



CSCI-6658-01

**ETHICAL HACKING**



Infoseclablearning Assignment-3

**Performing a Denial of Service Attack from the**

**WAN**

Student Info:

Name : Akhila Parankusham

Student ID: 00810899

Email: [apara7@unh.newhaven.edu](mailto:apara7@unh.newhaven.edu)

## **TABLE OF CONTENTS**

<b>Executive Summary</b> .....	02
Highlights.....	02
Objectives.....	02
<b>Lab Description Details</b> .....	02
<b>Supporting Evidence</b> .....	03
<b>Conclusion &amp; Wrap-up</b> .....	16

## Executive Summary

### Highlights

This hands-on lab provides hands-on experience conducting basic denial-of-service (DoS) attacks from a distant location.

- The lab employs the Low Orbit Ion Cannon (LOIC) technology to demonstrate TCP flood, UDP flood, and HTTP flood DoS attacks.
- Using Tcpdump, a Linux sniffer system is used to collect the attack communication.
- The capinfos command displays packet capture statistics.
- SYN packets are transmitted in the TCP flood attack to flood the target with open connections.
- The UDP flood involves sending UDP datagrams to random ports.
- The HTTP flood causes a web server to become overloaded with GET and POST requests.
- The HTTP flood is more effective when the "Wait for reply" function in LOIC is off.
- The primary goal of these attacks is to drain server resources and prevent authorized users from accessing services.
- Mastering traffic analysis and network sniffing is a critical skill set for ethical hackers.
- For instructional and penetration testing purposes, the lab gives hands-on experience with typical DoS attack tools and techniques.

### Objectives

The major goal of this lab is to obtain hands-on experience with various denial-of-service (DoS) attacks on a wide area network (WAN). The lab's specific goals are as follows:

- The use of a TCP flood to simulate a DoS attack.
- A DoS attack using a UDP flood is demonstrated.
- Using an HTTP flood to simulate a DoS attack.

### Lab Description Details

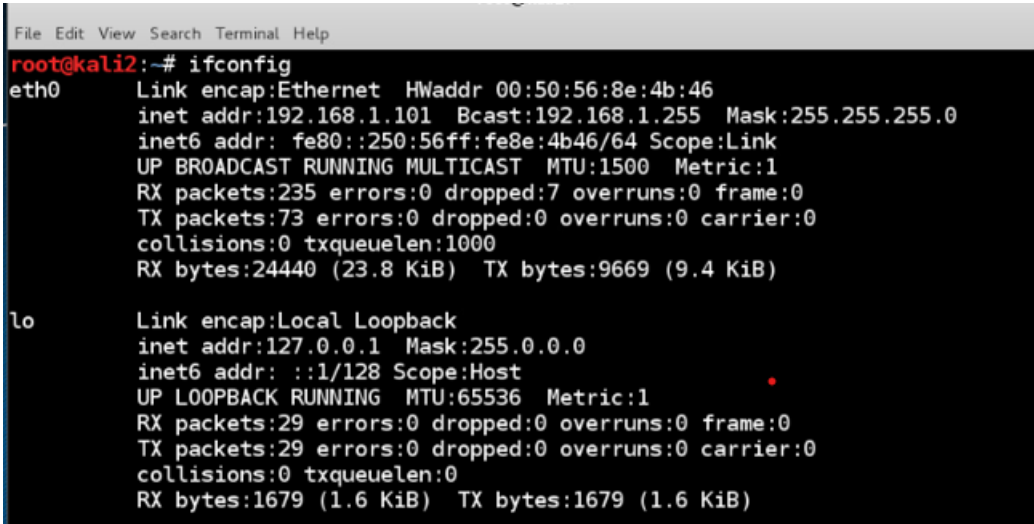
- **TCP Flood:** A DoS attack that uses TCP's three-way handshake to flood a target with connection requests.

