

Configuration Manual

MSc Research Project
MSc in Cybersecurity

Nagraj Merala
Student ID: x20180985

School of Computing
National College of Ireland

Supervisor: Rohit Verma

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name:NAGRAJ MERALA.....

Student ID:x20180985.....

Programme:MSc. Cybersecurity..... **Year:** ...2021-22....

Module:MSc Research Project.....

Supervisor:Rohit Verma.....

Submission Due Date:19th September 2022.....

Project Title: Defending IoT against escalating cyber threats like botnet attacks, data privacy issues and inadequate patch management capabilities

Word Count:1532..... **Page Count:**.....26.....

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:NAGRAJ MERALA.....

Date:19th September 2022.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

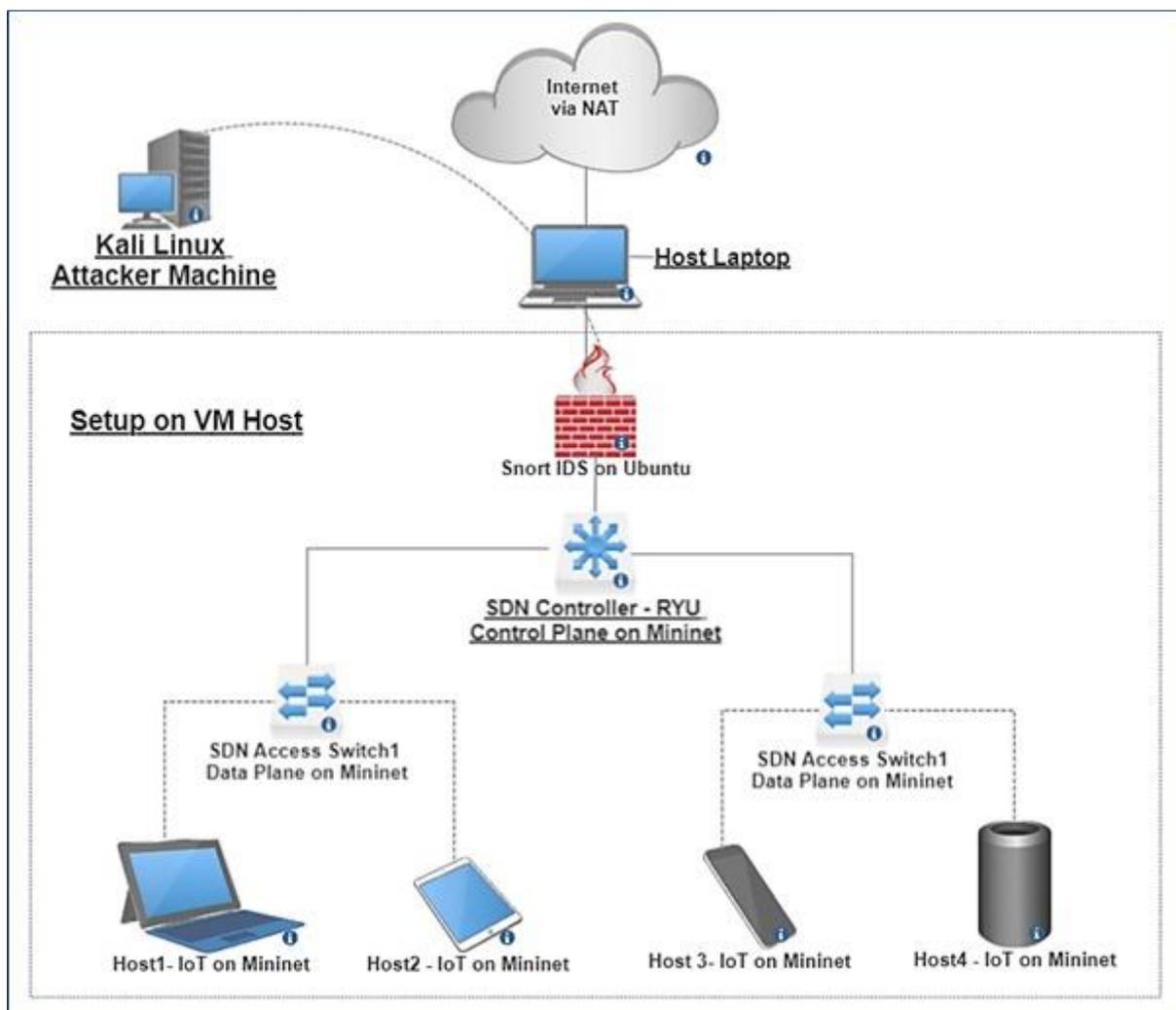
Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

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

Configuration Manual

Nagraj Merala
Student ID: x20180985

Network architecture diagram:



1 Project Dependencies

1. Oracle Virtual Box
2. Ubuntu
3. Kali Linux
4. Mininet-SDN
5. RYU SDN Controller
6. Snort-IDS
7. Blockchain implementation

1.1 System Configuration

1. Host Machine:
 - Operating System: Windows 11 64-bit
 - Processor: AMD Ryzen 9 5900HX
 - RAM: 16GB
 - NVIDIA: 4GB – GeForce RTX 3050
 - Storage: 1TB SSD

2. Virtual Machines:

Machine 1:

- OS: Kali Linux
- Processor Allocated: 2
- Storage: 80GB
- RAM: 4GB

Machine 2:

- OS: Ubuntu LTS 22.04
- Processor Allocated: 2
- Storage: 20GB
- RAM: 4GB

- 2.2 Tools:

Mininet:

RYU SDN Controller:

Snort-IDS:

Nmap:

Wireshark:

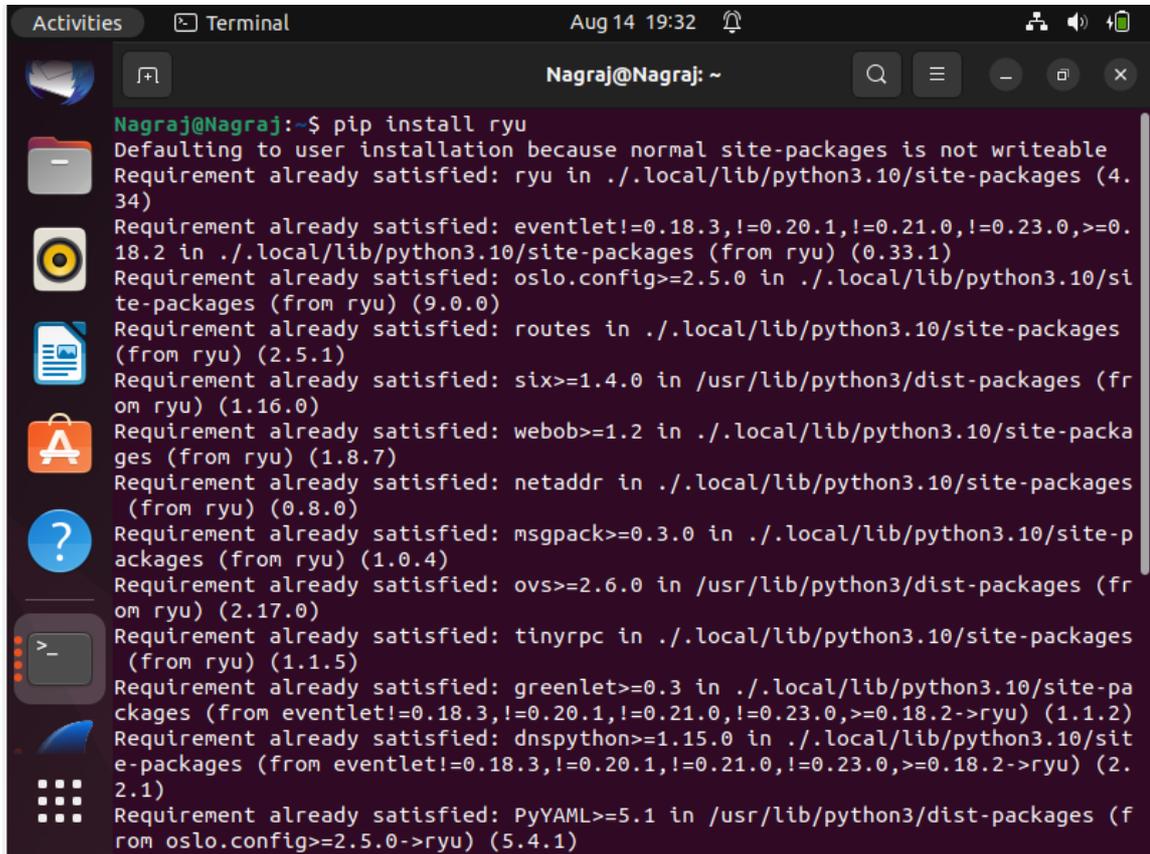
2 Implementation steps

1. Downloading and installation of Virtual box
2. Downloading and installation of Kali Linux
3. Downloading and installation of Ubuntu LTS 22.04
4. Downloading and installation of Wireshark
5. Downloading and installation of Mininet
7. Downloading and installation of RYU SDN Controller
8. Downloading and installation of Snort-IDS
9. Install python and all the required libraries.
10. Writing the customize rules in snort using text editor
11. Testing and evaluation of the setup, rules by performing the penetration testing.

3 Installation and configuration of RYU SDN Controller

- As a first step, we will download and install the RYU SDN Controller using the command:

pip install ryu

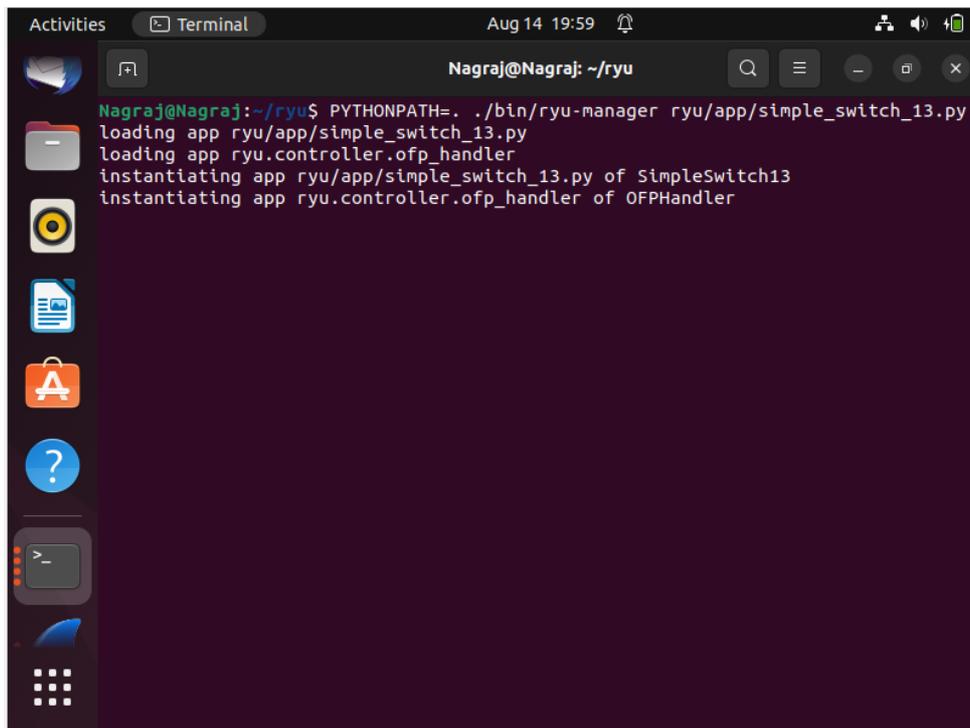


```
Nagraj@Nagraj:~$ pip install ryu
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: ryu in ./local/lib/python3.10/site-packages (4.34)
Requirement already satisfied: eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2 in ./local/lib/python3.10/site-packages (from ryu) (0.33.1)
Requirement already satisfied: oslo.config>=2.5.0 in ./local/lib/python3.10/site-packages (from ryu) (9.0.0)
Requirement already satisfied: routes in ./local/lib/python3.10/site-packages (from ryu) (2.5.1)
Requirement already satisfied: six>=1.4.0 in /usr/lib/python3/dist-packages (from ryu) (1.16.0)
Requirement already satisfied: webob>=1.2 in ./local/lib/python3.10/site-packages (from ryu) (1.8.7)
Requirement already satisfied: netaddr in ./local/lib/python3.10/site-packages (from ryu) (0.8.0)
Requirement already satisfied: msgpack>=0.3.0 in ./local/lib/python3.10/site-packages (from ryu) (1.0.4)
Requirement already satisfied: ovs>=2.6.0 in /usr/lib/python3/dist-packages (from ryu) (2.17.0)
Requirement already satisfied: tinyrpc in ./local/lib/python3.10/site-packages (from ryu) (1.1.5)
Requirement already satisfied: greenlet>=0.3 in ./local/lib/python3.10/site-packages (from eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2->ryu) (1.1.2)
Requirement already satisfied: dnspython>=1.15.0 in ./local/lib/python3.10/site-packages (from eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2->ryu) (2.2.1)
Requirement already satisfied: PyYAML>=5.1 in /usr/lib/python3/dist-packages (from oslo.config>=2.5.0->ryu) (5.4.1)
```

