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)	□
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.	□

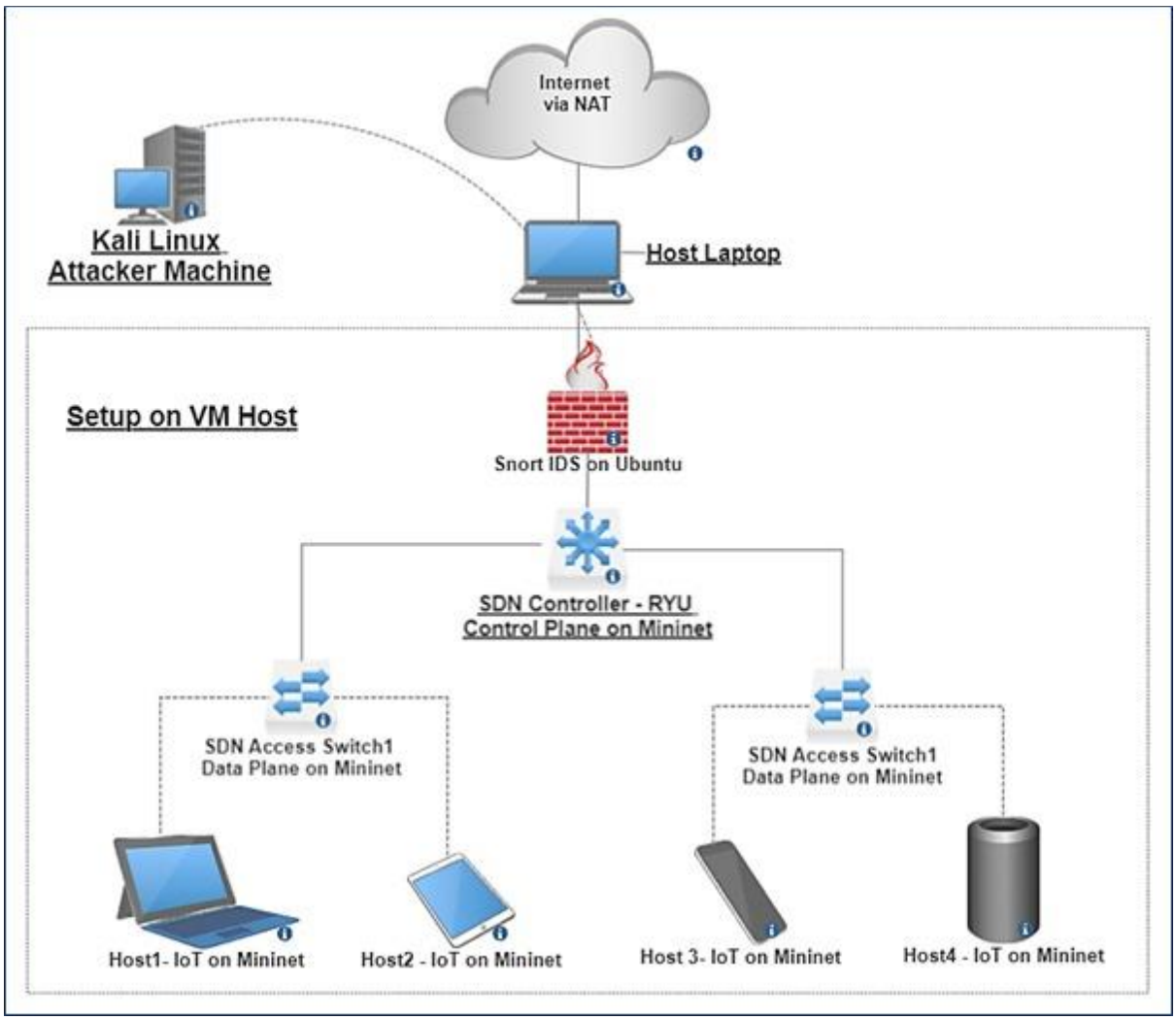
Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

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

Configuration Manual

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:

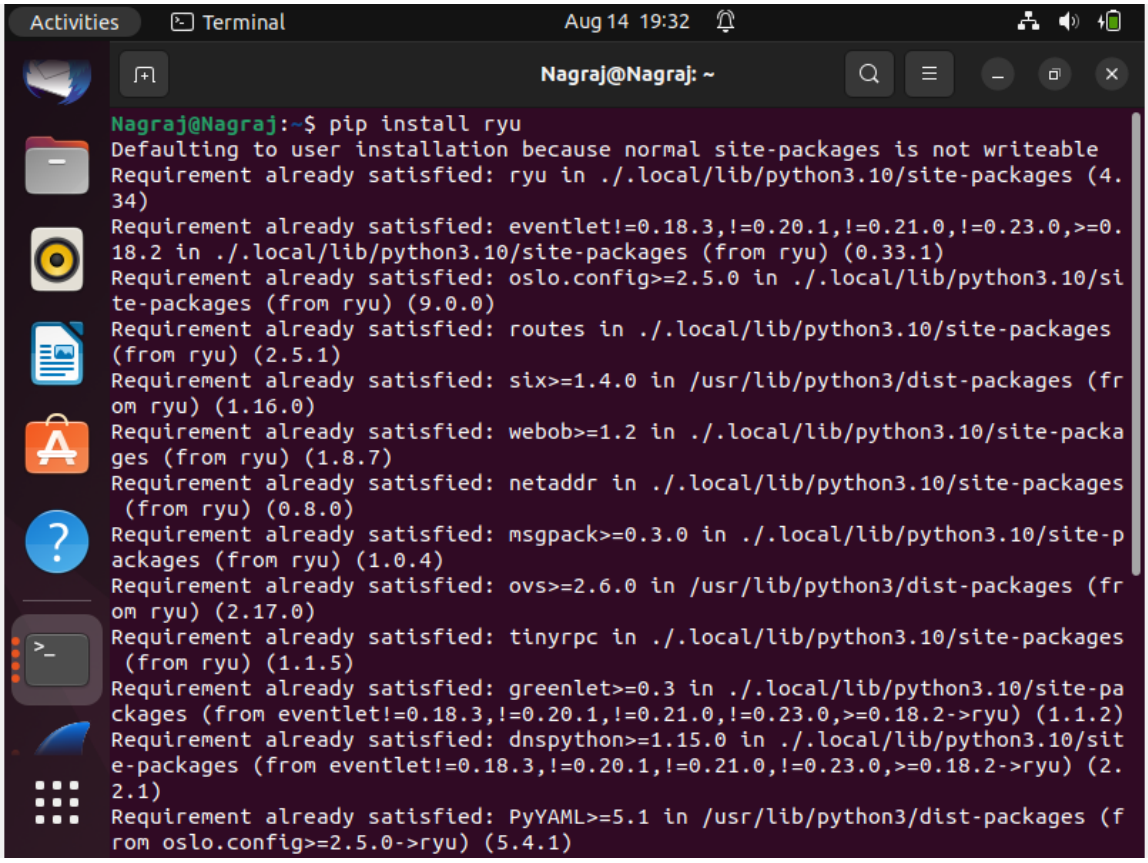
2 Implementation steps

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

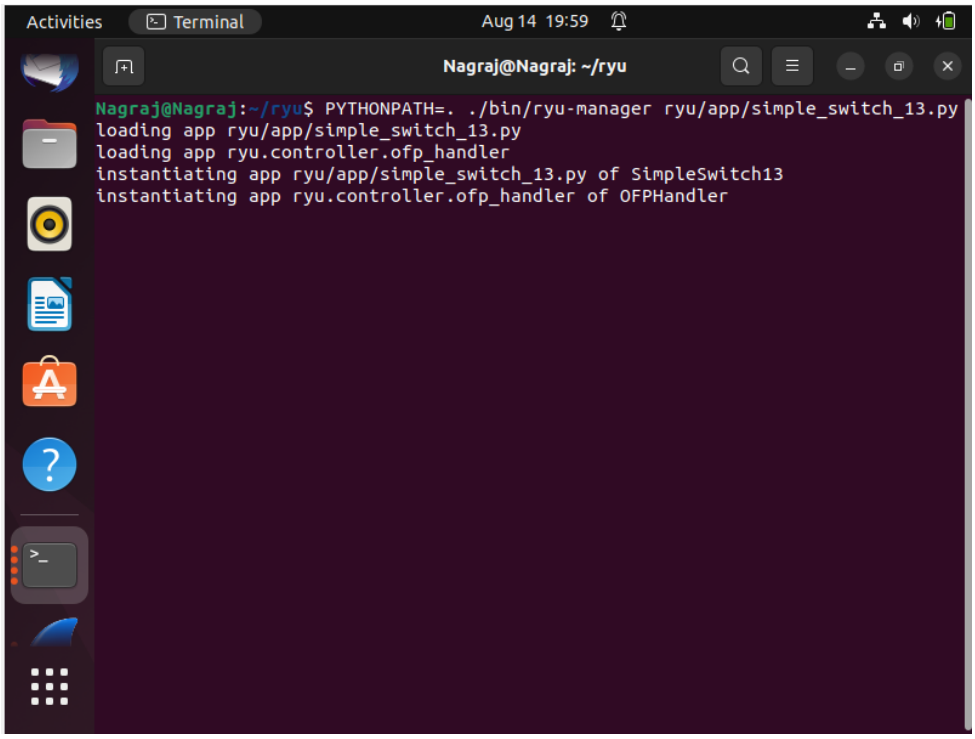


- Follow the below given link for other pre-requisites and