- **UDP Flood:** A DoS attack that floods the victim with UDP datagrams on random ports.
- **HTTP Flood:** Shows a DoS attack on web servers by assaulting them with a large number of genuine GET and POST requests.
- **Ifconfig:** Enumerates and configures network interfaces on a Linux system.
- **Tcpdump:** It is a Linux command-line application for collecting and analyzing network data.
- **Denial of Service (DoS):** Describes the nature of assaults aimed at disrupting network services and rendering resources unavailable.
- **LOIC (Low Orbit Ion Cannon):** An attacker-favored network stress testing and DoS tool.
- **Capinfos:** A program that displays data regarding capture files, such as traffic analysis during DoS assaults.

## Supporting Evidence

**Step 1:** Open the terminal and check for the IP address of the system.

```
# ifconfig
```



```
File Edit View Search Terminal Help
root@kali2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:8e:4b:46
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe8e:4b46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:235 errors:0 dropped:7 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24440 (23.8 KiB)  TX bytes:9669 (9.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1679 (1.6 KiB)  TX bytes:1679 (1.6 KiB)
```

**Step 2:** Save the IP address configuration and view its contents from the file.

```
# ifconfig > ip1.txt
```

```
# cat ip1.txt
```

```
File Edit View Search Terminal Help
collisions:0 txqueuelen:0
RX bytes:1679 (1.6 KiB) TX bytes:1679 (1.6 KiB)

root@kali2:~# ifconfig > ip1.txt
root@kali2:~# cat ip1.txt
eth0      Link encap:Ethernet  HWaddr 00:50:56:8e:4b:46
          inet addr:192.168.1.101 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe8e:4b46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:253 errors:0 dropped:7 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25520 (24.9 KiB) TX bytes:9669 (9.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1679 (1.6 KiB) TX bytes:1679 (1.6 KiB)
```

**Step 3:** View the contents of ip2.txt file

# cat ip2.txt

```
File Edit View Search Terminal Help
TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1679 (1.6 KiB) TX bytes:1679 (1.6 KiB)

root@kali2:~# cat ip2.txt
eth0      Link encap:Ethernet  HWaddr 00:0c:29:07:07:1c
          inet addr:192.168.1.101 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe07:71c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7439 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:827225 (807.8 KiB) TX bytes:8005 (7.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          sample flag: 999818
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1200 (1.1 KiB) TX bytes:1200 (1.1 KiB)
```

**Step 4:** Solve the sample challenges using cat command.



## SAMPLE CHALLENGE



## CHALLENGE #1

```
root@kali2:~# cat ip3.txt
eth0  Link encap:Ethernet  HWaddr 00:0c:29:07:07:1c
      inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
      inet6 addr: fe80::20c:29ff:fe07:71c/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:7439 errors:0 dropped:0 overruns:0 frame:0
      TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:827225 (807.8 KiB)  TX bytes:8005 (7.8 KiB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      flag:123457
```

**Step 5:** Verify that no IPv4 address is listed for eth0.

```
# ifconfig eth0 0.0.0.0 up
```

```
# ifconfig
```

```
root@kali2:~# ifconfig eth0 0.0.0.0 up
root@kali2:~# ifconfig
eth0    Link encap:Ethernet  HWaddr 00:50:56:8e:4b:46
      inet6 addr: fe80::250:56ff:fe8e:4b46/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:325 errors:0 dropped:7 overruns:0 frame:0
      TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:31306 (30.5 KiB)  TX bytes:10275 (10.0 KiB)

lo      Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:39 errors:0 dropped:0 overruns:0 frame:0
      TX packets:39 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:2179 (2.1 KiB)  TX bytes:2179 (2.1 KiB)
```

**Step 6:** Check all the options available for tcpdump.

```
# tcpdump --help
```

```
root@kali2:~# tcpdump --help
tcpdump version 4.6.2
libpcap version 1.6.2
OpenSSL 1.0.1k 8 Jan 2015
Usage: tcpdump [-aAbdDefhHIJKlLnOpqRStuUvX#] [-B size] [-c count]
      [-C file_size] [-E algo:secret] [-F file] [-G seconds]
      [-i interface] [-j timestamp] [-M secret] [--number]
      [-Q in|out|inout]
      [-r file] [-s snaplen] [--time-stamp-precision precision]
      [-T type] [--version] [-V file]
      [-w file] [-W filecount] [-y datalinktype] [-z command]
      [-Z user] [expression]
```