- Follow the below given link for other pre-requisites and dependencies required for installation:

https://ryu.readthedocs.io/en/latest/getting_started.html

- Start the Ryu Controller using command
cd ryu
PYTHONPATH=. ./bin/ryu-manager ryu/app/simpleswitch_13.py

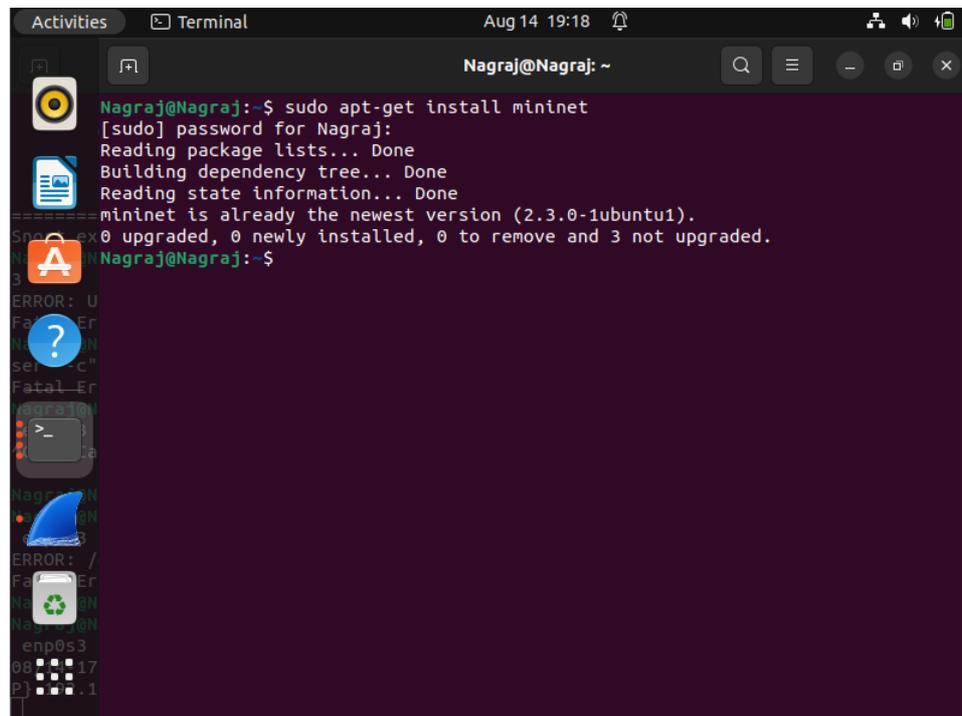


```
Aug 14 19:59
Nagraj@Nagraj: ~/ryu
Nagraj@Nagraj:~/ryu$ PYTHONPATH= ./bin/ryu-manager ryu/app/simple_switch_13.py
loading app ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

- Once the RYU SDN Controller is switched on, we will create network topology using Mininet

4 Starting with Mininet configuration:

- Downloading and installing the mininet SDN by using command:
sudo apt-get install mininet



```
Aug 14 19:18
Nagraj@Nagraj: ~
Nagraj@Nagraj:~$ sudo apt-get install mininet
[sudo] password for Nagraj:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mininet is already the newest version (2.3.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Nagraj@Nagraj:~$
```

- After installing the mininet we will create a SDN network topology inside mininet containing:

3-host, 1- openflow switch and 1- remote controller using RYU SDN Controller tool using the following command:

sudo mn --topo single,3 --mac --switch ovsk --controller remote

```

Nagraj@Nagraj:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
[sudo] password for Nagraj:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
c0
mininet>

```

5 Installation and configuration of Snort-IDS (Intrusion Detection System)

- Before installing the Snort-IDS you will need some pre-requisite software. For this follow the below given link:

<https://upcloud.com/resources/tutorials/install-snort-ubuntu>

- Download the Snort-IDS by using the following command:

sudo wget https://www.snort.org/downloads/snort/snort-2.9.16.tar.gz

```

Nagraj@Nagraj:~$ sudo wget https://www.snort.org/downloads/snort/snort-2.9.20.tar.gz
--2022-08-14 20:15:48-- https://www.snort.org/downloads/snort/snort-2.9.20.tar.gz
Resolving www.snort.org (www.snort.org)... 2606:4700::6812:8b09, 2606:4700::6812:8a09, 104.18.139.9, ...
Connecting to www.snort.org (www.snort.org)|2606:4700::6812:8b09|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/025/687/original/snort-2.9.20.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAU7AK5ITMJQBJPBJ%2F20220814%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220814T191549Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=8e01a2c076cdbadd57f623a1faab9233437f4e62f0f53982b93e7ed6ce0e50ef [following]
--2022-08-14 20:15:49-- https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/025/687/original/snort-2.9.20.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAU7AK5ITMJQBJPBJ%2F20220814%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220814T191549Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=8e01a2c076cdbadd57f623a1faab9233437f4e62f0f53982b93e7ed6ce0e50ef
Resolving snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)... 54.231.171.105
Connecting to snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)|54.231.171.105|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7009894 (6.7M) [binary/octet-stream]
Saving to: 'snort-2.9.20.tar.gz.1'

```

- Refer the below link for configuration of Snort to run in NIDS mode and configuring the network settings:

<https://upcloud.com/resources/tutorials/install-snort-ubuntu>

- Snort should be configured and placed inline to monitor the active traffic and internet-based traffic and to do that use command:

ip link set enp0s3 promisc on

- Once the installation is complete, we will create rules to provide relevant alert. For creating the rules, edit the file local.rules.

sudo nano /etc/snort/rules/local.rules

- We have created four rules.

```

GNU nano 6.2 /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp any any -> 192.168.0.221 any (msg:"NMAP ping sweep Scan";dsize:0;si
alert tcp any any -> $HOME_NET 22 (msg:"NMAP TCP Scan"; sid:10000005; rev:2;)
alert tcp any any -> 192.168.0.221 22 (msg:"NMAP XMAS Tree Scan";flags:FPU; si
alert tcp any any -> 192.168.0.221 22 (msg:"NMAP FIN Scan";flags:F; sid:100000

```

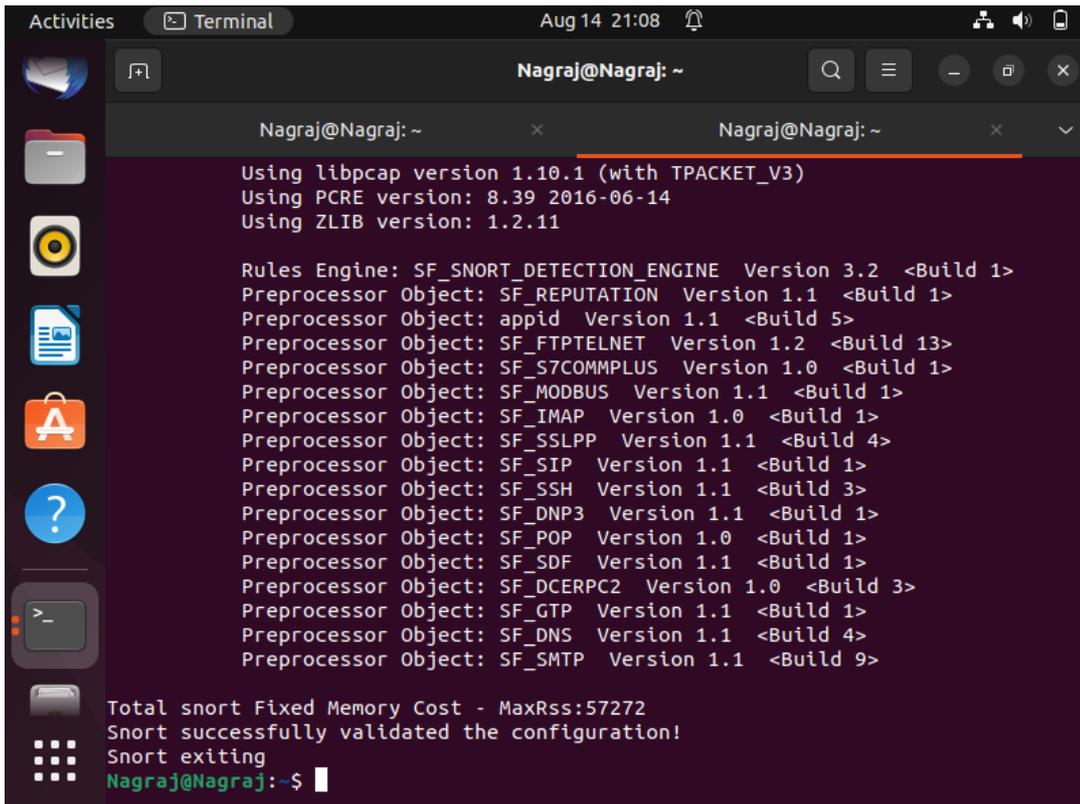
- Initializing the Snort-IDS in test mode to know if it has been installed correctly.
sudo snort -T -c /etc/snort/snort.conf

```

Nagraj@Nagraj:~$ sudo snort -T -c /etc/snort/snort.conf
Running in Test mode

---== Initializing Snort ===-
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414 1741 1830
 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 777
7 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 83
00 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50
002 55555 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined : [ 80:81 110 143 311 383 591 593 901 1220 1
414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:
7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181
8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:3
4444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined : [ 2123 2152 3386 ]
Detection:
  Search-Method = AC-Full-Q
  Split Anv/Anv group = enabled

```

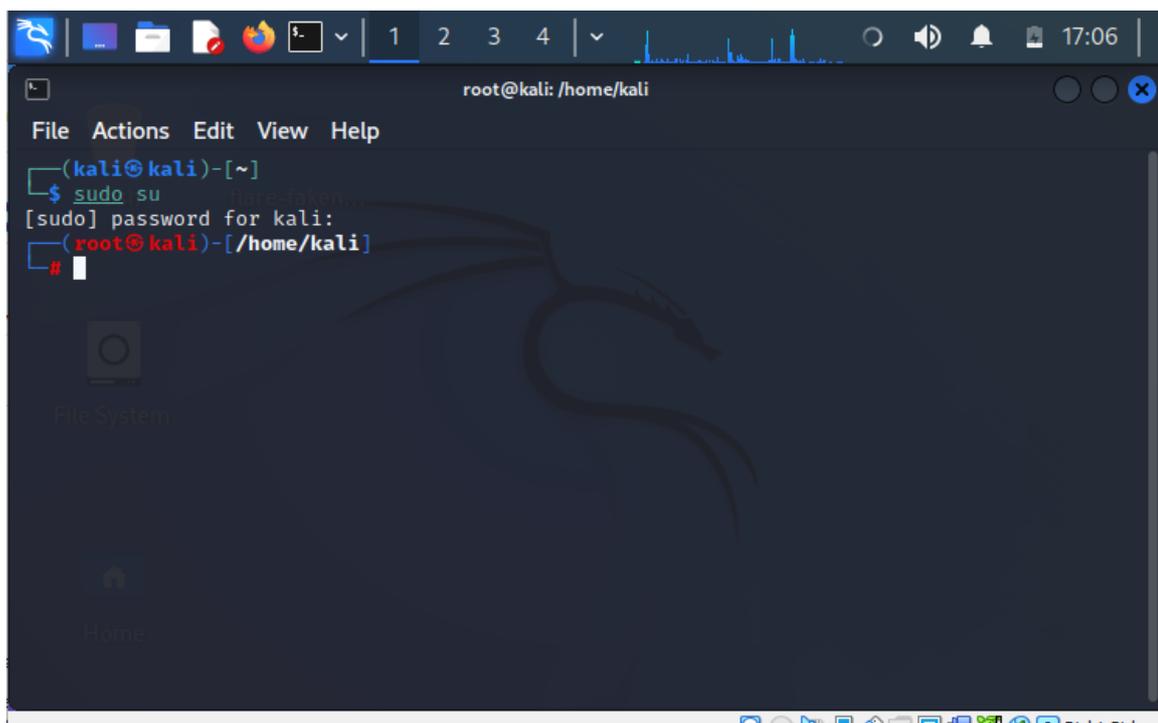


Activities Terminal Aug 14 21:08 Nagraj@Nagraj: ~

```
Nagraj@Nagraj: ~  
Using libpcap version 1.10.1 (with TPACKET_V3)  
Using PCRE version: 8.39 2016-06-14  
Using ZLIB version: 1.2.11  
  
Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.2 <Build 1>  
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>  
Preprocessor Object: appid Version 1.1 <Build 5>  
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>  
Preprocessor Object: SF_S7COMMPLUS 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_SSLPP Version 1.1 <Build 4>  
Preprocessor Object: SF_SIP Version 1.1 <Build 1>  
Preprocessor Object: SF_SSH Version 1.1 <Build 3>  
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>  
Preprocessor Object: SF_POP Version 1.0 <Build 1>  
Preprocessor Object: SF_SDF Version 1.1 <Build 1>  
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>  
Preprocessor Object: SF_GTP Version 1.1 <Build 1>  
Preprocessor Object: SF_DNS Version 1.1 <Build 4>  
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>  
  
Total snort Fixed Memory Cost - MaxRss:57272  
Snort successfully validated the configuration!  
Snort exiting  
Nagraj@Nagraj:~$
```

6 Evaluation

- Starting the Kali Linux virtual machine, it will be used as an external attacker to simulate penetration testing and for evaluation of security testing on configured network.



root@kali: /home/kali

```
File Actions Edit View Help  
root@kali:~  
└─(kali@kali)-[~]  
└─$ sudo su  
[sudo] password for kali:  
└─(root@kali)-[/home/kali]  
└─#
```

- Use command ifconfig to know the ip address of the machine.

