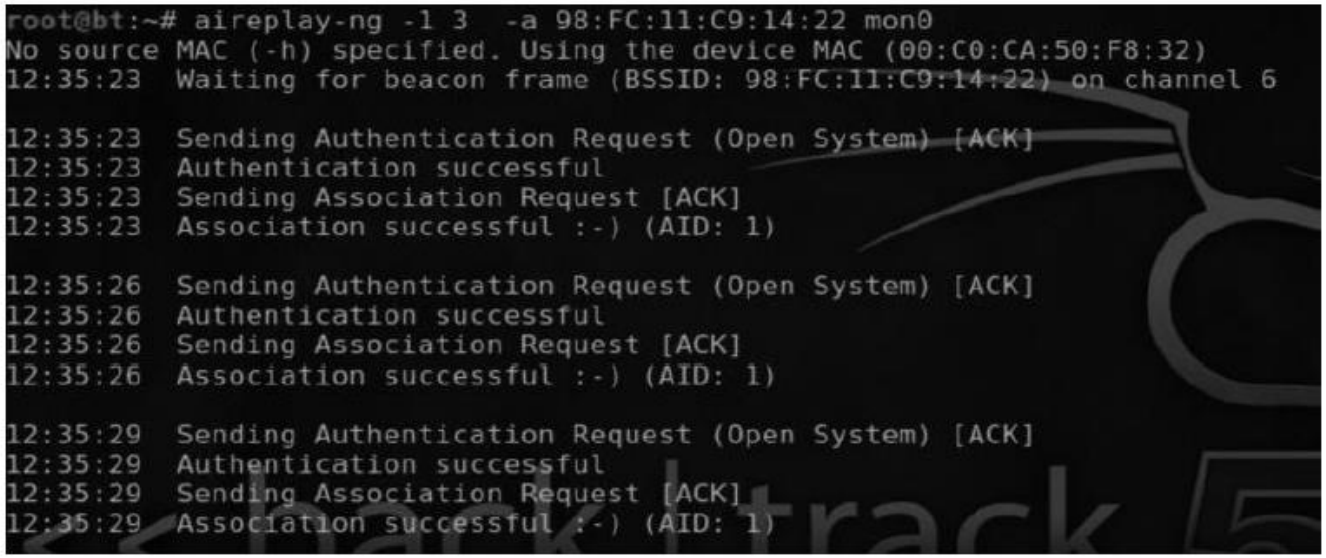
We had to specify the bssid of the target that we learnt from the previous step, followed by the channel that the access point is on, which we also learnt from previous step (channel 6). The reason we want to restrict it to channel 6 is that we don't want our wireless card to switch channels. Then we instruct it to write the results to a file called RHAWEP. The file would be in several formats, such as kismet, cap, etc., so that we can analyze it using different tools. What we are interested in is the contents of the cap file.

Speeding Up the Cracking Process

Step 4—In order to decrypt the wep key, we would need data packets, but waiting to collect them would be time consuming. To speed up this process, we can use a fake authentication attack which will associate our MAC address with the access point. This attack is only useful in the case where we have no clients associated with the access point.

Structure
aireplay-ng - 1 3 –a (bssid of the target) (interface)

Command:
`aireplay-ng -1 3 -a 98:fc:11:c9:14:22 mon0`



Cracking the WEP

Step 6—Finally, it's the time to decrypt the contents of the RHAWEP-0.1-cap file. We will use aircrack-ng to do this.

Command:

```
aircrack-ng RHAWEP-0.1-cap
```

Cracking a WPA/WPA2 Wireless Network Using Aircrack-ng

- Let's see how we can use aircrack-ng to crack a WPA/WPA2 network:
- *Step 1*—First of all, ensure that your network card is inside the monitoring mode.
- *Step 2*—Next, we would listen on the mon0 interfaces for other access points having encryption
- set to either wpa or wpa2. We would use the "airmon-ng mon0" command to do it.

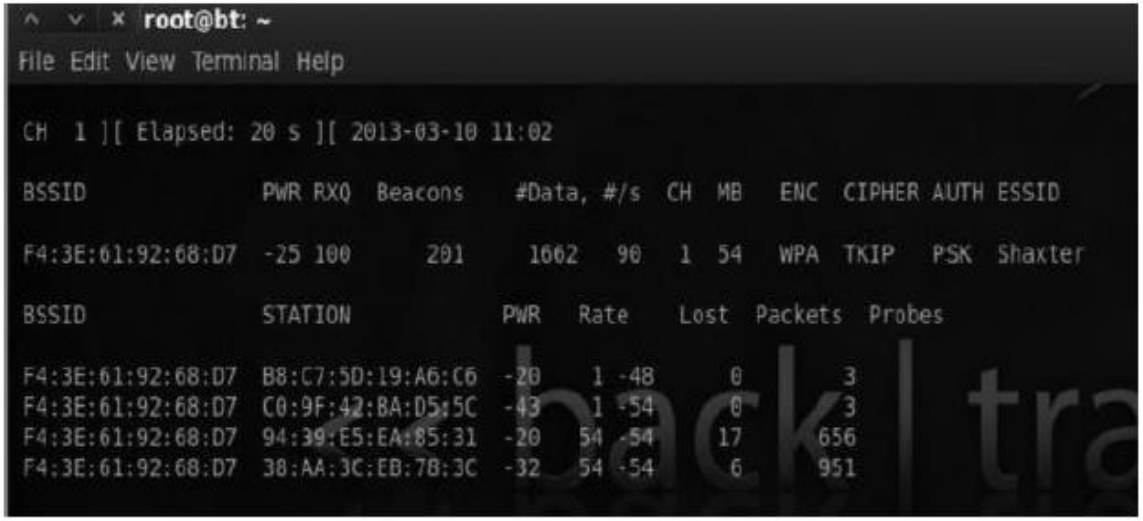
Capturing Packets

Step 3—Next, we need to save the data associated with our access point to a specific file. The inputs we need to specify are the channel, the bssid, and the file name to write.