## TCP Flood:

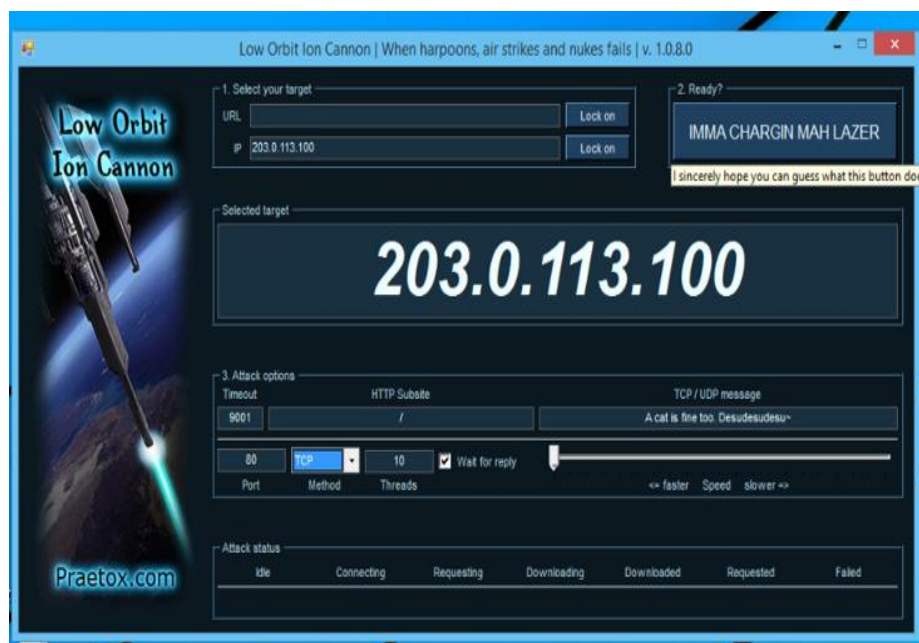
**Step 7:** Check for the tcpdump sniffing on the eth0 interface.

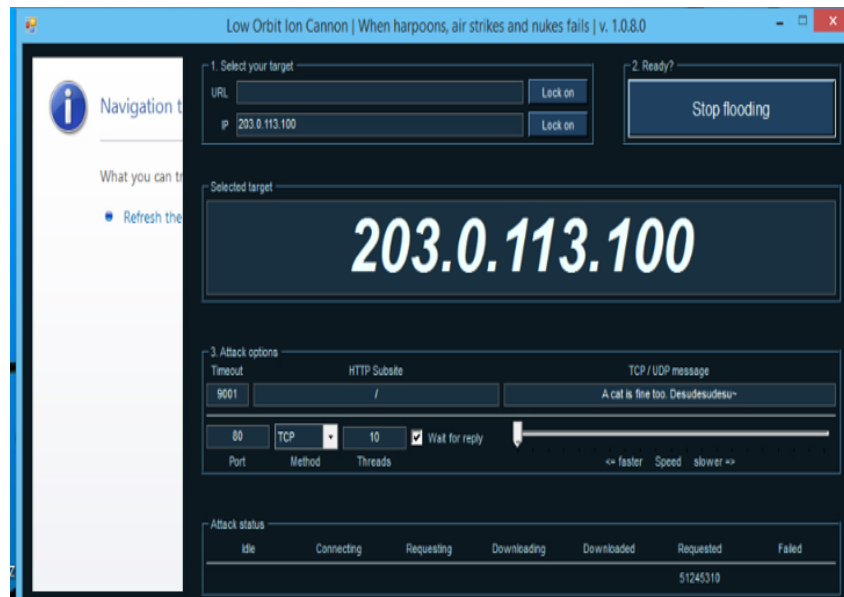
```
# tcpdump -i eth0 -nntttt -s 0 -w TCPcapture.cap
```

```
root@kali2:~# tcpdump -i eth0 -nntttt -s 0 -w TCPcapture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

**Step 8:** Launch the Windows 8.1 Attack Machine button with external IP Address 175.45.176.200 and then start the packet traffic utilizing the Low Orbit Cannon.

**Step 9:** Launch LOIC.exe and set the target. Then click on lock on. Select Protocol as TCP from the dropdown menu. Click IMMA CHARGIN MAH LAZER button. Wait for 30 seconds and click on stop flooding button and return back to Linux Sniffer.





**Step 10:** Stop the capture, view and examine the total number of packets in the TCPcapture file.