```

root@kali: /home/kali
File Actions Edit View Help

(root@kali)-[/home/kali]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 08:00:27:88:f3:c6 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 11416 (11.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.130 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 2a02:8084:2147:ef00:d4fa:2028:85f9:7a38 prefixlen 64 scopeid 0<global>
    inet6 2a02:8084:2147:ef00:a00:27ff:fe6c:ee0f prefixlen 64 scopeid 0<global>
    inet6 fe80::a00:27ff:fe6c:ee0f prefixlen 64 scopeid 0<link>
    ether 08:00:27:6c:ee:0f txqueuelen 1000 (Ethernet)
    RX packets 11404 bytes 7920294 (7.5 MiB)
    RX errors 0 dropped 1 overruns 0 frame 0
    TX packets 6068 bytes 990455 (967.2 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 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 22 bytes 1100 (1.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 1100 (1.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

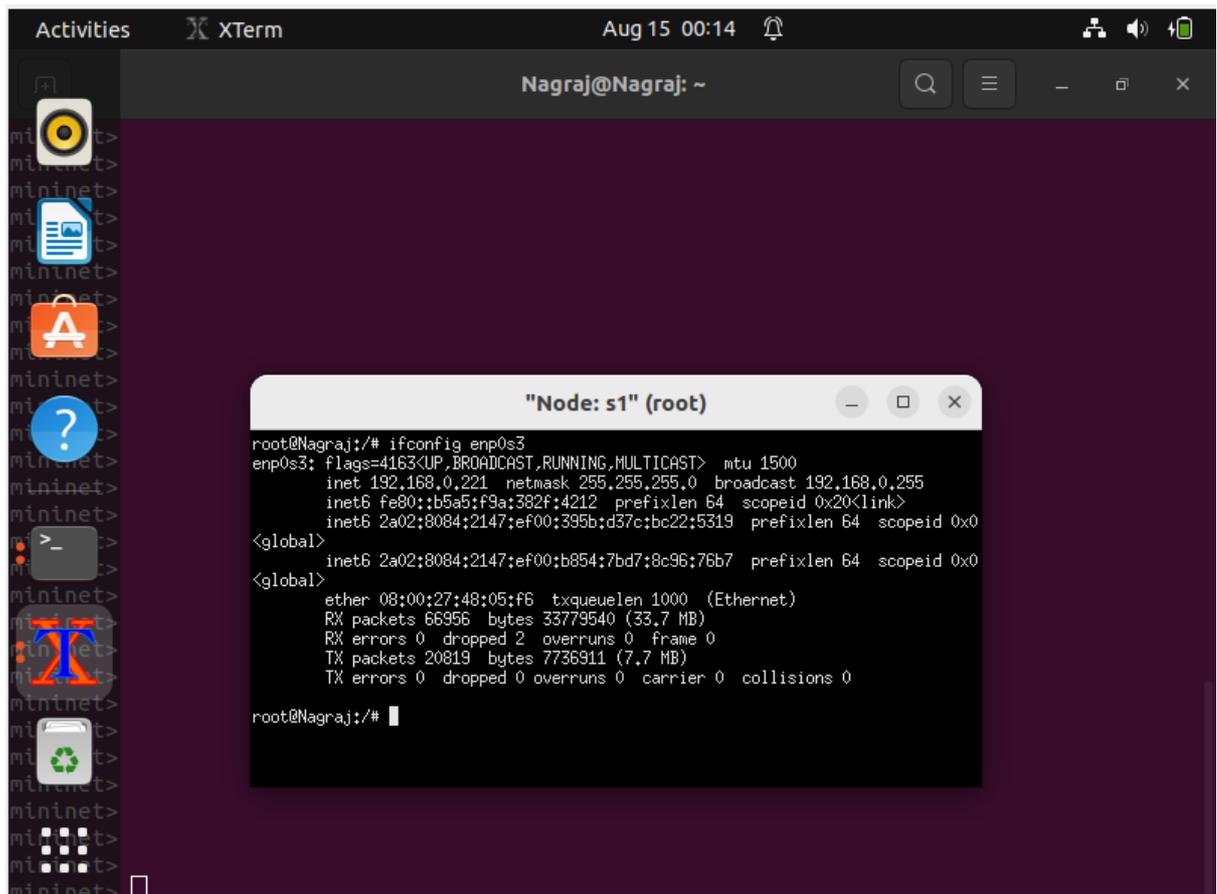
(root@kali)-[/home/kali]
#

```

- Using mininet SDN on Ubuntu machine we have set up two hosts, h1 and h2. To know the ip address of the node h1 first use following command which will open terminal of host 1.

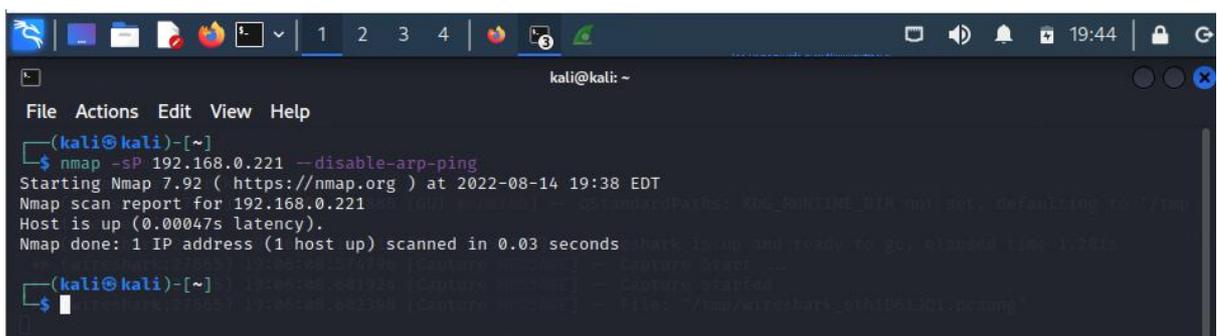
xterm h1

- Once the terminal is up use command ifconfig to know the ip address of it



- We have set rules on Snort-IDS to alert the administrator, we will test the rules using nmap on kali linux.
- We will use below command to send icmp packages to host machine to know whether the host is up or not

`nmap -sP 192.168.0.221 --disable-arp-ping`



- We have set the rule for icmp packet to alert the administrator when coming from any external network to our home network. Following is the rule for icmp alert.

```

GNU nano 6.2 /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp any any -> 192.168.0.221 any (msg:"NMAP ping sweep Scan";dsize:0;si
alert tcp any any -> $HOME_NET 22 (msg:"NMAP TCP Scan"; sid:10000005; rev:2;)
alert tcp any any -> 192.168.0.221 22 (msg:"NMAP XMAS Tree Scan";flags:FPU; si
alert tcp any any -> 192.168.0.221 22 (msg:"NMAP FIN Scan";flags:F; sid:100000

```

- Now using the below command initialize the Snort-IDS.

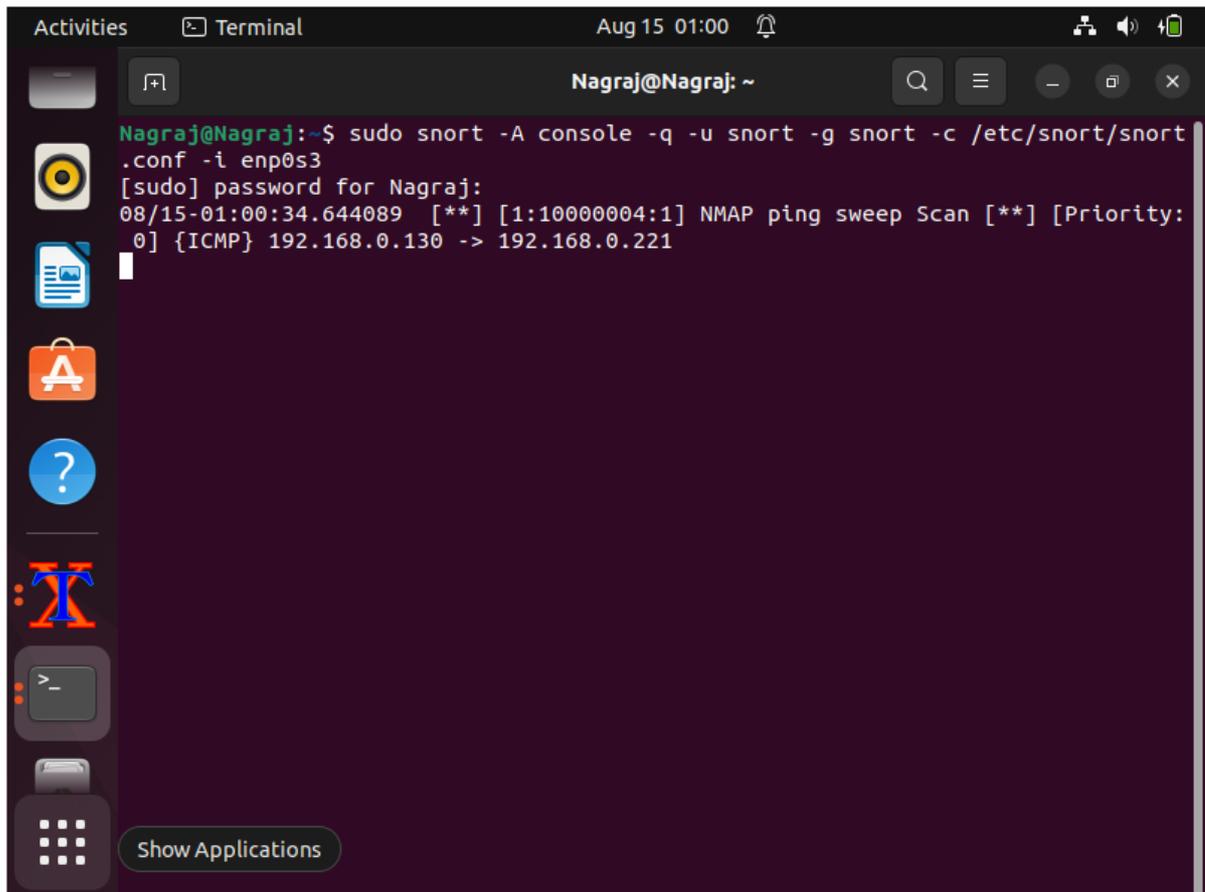
sudo snort -A console -q -u snort 0g snort -c /etc/snort/snort.conf -i enp0s3

