Configuration Manual

MSc Research Project

MSc Cybersecurity

Eldhose Shaji

Student ID: x21195986

School of Computing

National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	**Eldhose Shaji**
Student ID:	**X21195986**
Programme:	**MSc Cybersecurity** **Year:** **2023**
Module:	**MSc Research Project**
Supervisor:	**Noel Cosgrave**
Submission Due Date:	**14/08/2023**
Project Title:	**Resource Isolation to Mitigate Denial-of-Service and DDoS Attacks in Cloud Computing**
Word Count:	**1608** **Page Count:13**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project. ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: **Eldhose Shaji**

Date: **14/08/2023**

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Eldhose Shaji
Student ID: x21195986

1 Introduction

The configuration manual is the list of specific configurations that were done in the host system as well as in the lab environment of the research project. The document also entails the details of the software tools used in the research and also the hardware requirements along with the configurations of it. The aim of the project is to properly secure the cloud resources by isolating them and thereby mitigate the DoS/DDoS attacks. The proper isolation can be helpful in not only mitigating the DDoS/DoS attacks but also helping to eliminate the chances of occurrence of co-process interference between tenants in the multi-cloud environment. The lab of the project is designed to match the exact real-time cloud environment but in the simulated and controlled environment. Thus, it will be helpful in assessing the robustness of the proposed method and to evaluate its efficiency.

2 Hardware Requirements

The lab environment is configured in the local laptop and the configurations of the same are listed in the table below.

Operating System	Windows 11 Home Edition
OS Version and Build	22H2, 22621.1992
Processor	11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 2.70 GHz
Storage	512 GB PCIe® NVMe™ M.2 SSD
RAM	16.0 GB (15.7 GB usable)
System Type	64-bit operating system, x64-based processor
System Make and Model	HP-Pavilion Series

3 Software Requirements and Configurations

Below is the list of software tools and their specifications used in the lab testing and evaluation of the proposed methodology for resource isolation.

i. Wireshark

Version: Version 4.0.0 (v4.0.0-0-g0cbe09cd796b).

Wireshark is a free and open-source packet analyser used for network troubleshooting, analysis, software development, and communications protocol research. Wireshark is used in this project to track packets and filter them by specific rules. In this project, Wireshark has been employed to monitor the entire packets during the event of an attack and also at the normal functioning of the system. Also, Wireshark has been employed to find out the packets that are really causing the DDoS/DoS attacks. The configurations enabled in Wireshark and the screen snip of the related data are shown below.

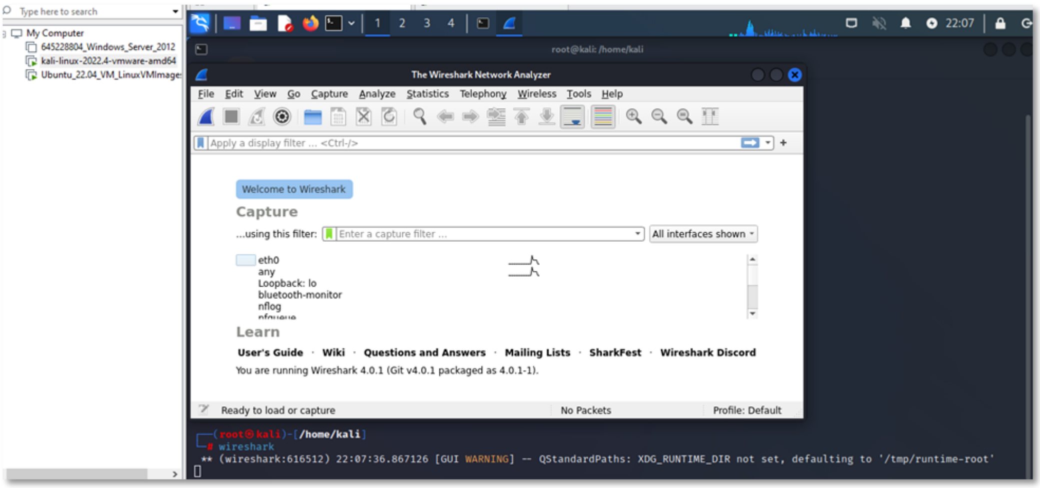


Figure 1 Wireshark Initialisation

Filter Name	Filter Expression
Ethernet address 00:00:5e:00:53:00	eth.addr == 00:00:5e:00:53:00
Ethernet type 0x0806 (ARP)	eth.type == 0x0806
Ethernet broadcast	eth.addr == ff:ff:ff:ff:ff:ff
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	ip.addr == 192.0.2.1
IPv4 address isn't 192.0.2.1	ip.addr != 192.0.2.1
IPv6 only	ipv6
IPv6 address 2001:db8::1	ipv6.addr == 2001:db8::1
TCP only	tcp
UDP only	udp
Non-DNS port	!(udp.port == 53 || tcp.port == 53)
TCP or UDP port is 80 (HTTP)	tcp.port == 80 || udp.port == 80
HTTP	http
No ARP and no DNS	not arp and not dns
Non-HTTP and non-SMTP to/from 192.0.2.1	ip.addr == 192.0.2.1 and tcp.port not in {80, 25}

Figure 2. Packet filtering rules applied.