```
# capinfos TCPcapture.cap
```

```

root@kali2: ~
File Edit View Search Terminal Help
es
^C873885 packets captured
873885 packets received by filter
0 packets dropped by kernel
root@kali2:~# capinfos TCPcapture.cap
File name: TCPcapture.cap
File type: Wireshark/tcpdump/... - pcap
File encapsulation: Ethernet
Packet size limit: file hdr: 262144 bytes
Number of packets: 873 k
File size: 67 MB
Data size: 53 MB
Capture duration: 259 seconds
Start time: Mon Oct 16 18:46:18 2023
End time: Mon Oct 16 18:50:37 2023
Data byte rate: 206 kBps
Data bit rate: 1648 kbps
Average packet size: 61.07 bytes
Average packet rate: 3373 packets/sec
SHA1: 12537e7b658b1ef6c938e5d1a5572942cc4c260d
RIPEMD160: 815eef5a03ad8aaf448accd373eba4cb65c23150
MD5: 7439f61ea57b1aa7c13ac508fed8f66a
Strict time order: False
root@kali2:~#

```

**UDP Flood:**

**Step 11:** Check for the tcpdump sniffing on the eth0 interface.

```
# tcpdump -i eth0 -nntttt -s 0 -w UDPcapture.cap
```



```
root@kali2:~# tcpdump -i eth0 -nntttt -s 0 -w UDPcapture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

**Step 12:** Launch the Windows 8.1 Attack Machine button with external IP Address 175.45.176.200 and then start the packet traffic utilizing the Low Orbit Cannon.

**Step 13:** Select Protocol as UDP from the dropdown menu. Click IMMA CHARGIN MAH LAZER button. Wait for 30 seconds and click on stop flooding button and return to Linux Sniffer.



**Step 14:** Stop the capture, view and examine the total number of packets captured in the UDPcapture file.

```
# capinfos UDPcapture.cap
```

```
root@kali2: ~
File Edit View Search Terminal Help
es
^C84 packets captured
84 packets received by filter
0 packets dropped by kernel
root@kali2:~# capinfos UDPcapture.cap
File name: UDPcapture.cap
File type: Wireshark/tcpdump/... - pcap
File encapsulation: Ethernet
Packet size limit: file hdr: 262144 bytes
Number of packets: 84
File size: 7872 bytes
Data size: 6504 bytes
Capture duration: 88 seconds
Start time: Mon Oct 16 18:52:22 2023
End time: Mon Oct 16 18:53:50 2023
Data byte rate: 73 bytes/s
Data bit rate: 589 bits/s
Average packet size: 77.43 bytes
Average packet rate: 0 packets/sec
SHA1: 3dcecb78a3e28460a53eed8c808d5ca94817bd0
RIPEMD160: 2a2899075ace38b1fddc7dfc0e22f241b95275c3
MD5: 2bc4880de85c4990a4940f43003ec54f
Strict time order: True
```