```

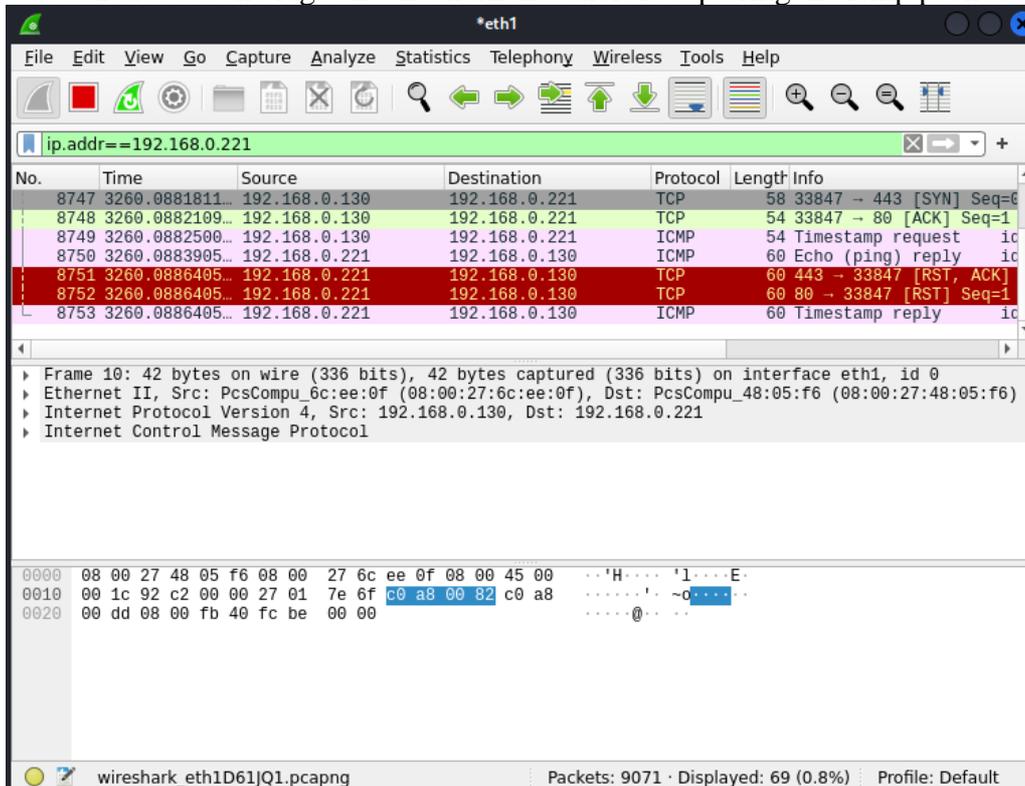
Nagraj@Nagraj: /etc/snort/rules
Nagraj@Nagraj: ~
Nagraj@Nagraj: /etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /
etc/snort/snort.conf -i enp0s3
[sudo] password for Nagraj:

```

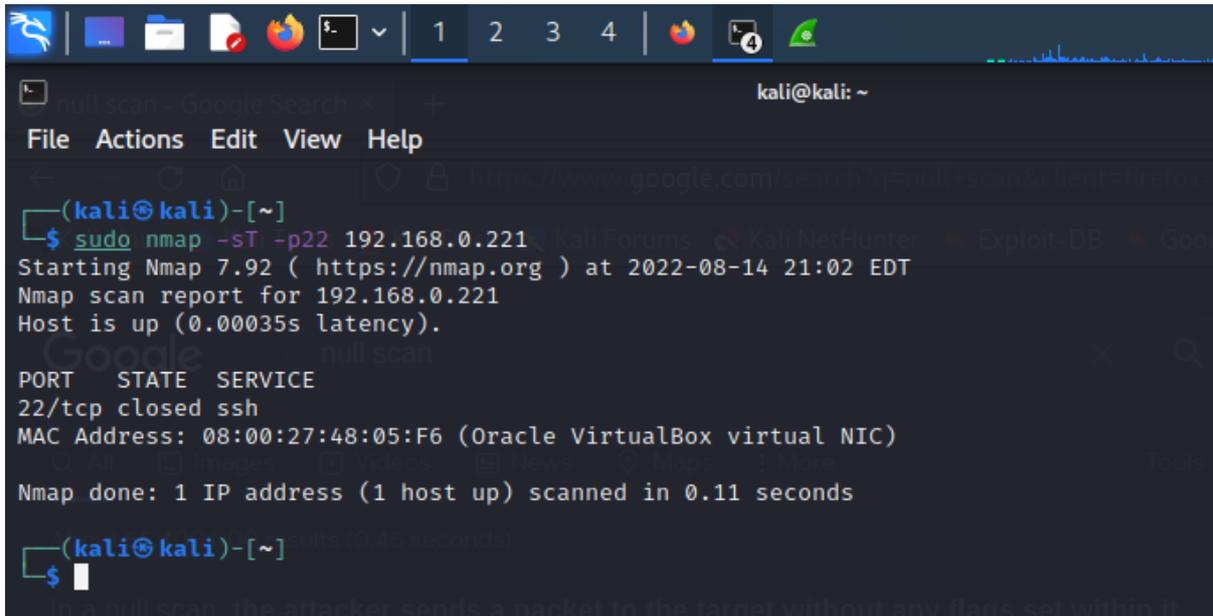
- Alert captured on Snort-IDS terminal



- We can also see the logs on Wireshark that snort is capturing the ICMP packets



- Following screenshots capture the alert for tcp packets alert coming from any external network using nmap tcp scan



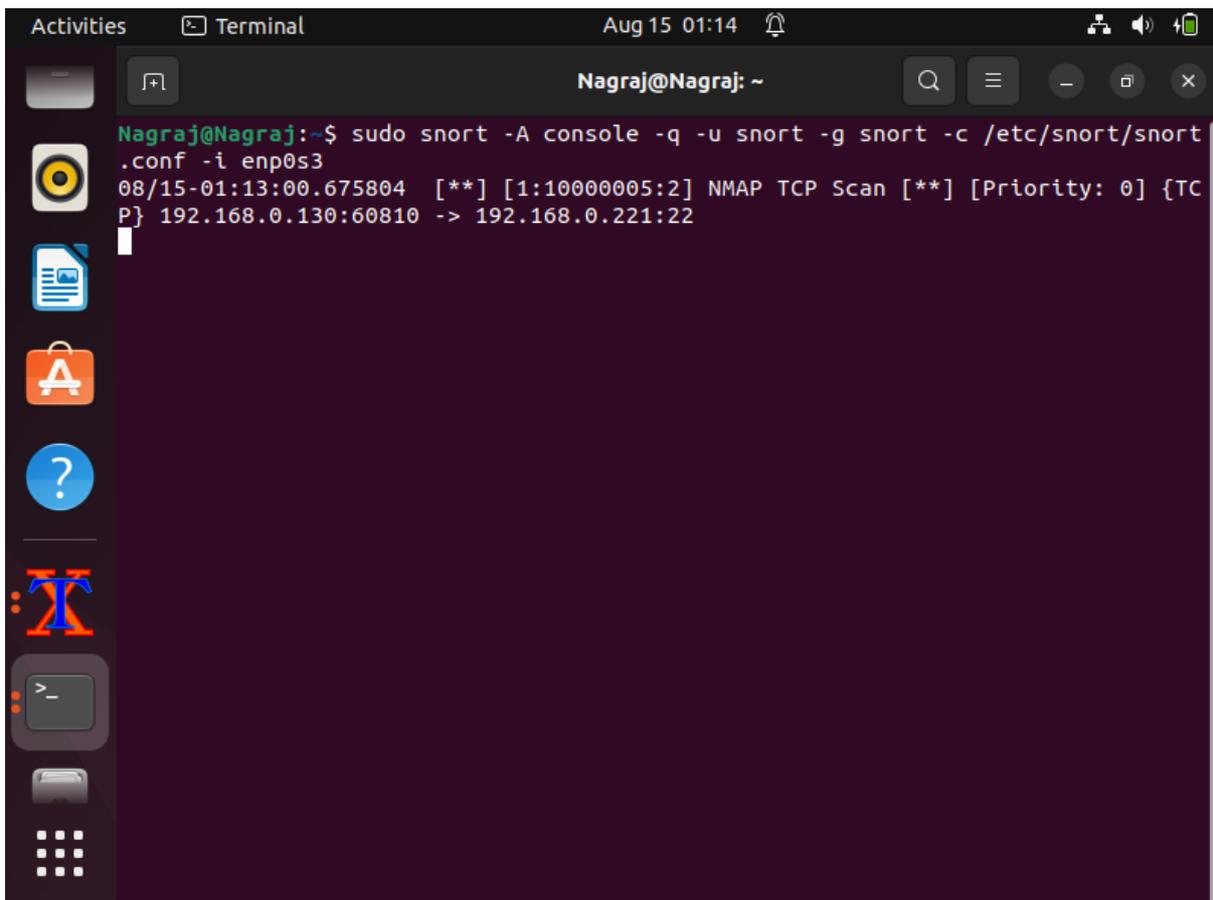
```
(kali@kali)-[~]
└─$ sudo nmap -sT -p22 192.168.0.221
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-14 21:02 EDT
Nmap scan report for 192.168.0.221
Host is up (0.00035s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:48:05:F6 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds

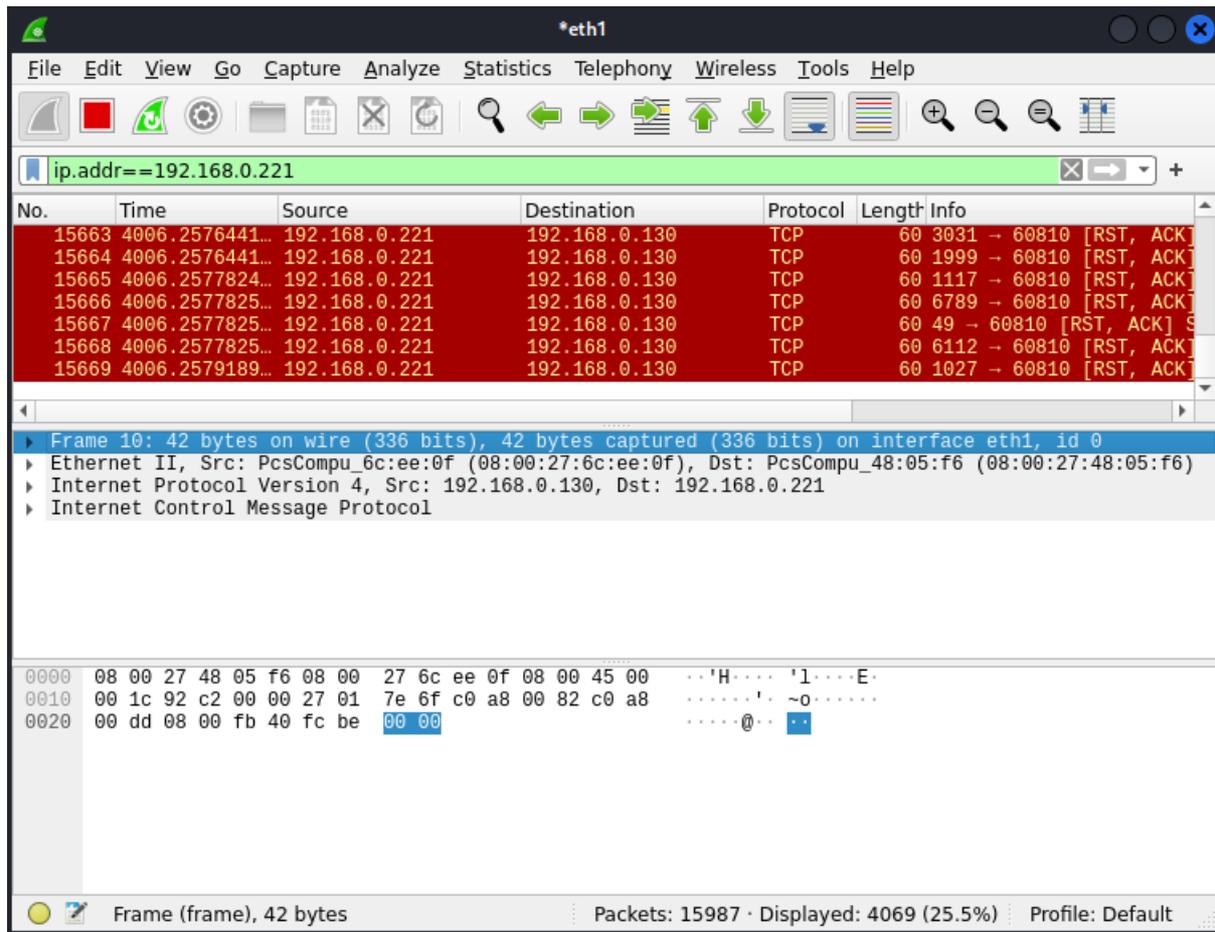
(kali@kali)-[~]
└─$
```

- We will see the TCP packet alert coming on Snort-IDS terminal in below screenshot:

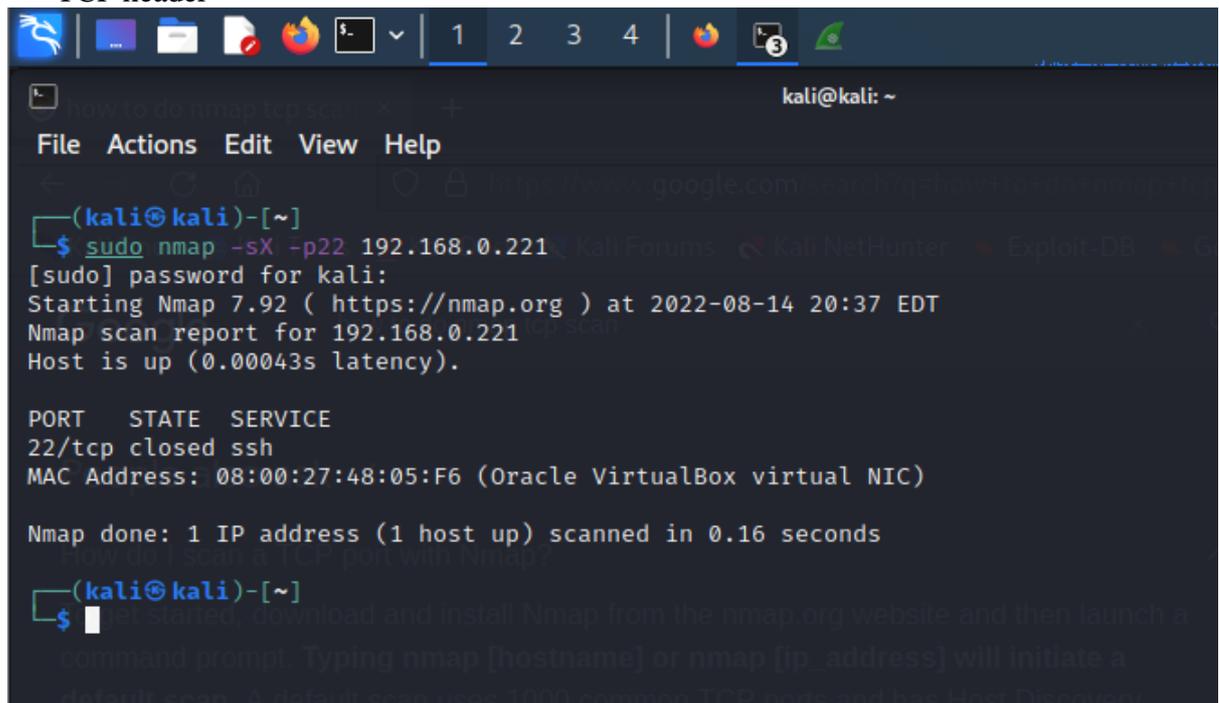