Command:

```
airodump-ng -c 1 -w rhawap --bssid F4:3E:61:92:68:D7 mon0
```

- –w—File to write
- –c—Channel



Capturing the Four-Way Handshake

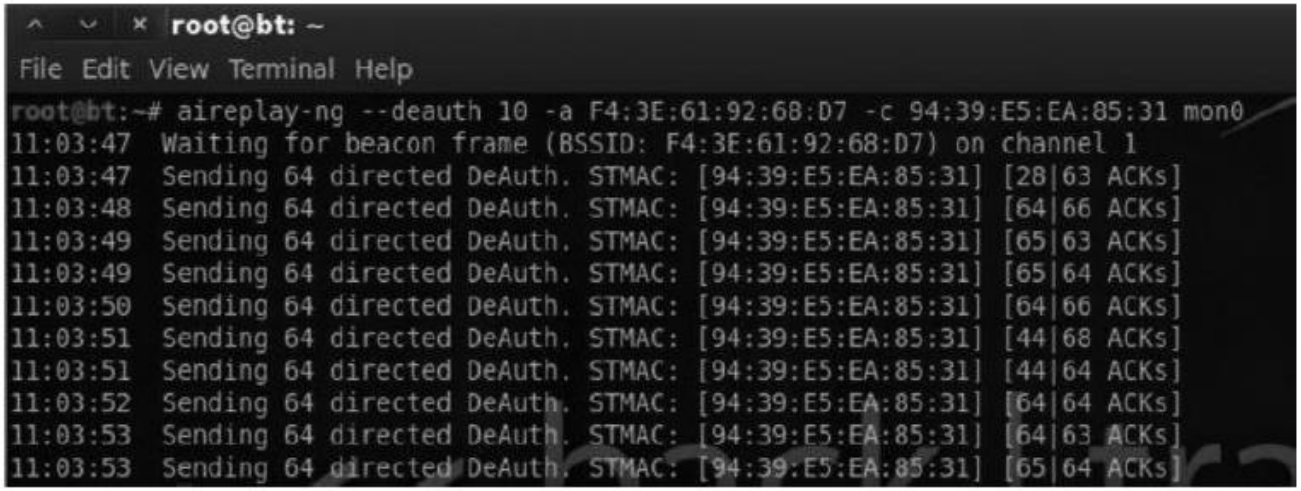
Step 4—In order to successfully crack WAP, we would need to capture the four-way handshake. As mentioned, to achieve this we could use a deauthentication attack to force clients to disconnect and reconnect with the access point.

Structure

aireplay-ng --deauth 10 –a ≤Target AP≥ –c ≤Mac address of Mon0≥mon0

Command:

```
aireplay-ng --deauth 10 -a F4:3E:61:92:68:D7 -c 94:39:E5:EA:85:31 mon0
```



After we have successfully performed a deauthentication attack, we will be able to capture the four-way handshake.

Cracking WPA/WAP2

Now that we have all the inputs required for cracking the WPA/WPA PSK, we will use aircrack-ng and specify a wordlist that would be used against the rhawap.cap file that was generated earlier. Remember that in order for us to successfully crack the WPA/WPA2 PSK, we need to make sure that our file contains the four-way handshake.

Structure
aircrack-ng –w Wordlist 'capture_file'.cap

Command:
```
aircrack-ng rhawap.cap -w/pentest/passwords/wordlists/darkc0de.lst
```

So, now this will start the dictionary attack against the rhawap.cap file, and if the key is found in the dictionary, it will reveal it to us.

Web Hacking

Username Enumeration

Sometimes it's possible to check if a current user exists in the database or not based upon the error messages that the application displays. This could be very helpful in cases where you want to conduct a brute force attack or an attack against a particular user. It could also aid you when exploiting the password reset feature. Let's take a look at an example of how this works.

Invalid Username with Invalid Password

We have a popular website xyz.com. When we enter an invalid username with an invalid password, the following error is displayed:

"Username is invalid," indicating that the particular username was not found in the website's database.

Valid Username with Invalid Password

When we enter a valid username with invalid password, the following error is displayed:

"Password is incorrect."

Not to mention, the website provided is well known; however, this isn't a big issue for them because most of their usernames are already public in their forums, listings, and market places, but certainly, this can still be an issue in several other applications.

Enabling Browser Cache to Store Passwords

Another bad security practice that is often followed is developers using autocomplete function for password fields, which enables the passwords to be saved in browser cache allowing an attacker to access the password if he can somehow access the browser cache.

We can check if autocomplete is enabled with the following command:

```
<input type="text" name="foo" autocomplete="on"/>
```

To protect against this issue, it's recommended that the autocomplete be disabled.

Brute Force and Dictionary Attacks

- **Types of Authentication**
- **HTTP Basic Authentication:**
- HTTP basic authentication is one of the first authentication mechanisms that were introduced.It works as follows:
- When we send a GET request to the protected resource, the webserver would respond with a log-in screen, which would set a “WWW-Authenticate” header also known as the authorization header.
- Our credentials are then sent to the server via the authorization header in the *base64-encoded* form. Upon receiving the header, the server would decode the base64 string to plain text and compare it with the information stored in the authorization file.

HTTP-Digest Authentication

- HTTP-Digest authentication was the modified and improved version of HTTP basic authentication.
- One of the major improvements was that it sent the password in an encrypted form.

Form-Based Authentication

- Form-based authentication is the recommended method for authenticating a user. The credentials are submitted by either POST or GET method over an HTTP or HTTPS protocol.
- it’s not a good security practice to send sensitive credentials by GET method as they can be easily leaked via referrer header or other attack, we still see it being used.

Attacking Form-Based Authentication

Step 1—Our first step would be to perform username enumeration; this can be easily done by entering an incorrect password with the username you want to check is present in the database. In this case, we found that the username "admin" exists.

Attacking Form-Based Authentication

- *Step 2*—Next, we would trap the authentication request with burp suite and then press “Ctrl+I” to send it to the intruder
- *Step 3*—Burp would automatically highlight the input fields that you can try to run your attack against; however, we are interested only in the password field with the parameter (pwd). So we will click on the “Clear” button at the right to clear all the inputs and click the “Add” button twice
- *Step 4*—We will now move to the “payloads” tab, and under payloads options, we will load our wordlist against which we want to test this particular form. For demonstration purpose, I would use the list of top 500 worst passwords by Symantec, for which I will provide the link later.
- *Step 5*—Once we have everything set up, we will click on “Intruder” at the top and click on “Start Attack,” and it will try the wordlist against our target.

