SECURITY TOOLS

COMPUTER SECURITY I
DVGC19

19 NOVEMBER 2023

AHMAD SHAMMOUT : SHMOOT001@GMAIL.COM
JOHANNA OLSSON : JOHANNAMARIA1@LIVE.SE

1. Security Tools

The goal for this experiment was gaining first-hand experience on three different security tools. These tools are designed for troubleshooting, network discovery, finding vulnerabilities and/or misconfigurations, to test the security of programs and detect bad security decisions. The tools used for the experiment were Nmap, Wireshark, and Ettercap.

1.1 Nmap

Nmap, short for "Network Mapper," is a powerful open-source tool designed for network exploration and security auditing. It is used to discover devices and services on a computer network, thus creating a map of the network's structure. Nmap's capabilities include port scanning, version detection, and vulnerability identification, making it a valuable asset for network administrators and security professionals to assess and secure their networks.

1.2 Wireshark

Wireshark is a widely-used open-source network protocol analyzer that allows users to capture and inspect the data traveling back and forth on a computer network in real-time. It is used for troubleshooting network issues, analyzing network traffic patterns, and identifying security vulnerabilities by examining the packet-level details of communication. Wireshark supports a variety of protocols and provides a detailed view of network activities, making it an essential tool for network administrators and security analysts.

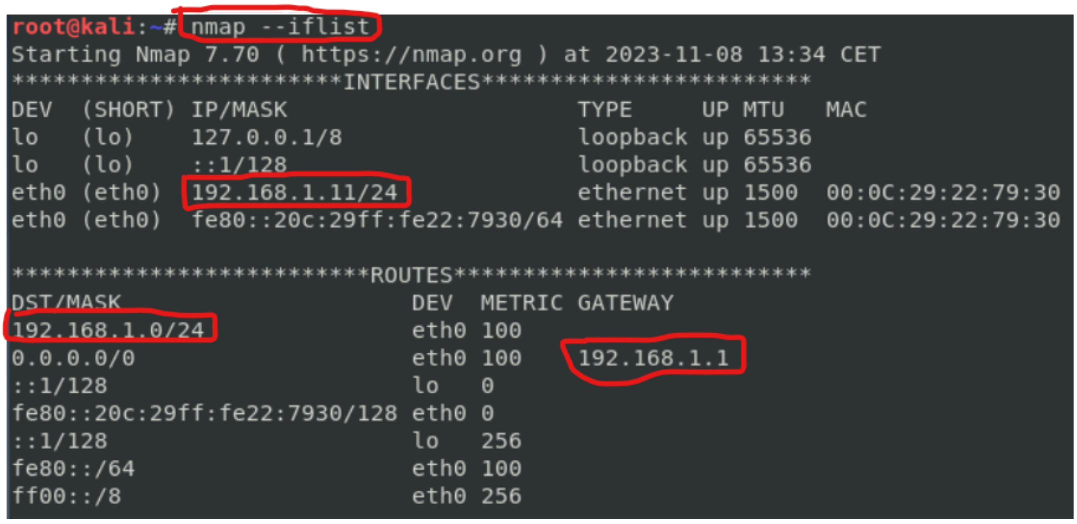
1.3 Ettercap

Ettercap is a free and open-source network security tool that operates as a comprehensive suite for man-in-the-middle (MITM) attacks on computer networks. It allows users to intercept, log, and analyze communication between hosts on a network, facilitating various security assessments. Ettercap is commonly used for tasks such as network sniffing, password interception, and protocol analysis, making it a valuable tool for both security professionals and malicious actors for educational and ethical hacking purposes.

2. Performed Tasks

2.1 Nmap

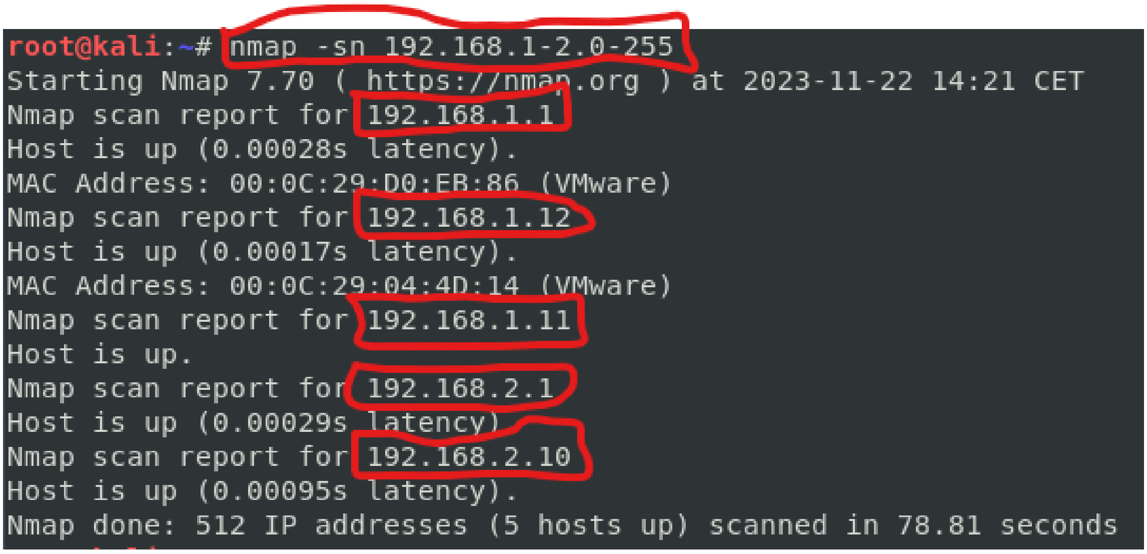
Initially, we would display the interfaces and routes of the Kali host. To achieve this, we employed the "*nmap – iflist*" command, which provided information about the interfaces and routes on Kali. Running "*nmap –iflist*" revealed that Kali's IP address was "*192.168.1.11/24*", and it also displayed the IP addresses of the routes as "*192.168.1.0/24 & 192.168.1.1*"



```
root@kali:~# nmap --iflist
Starting Nmap 7.70 ( https://nmap.org ) at 2023-11-08 13:34 CET
************************INTERFACES************************
DEV  (SHORT) IP/MASK                     TYPE     UP MTU   MAC
lo   (lo)    127.0.0.1/8                 loopback up 65536
lo   (lo)    ::1/128                     loopback up 65536
eth0 (eth0)  192.168.1.11/24             ethernet up 1500  00:0C:29:22:79:30
eth0 (eth0)  fe80::20c:29ff:fe22:7930/64 ethernet up 1500  00:0C:29:22:79:30

**************************ROUTES**************************
DST/MASK                      DEV  METRIC GATEWAY
192.168.1.0/24                eth0 100
0.0.0.0/0                     eth0 100    192.168.1.1
::1/128                       lo   0
fe80::20c:29ff:fe22:7930/128  eth0 0
::1/128                       lo   256
fe80::/64                     eth0 100
ff00::/8                      eth0 256
```