```
Aug 15 01:14
Nagraj@Nagraj: ~
└─$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
08/15-01:13:00.675804 [**] [1:10000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP} 192.168.0.130:60810 -> 192.168.0.221:22
```

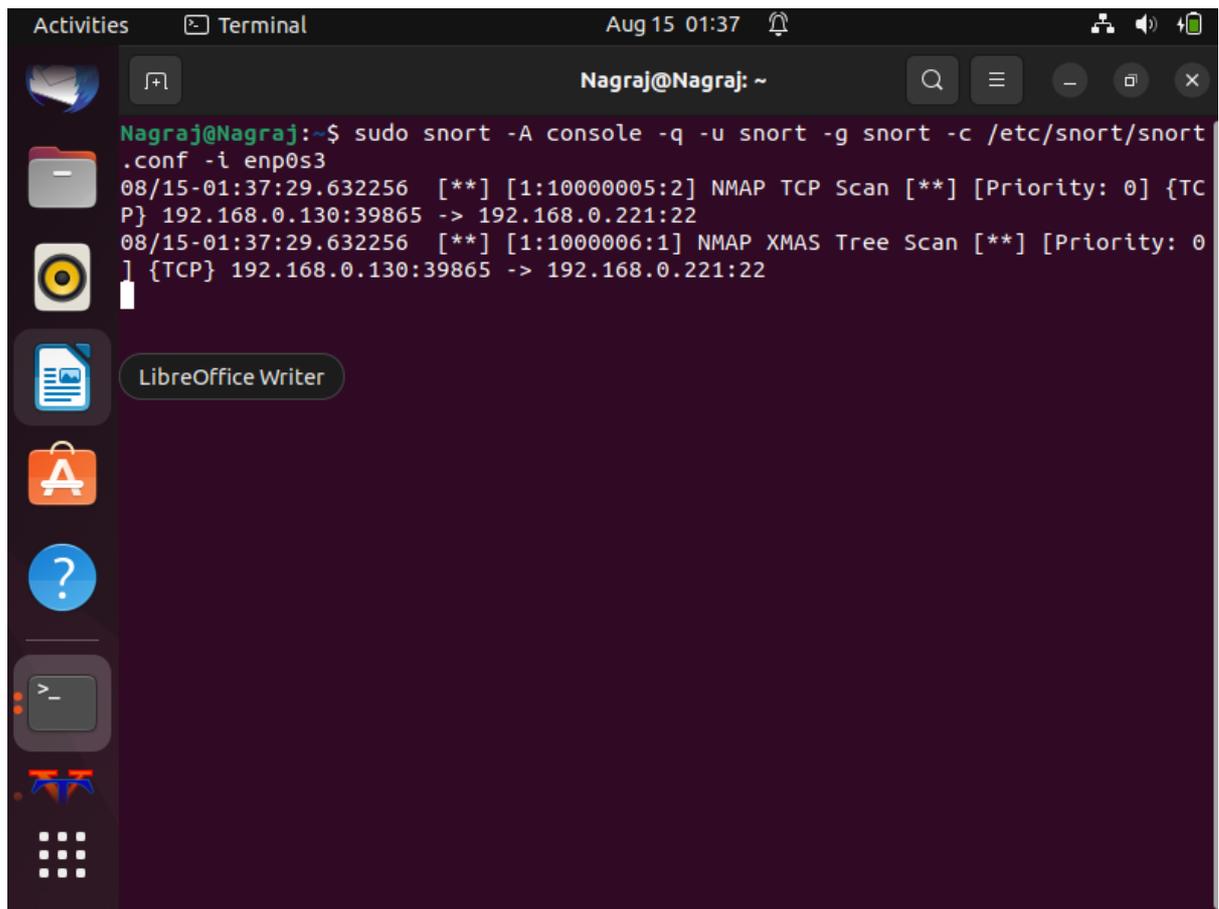
- Using Wireshark for validating the TCP packets.



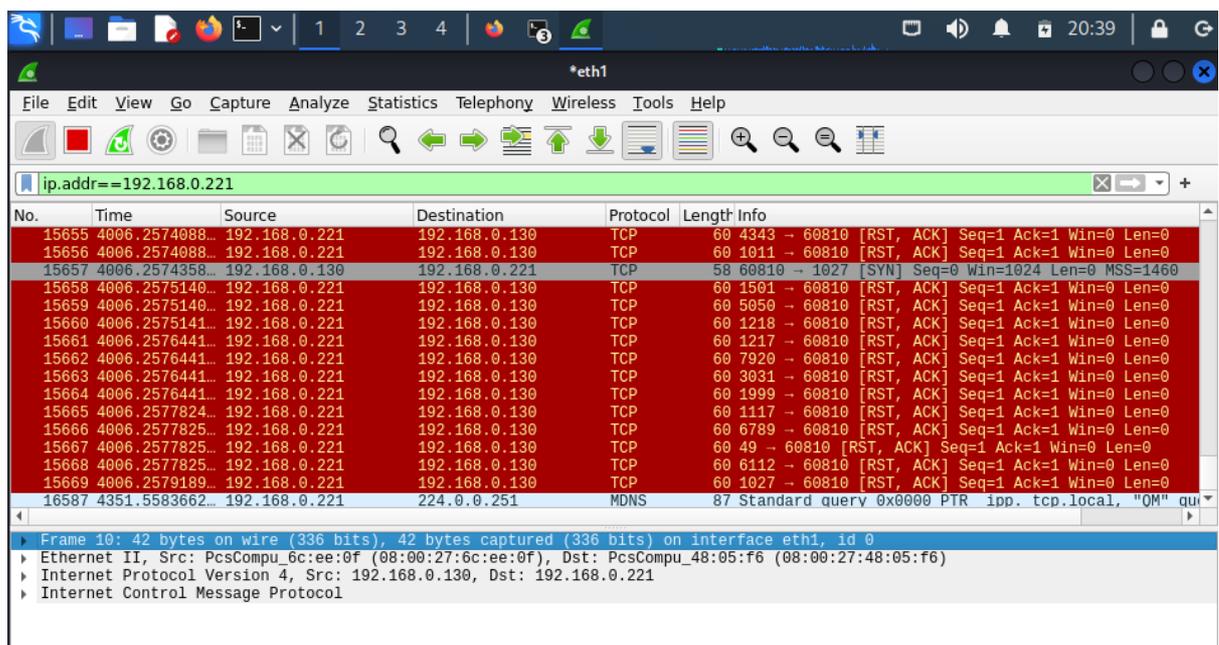
- Below screenshot captures alert for XMAS Scan in which the attacker manipulates the TCP header



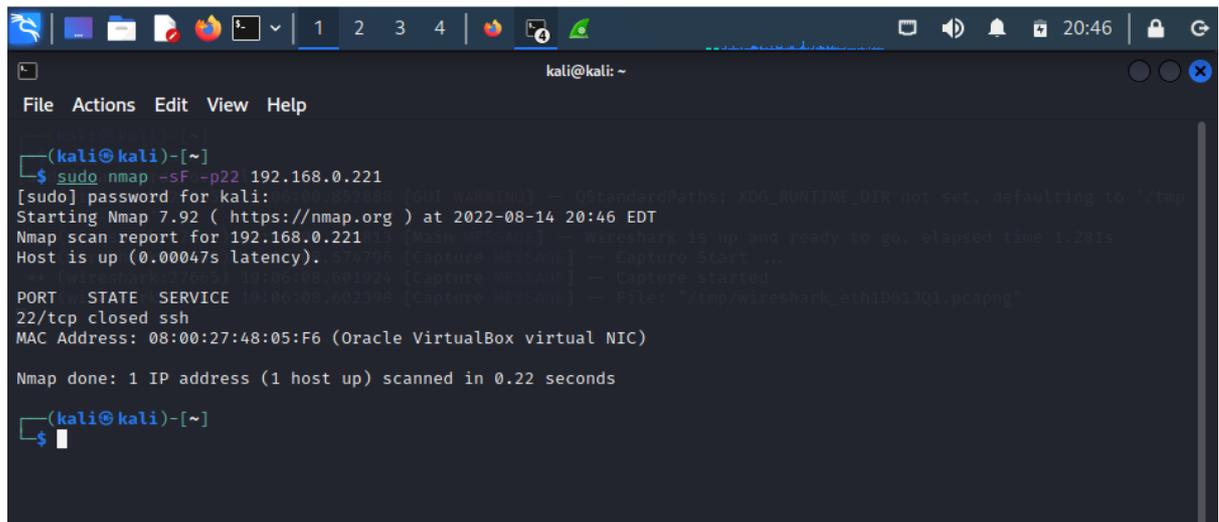
- Refer below screenshot for Snort-IDS terminal capturing the XMAS Scan:



- Analyzing the wireshark output:

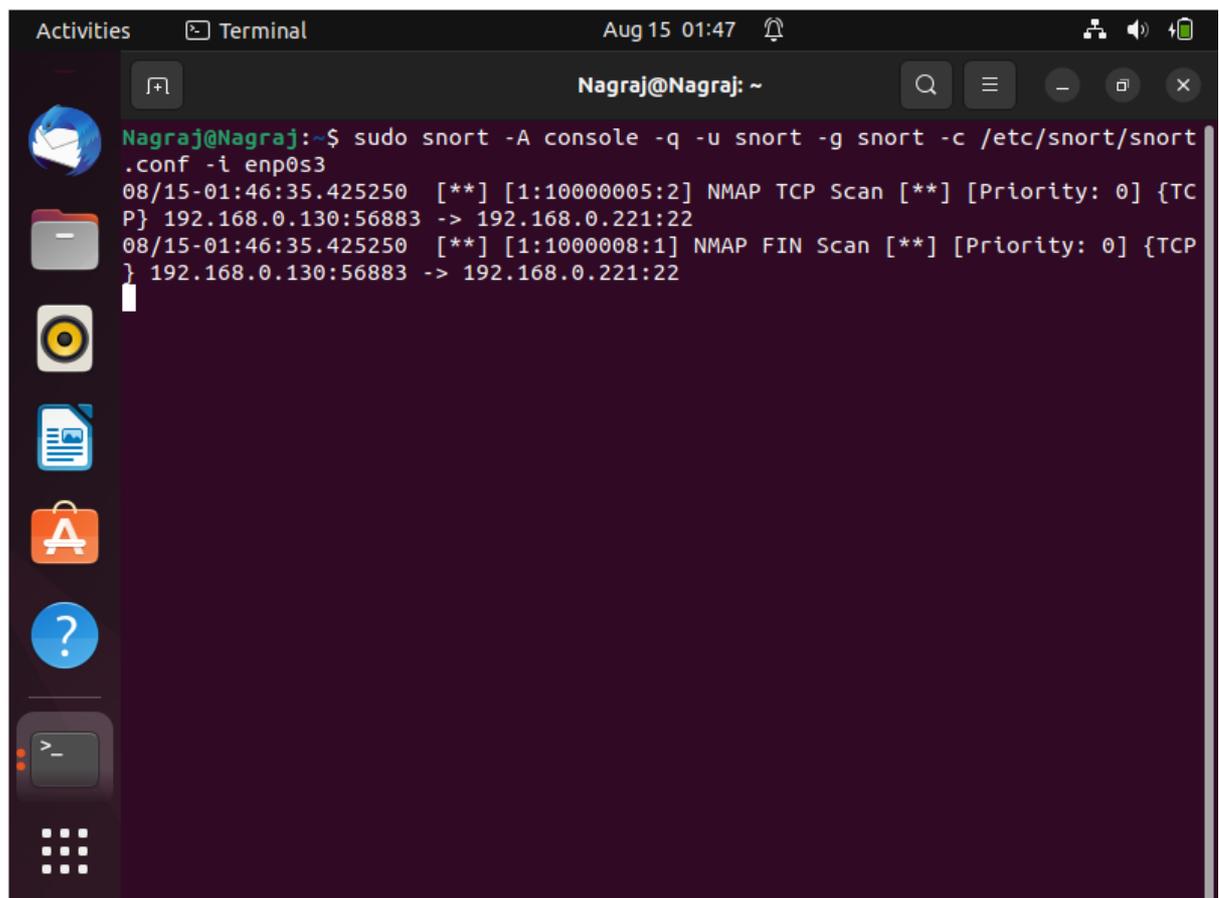


- Getting alert for FIN Scan which is used to terminate TCP connection after completion of the data transfer.



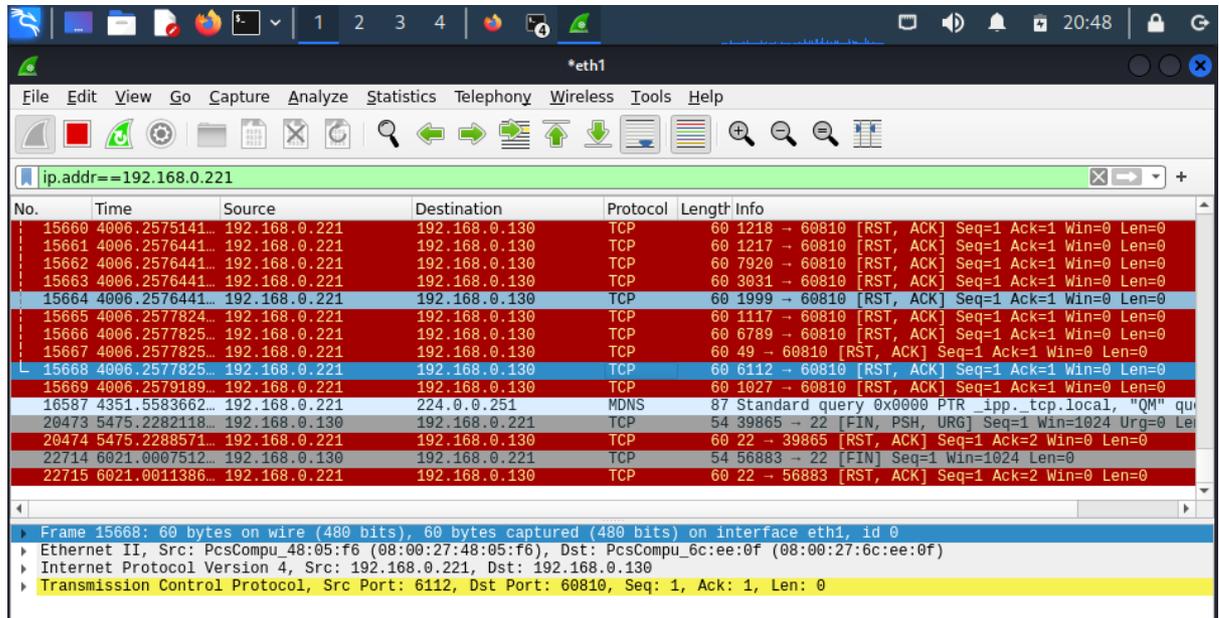
```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ sudo nmap -sF -p22 192.168.0.221  
[sudo] password for kali:  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-14 20:46 EDT  
Nmap scan report for 192.168.0.221  
Host is up (0.00047s latency).  
  