Brute Force Attack

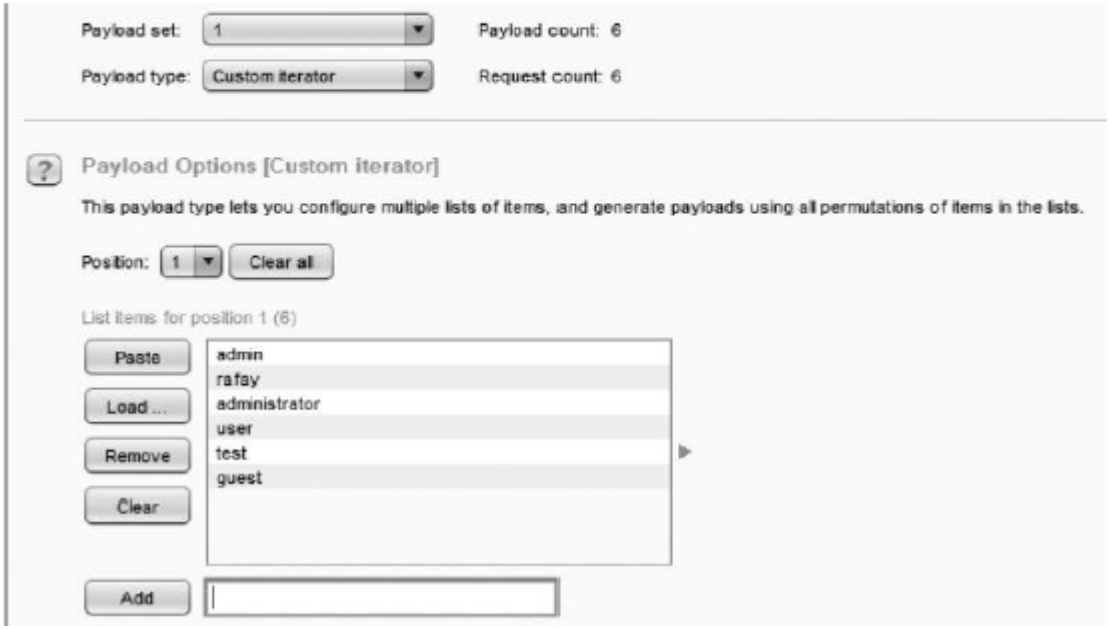
Attacking HTTP Basic Auth

Step 1—We will start by intercepting the authentication, and then send it to burp intruder.

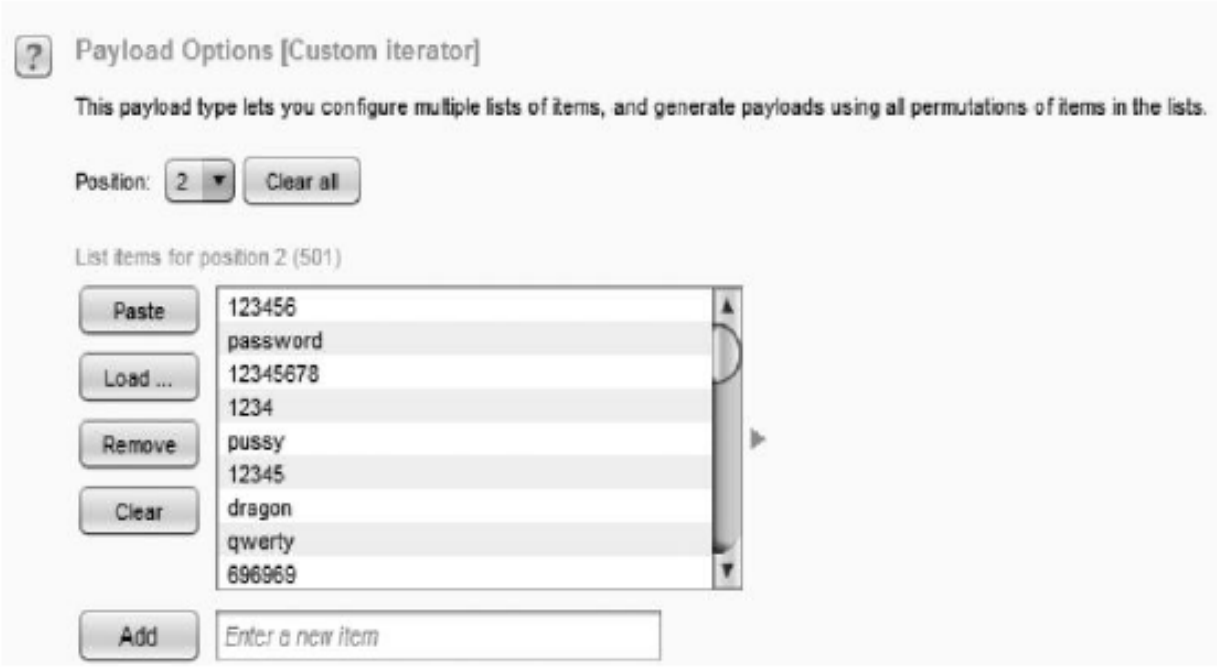
Step 2—Again, by default, burp intruder would pinpoint the possible positions to be brute-forced; however, we are interested in attacking only the authorization header that would be sent to the server, so we would click the "Add" button to lock the position.

```
GET /blog/
Host: www.target.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/:
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*.
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
uaid=uaid%3De8ng3-TnLxGuPmuy6-HARe639U7J%26_now%3D1378507039%26_
1%26_mac%3DTVU1_oyxhTGxNQRwsYIV8_HM8ncFkyed01b4As3dB-s.;
etala=111461200.1055599790.1378507039.1378507039.1378507039.1.0
last_browse_page=%2F; user_prefs=1&2596706699&qOtPzM1JLaoEAA==;
__utma=111461200.130110084.1378507032.1378507032.1378507032.1;
__utmc=111461200; __utmz=111461200.1378507032.1.1.utmcsr=(direc
Authorization: Basic §YWRtaW46cGFzcw==§
Connection: keep-alive
Cache-Control: max-age=0
```

Step 3—The next step would be to define the usernames that would be used to brute force. We would choose the payload type to *custom iterator* so we can add our separator and add the usernames that we want to test. Also, in the "Separator for Position 1," we will add a colon.