dependencies required for installation:

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

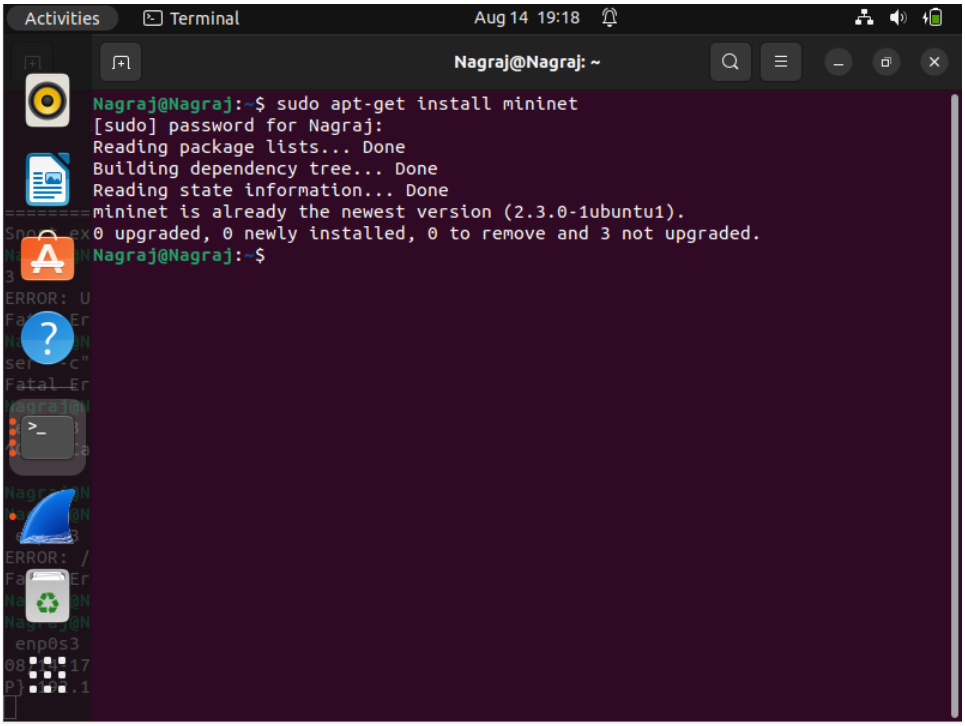


- Once the RYU SDN Controller is switched on, we will create network topology using Mininet