## HTTP Flood:

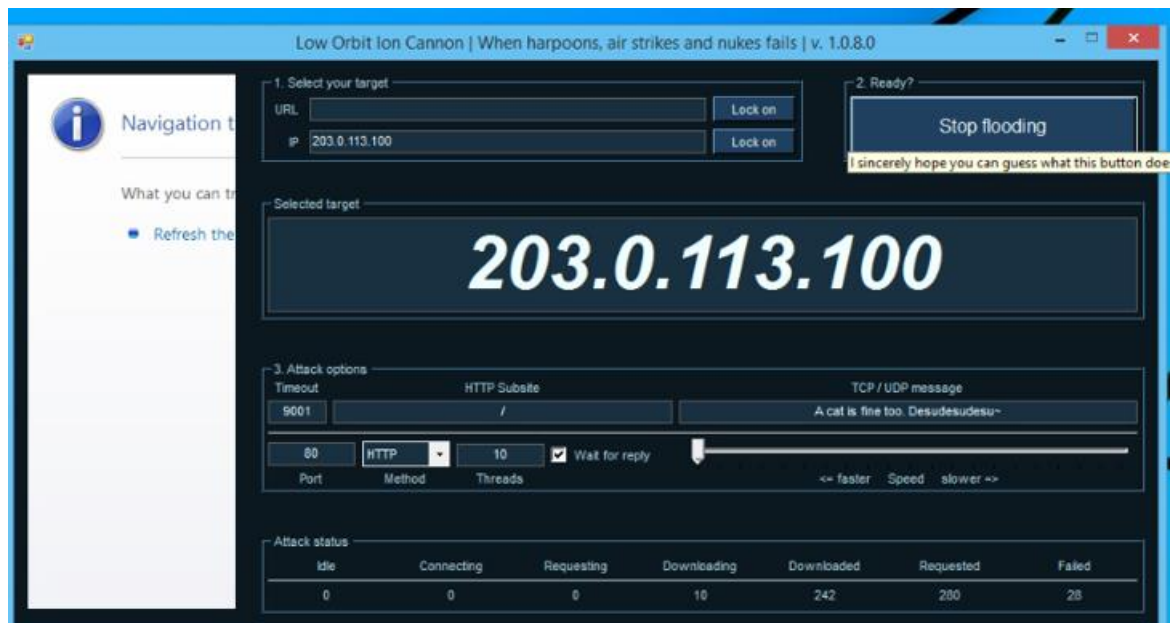
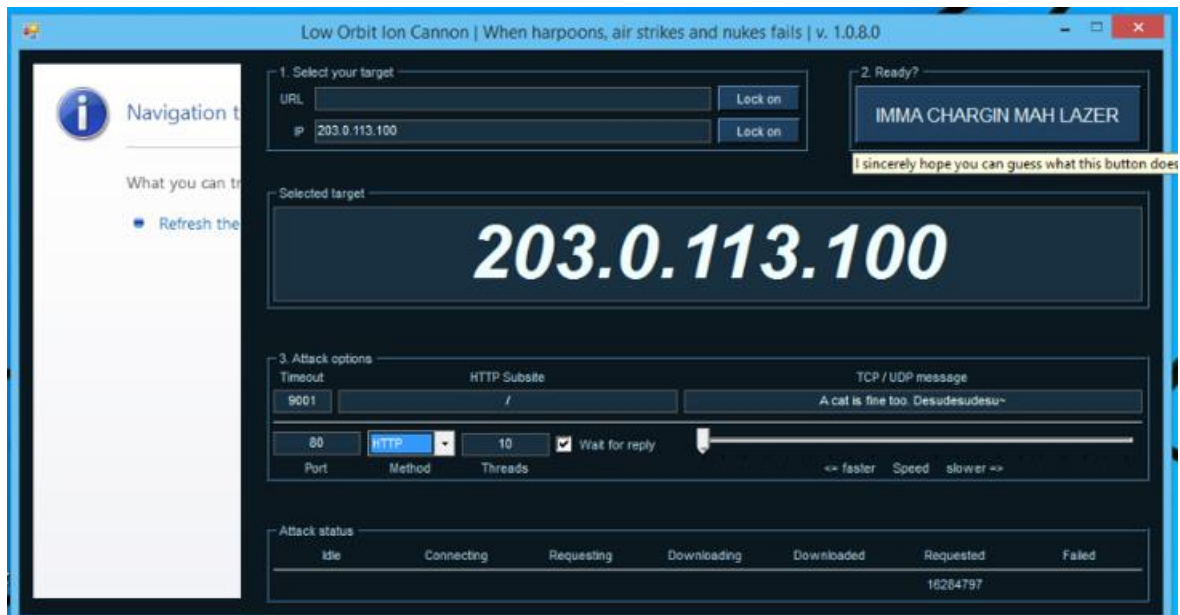
**Step 15:** Check for the tcpdump sniffing on the eth0 interface.

```
# tcpdump -i eth0 -nntttt -s 0 -w HTTPcapture.cap
```

```
root@kali2:~# tcpdump -i eth0 -nntttt -s 0 -w HTTPcapture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
es
```

**Step 16:** Launch the Windows 8.1 Attack Machine button with external IP Address 175.45.176.200 and then start the packet traffic utilizing the Low Orbit Cannon.