PORT      STATE SERVICE  
22/tcp    closed ssh  
MAC Address: 08:00:27:48:05:F6 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds  
  
(kali@kali)-[~]  
└─$
```

- Getting the alerts for FIN Scan

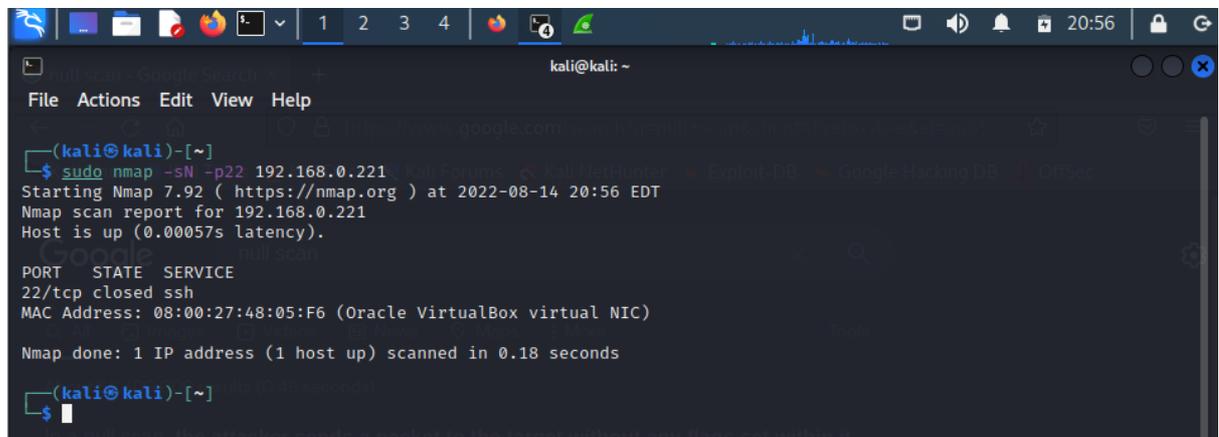


```
Activities Terminal Aug 15 01:47  
Nagraj@Nagraj: ~  
Nagraj@Nagraj:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3  
08/15-01:46:35.425250  [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP  
P} 192.168.0.130:56883 -> 192.168.0.221:22  
08/15-01:46:35.425250  [**] [1:1000008:1] NMAP FIN Scan [**] [Priority: 0] {TCP  
} 192.168.0.130:56883 -> 192.168.0.221:22
```

- Validating using Wireshark



- Capturing NULL Scan in which the packets forwarded by the attacker are without flags



- Getting the alert message on Snort-IDS terminal

```

Nagraj@Nagraj:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
08/15-01:56:17.663552  [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] [TC
P] 192.168.0.130:35854 -> 192.168.0.221:22
08/15-01:56:17.663552  [**] [1:1000009:1] NMAP NULL Scan [**] [Priority: 0] [TC
P] 192.168.0.130:35854 -> 192.168.0.221:22
  
```

- The line highlighted in the blue is a FIN packet

No.	Time	Source	Destination	Protocol	Length	Info
15662	4006.2576441...	192.168.0.221	192.168.0.130	TCP	60	7920 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15663	4006.2576441...	192.168.0.221	192.168.0.130	TCP	60	3031 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15664	4006.2576441...	192.168.0.221	192.168.0.130	TCP	60	1999 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15665	4006.2577824...	192.168.0.221	192.168.0.130	TCP	60	1117 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15666	4006.2577825...	192.168.0.221	192.168.0.130	TCP	60	6789 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15667	4006.2577825...	192.168.0.221	192.168.0.130	TCP	60	49 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15668	4006.2577825...	192.168.0.221	192.168.0.130	TCP	60	6112 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15669	4006.2579189...	192.168.0.221	192.168.0.130	TCP	60	1027 → 60810 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16587	4351.5583662...	192.168.0.221	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipp._tcp.local, "QM" qu
20473	5475.2282118...	192.168.0.130	192.168.0.221	TCP	54	39865 → 22 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
20474	5475.2288571...	192.168.0.221	192.168.0.130	TCP	60	22 → 39865 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
22714	6021.0007512...	192.168.0.130	192.168.0.221	TCP	54	56883 → 22 [FIN] Seq=1 Win=1024 Len=0
22715	6021.0011386...	192.168.0.221	192.168.0.130	TCP	60	22 → 56883 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
25192	6603.2116510...	192.168.0.130	192.168.0.221	TCP	54	35854 → 22 [<None>] Seq=1 Win=1024 Len=0
25193	6603.2120336...	192.168.0.221	192.168.0.130	TCP	60	22 → 35854 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 25192: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth1, id 0
 Ethernet II, Src: PcsCompu_6c:ee:0f (08:00:27:6c:ee:0f), Dst: PcsCompu_48:05:f6 (08:00:27:48:05:f6)
 Internet Protocol Version 4, Src: 192.168.0.130, Dst: 192.168.0.221
 Transmission Control Protocol, Src Port: 35854, Dst Port: 22, Seq: 1, Len: 0

- Simulating DoS attack and replication a simple botnet attack: Getting alert for DOS Ping-Flood attack. For this, first using following command on attacker machine to send packets to victim machine:

sudo hping3 -1 -fast 192.168.0,221

```
kali@kali: ~  
File Actions Edit View Help  
google.com  
(kali@kali)-[~]  
└─$ sudo hping3 -i 1 --fast 192.168.0.221  
HPING 192.168.0.221 (eth1 192.168.0.221): icmp mode set, 28 headers + 0 data bytes  
len=46 ip=192.168.0.221 ttl=64 id=13366 icmp_seq=0 rtt=4.7 ms  
len=46 ip=192.168.0.221 ttl=64 id=13379 icmp_seq=1 rtt=11.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=13380 icmp_seq=2 rtt=3.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=13387 icmp_seq=3 rtt=6.1 ms  
len=46 ip=192.168.0.221 ttl=64 id=13411 icmp_seq=4 rtt=5.1 ms  
len=46 ip=192.168.0.221 ttl=64 id=13417 icmp_seq=5 rtt=4.4 ms  
len=46 ip=192.168.0.221 ttl=64 id=13419 icmp_seq=6 rtt=7.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13439 icmp_seq=7 rtt=10.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13448 icmp_seq=8 rtt=5.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13462 icmp_seq=9 rtt=8.1 ms  
len=46 ip=192.168.0.221 ttl=64 id=13487 icmp_seq=10 rtt=5.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13493 icmp_seq=11 rtt=8.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13509 icmp_seq=12 rtt=4.7 ms  
len=46 ip=192.168.0.221 ttl=64 id=13510 icmp_seq=13 rtt=7.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13520 icmp_seq=14 rtt=10.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=13524 icmp_seq=15 rtt=9.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=13543 icmp_seq=16 rtt=1.2 ms
```

```
kali@kali: ~  
File Actions Edit View Help  
1 2 3 4  
len=46 ip=192.168.0.221 ttl=64 id=19062 icmp_seq=423 rtt=6.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=19085 icmp_seq=424 rtt=4.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=19098 icmp_seq=425 rtt=6.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=19116 icmp_seq=426 rtt=10.0 ms  
len=46 ip=192.168.0.221 ttl=64 id=19142 icmp_seq=427 rtt=7.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=19159 icmp_seq=428 rtt=10.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=19178 icmp_seq=429 rtt=6.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=19180 icmp_seq=430 rtt=10.0 ms  
len=46 ip=192.168.0.221 ttl=64 id=19203 icmp_seq=431 rtt=2.2 ms  
len=46 ip=192.168.0.221 ttl=64 id=19218 icmp_seq=432 rtt=5.8 ms  
len=46 ip=192.168.0.221 ttl=64 id=19239 icmp_seq=433 rtt=7.7 ms  
len=46 ip=192.168.0.221 ttl=64 id=19263 icmp_seq=434 rtt=3.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=19272 icmp_seq=435 rtt=3.9 ms  
len=46 ip=192.168.0.221 ttl=64 id=19295 icmp_seq=436 rtt=3.9 ms  
^C  
— 192.168.0.221 hping statistic —  
438 packets transmitted, 437 packets received, 1% packet loss  
round-trip min/avg/max = 0.4/6.3/12.9 ms  
(kali@kali)-[~]  
└─$
```

- Refer below screenshot for Snort_IDS capturing the packets and giving alert on its terminal