Step 4—Next, we would need to select the password that we are testing the usernames against; for that, we select number "2" from the drop-down menu holding the name "positions."



Step 5—Finally, we need to encode our payload with base64 encoding, for which we need to define a rule under the "Payload Processing" tab. To add a rule, select rule type to "Encode" and encoding type to "Base64-encode."

That's all you need to do for attacking http basic authentication.

Log-In Protection Mechanisms

CAPTCHA Validation Flaw

- One of the common flaws in CAPTCHA is validation; even if CAPTCHA is in place, we are still able to determine if we have guessed the correct password just by observing the error messages or responses. This happens due to poor handling of error messages or due to weak CAPTCHA implementation.

Submitting a wrong password

As Ajay submitted a wrong password, the following error appeared:

"Password is incorrect."

Take a look at the following picture:

CAPTCHA Reset Flaw

- Another issue, which I often test CAPTCHA against, is the counter reset flaw. This can be tested by sending a series of incorrect log-in attempts followed by a correct log-in attempt and see if CAPTCHA shows up or not.

Let's take a look at a real-world example of this reset bug, again in etsy.com, due to a weak CAPTCHA implementation. This bug was found by a security researcher with nickname "pwn-dizzle"; he discovered two issues while testing CAPTCHA's implementation.

The first issue he found was a 10 s delay, which occurred after the 20th unsuccessful attempt, which was being performed on a per-IP basis.

Manipulating User-Agents to Bypass CAPTCHA and Other Protections

Authentication Bypass Attacks

SQL injection is one of the first methods that you should test a log-in form against; the vulnerability occurs due to lack of input validation/filtering. The attacker's input query, which allows the attacker to do multiple things such as data retrieval and reading system files such as /etc/passwd; however, here our only focus is using SQL Injection to bypass the authentication mechanism.

Let's take a look at a potentially vulnerable code that would result in an SQL injection:

Code

```
<?php
$query="SELECT * FROM users WHERE username='".$_POST['username']. "' AND
  password='". $POST_['password']."'"
response=mysql_query($query);
?>
```

This is how the query would be executed:

```
SELECT * FROM users WHERE username = 'administrator' AND password =
  'mypass'
```

This query would retrieve the details of username "administrator" with the password "mypass" from the table users.

Testing for SQL Injection Auth Bypass

Since our input is not properly being filtered or validated, we can insert the following SQL query in the user input to bypass authentication:

```
' or '1'='1
```

Since this statement is always true—1 is always equal to 1—it will result in bypassing authentication. Assuming that the password parameter is vulnerable and the username that we are trying is "administrator," the following query would be executed:

```
SELECT * FROM users WHERE username = 'administrator' AND password = '' or
  '1'='1'
```

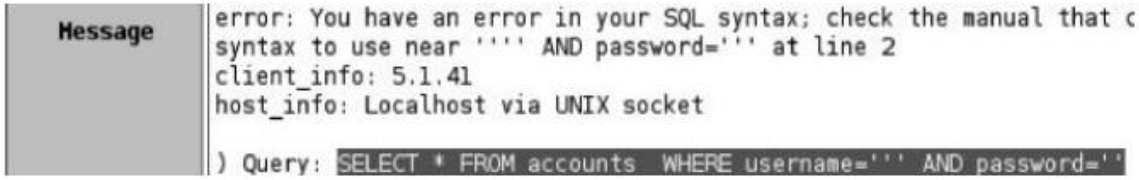
Alternatively, you can use an SQL comment to ignore everything after your query resulting in bypassing authentication.

```
' or '1'='1' --
' or '1'='1' #
```

Let's now see this in action. For demonstration, I will use the OWASP Mutillidae project, which contains the most popular vulnerabilities found in web applications. It contains the owasp top 10 vulnerabilities and others.

Testing for SQL Injection Auth Bypass

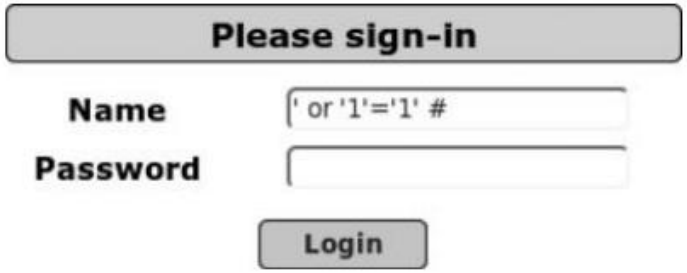
We will insert an apostrophe (') in the "Name" field to look for a typical SQL injection and see if we are able to break the query.



We get an sql error, which means that we have successfully managed to break the query.

Next we would have to use true statements in order to bypass authentication. We will use sql comments to ignore everything after username. We will insert the following command:

```
' or '1'='1' #
```



Testing for SQL Injection Auth Bypass

Step 1—We will intercept the request and send it to burp intruder (Ctrl+I). Under burp intruder, we will choose “Sniper” as an attack type and will choose to fuzz both username and password parameters.

Step 2—Next, we will load the cheat sheet in burp intruder, which would be used to test the form against

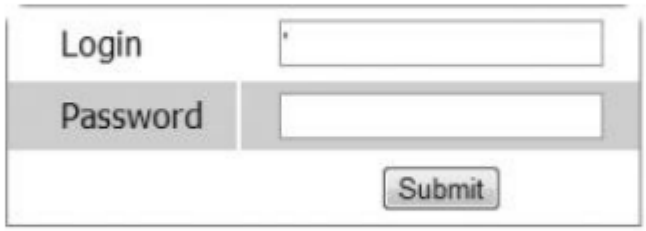
Step 3—Finally, we will start the intruder attack and take a note of the content length to see where we have been able to bypass the authentication mechanism.

Authentication Bypass Using XPATH Injection

XPATH injection is an attack where an attacker injects xpath queries to bypass the log-in mechanism by making the overall statements true. XPATH is a standard way of querying XML databases. It's similar to SQL queries used to query mysql and mssql databases.