Figure 1 : Showing Kali host interfaces and routes.

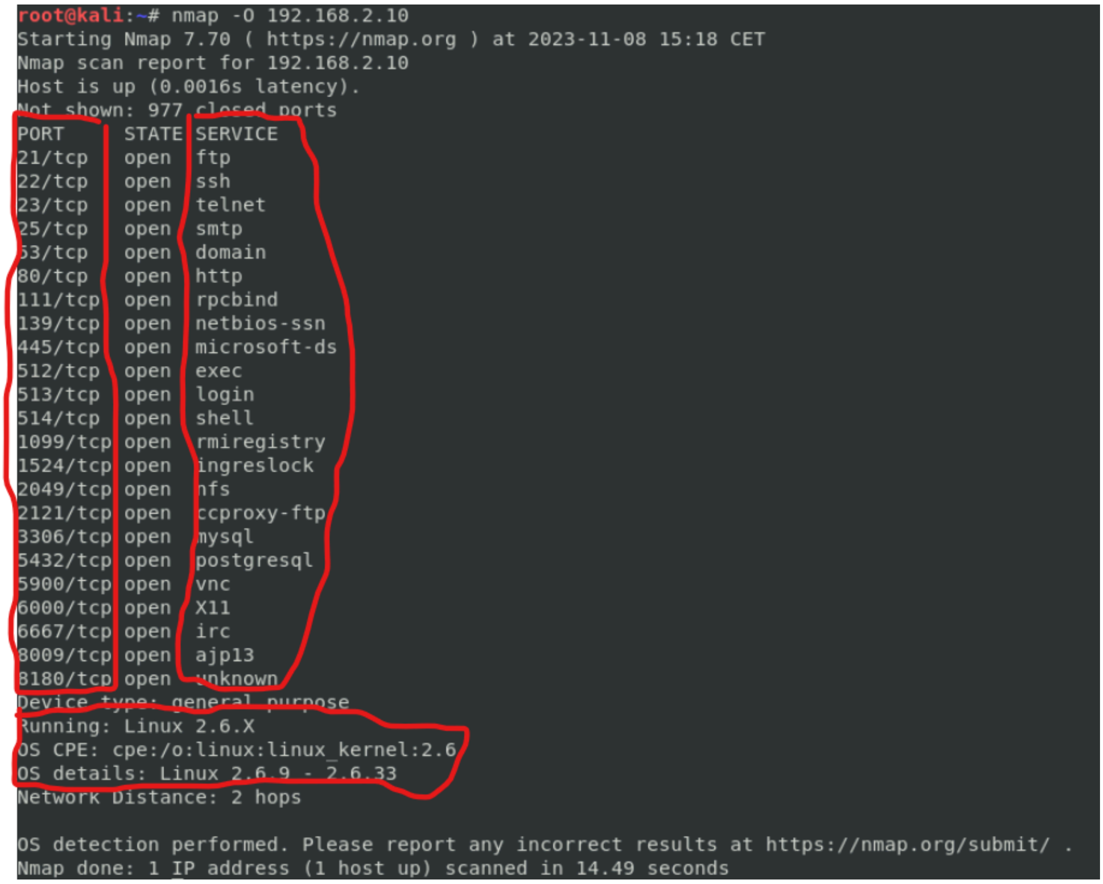
To identify active devices, we utilized the command "*nmap -sn 192.168.1-2.0-255*", where "*nmap -sn*" serves as a host discovery tool in Nmap, specifically designed for a "*ping scan*". This method involves sending ICMP Echo Request (ping) packets to target hosts, allowing us to determine their online/offline status without engaging in a comprehensive port scan. The discovered networks included "*192.168.1.1*", "*192.168.1.12*", "*192.168.1.11*", *"192.168.2.1"* and *"192.168.2.10"*.



```
root@kali:~# nmap -sn 192.168.1-2.0-255
Starting Nmap 7.70 ( https://nmap.org ) at 2023-11-22 14:21 CET
Nmap scan report for 192.168.1.1
Host is up (0.00028s latency).
MAC Address: 00:0C:29:D0:EB:86 (VMware)
Nmap scan report for 192.168.1.12
Host is up (0.00017s latency).
MAC Address: 00:0C:29:04:4D:14 (VMware)
Nmap scan report for 192.168.1.11
Host is up.
Nmap scan report for 192.168.2.1
Host is up (0.00029s latency).
Nmap scan report for 192.168.2.10
Host is up (0.00095s latency).
Nmap done: 512 IP addresses (5 hosts up) scanned in 78.81 seconds
```

Figure 2 : Scanning the network to find the devices that are up and running.