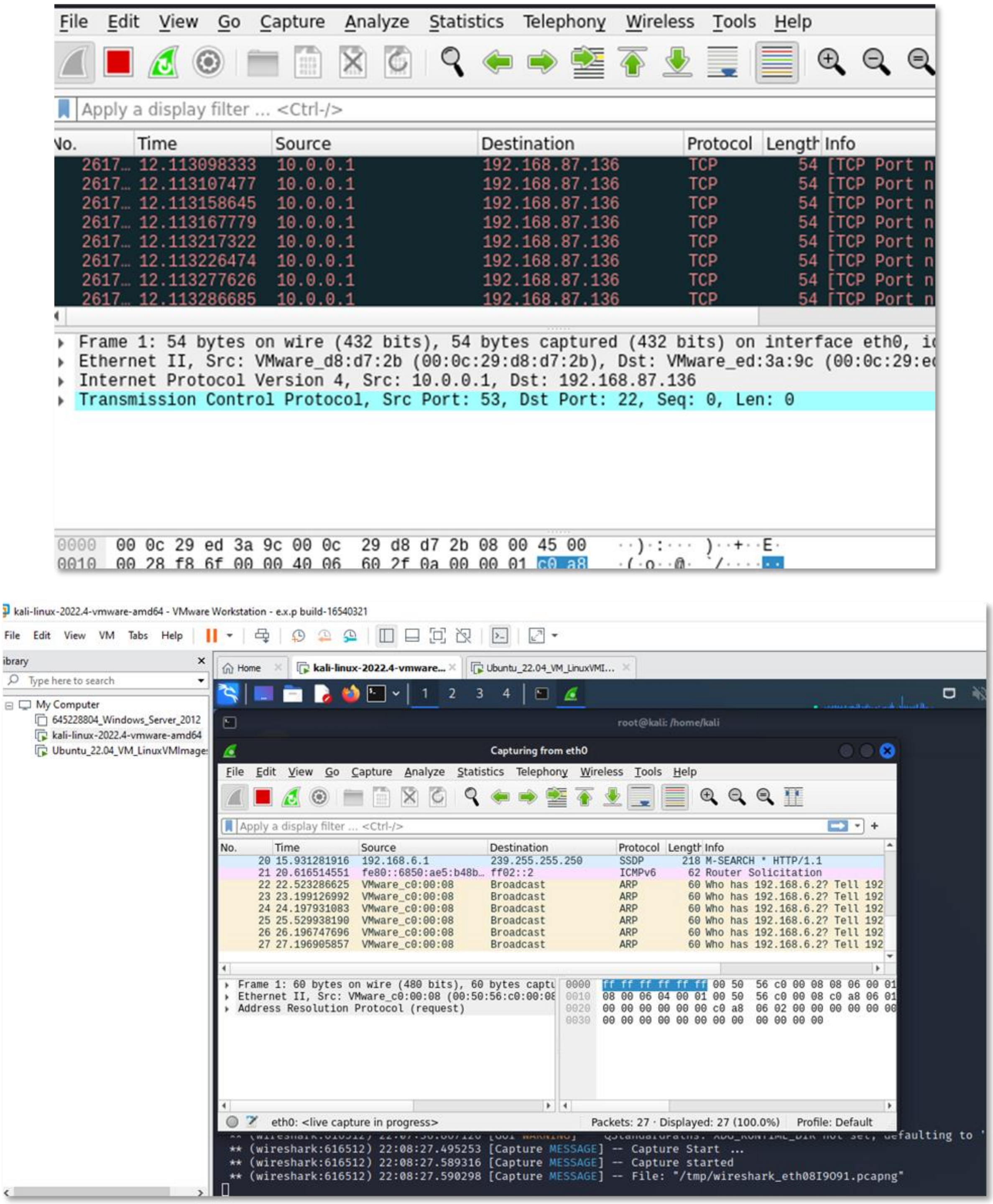
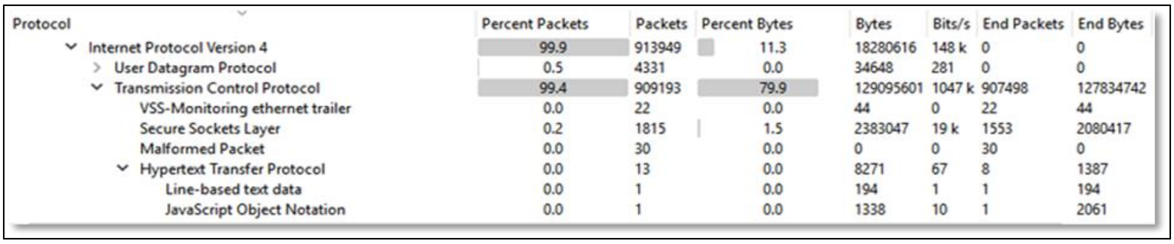
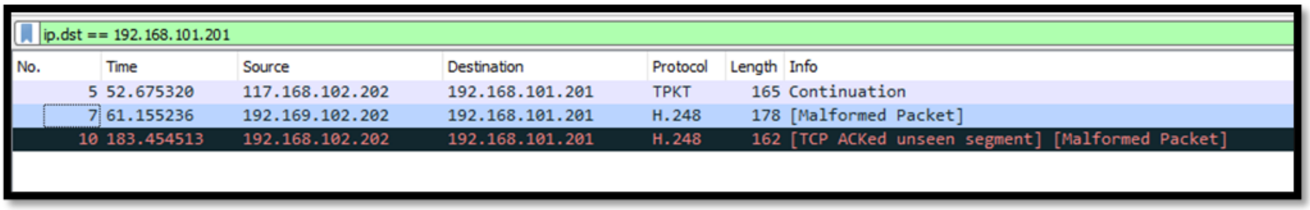


Figure 3. Initiated the Packet Tracing



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Internet Protocol Version 4	99.9	913949	11.3	18280616	148 k	0	0
User Datagram Protocol	0.5	4331	0.0	34648	281	0	0
Transmission Control Protocol	99.4	909193	79.9	129095601	1047 k	907498	127834742
VSS-Monitoring ethernet trailer	0.0	22	0.0	44	0	22	44
Secure Sockets Layer	0.2	1815	1.5	2383047	19 k	1553	2080417
Malformed Packet	0.0	30	0.0	0	0	30	0
Hypertext Transfer Protocol	0.0	13	0.0	8271	67	8	1387
Line-based text data	0.0	1	0.0	194	1	1	194
JavaScript Object Notation	0.0	1	0.0	1338	10	1	2061

Figure 4. Statistics of Inbound and Outbound Traffic



ip.dst == 192.168.101.201

No.	Time	Source	Destination	Protocol	Length	Info
5	52.675320	117.168.102.202	192.168.101.201	TPKT	165	Continuation
7	61.155236	192.169.102.202	192.168.101.201	H.248	178	[Malformed Packet]
10	183.454513	192.168.102.202	192.168.101.201	H.248	162	[TCP ACKed unseen segment] [Malformed Packet]