**Step 17:** Select Protocol as HTTP from the dropdown menu. Click IMMA CHARGIN MAH LAZER button. Wait for 30 seconds and click on stop flooding button and return to Linux Sniffer.



**Step 18:** Stop the capture, view and examine the total number of packets captured in the HTTPcapture file.

```
# capinfos HTTPcapture.cap
```

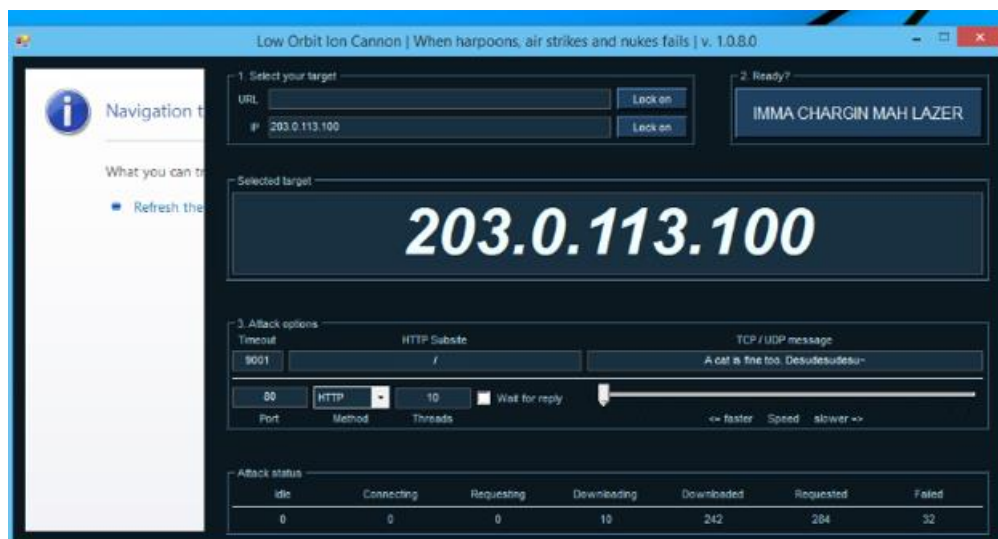
```
File Edit View Search Terminal Help
es
^C631 packets captured
631 packets received by filter
0 packets dropped by kernel
root@kali2:~# capinfos HTTPcapture.cap
File name:      HTTPcapture.cap
File type:      Wireshark/tcpdump/... - pcap
File encapsulation: Ethernet
Packet size limit: file hdr: 262144 bytes
Number of packets: 631
File size:      490 kB
Data size:      480 kB
Capture duration: 85 seconds
Start time:     Mon Oct 16 18:55:39 2023
End time:       Mon Oct 16 18:57:04 2023
Data byte rate: 5624 bytes/s
Data bit rate:  44 kbps
Average packet size: 761.53 bytes
Average packet rate: 7 packets/sec
SHA1:           29e1503a242edb071dba51ef92236a3cfabe87a7
RIPEMD160:      cf23026204b7acc056331ac8ee427f62040443be
MD5:            e02c9ebae68b822c3ea17b58048f205f
Strict time order: True
```

**Step 19:** Check for the tcpdump sniffing on the eth0 interface.

```
# tcpdump -i eth0 -nntttt -s 0 -w HTTP2capture.cap
```

```
root@kali2:~# tcpdump -i eth0 -nntttt -s 0 -w HTTP2capture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

**Step 20:** Launch the Windows 8.1 Attack Machine and uncheck the wait for reply button. Click on IMMA CHARGIN MAH LAZER button. Wait for 30 seconds and click on stop flooding button and return to Linux Sniffer.





**Step 21:** Stop the capture, view and examine the total number of packets captured in the HTTP2capture file.