4 Starting with Mininet configuration:

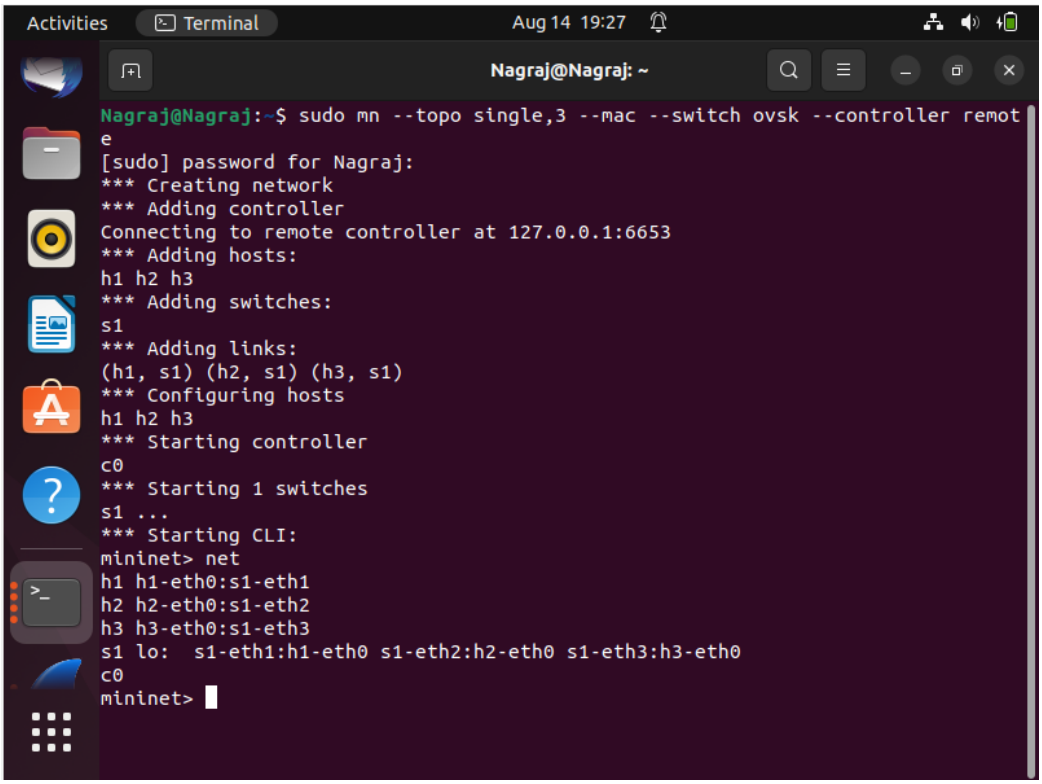
- Downloading and installing the mininet SDN by using command:

sudo apt-get install mininet



- 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



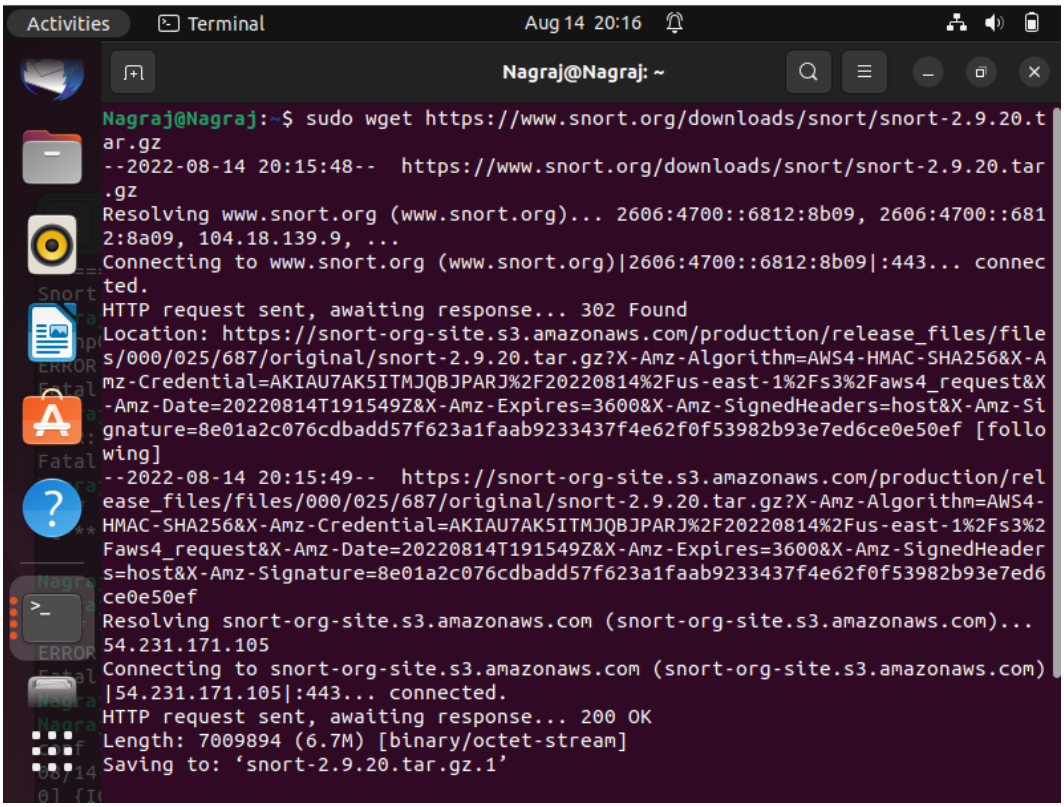
5 Installation and configuration of Snort-IDS (Intrusion Detection System)