Figure 5 Blocked TCP Packets

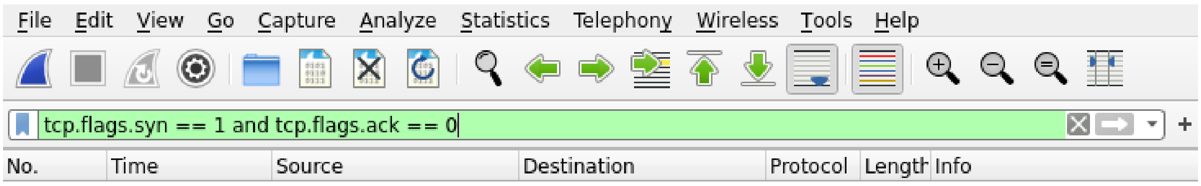


Figure 6. Filter Configurations for TCP Packets

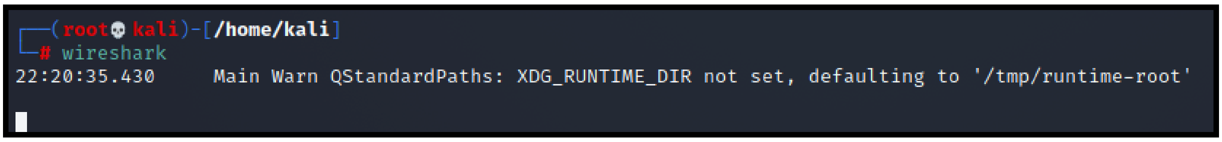
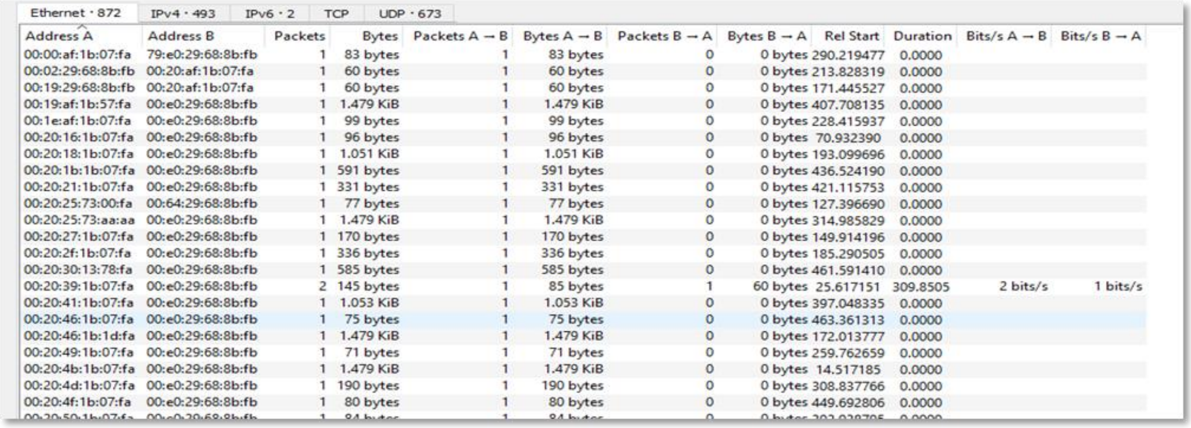
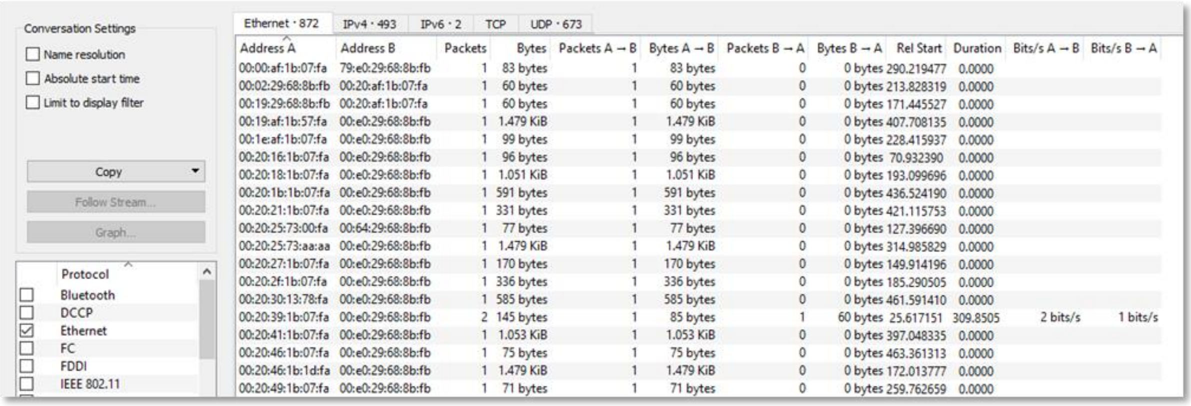