To scan the Metasploitable server for its operating system, open ports and running services, we employed the command "*nmap -O 192.168.2.10*". This command facilitated the detection of the machine's operating system, open ports, and active services.



```
root@kali:~# nmap -O 192.168.2.10
Starting Nmap 7.70 ( https://nmap.org ) at 2023-11-08 15:18 CET
Nmap scan report for 192.168.2.10
Host is up (0.0016s latency).
Not shown: 977 closed ports
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 14.49 seconds
```

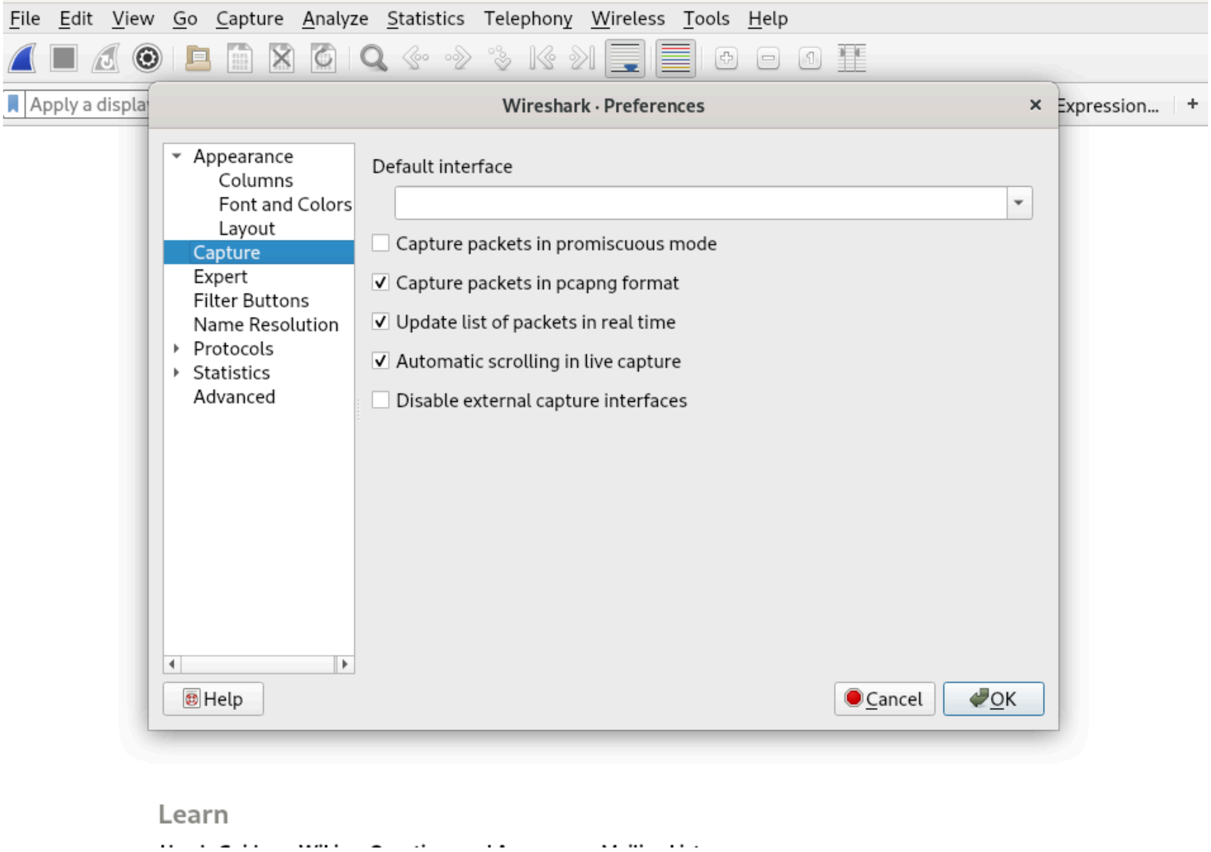
Figure 3 : Scanning the Metasploitable server for its operating system, open ports and running services.

Further information about the attack surface could be gathered by implementing more in-depth scans on the identified devices. For instance, performing detailed port scans (beyond a ping scan) to reveal open ports and potentially vulnerable services on the Windows XP system and the Metasploitable server.

The lesson that we have learnt from this task is that we can gain information about other devices that are connected to the same network, this can be useful to know which devices are connected to your network and prevent attacks from unknown devices.

2.2 Wireshark

Here we just turned off the promiscuous mode.