Bypassing an authentication with xpath injection is a bit more difficult than SQL injection. We will have to satisfy the two conditions:

Step 1—We have a form that we need to test for an XPATH injection. We will simply submit an apostrophe (') via the input parameters and look for an error:

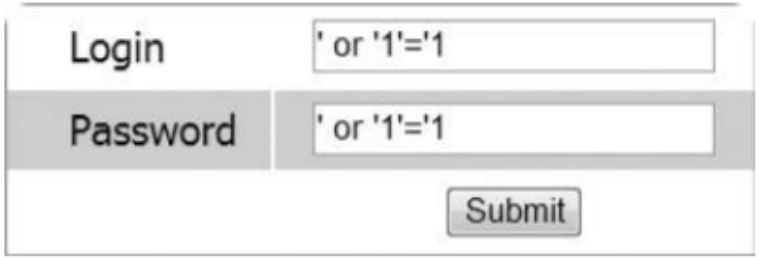


An error occurred while processing the XPath query

Authentication Bypass Using XPATH Injection

Step 2—Since, as mentioned before, we need to make sure that our statement is true, we would insert the following true statements in the inputs.

Login: ' or '1' = '1
Password: ' or '1' = '1



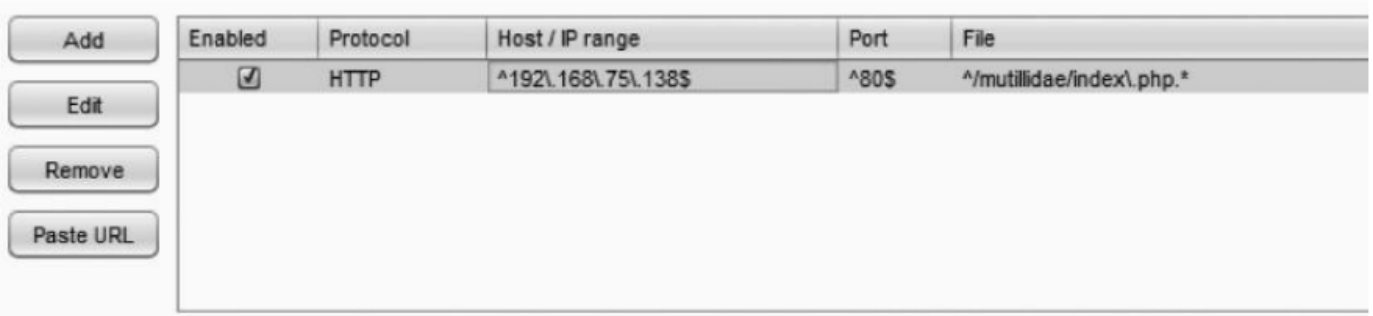
The overall query becomes true, and we can successfully bypass the log-in form.

Authentication Bypass Using Response Tampering

Crawling Restricted Links

The best way of finding this vulnerability is by crawling all the pages of a particular website and taking note of all the restricted links not accessible by normal users. Acunetix web vulnerability scanner has a great crawler that you can use; alternatively, burp suite's spider feature is a great way to crawl a website for pages that are not publicly accessible.

To use the burp spider effectively, we first need to set the scope to crawl our defined target only. To set the scope, simply copy the url and click on "Paste URL", and burp would adjust the settings automatically.

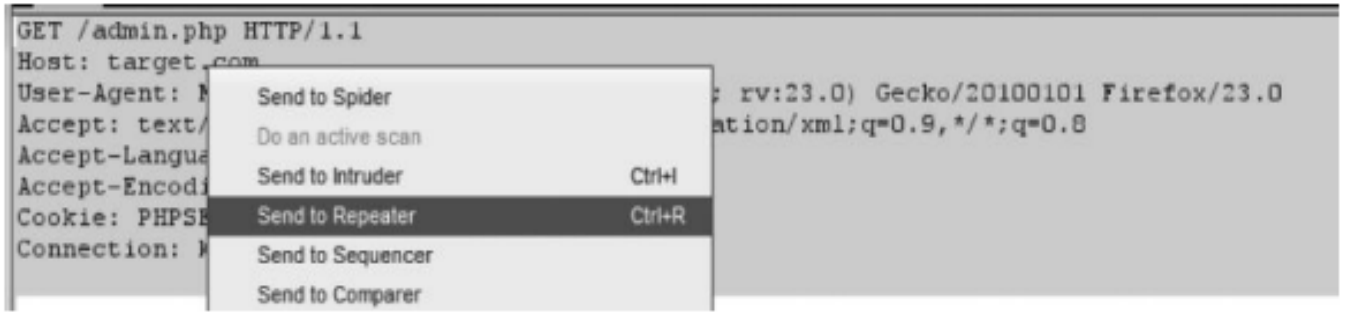


Next, we right click the place where we want to spider from and click on "Spider this branch" if it's a branch or "Spider from here" if it's a webpage.

Testing for the Vulnerability

To test for this vulnerability, you need to take a look at the response that you get when sending an HTTP request to the restricted page. Imagine a website, target.com, with a restricted page admin.php. On submitting a GET request to admin.php, we get a "302 Moved Temporarily" error. You may also get a "302 found" response or any other response depending upon the content. The important point to note is if the response body contains the restricted resource.

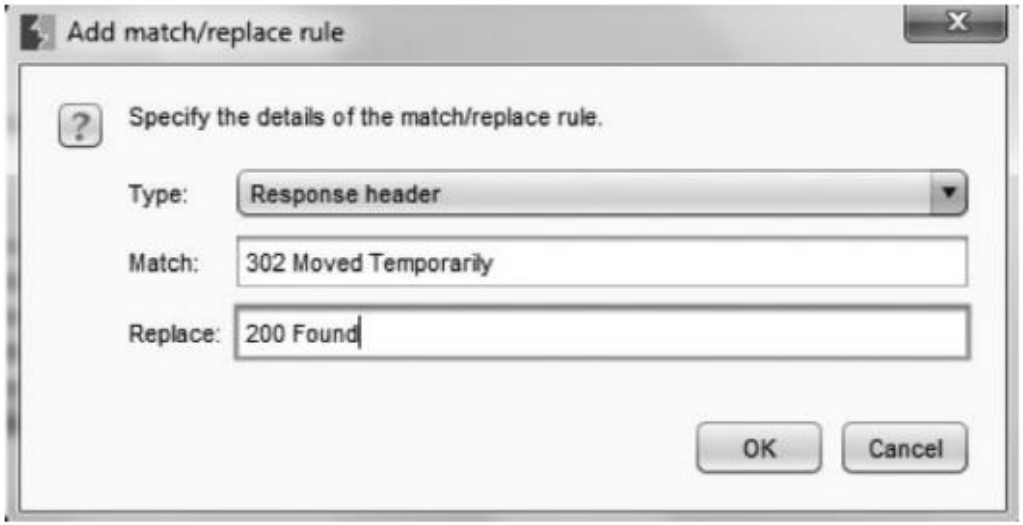
In order to analyze the request and response, we will send the request to burp repeater:



We can see that, on accessing the admin.php page, we are getting a "302 Moved Temporarily" error.