Figure 7 Initiating Wireshark from Kali VM



Ethernet · 872 | IPv4 · 493 | IPv6 · 2 | TCP | UDP · 673

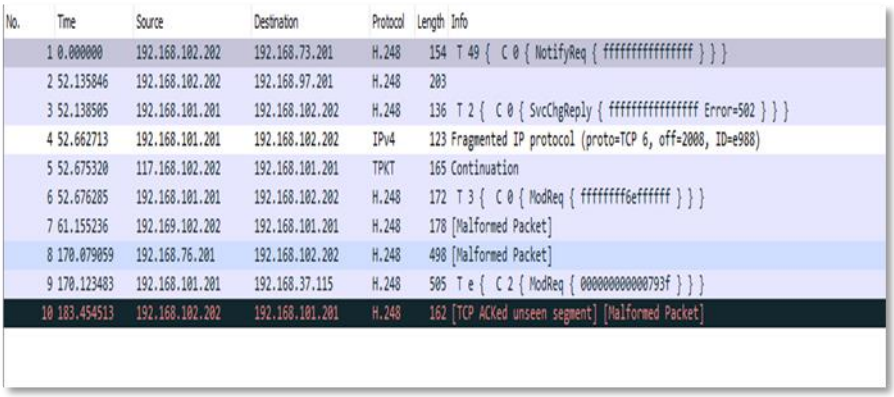
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:00:af:1b:07:fa	79:e0:29:68:8b:fb	1	83 bytes	1	83 bytes	0	0 bytes	290.219477	0.0000		
00:02:29:68:8b:fb	00:20:af:1b:07:fa	1	60 bytes	1	60 bytes	0	0 bytes	213.828319	0.0000		
00:19:29:68:8b:fb	00:20:af:1b:07:fa	1	60 bytes	1	60 bytes	0	0 bytes	171.445527	0.0000		
00:19:af:1b:57:fa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	407.708135	0.0000		
00:1e:af:1b:07:fa	00:e0:29:68:8b:fb	1	99 bytes	1	99 bytes	0	0 bytes	228.415937	0.0000		
00:20:16:1b:07:fa	00:e0:29:68:8b:fb	1	96 bytes	1	96 bytes	0	0 bytes	70.932390	0.0000		
00:20:18:1b:07:fa	00:e0:29:68:8b:fb	1	1.051 KiB	1	1.051 KiB	0	0 bytes	193.099696	0.0000		
00:20:1b:1b:07:fa	00:e0:29:68:8b:fb	1	591 bytes	1	591 bytes	0	0 bytes	436.524190	0.0000		
00:20:21:1b:07:fa	00:e0:29:68:8b:fb	1	331 bytes	1	331 bytes	0	0 bytes	421.115753	0.0000		
00:20:25:73:00:fa	00:64:29:68:8b:fb	1	77 bytes	1	77 bytes	0	0 bytes	127.396690	0.0000		
00:20:25:73:aa:aa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	314.985829	0.0000		
00:20:27:1b:07:fa	00:e0:29:68:8b:fb	1	170 bytes	1	170 bytes	0	0 bytes	149.914196	0.0000		
00:20:2f:1b:07:fa	00:e0:29:68:8b:fb	1	336 bytes	1	336 bytes	0	0 bytes	185.290505	0.0000		
00:20:30:13:78:fa	00:e0:29:68:8b:fb	1	585 bytes	1	585 bytes	0	0 bytes	461.591410	0.0000		
00:20:39:1b:07:fa	00:e0:29:68:8b:fb	2	145 bytes	1	85 bytes	1	60 bytes	25.617151	309.8505	2 bits/s	1 bits/s
00:20:41:1b:07:fa	00:e0:29:68:8b:fb	1	1.053 KiB	1	1.053 KiB	0	0 bytes	397.048335	0.0000		
00:20:46:1b:07:fa	00:e0:29:68:8b:fb	1	75 bytes	1	75 bytes	0	0 bytes	463.361313	0.0000		
00:20:46:1b:1d:fa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	172.013777	0.0000		
00:20:49:1b:07:fa	00:e0:29:68:8b:fb	1	71 bytes	1	71 bytes	0	0 bytes	259.762659	0.0000		
00:20:4b:1b:07:fa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	14.517185	0.0000		
00:20:4d:1b:07:fa	00:e0:29:68:8b:fb	1	190 bytes	1	190 bytes	0	0 bytes	308.837766	0.0000		
00:20:4f:1b:07:fa	00:e0:29:68:8b:fb	1	80 bytes	1	80 bytes	0	0 bytes	449.692806	0.0000		



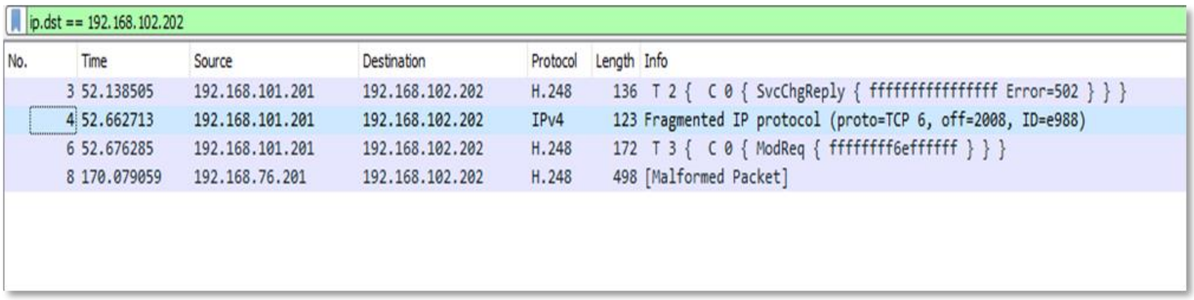
Ethernet · 872 | IPv4 · 493 | IPv6 · 2 | TCP | UDP · 673

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:00:af:1b:07:fa	79:e0:29:68:8b:fb	1	83 bytes	1	83 bytes	0	0 bytes	290.219477	0.0000		
00:02:29:68:8b:fb	00:20:af:1b:07:fa	1	60 bytes	1	60 bytes	0	0 bytes	213.828319	0.0000		
00:19:29:68:8b:fb	00:20:af:1b:07:fa	1	60 bytes	1	60 bytes	0	0 bytes	171.445527	0.0000		
00:19:af:1b:57:fa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	407.708135	0.0000		
00:1e:af:1b:07:fa	00:e0:29:68:8b:fb	1	99 bytes	1	99 bytes	0	0 bytes	228.415937	0.0000		
00:20:16:1b:07:fa	00:e0:29:68:8b:fb	1	96 bytes	1	96 bytes	0	0 bytes	70.932390	0.0000		
00:20:18:1b:07:fa	00:e0:29:68:8b:fb	1	1.051 KiB	1	1.051 KiB	0	0 bytes	193.099696	0.0000		
00:20:1b:1b:07:fa	00:e0:29:68:8b:fb	1	591 bytes	1	591 bytes	0	0 bytes	436.524190	0.0000		
00:20:21:1b:07:fa	00:e0:29:68:8b:fb	1	331 bytes	1	331 bytes	0	0 bytes	421.115753	0.0000		
00:20:25:73:00:fa	00:64:29:68:8b:fb	1	77 bytes	1	77 bytes	0	0 bytes	127.396690	0.0000		
00:20:25:73:aa:aa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	314.985829	0.0000		
00:20:27:1b:07:fa	00:e0:29:68:8b:fb	1	170 bytes	1	170 bytes	0	0 bytes	149.914196	0.0000		
00:20:2f:1b:07:fa	00:e0:29:68:8b:fb	1	336 bytes	1	336 bytes	0	0 bytes	185.290505	0.0000		
00:20:30:13:78:fa	00:e0:29:68:8b:fb	1	585 bytes	1	585 bytes	0	0 bytes	461.591410	0.0000		
00:20:39:1b:07:fa	00:e0:29:68:8b:fb	2	145 bytes	1	85 bytes	1	60 bytes	25.617151	309.8505	2 bits/s	1 bits/s
00:20:41:1b:07:fa	00:e0:29:68:8b:fb	1	1.053 KiB	1	1.053 KiB	0	0 bytes	397.048335	0.0000		
00:20:46:1b:07:fa	00:e0:29:68:8b:fb	1	75 bytes	1	75 bytes	0	0 bytes	463.361313	0.0000		
00:20:46:1b:1d:fa	00:e0:29:68:8b:fb	1	1.479 KiB	1	1.479 KiB	0	0 bytes	172.013777	0.0000		
00:20:49:1b:07:fa	00:e0:29:68:8b:fb	1	71 bytes	1	71 bytes	0	0 bytes	259.762659	0.0000		