```

Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Commencing packet processing (pid=36080)
08/15-02:39:49.098958  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:49.098958  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:49.098958  [**] [1:384:5] ICMP PING [**] [Classification: Misc acti
vity] [Priority: 3] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:50.099722  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:50.099722  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:50.099722  [**] [1:384:5] ICMP PING [**] [Classification: Misc acti
vity] [Priority: 3] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:51.100530  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:51.100530  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:39:51.100530  [**] [1:384:5] ICMP PING [**] [Classification: Misc acti
vity] [Priority: 3] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:44:07.099149  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:44:07.099149  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:44:07.099149  [**] [1:384:5] ICMP PING [**] [Classification: Misc acti
vity] [Priority: 3] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:44:07.200018  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.0.130 -> 192.168.0.221
08/15-02:44:07.200018  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Atte

```

- Using Wireshark for capturing and observing the packet transfer:

The screenshot shows the Wireshark interface with the following details:

- Filter:** ip.addr==192.168.0.221
- Packet List Table:**

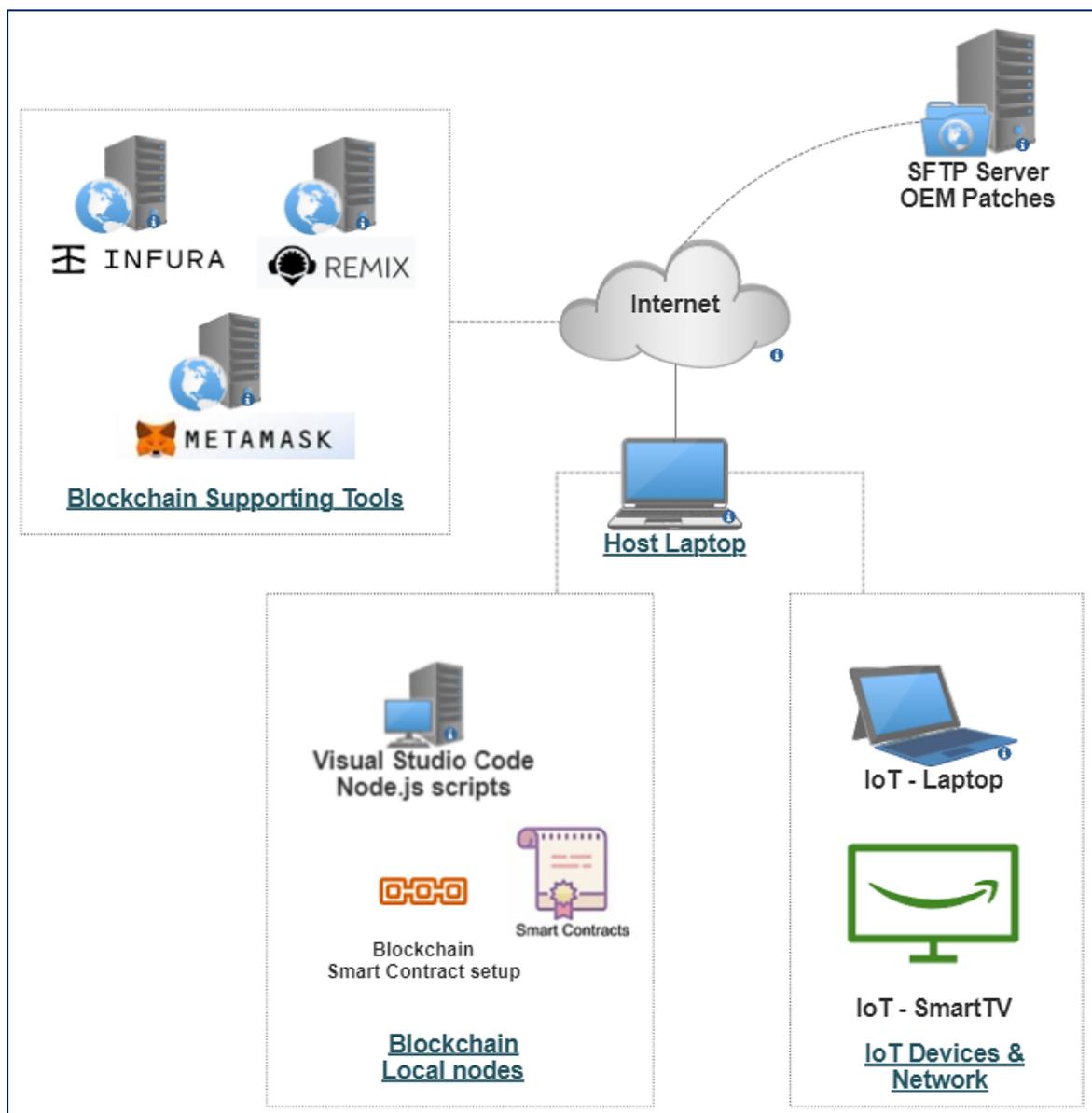
No.	Time	Source	Destination	Protocol	Length	Info
34097	9516.0655328...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34098	9516.1655202...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34099	9516.1658601...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34100	9516.2662249...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34101	9516.2666090...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34102	9516.3671508...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34103	9516.3674554...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34104	9516.4675007...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34105	9516.4679374...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34106	9516.5711100...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34107	9516.5715135...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34108	9516.6714561...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34109	9516.6717904...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
34110	9516.7951995...	192.168.0.130	192.168.0.221	ICMP	42	Echo (ping) request id:
34111	9516.7955832...	192.168.0.221	192.168.0.130	ICMP	60	Echo (ping) reply id:
- Packet Details:**
 - Frame 25192: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth1, id 0
 - Ethernet II, Src: PcsCompu_6c:ee:0f (08:00:27:6c:ee:0f), Dst: PcsCompu_48:05:f6 (08:00:27:48:05:f6)
 - Internet Protocol Version 4. Src: 192.168.0.130. Dst: 192.168.0.221

Blockchain Implementation Steps:

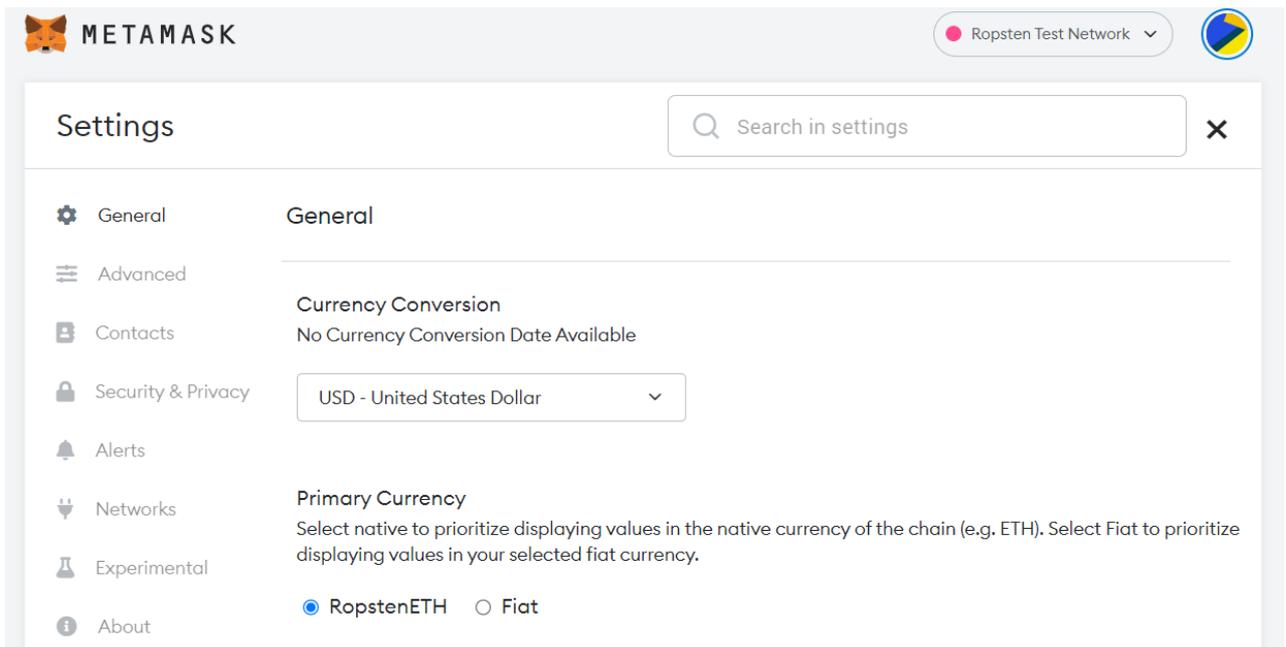
Below are the steps used for implementation of Blockchain setup for this Project, this will be followed with some relevant screenshots and inputs:

- Creation of Metamask ID
- Add gas using Faucet free tokens
- Test Smart contract in Remix
- Create Infura account
- Create scripts in Node.js for auto smart contracts and get function

Blockchain setup architecture diagram:



1. Creation of Metamask account and selecting Ropsten network



2. Adding Free Faucet Tokens – required to execute smart contracts and simulate tests.

Ropsten testnet faucet

Your Ropsten address

[Give me Ropsten ETH!](#)

Successfully queued **10.0000** to
0x90E5676487A0371e030d33F348d68fc1E4FB054f.
Please expect rETH to your address within **half an hour**.

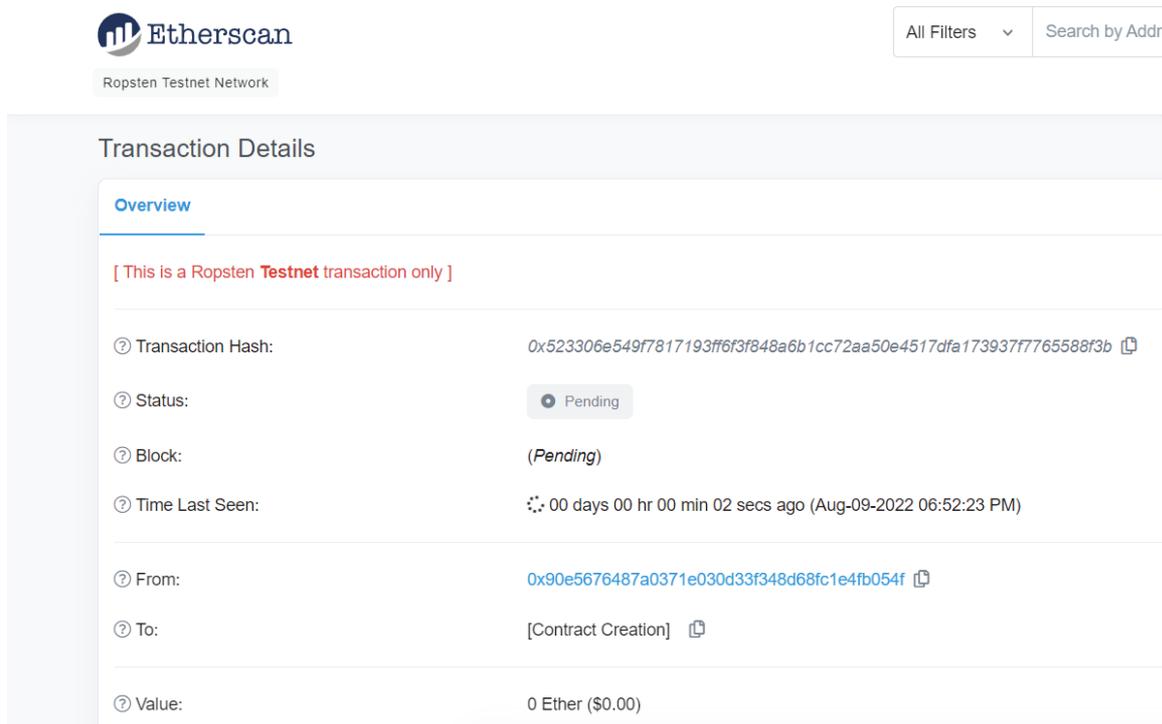
Need more rETH?

Faucet stats

93,633,589.3986 rETH available
10.0000 rETH daily limit per address
5 recipients queued
Faucet [0x7917 ... D36B](#)
Currently at block 12753705

Did not receive your rETH?

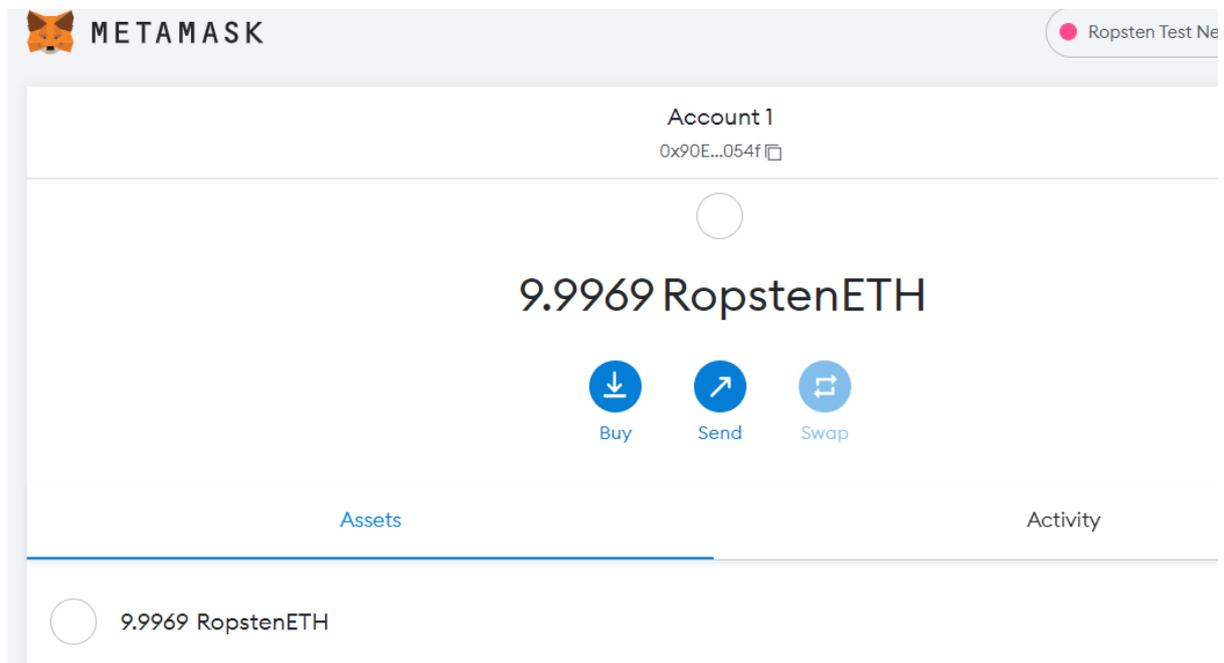
3. Etherscan for validation



The screenshot shows the Etherscan interface for a transaction on the Ropsten Testnet. The transaction is pending. The details are as follows:

Field	Value
Transaction Hash	0x523306e549f7817193ff6f3f848a6b1cc72aa50e4517dfa173937f7765588f3b
Status	Pending
Block	(Pending)
Time Last Seen	00 days 00 hr 00 min 02 secs ago (Aug-09-2022 06:52:23 PM)
From	0x90e5676487a0371e030d33f348d68fc1e4fb054f
To	[Contract Creation]
Value	0 Ether (\$0.00)

4. Receipt of Tokens on Metamask



The screenshot shows the Metamask wallet interface for Account 1. The balance is 9.9969 RopstenETH. The interface includes navigation options for Buy, Send, and Swap, and tabs for Assets and Activity.

Account 1
0x90E...054f

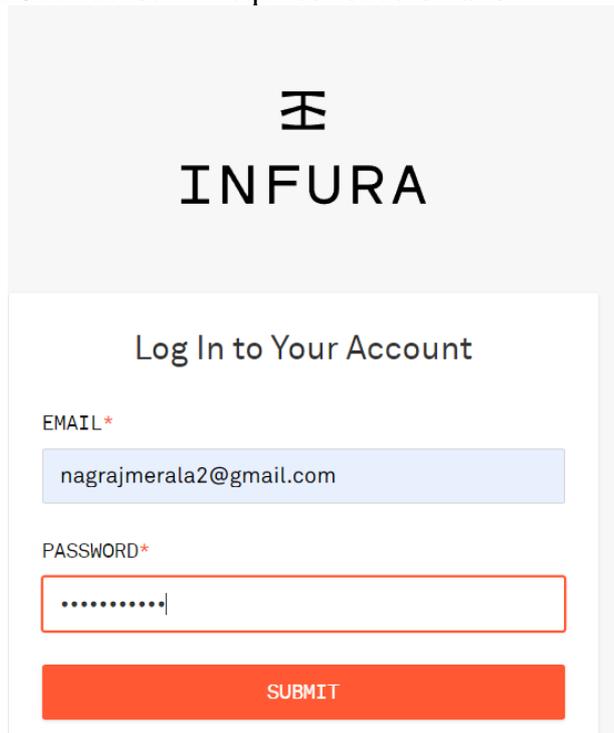
9.9969 RopstenETH

Buy Send Swap

Assets Activity

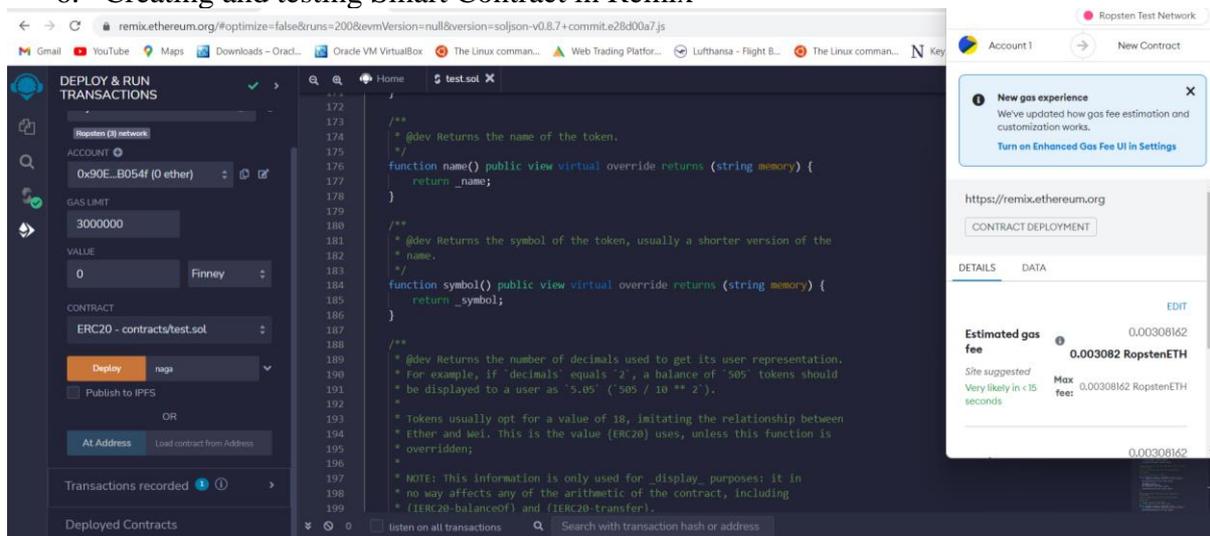
9.9969 RopstenETH

5. Creation of INFURA account - required for automation



The image shows the INFURA login page. At the top is the INFURA logo, which consists of a stylized symbol above the word "INFURA". Below the logo is a white box with the heading "Log In to Your Account". Underneath, there are two input fields: "EMAIL*" with the value "nagrajmerala2@gmail.com" and "PASSWORD*" with a masked password of ten dots. A red "SUBMIT" button is located at the bottom of the form.

6. Creating and testing Smart Contract in Remix



The image displays the Remix IDE interface. On the left, the "DEPLOY & RUN TRANSACTIONS" panel is visible, showing the "Ropsten (E) network" selected, an account with address "0x90E...B054f", a gas limit of "3000000", and a value of "0" in "Finney". The "CONTRACT" dropdown is set to "ERC20 - contracts/test.sol", and the "Deploy" button is highlighted. The main editor shows a Solidity smart contract for an ERC20 token. The code includes comments and functions: `name()` returning the token name, `symbol()` returning the token symbol, and `decimals()` returning the number of decimals. The right sidebar shows the "CONTRACT DEPLOYMENT" details, including the "Estimated gas fee" of "0.003082 RopstenETH" and a "Max fee" of "0.00308162 RopstenETH".

7. Using Node.js for automated script on Smart Contract creation



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
INFURA_BASE_URL="1d0875484eef4ddd882486cf0e730dce"
ACCOUNT_MNEMONIC="0x90E5676487A0371e030d33F348d68fc1E4FB054f"
~
~
~
```