Automating It with Burp Suite

To automate this process, you can ask burp suite to change all the responses from "302 Moved Temporarily" to "200 OK." To do this, navigate to Proxy → Options and in the Math and Replace section, click on "Add a new rule" and enter details as follows:



The next time, burp looks at any "302 Moved Temporarily" header, it will replace it with "200 OK" automatically.

Session Attacks

Guessing Weak Session ID

It’s very important to make sure that the session ID is random and cannot be predicted or guessed by brute force attacks. It should expire after a certain time of inactivity; also a single session should be locked to a single IP address, making it even more difficult for an attacker to reuse the session ID.

Let’s talk about how we can analyze the randomness of tokens by using burp suite’s sequencer tool.

Guessing Weak Session ID

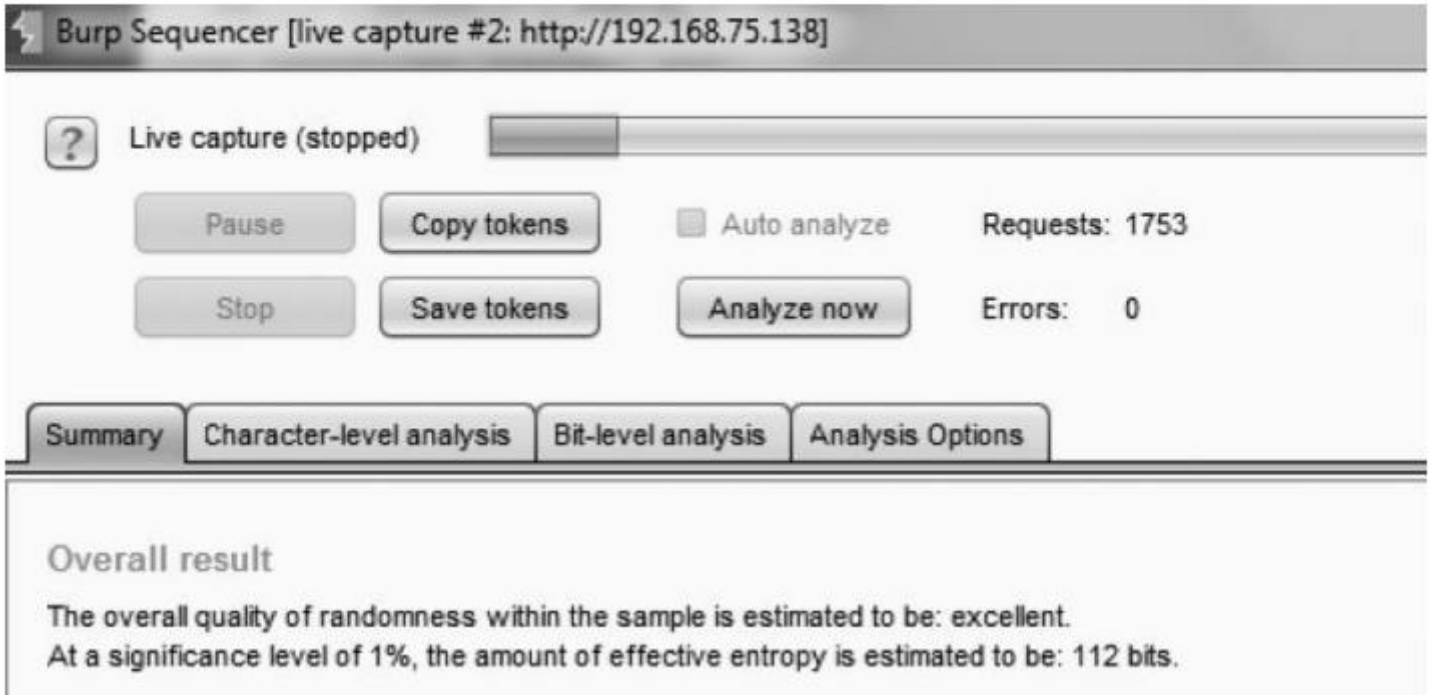
Step 1—Our first step would be to capture the response from the target application, which would contain the set-cookie header having our session ID.

Guessing Weak Session ID

Step 2—Next, we would feed the response in burp sequencer, and it will automatically extract the session token from it. If it doesn't, select the session ID from the cookie field.

Guessing Weak Session ID

Step 3—Next, we will click on "Start Live Capture," and it will start capturing the tokens; it will strip the set-cookie header from the http request, and as the response comes from the webserver, it would contain a newly generated session token.



Guessing Weak Session ID

Step 4—Once it generates a minimum of 1000 tokens, click on "Analyze now"; the more the number of the tokens generated, the better the analysis would be.

SQL Injection Attacks

An SQL injection occurs when the user-supplied input or query is considered as a database query; in simple words, the input is not filtered by the application, which means that an attacker could inject malicious code in the application that would be parsed by the interpreter as an SQL statement resulting in an SQL injection flaw.

This will then allow an attacker to conduct a wide variety of attacks. SQL, LDAP, and XPath injection all fell down in the “Injection attacks” category which secure the first spot inside the OWASP 2013 Top 10 attacks.

Types of SQL Injection

The following are the three types of SQL injection attacks:

Union-Based SQL Injection

This is the most common type of SQL injection. It comes from the class of inband SQL injection, and this type of attack utilizes the use of a UNION statement, which is the combination of two select statements, to extract information from the database. We will discuss this attack in detail later.

Error-Based SQL Injection

An error-based SQL injection is the easiest; however, the only problem with this technique is that it works only with MS-SQL Server. In this technique, we cause an application to throw an error to extract the database. Typically, you ask a question to the database, and it returns with an error containing the information you asked for.