Figure 8. SYN flooding packets Transferred.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.102.202	192.168.73.201	H.248	154	T 49 { C 0 { NotifyReq { ffffffffffffffff } } }
2	52.135846	192.168.102.202	192.168.97.201	H.248	203	
3	52.138505	192.168.101.201	192.168.102.202	H.248	136	T 2 { C 0 { SvcChgReply { ffffffffffffffff Error=502 } } }
4	52.662713	192.168.101.201	192.168.102.202	IPv4	123	Fragmented IP protocol (proto=TCP 6, off=2008, ID=e988)
5	52.675320	117.168.102.202	192.168.101.201	TPKT	165	Continuation
6	52.676285	192.168.101.201	192.168.102.202	H.248	172	T 3 { C 0 { ModReq { ffffffff6effffff } } }
7	61.155236	192.169.102.202	192.168.101.201	H.248	178	[Malformed Packet]
8	170.079059	192.168.76.201	192.168.102.202	H.248	498	[Malformed Packet]
9	170.123483	192.168.101.201	192.168.37.115	H.248	505	T e { C 2 { ModReq { 000000000000793f } } }
10	183.454513	192.168.102.202	192.168.101.201	H.248	162	[TCP ACKed unseen segment] [Malformed Packet]



ip.dst == 192.168.102.202

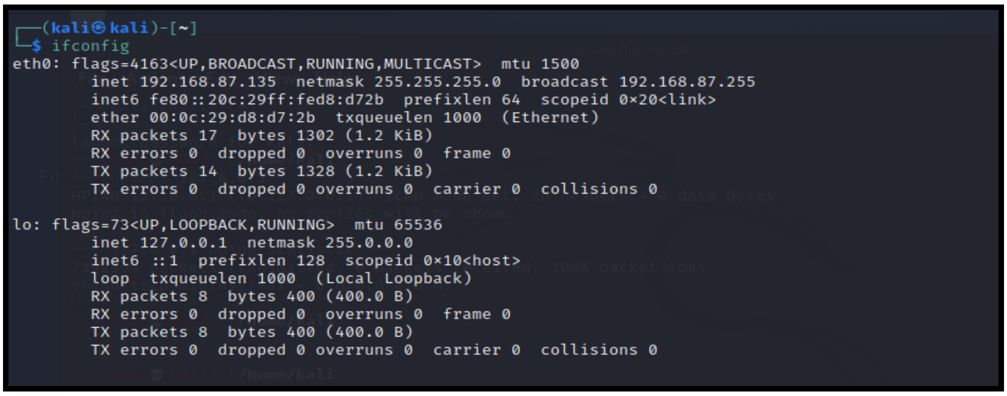
No.	Time	Source	Destination	Protocol	Length	Info
3	52.138505	192.168.101.201	192.168.102.202	H.248	136	T 2 { C 0 { SvcChgReply { ffffffffffffffff Error=502 } } }
4	52.662713	192.168.101.201	192.168.102.202	IPv4	123	Fragmented IP protocol (proto=TCP 6, off=2008, ID=e988)
6	52.676285	192.168.101.201	192.168.102.202	H.248	172	T 3 { C 0 { ModReq { ffffffff6effffff } } }
8	170.079059	192.168.76.201	192.168.102.202	H.248	498	[Malformed Packet]

Figure 9. Samples of malicious packets detected.

ii. Kali Linux

Version: Kali-Linux-2022.4-vmware-amd64.

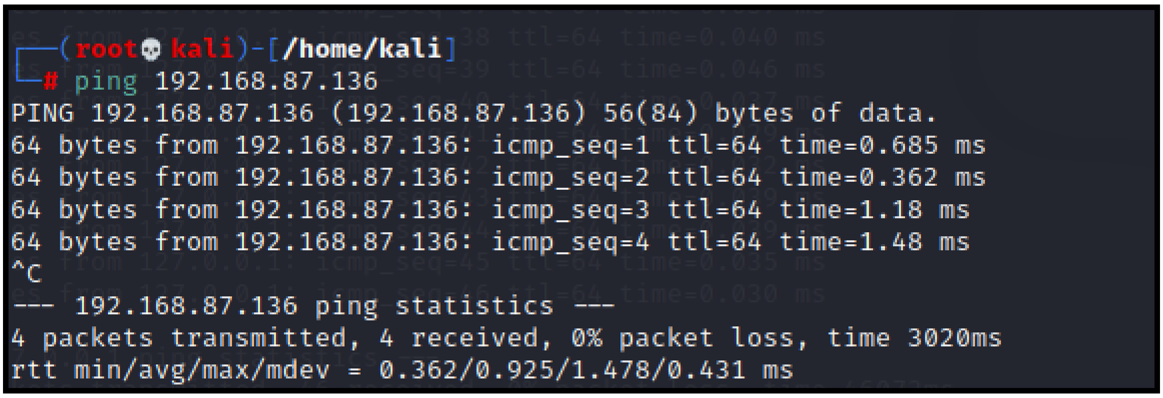
Kali is a popular OS distribution from Linux, which allows us to conduct Pentesting, ethical hacking, digital forensics and much more. Kali Linux is employed as the attacking virtual machine in this project and it is used to flood the targeted VM to multiple TCP requests in a limited time period. To launch the attack and to generate a huge volume of data traffic or requests to the target VM, the 'Hping3' command is being used. The **'hping3 -S -p <port> --flood <target_ip>**' command sends a flood of TCP SYN packets to the target IP and port, simulating a Denial of Service (DoS) attack. The screen snips of the configurations done in Kali Linux are as shown below.