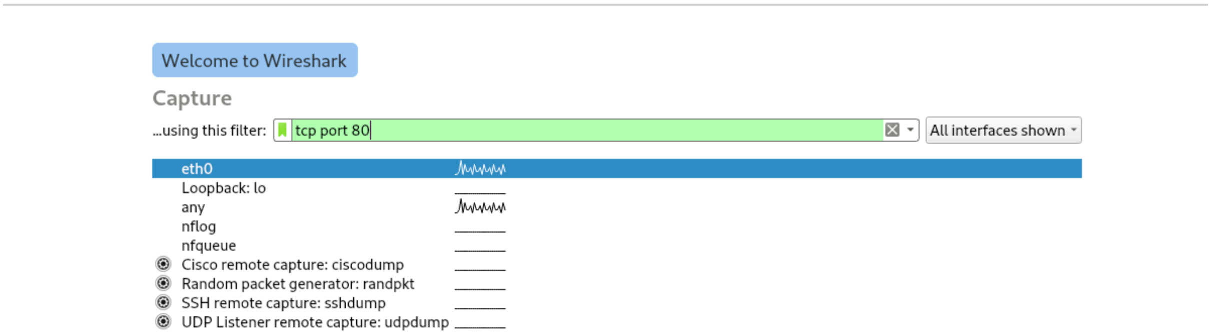


Default interface

- ☐ Capture packets in promiscuous mode
- ☑ Capture packets in pcapng format
- ☑ Update list of packets in real time
- ☑ Automatic scrolling in live capture
- ☐ Disable external capture interfaces

Figure 4 : Turning off the promiscuous mode in the Wireshark.

Then we set the capture filter to tcp port 80.