Blind SQL Injection

The blind SQL injection is the hardest of them all. In this technique, no error messages are received from the database; therefore, we extract the data by asking questions to the database. The blind SQL injection is further divided into two categories:

1. Boolean-based SQL injection
2. Time-based SQL injection

Both of these methods can be used to extract the database by either asking a question or inducing a time delay. We will discuss more about them later.

Detecting SQL Injection

To identify an SQL injection, we would need to test every user input to see if it's been filtered out right or not. Input parameters such as "GET, POST" are the ones commonly vulnerable to this attack. However, "cookie" values and "http headers" can also be used to conduct SQL injection attacks, where any one of the http headers or cookie values would be inserted in the database and would be displayed at some point of time. If they are not filtering it out correctly, it could result in an SQL injection.

To test this, you could insert one of following inputs and hope to break the existing query: Single quote (‘), double quotes (“), or backtick/accent grave (`)

In most cases, the single quote would work; however, it doesn't hurt to test the others. In the case you are entering a single quote, if an error is displayed, there is a good chance that it's vulnerable to an SQL injection. Next, enter another single quote; if no error is displayed, it's most probably vulnerable to an SQL injection. Similarly, probe the user inputs with double quotes and backtick.

Note: This is the case when the application is returning an error. If it doesn't, it doesn't always mean that the application is not vulnerable to SQL injection. We will look into this in detail when we discuss blind sql injection attacks.

XSS (Cross-Site Scripting)

XSS is an input validation issue just like SQL injection. XSS occurs when the user input is not properly filtered or sanitized before it's reflected back to the user.

How to Identify XSS Vulnerability

Since XSS is an input validation problem, we will probe all the inputs and try to figure out any input that is not sanitized such as url parameters, forms, cookies, and file uploads before it's returned to the user.

The basic test for finding if a website that is prone to XSS vulnerability is to inject the following piece of code, which is a minor variation of the XSS locator code found on "OWASP XSS Filter Cheat Sheet."

'"<>();[]{}XSS

Once you inject this payload into every possible input, view the source of the page that was rendered back. Then, try finding the word "XSS" in the source; how do you see it reflected back? If any one of these characters is not escaped, then the website is probably vulnerable to an XSS.

Types of Cross-Site Scripting

Primarily, there are three types of cross site scripting vulnerabilities:

1. Reflected/nonpersistent XSS
2. 2. Stored/persistent XSS
3. 3. DOM-based XSS

Reflected/Nonpersistent XSS

This is one of the most common forms of a cross-site scripting vulnerability that you would find in a reflected XSS attack. The input is reflected back to the user, and it's not stored on the server or the database. These types of XSS attacks are a bit harder to exploit, since we need the victim to click our specially crafted payload.

Let's talk about an example of a simple cross-site scripting vulnerability. I will use dvwa to demonstrate the attacks on low, medium, and high security levels. Let's start by looking at the underlying vulnerable code for a low security level.

Vulnerable Code

```
echo '<pre>';
echo 'Hello ' . $_GET['name'];
echo '</pre>';

}

?>
```

Medium Security

Next, we will look at medium security level for dvwa. Let's start with the vulnerable code.

Vulnerable Code

```
echo '<pre>';
echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
echo '</pre>';
```

The code is simply using the `str _ replace` function to strip out <script> tags before it's reflected back, again a poor approach to security "blacklists." Since there are a huge number of ways to inject JavaScript code in an input, filters based upon blacklists have constantly failed. In this case, an attacker can execute any one of the following payloads to bypass the blacklist.

<img src=x onerror=alert(0);>

<iframe/onload=alert(0);>

Cross-Site Request Forgery (CSRF)

A CSRF attack also known as XSRF or session ridding is yet another commonly found vulnerability in web applications.

It is often confused with XSS attacks though it's completely different. In a CSRF attack, an attacker forces the browser to make an unintended request on behalf of the victim.

Changing a user's password, sending message on behalf of the victim, logging off the victim, etc., are the common examples of a CSRF attack.

Why Does a CSRF Attack Work?

CSRF attacks work because the website never verifies whether the request came from a legitimate user; instead, it just verifies that the request came from the browser of the authorized user. The attack works as follows:

Step 1—A user is authenticated on a website, say, paypal.com.

Step 2—The attacker tricks the victim into visiting his controlled domain, say, attacker.com. The attacker.com contains the malicious code, which actually sends a request to paypal.com to perform a specific action, say, changing the victim's password.

Step 3—paypal.com assumes that the request was sent from the victim's browser and does not verify it, and hence changes the victim's password.

GET-Based CSRF

Let's assume that the website target.com utilizes a GET request to change the password. The request looks like the following:

http://target.com/password.php?newpass=abcd&confpass=abcd

The attacker can now modify the newpass and confpass parameters with his own password and force the victim's browser to perform a GET request and hence the passwords would be changed to what the attacker sets up. The code for forcing the victim's browser to make a get request would look something like this:

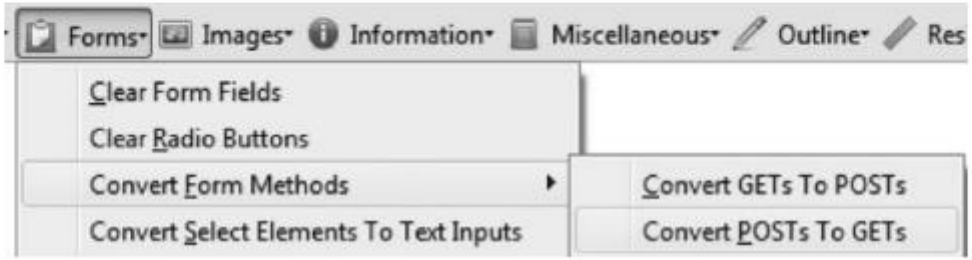
```
<img src="http://target.com/password.php?newpass=12345&confpass=12345"
  width="100" height="100">
```

POST-Based CSRF

There is a common myth among web developers that using POST request to submit a form would prevent a cross site request forgery; however, this is completely wrong. Performing a CSRF attack on POST-based form just takes additional lines of the code.