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.87.135  netmask 255.255.255.0  broadcast 192.168.87.255
        inet6 fe80::20c:29ff:fed8:d72b  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:d8:d7:2b  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 1302 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1328 (1.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 400 (400.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 400 (400.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 10. Kali VM IP configurations



```
┌──(root💀kali)-[/home/kali]
└─# ping 192.168.87.136
PING 192.168.87.136 (192.168.87.136) 56(84) bytes of data.
64 bytes from 192.168.87.136: icmp_seq=1 ttl=64 time=0.685 ms
64 bytes from 192.168.87.136: icmp_seq=2 ttl=64 time=0.362 ms
64 bytes from 192.168.87.136: icmp_seq=3 ttl=64 time=1.18 ms
64 bytes from 192.168.87.136: icmp_seq=4 ttl=64 time=1.48 ms
^C
--- 192.168.87.136 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3020ms
rtt min/avg/max/mdev = 0.362/0.925/1.478/0.431 ms
```

Figure 11 Kali VM ping to Victim (Ubuntu VM)

iii. Ubuntu

Version: kali-linux-2023.1-VirtualBox-amd64

Ubuntu is also a popular Linux distribution that is being used in most servers, because of its user-friendly GUI and security offered by the Linux platform. Ubuntu is picked because of most the servers running in the cloud are using Ubuntu OS and using the same OS in the simulation will make the lab setup more connected to the real-world infrastructure. Thereby evaluation of the proposed methodology can be more impactful.

The configurations made in the Ubuntu VM are depicted by the following screen snips.