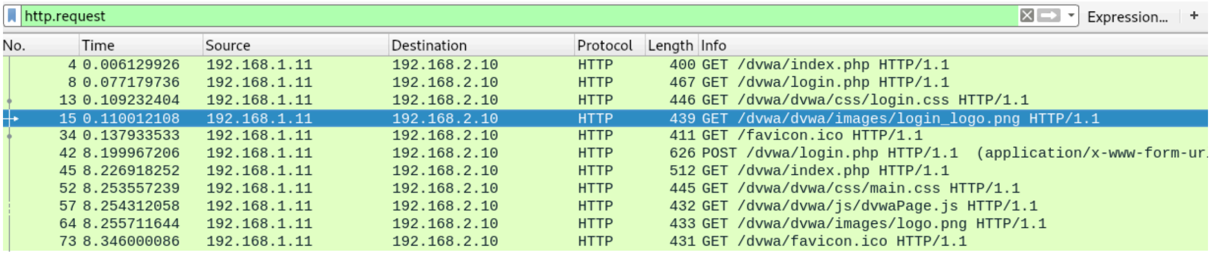


...using this filter: tcp port 80

```
eth0
Loopback: lo
any
nflog
nfqueue
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

Figure 5 : Setting capture filter to tcp port 80.

Then in Kali VM, we opened the browser and connected to http://192.168.2.10/dvwa/index.php with a username and password to generate som HTTP traffic. Then we filtered the http request by using “http.request”.

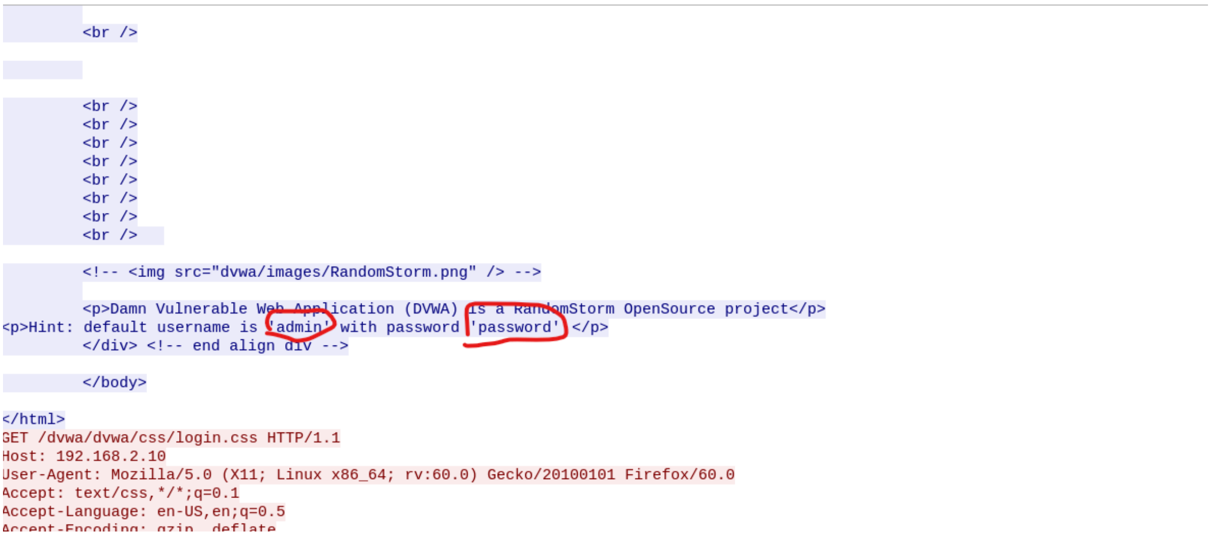


http.request

No.	Time	Source	Destination	Protocol	Length	Info
4	0.006129926	192.168.1.11	192.168.2.10	HTTP	400	GET /dvwa/index.php HTTP/1.1
8	0.077179736	192.168.1.11	192.168.2.10	HTTP	467	GET /dvwa/login.php HTTP/1.1
13	0.109232404	192.168.1.11	192.168.2.10	HTTP	446	GET /dvwa/dvwa/css/login.css HTTP/1.1
15	0.110012108	192.168.1.11	192.168.2.10	HTTP	439	GET /dvwa/dvwa/images/login_logo.png HTTP/1.1
34	0.137933533	192.168.1.11	192.168.2.10	HTTP	411	GET /favicon.ico HTTP/1.1
42	8.199967206	192.168.1.11	192.168.2.10	HTTP	626	POST /dvwa/login.php HTTP/1.1 (application/x-www-form-ur.
45	8.226918252	192.168.1.11	192.168.2.10	HTTP	512	GET /dvwa/index.php HTTP/1.1
52	8.253557239	192.168.1.11	192.168.2.10	HTTP	445	GET /dvwa/dvwa/css/main.css HTTP/1.1
57	8.254312058	192.168.1.11	192.168.2.10	HTTP	432	GET /dvwa/dvwa/js/dvwaPage.js HTTP/1.1
64	8.255711644	192.168.1.11	192.168.2.10	HTTP	433	GET /dvwa/dvwa/images/logo.png HTTP/1.1
73	8.346000086	192.168.1.11	192.168.2.10	HTTP	431	GET /dvwa/favicon.ico HTTP/1.1

Figure 6 : Filtering the http request using “http.request”

To analyze the traffic, we used “Follow TCP Stream” to get information, and from that we could see the username and password used on the web browser.

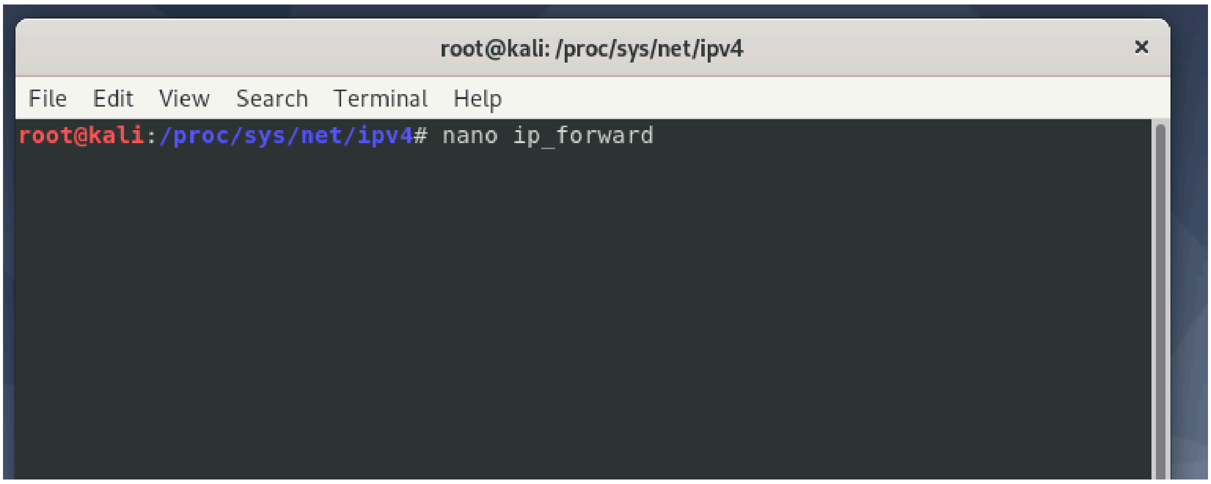


```
        <br />


        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />

        <!-- <img src="dvwa/images/RandomStorm.png" /> -->

        <p>Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project</p>
<p>Hint: default username is 'admin' with password 'password' </p>
        </div> <!-- end align div -->