Assume that the victim's website is using POST method to submit "change password" request to the victim. The options are as follows:

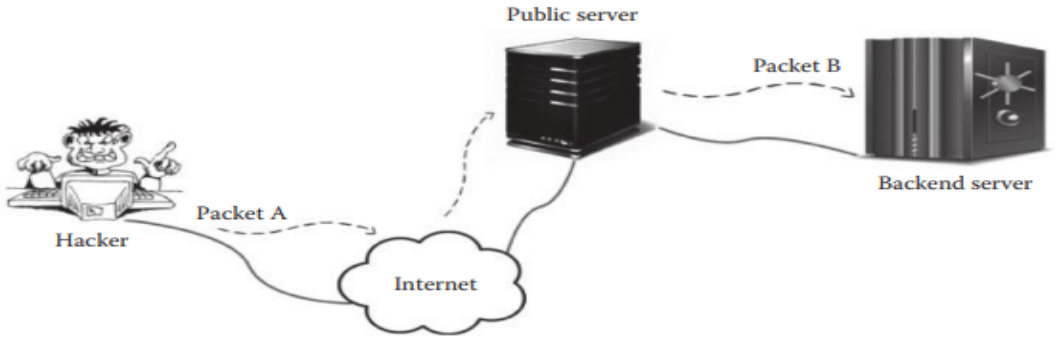
- In the case the application is accepting POST request via GET method, we can convert the POST request to a GET request and use the earlier POC to conduct the attack. We can utilize a Firefox plug-in called "Web Developer toolbar," which makes it easier for us to convert a POST request to a GET request.



SSRF Attacks

SSRF stands for (server side request forgery). SSRF itself is not a new vulnerability;, and however, it's a class of different vulnerabilities. SSRF vulnerability occurs due to unsafe use of functions that are used to open sockets and fetch data (image, text, and content) from a webserver. An example of these functions would be the use of "`Curl`," "`file _ get _ Contents`," "`fsockopen()`," etc., in PHP; such functions exist in almost every programming language.

If these functions are used unsafely and the developer does not sanitisze the inputs and response, an attacker may be able to use public-facing servers as a pivot to exploit the application running on the internal network, since all of the traffic to the back end server would be sent via the public server. Hence SSRF can be used to bypass Firewall's/IDS and IPS protections.



This diagram demonstrates how an SSRF vulnerability works. An attacker sends a specially crafted "Packet A" to the Internet-facing webserver and that webserver then sends "packet B" on behalf of the attacker to the back end server running on the internal network. In this way, an attacker could sometimes bypass Firewall restrictions because the back end server would trust the packet coming from the webserver as it is on the same internal network as the back end server.

Depending upon the parser, vulnerable application and the function such as (CURL) for opening sockets an attacker may be other URL schemas such as "gopher" to communicate queries the internal web servers. The popular URI schemas include the following:

Remote SSRF

Remote SSRF is what we have discussed so far. According to the paper, a remote SSRF can be divided into three main categories:

1. Simple SSRF
2. Partial SSRF
3. Full SSRF

Simple SSRF

In a simple SSRF, we are not able to control the data of "packet B" that are sent to the application in a trusted internal network; all we can do is to control the remote IP and the remote port.

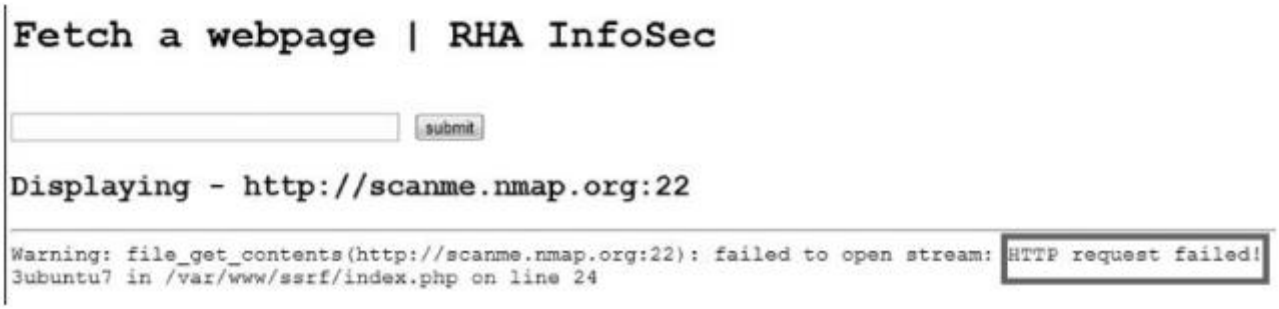
For all of our SSRF tests, we would use a site set up by nmap ("scanme.nmap.org"), which has known ports 22, 80, and 9929 open. We will feed the URL followed by a colon and an open port and note down the response, and would do the same for a closed ports such as (51, 52) etc. If both responses differ from each other, this means that we have a way to figure out if a certain port is open or not. The error messages are the most common form of response; however, you may also want to compare the timings, response sizes to check if the port is open or closed.

Let's test for SSRF on our vulnerable application:

We will test for an open port first:

Command

```
http://scanme.nmap.org:22
```



Partial SSRF

In a partial SSRF, we control only certain parts of packet B that arrive internal application; this type of vulnerability can be used to read local system files such as /etc/passwd, `/etc/hosts`, and many others. We can leverage file://protocol to read local files on the system.

Command

- `file:///etc/passwd`
- `file:///etc/hosts`

Full SSRF

In the case of a full ssrf vulnerability, we have complete control over packet B; this means that we can exploit the vulnerable services running on the internal network. In the case of schemas such as file://, we have a limited control over packet B. However, with schemas such as dict://, http://, and gopher://, we can send our malicious payload to any application running on any port.

dict://

Let's talk about the dict:// schema first. Consider that a public webserver is vulnerable to SSRF. By enumerating, we found that the webserver is running memcached on the internal network, which has a default port of 11211.

	TCP					
	HTTP	memcached	fastcgi	zabbix	nagios	MySQL
gopher	cURL, Java, LWP, ASP.Net	cURL, LWP, Java, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net
http	All	if LF available	-	-	-	-
dict	-	cURL	-	-	-	-
ldap	LWP	LWP	-	-	-	-
tftp	-	-	-	-	-	-