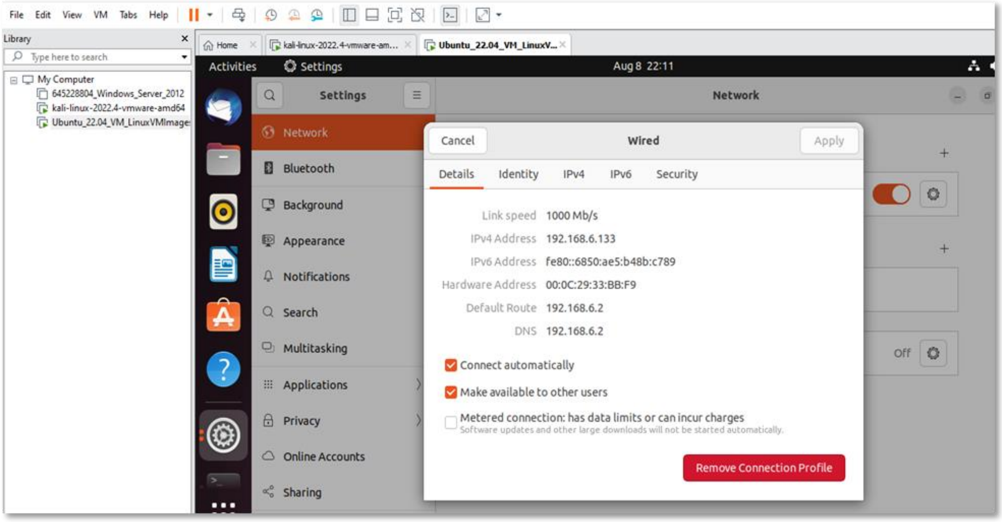


Figure 12. Ubuntu network configuration

```
ubuntu@ubuntu2004:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.87.136  netmask 255.255.255.0  broadcast 192.168.87.255
        inet6 fe80::b3b2:f279:334f:f148  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ed:3a:9c  txqueuelen 1000  (Ethernet)
        RX packets 69  bytes 5906 (5.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189  bytes 14835 (14.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 652  bytes 58697 (58.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 652  bytes 58697 (58.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 13. Ubuntu VM IP configuration

iv. VMware

Version: VMware-workstation-17.0.0-20800274

VMware is the virtualisation software tool used for hosting the attacking Kali VM and the victim Ubuntu VM. It allows running multiple VMs on the same local system seamlessly and offers more customisable settings to be done at any of the installed VM packages. The above-stated are the main reasons for selecting VMware as the virtualisation tool. The screen snip of the same is as shown below.

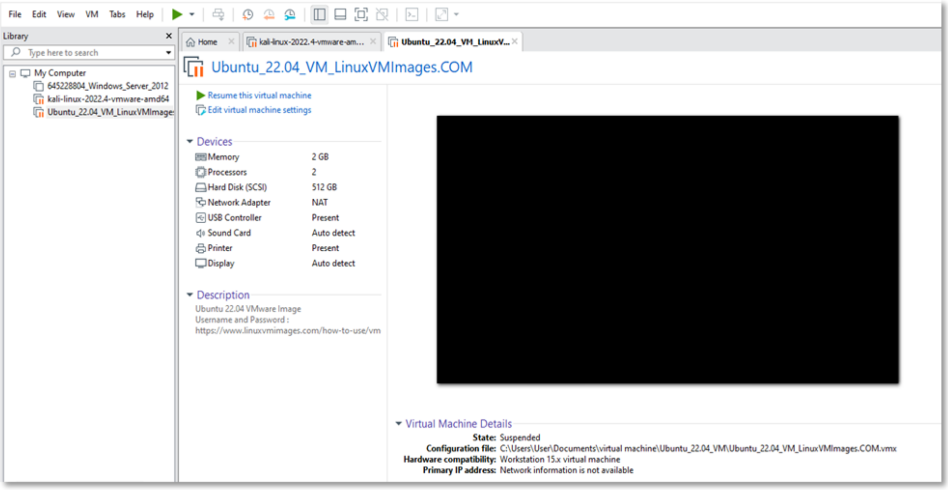


Figure 14. VMware home screen with both Kali and Ubuntu VMs installed.

4 Dataset & System Load

The dataset used in the project is the primary data that has been obtained as a result of the implementation. The primary data used in the project is the IP packets found as the attacking packet and those were not the attacking packets out of the whole packets sent. The system utilisation during the event of a DDoS attack can be found by analysing the resource utilisation graph of the local system. The system will be working under stress conditions when a DDoS or DoS attack hits and due to the enormous system utilisation, the whole system gets collapsed. The image of the same is attached below.

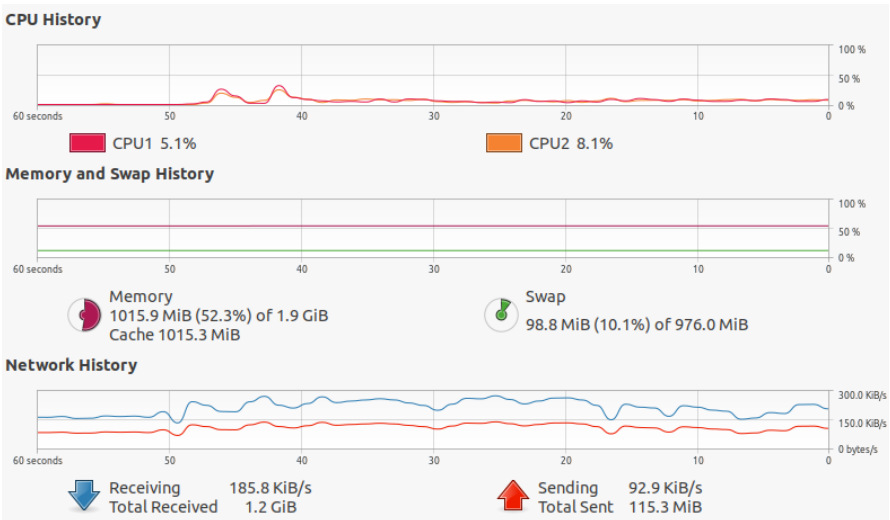


Figure 15. System Utilisation

Packet Number	Source IP	Destination IP	Protocol	Length	Description
1	192.168.87.135	192.168.87.136	TCP	120	SYN packet (Attack)
2	192.168.87.135	192.168.87.136	TCP	120	SYN packet (Attack)
...	...	...	...	...	...
1123	192.168.87.135	192.168.87.136	TCP	1500	Normal Data Packet
1124	192.168.87.135	192.168.87.136	TCP	1500	Normal Data Packet
...	...	...	...	...	...
15000	192.168.87.135	192.168.87.136	TCP	120	SYN packet (Attack)

Table 1 Packets identified to be successful in DDoS attack

5 Mitigation.

The proposed mitigation technique using the iptables are shown below. Below are the detailed screenshots of the configurations made in the Ubuntu VM for making the entries in the iptable.

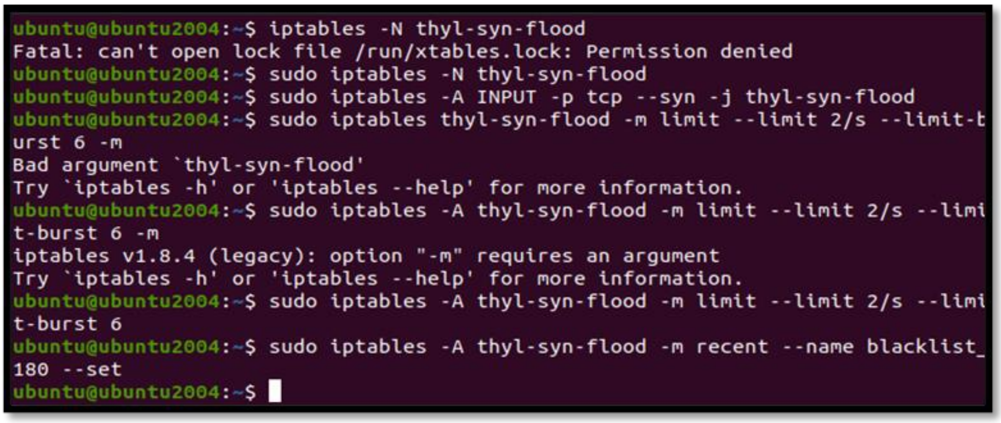


Figure 16 Mitigating DDoS attack using iptables.

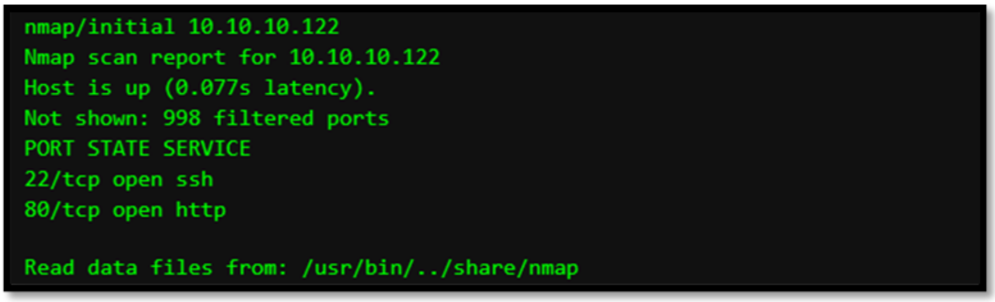
```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu2004:/home/ubuntu/Downloads# iptables -A INPUT -s 127.0.0.1   -p 80 -j DROP
root@ubuntu2004:/home/ubuntu/Downloads# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

^[[A^[[BDROP      80   --  127.0.0.1           anywhere
DROP       80   --  127.0.0.1            anywhere
DROP       80   --  127.0.0.1            anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

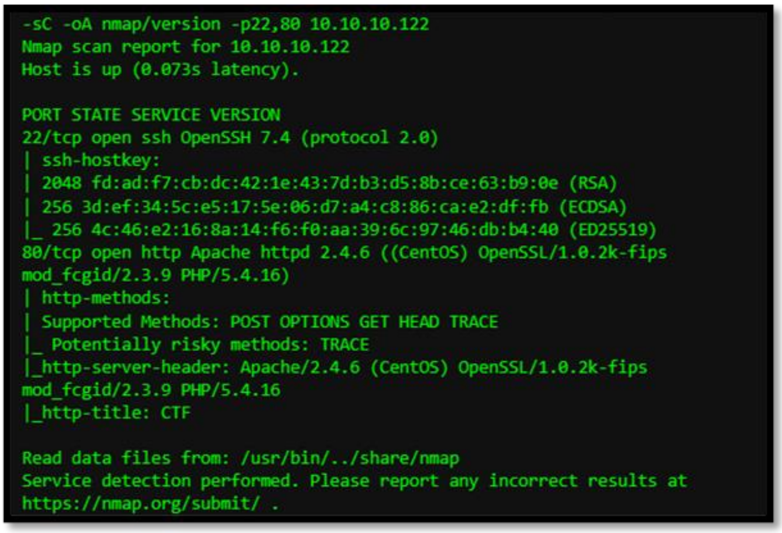
Figure 17 Configuring the iptable



```
nmap/initial 10.10.10.122
Nmap scan report for 10.10.10.122
Host is up (0.077s latency).
Not shown: 998 filtered ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http

Read data files from: /usr/bin/../share/nmap
```