```
# capinfos HTTP2capture.cap
```

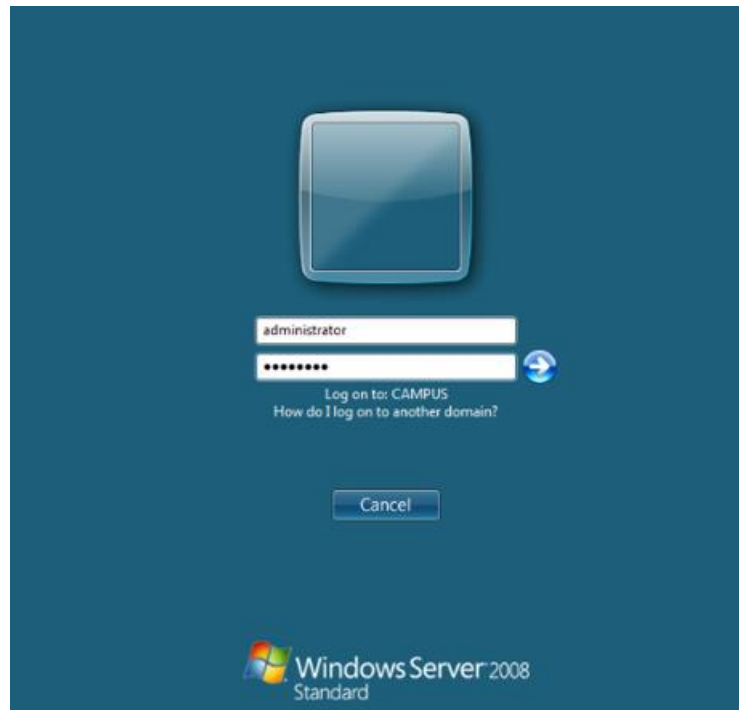
```