        </body>

</html>
GET /dvwa/dvwa/css/login.css HTTP/1.1
Host: 192.168.2.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
```

Figure 7 : Showing the username and password in Wireshark.

2.3 Ettercap

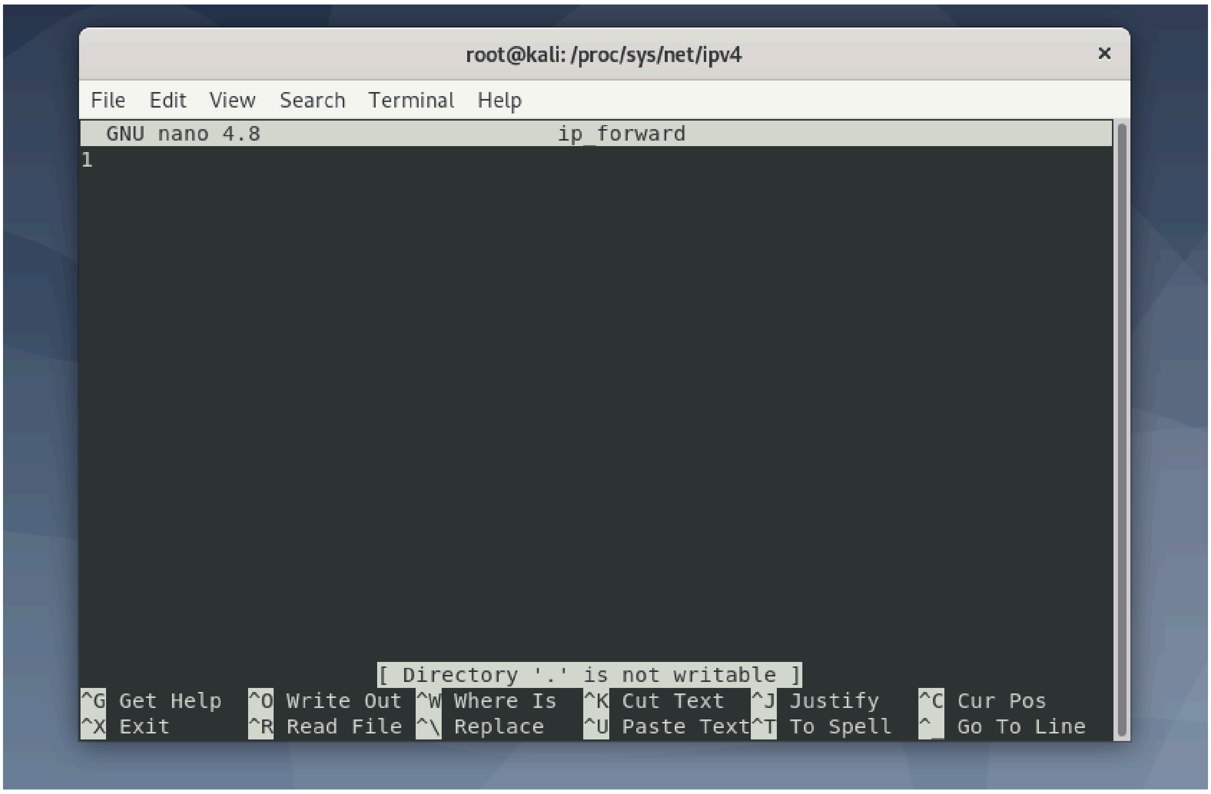
Here we are opening the ip_forward file using nano.



```
root@kali:/proc/sys/net/ipv4# nano ip_forward
```

Figure 8 : Opening the ip_forward file using nano.

We changed the value in the file ip_forward to a 1 instead of a 0.

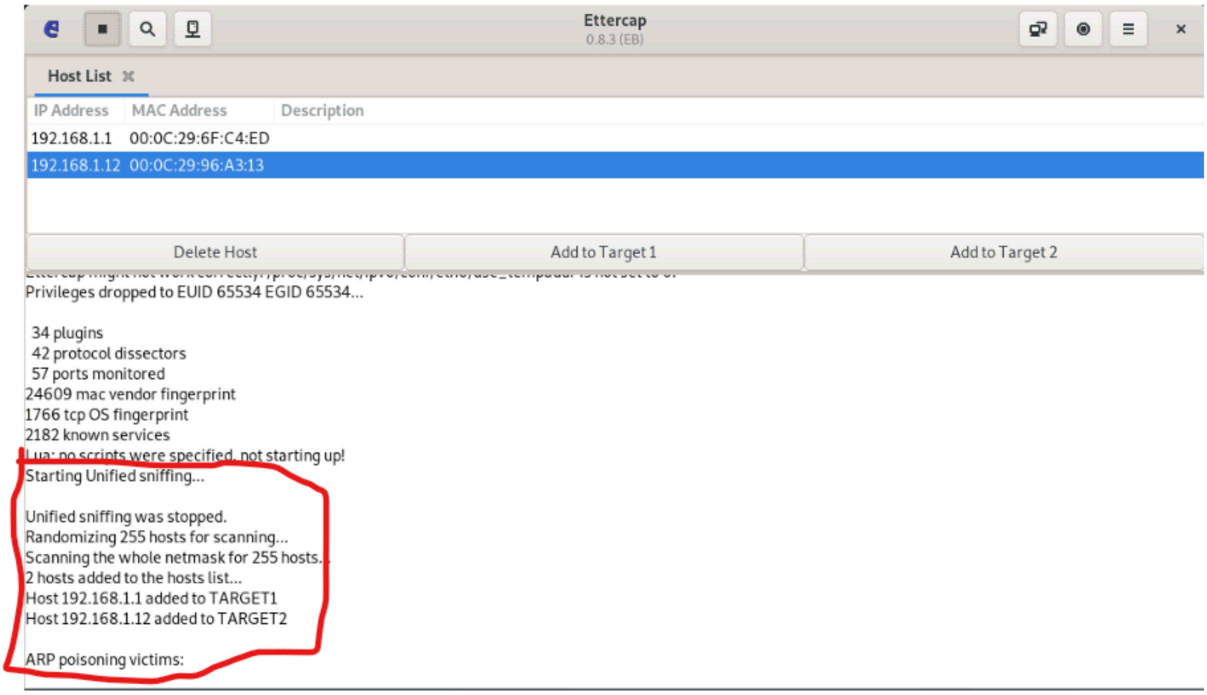


root@kali: /proc/sys/net/ipv4
File Edit View Search Terminal Help