Figure 18 Initializing Nmap for RSA Token isolation.



```
-sC -oA nmap/version -p22,80 10.10.10.122
Nmap scan report for 10.10.10.122
Host is up (0.073s latency).

PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
| 2048 fd:ad:f7:cb:dc:42:1e:43:7d:b3:d5:8b:ce:63:b9:0e (RSA)
| 256 3d:ef:34:5c:e5:17:5e:06:d7:a4:c8:86:ca:e2:df:fb (ECDSA)
|_ 256 4c:46:e2:16:8a:14:f6:f0:aa:39:6c:97:46:db:b4:40 (ED25519)
80/tcp open http Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips
mod_fcgid/2.3.9 PHP/5.4.16)
| http-methods:
| Supported Methods: POST OPTIONS GET HEAD TRACE
|_ Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
mod_fcgid/2.3.9 PHP/5.4.16
|_http-title: CTF

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```

Figure 19 Involving the Nsa script for Rsa Token isolation.

The above figure represents the overall RSA token isolation process. By applying the NSA script, the isolation of the RSA Token is completed.

6 Evaluation

The project evaluation can be done by the formulation of a confusion matrix and the same can be executed in the Jupiter notebook to get the precision, accuracy, misclassification rate and prevalence. Jupiter notebook has been used for the calculation of the same and below are the code snippets and the related outputs.

Below shown is the confusion matrix tabulated for evaluating the accuracy and precision of the DDoS attack mitigation solution. The table consists of the data like true negative, false positive, false negative and true positive. In this case of the project, the vectors are the IP packets detected as the cause of the DDoS attack and those were not. Fifteen thousand IP packets were used in the simulation and the confusion matrix was made out of the observed data. The details of the attacking packets are mentioned in the main report in a tabular format.

Below is the confusion matrix of the dataset.

n = 15000	Packets failed to Attack	Packets succeeded in Attack	
Packets failed to Attack	TN =2202	FP =1798	**4000**
Packets succeeded in Attack	FN =478	TP =10522	**11000**
	2680	**12320**	

Table 1 Confusion Matrix

The flowing are the codes executed in the Jupiter notebook, for generating the confusion matrix so as to evaluate the accuracy and precision of the proposed solution. The code snippets along with the obtained output are attached as follows.

```
In [12]: import numpy as np
         import matplotlib.pyplot as plt
         import itertools

         # Assigning the data
         TN = 2202
         FP = 1798
         FN = 478
         TP = 10522
```

Figure 20 Importing Libraries

Numpy is the mathematical library used in Python (Jupiter Notebook) for scientific calculations. ***Matplotlib.pyplot*** is the Python library that is used for making the different types of plots that are required in the evaluation of the project. Here this library is used for generating the confusion matrix and to derive the conclusion from it.

```
In [33]: # Creating confusion matrix
cm = np.array([[TN, FP], [FN, TP]])

# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes, title='Confusion Matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# defining class labels
class_labels = ['Packets Failed to Attack', 'Packets Succeeded in Attack']

# Plotting confusion matrix
plt.figure(figsize=(8, 6))
plot_confusion_matrix(cm, classes=class_labels, title='DDoS Attack Confusion Matrix')
plt.show()
```

Figure 21 Code for generating the confusion matrix.

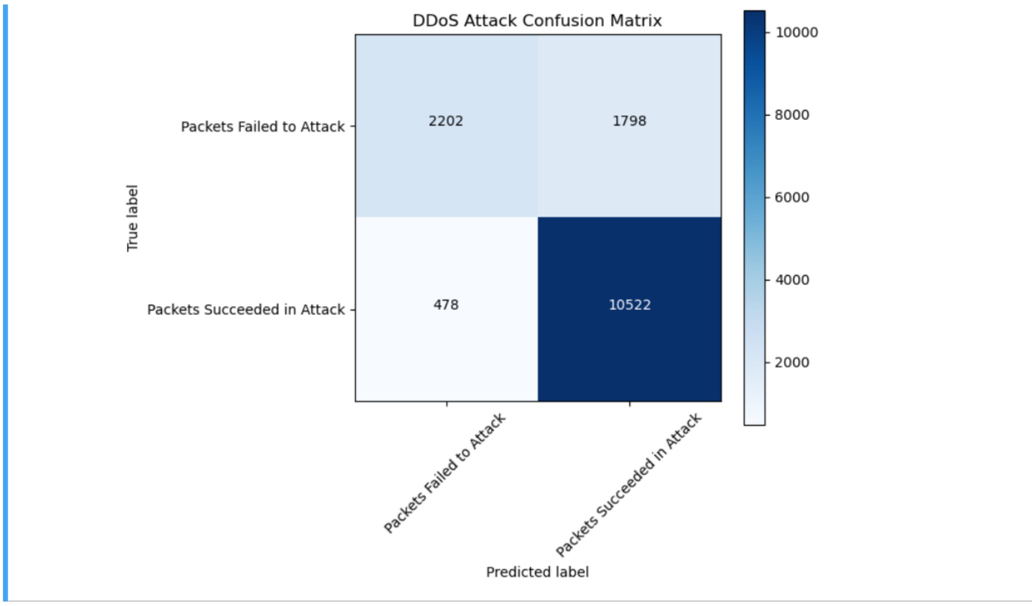


Figure 22 Confusion matrix

```
In [14]: # Calculating Accuracy
         accuracy = (TP + TN) / (TP + TN + FP + FN)
         print("Accuracy:", accuracy)

         Accuracy: 0.8482666666666666

In [15]: # Calculating Misclassification Rate
         misclassification_rate = (FP + FN) / (TP + TN + FP + FN)
         print("Misclassification Rate:", misclassification_rate)

         Misclassification Rate: 0.15173333333333333

In [34]: # Calculating True Positive Rate (Sensitivity/Recall)
         true_positive_rate = TP / (TP + FN)
         print("True Positive Rate:", true_positive_rate)

         True Positive Rate: 0.9565454545454546
```

Figure 23 Codes for generating the Accuracy, Misclassification Rate and True positive Rate

```
In [35]: # Calculating False Positive Rate
         false_positive_rate = FP / (FP + TN)
         print("False Positive Rate:", false_positive_rate)

         False Positive Rate: 0.4495

In [36]: # Calculating True Negative Rate (Specificity)
         true_negative_rate = TN / (TN + FP)
         print("True Negative Rate:", true_negative_rate)

         True Negative Rate: 0.5505

In [20]: # Calculating Precision
         precision = TP / (TP + FP)
         print("Precision:", precision)

         Precision: 0.8540584415584416
```

Figure 24 Codes for generating the False positive Rate, True Negative Rate and Precision

```
In [21]: # Calculating Prevalence
         prevalence = (TP + FN) / (TP + TN + FP + FN)
         print("Prevalence:", prevalence)

         Prevalence: 0.7333333333333333
```

Figure 25 Codes for generating the Prevalence.