8. Deploying Smart Contract:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

$

nagra@LAPTOP-GL3A1IPG MINGW64 ~/simple-smart-contracts-main
$

nagra@LAPTOP-GL3A1IPG MINGW64 ~/simple-smart-contracts-main
$ npm run deploy

> simple-smart-contracts@1.0.0 deploy
> node deploy.js

Using account 0x90E5676487A0371e030d33F348d68fc1E4FB054f for deployment...
Contract Inbox deployed to address : 0x93Cc38d2553Ba7DDdc92fc46CaA58a9bD970e9a7
```

9. Verifying transaction on Etherscan:

The screenshot shows the Etherscan interface for the Ropsten Testnet Network. The address 0x90E5676487A0371e030d33F348d68fc1E4FB054f is selected. The account has a balance of 9.993321922481301383 Ether. The transaction history shows three recent transactions, all of which are Contract Creation transactions with a value of 0 Ether and a transaction fee of approximately 0.00273368 Ether.

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x4aba6e6e5f9b309812...	0x60806040	12753851	1 min ago	0x90e5676487a0371e03...	OUT Contract Creation	0 Ether	0.00273368
0x92cfc52c45e2f610533...	0x60806040	12753849	2 mins ago	0x90e5676487a0371e03...	OUT Contract Creation	0 Ether	0.00086277
0x523306e549f7817193f...	0x60806040	12753725	28 mins ago	0x90e5676487a0371e03...	OUT Contract Creation	0 Ether	0.00308161

10. Using DaPP like IPFS etc.:

IPFS (InterPlanetary File System): Integrating IPFS with Blockchain would have been the ideal next step to achieve an end-to-end secured File Transfer platform which can help achieve the objective and hypothesis of using Blockchain for secured Patch updates. We have been able to complete the first objective of deploying smart contract and maintaining a ledger of transaction for tracking and future root cause analysis purpose, we attempted at deploying an IPFS solution but needs more work in future to achieve this in a real-world scenario. We have looked at few research papers and found it to be a feasible option to deploy IPFS for achieving Patch downloads post Smart contract step.

Discussion: I have conducted several tests during the course of these implementations and evaluations to review the functionality of the deployed tools and services, same time have conducted Penetration testing to evaluate the security capability and alert mechanisms, based on the observations I can state that the proposed solutions have tremendous potential to secure IoT in a Novel way and these solutions are scalable in nature as well hence applicable to wide variety of use cases starting from Home based IoT Network to Industrial, Healthcare and other similar setups.

Conclusion: This work was aimed at securing and defending IoT using SDN, IDS and Blockchain capabilities and solutions. I have explored several possible solutions using these technologies and researched extensively, based on my research I have been able to establish that it is possible to achieve the objectives outlined in this research paper and research question.

I have used Mininet Tool to demonstrate an SDN based connectivity and combined it with Snort based IDS tool to provide a layer of security where Cyberthreats like Botnet attacks and Data privacy issues can be mitigated. This setup was tested using Penetration testing methodologies and we could observe the results where there is an enhanced level of protection provided by this setup compared to the standard home based IoT connectivity without any such security layer. Snort based IDS played a critical role in monitoring, analysing and detecting the potential security incidents.

I have used Blockchain based Smart contract solution and relevant tools to address the improper patch management capability and file download or transactions related security concerns. This particular objective has been achieved partially where we were able to demonstrate the deployment of Smart contract but could not utilise Blockchain as a standalone solution to transfer files, instead as per research we could observe other solutions or DAPP for achieving the final part of File transfer trigger.

These demonstrations were able to answer the maximum percentage of the Research question of security issues in IoT by using Blockchain, IDS and SDN solutions. The only pending part of transferring the Files using blockchain solution can be addressed using integration of Tools like IPFS as observed in some of the research papers and needs to be conducted as a future scope of work.

Future work: There is a scope to package all these components into a single VM or a lightweight tool etc. with an SOP which will make it easier for a layman or non-tech savvy people as well to utilise the security benefits of SDN, IDS and Blockchain in securing IoT Networks which includes Home IoT, Industrial IoT and Healthcare among others. Another future scope of work is to deploy a suitable IPFS (InterPlanetary File System) solution integrated with Blockchain to trigger a successful Patch download and File transfer mechanism for IoT.