```
  GNU nano 4.8                        ip_forward
1















                       [ Directory '.' is not writable ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
```

Figure 9 : Changing from 0 to 1 in the ip_forward file.

Then we opened the Ettercap, then chose "*Sniff*" from the menu and "*Unified sniffing*" in submenu. Then we set the network interface to eth0, and chose the hosts from the "HostsList" from the submenu, and added the IP "*192.168.1.1*" to target 1 and the IP "*192.168.1.12*" to Target 2. Then we chose "*Mitm*" from the the menu and "*ARP poisoning*" from the submenu, and chose "*Sniff remote connection*" when prompted, then we started sniffing.

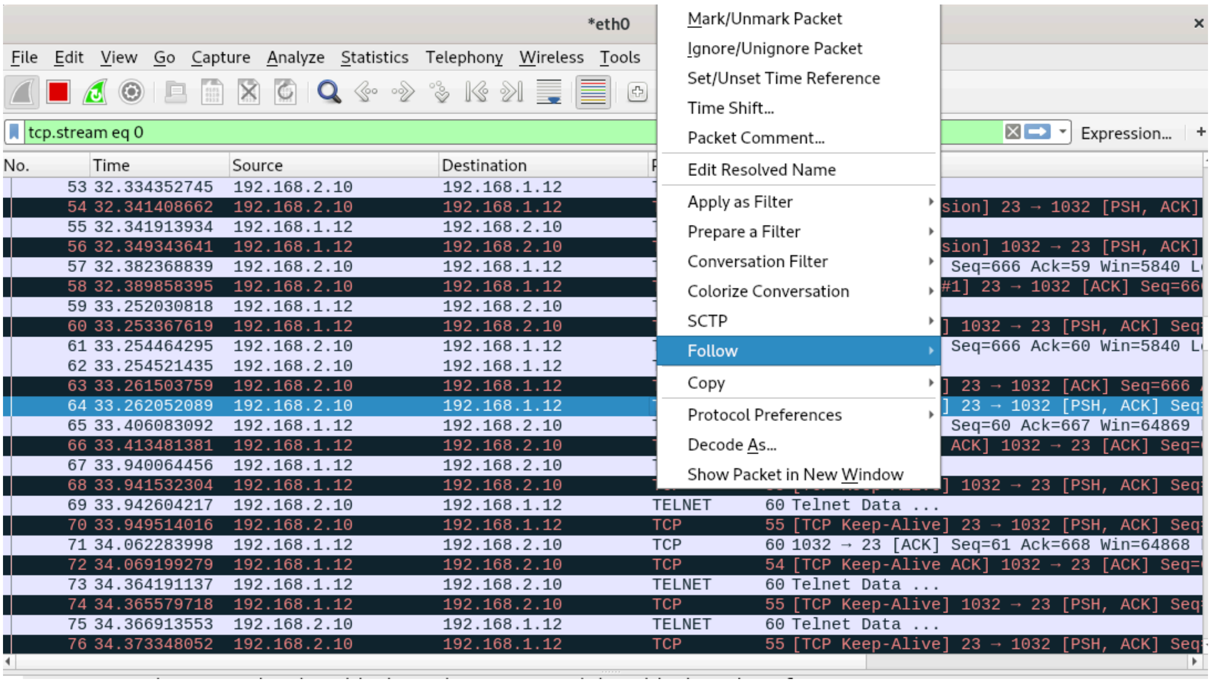


```
IP Address    MAC Address         Description
192.168.1.1   00:0C:29:6F:C4:ED
192.168.1.12  00:0C:29:96:A3:13
```

```
Privileges dropped to EUID 65534 EGID 65534...

 34 plugins
 42 protocol dissectors
 57 ports monitored
24609 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Unified sniffing was stopped.
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
2 hosts added to the hosts list...
Host 192.168.1.1 added to TARGET1
Host 192.168.1.12 added to TARGET2

ARP poisoning victims:
```

Figure 10: Starting Unified sniffing and adding the hosts to Target 1 and Target 2 and starting ARP poisoning.

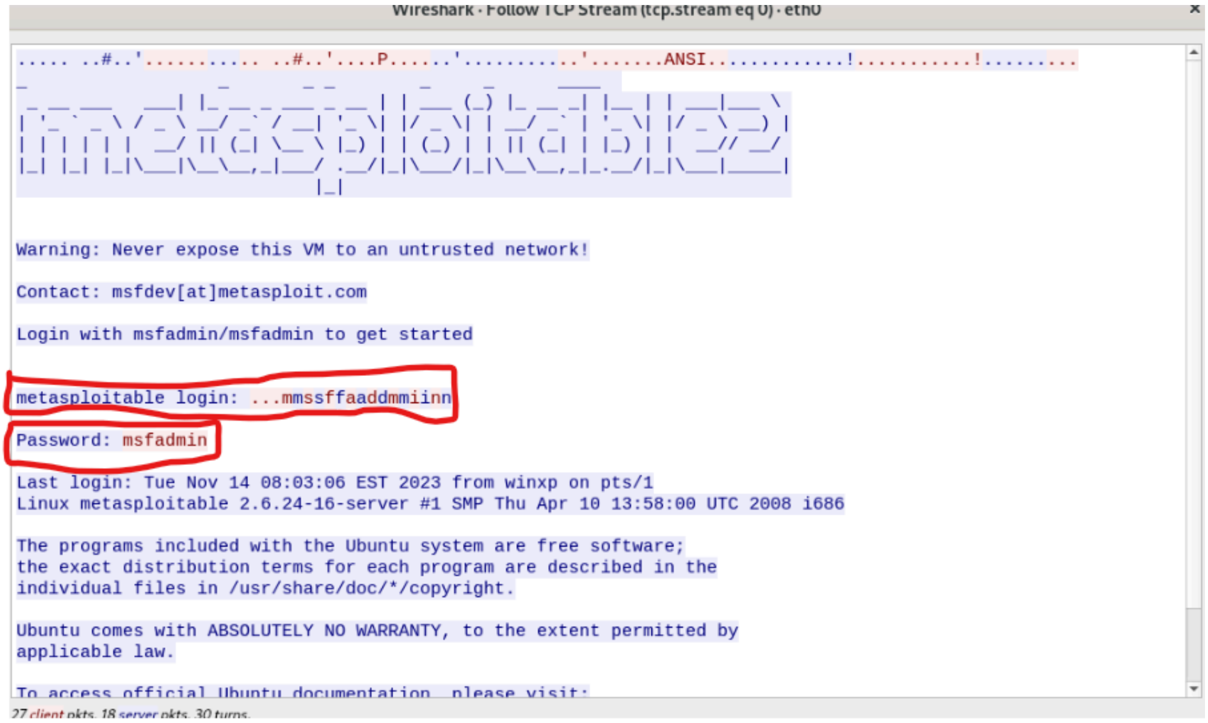
On Wireshark we started to capture traffic on eth0, and from the windows client machine we connected to the server (the Metasploitable machine) on its Telnet port, using the command - *telnet 192.168.2.10*. Then to examine the captured packets in Wireshark for the credentials we used Follow → TCP Stream.