root@kali2: ~
File Edit View Search Terminal Help
es
^C1494 packets captured
1494 packets received by filter
0 packets dropped by kernel
root@kali2:~# capinfos HTTP2capture.cap
File name: HTTP2capture.cap
File type: Wireshark/tcpdump/... - pcap
File encapsulation: Ethernet
Packet size limit: file hdr: 262144 bytes
Number of packets: 1494
File size: 597 kB
Data size: 573 kB
Capture duration: 108 seconds
Start time: Mon Oct 16 18:58:28 2023
End time: Mon Oct 16 19:00:17 2023
Data byte rate: 5288 bytes/s
Data bit rate: 42 kbps
Average packet size: 383.82 bytes
Average packet rate: 13 packets/sec
SHA1: a8c56e80ce97e6fc757468b6592a48746b398f63
RIPEMD160: 7ed10edff8be49a29e339b60aaa1e93088696a2a
MD5: 4aa0a164771f3d525ac9709671a2aa85
Strict time order: True
  
```

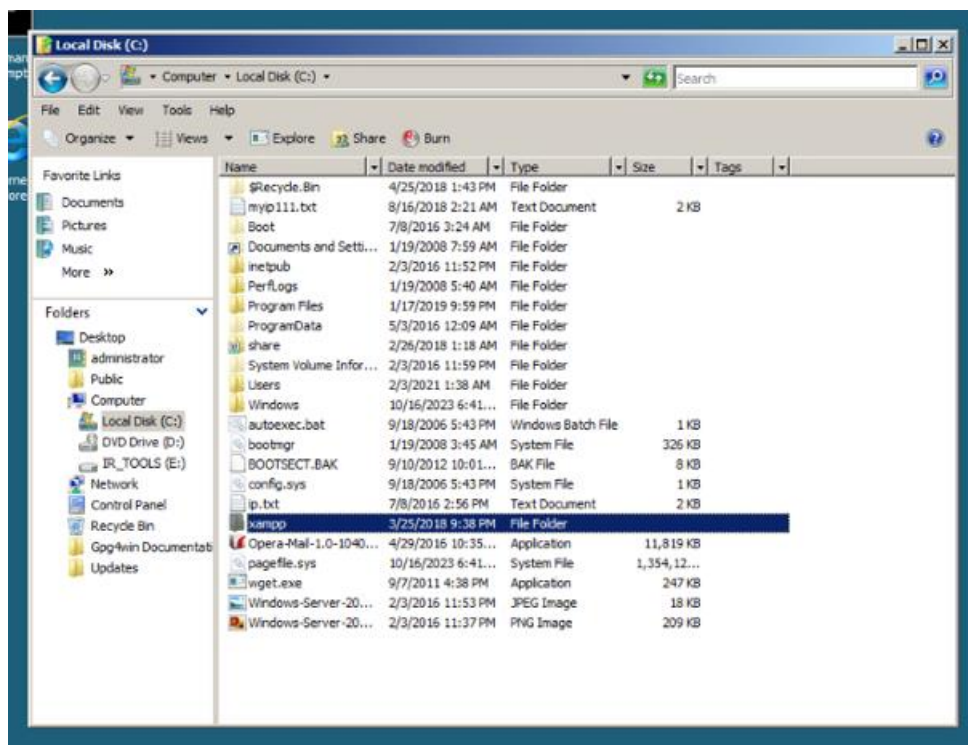
**Step 22:** Launch Windows Server. Once the machine is booted, log in to it using the credentials.

username: administrator

password: P@ssw0rd

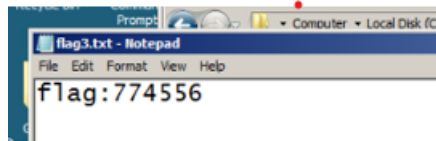


**Step 23:** Start button>Computer>C:>xampp folder>

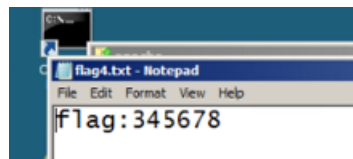




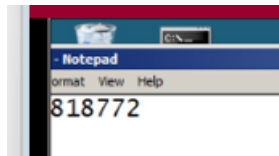
**Step 24:** Solve the challenge using flag3.txt file.



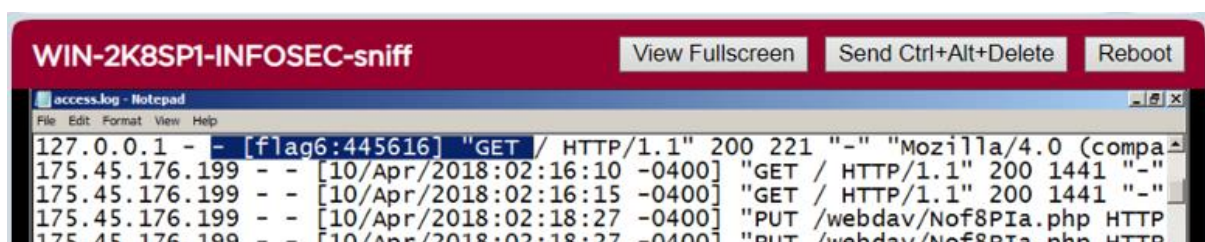
**Step 25:** Open apache folder and solve the challenge using the flag4.txt file.



**Step 26:** Open logs folder and solve the challenge using the flag5.txt file.



**Step 27:** Open access.log file and solve the challenge using the flag6.txt file.



**Step 28:** View the entries.

## Reboot

15



## Conclusion & Wrap-up

### Summary of Observations, Achievements, Setbacks, and Challenges

In this lab, we used the Low Orbit Ion Cannon (LOIC) software to launch denial-of-service assaults that included TCP SYN flood, UDP flood, and HTTP flood. We attentively watched the attack flow using tcpdump and measured packet size with capinfos. These attacks taxed resources and overwhelmed the target server, leaving its services unavailable. This practical training provided participants with hands-on exposure with common DoS attack tools and techniques across a wide area network (WAN).

In summary, the lab successfully accomplished its objectives by simulating various denial-of-service attacks over a WAN and measuring their effects using packet captures. The gained abilities and principles are applicable to ethical hacking and penetration testing.

#### Successes:

- TCP, UDP, and HTTP floods over the WAN were successfully initiated.
- Tcpdump was correctly setup to collect the attack flow.
- Capinfos was used to analyze the impact of the assaults using packet counts.

#### Failures:

- The HTTP flood's usefulness was restricted until the "Wait for reply" function of LOIC was deactivated.
- The victim server's service failure was not confirmed.

#### Challenges:

- It required some time to grasp the LOIC interface.
- The interpretation of tcpdump output and the application of filters posed a learning curve.
- Choosing the optimal assault length to enable successful captures without crashing the victim proved difficult.