```
tcp.stream eq 0

No.   Time          Source         Destination
  53 32.334352745  192.168.2.10   192.168.1.12
  54 32.341408662  192.168.2.10   192.168.1.12                       sion] 23 → 1032 [PSH, ACK]
  55 32.341913934  192.168.1.12   192.168.2.10
  56 32.349343641  192.168.1.12   192.168.2.10                       sion] 1032 → 23 [PSH, ACK]
  57 32.382368839  192.168.2.10   192.168.1.12                        Seq=666 Ack=59 Win=5840 L
  58 32.389858395  192.168.2.10   192.168.1.12                       #1] 23 → 1032 [ACK] Seq=66
  59 33.252030818  192.168.1.12   192.168.2.10
  60 33.253367619  192.168.1.12   192.168.2.10                       ] 1032 → 23 [PSH, ACK] Seq
  61 33.254464295  192.168.2.10   192.168.1.12                        Seq=666 Ack=60 Win=5840 L
  62 33.254521435  192.168.2.10   192.168.1.12
  63 33.261503759  192.168.2.10   192.168.1.12                       ] 23 → 1032 [ACK] Seq=666
  64 33.262052089  192.168.2.10   192.168.1.12                       ] 23 → 1032 [PSH, ACK] Seq
  65 33.406083092  192.168.1.12   192.168.2.10                        Seq=60 Ack=667 Win=64869
  66 33.413481381  192.168.1.12   192.168.2.10                        ACK] 1032 → 23 [ACK] Seq=
  67 33.940064456  192.168.1.12   192.168.2.10
  68 33.941532304  192.168.1.12   192.168.2.10                       ] 1032 → 23 [PSH, ACK] Seq
  69 33.942604217  192.168.2.10   192.168.1.12   TELNET     60 Telnet Data ...
  70 33.949514016  192.168.2.10   192.168.1.12   TCP        55 [TCP Keep-Alive] 23 → 1032 [PSH, ACK] Seq
  71 34.062283998  192.168.1.12   192.168.2.10   TCP        60 1032 → 23 [ACK] Seq=61 Ack=668 Win=64868
  72 34.069199279  192.168.1.12   192.168.2.10   TCP        54 [TCP Keep-Alive ACK] 1032 → 23 [ACK] Seq=
  73 34.364191137  192.168.1.12   192.168.2.10   TELNET     60 Telnet Data ...
  74 34.365579718  192.168.1.12   192.168.2.10   TCP        55 [TCP Keep-Alive] 1032 → 23 [PSH, ACK] Seq
  75 34.366913553  192.168.2.10   192.168.1.12   TELNET     60 Telnet Data ...
  76 34.373348052  192.168.2.10   192.168.1.12   TCP        55 [TCP Keep-Alive] 23 → 1032 [PSH, ACK] Seq
```

Figure 11 : Following the TCP Stream.

From following the TCP stream we got the results below, where we can recognize the username and password used to log in to the server.



```
..... ..#..'.......... ..#..'....P.......'...........'.......ANSI.............!...........!..........
_                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

metasploitable login: ...mmssffaaddmmiinn

Password: msfadmin

Last login: Tue Nov 14 08:03:06 EST 2023 from winxp on pts/1
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:
```

Figure 12 : Showing the username and password for Metasploitable machine.

Other possible malicious motives using Ettercap could be active attacks, such as altering system resources or destroying Another malicious motive could be session hijacking, where the attacker aims to take control of an established user session, or data interception for sensitive information such as financial transactions or personal data.

3. Possible Mistakes

From our results of the experiments, where we have gotten the expected outcome, we believe that not many mistakes were made.

For Nmap, there were a lot of possible commands to use, and therefore we might not have used the most usable/effective one. On the other hand, we believe that we got the right results from the commands we used and therefore there should not have been any major mistakes made. We used to scan the Metasploitable server for its operating system, open ports and running services we used the "*nmap -O* " command, but as description of this command, it should only know the operating system of the device, but we actually got the expected results when we used it, it got the open ports and running services to, so actually we should use another command to scan the open ports and running servers, for example : "*nmap - sS*".

For both Wireshark and Ettercap, the instructions for the experiments were very straightforward and it was hard to make any major mistakes. Since we got the expected results we believe there were no mistakes made.