- 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



- 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

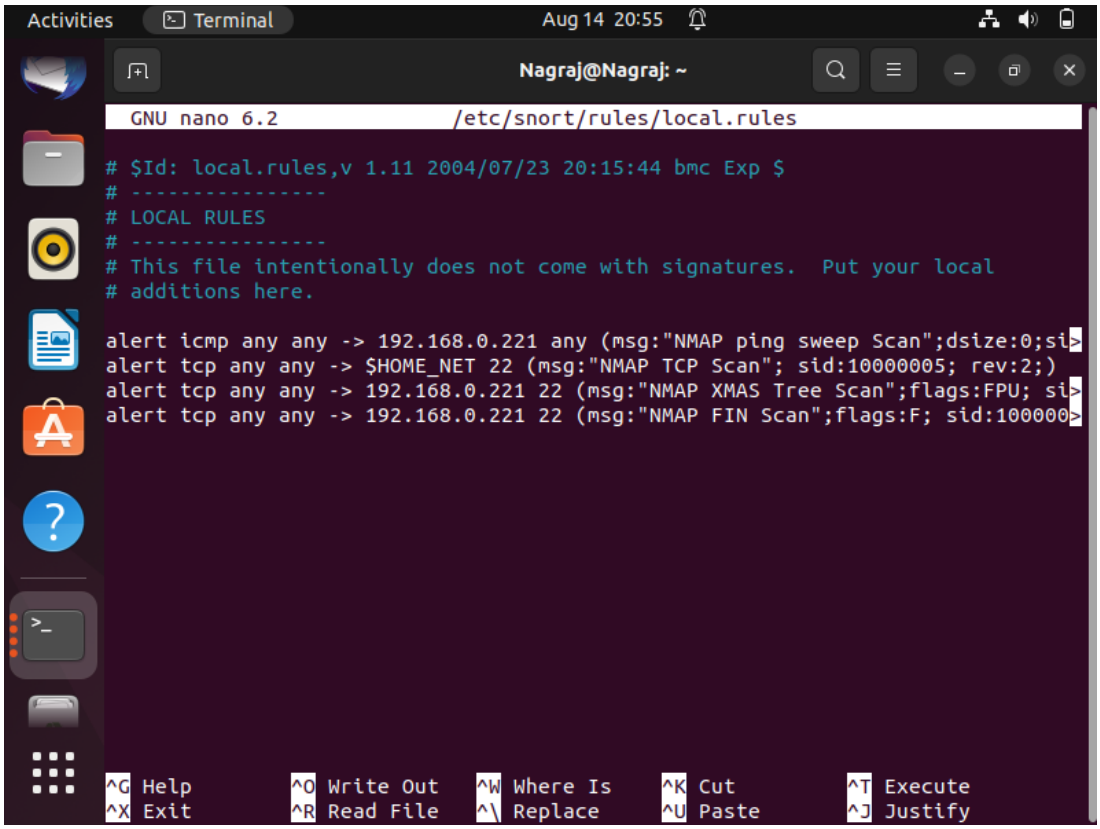
- Snort should be configured and placed inline to monitor the active traffic and internet-based traffic and to do that use command:

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.

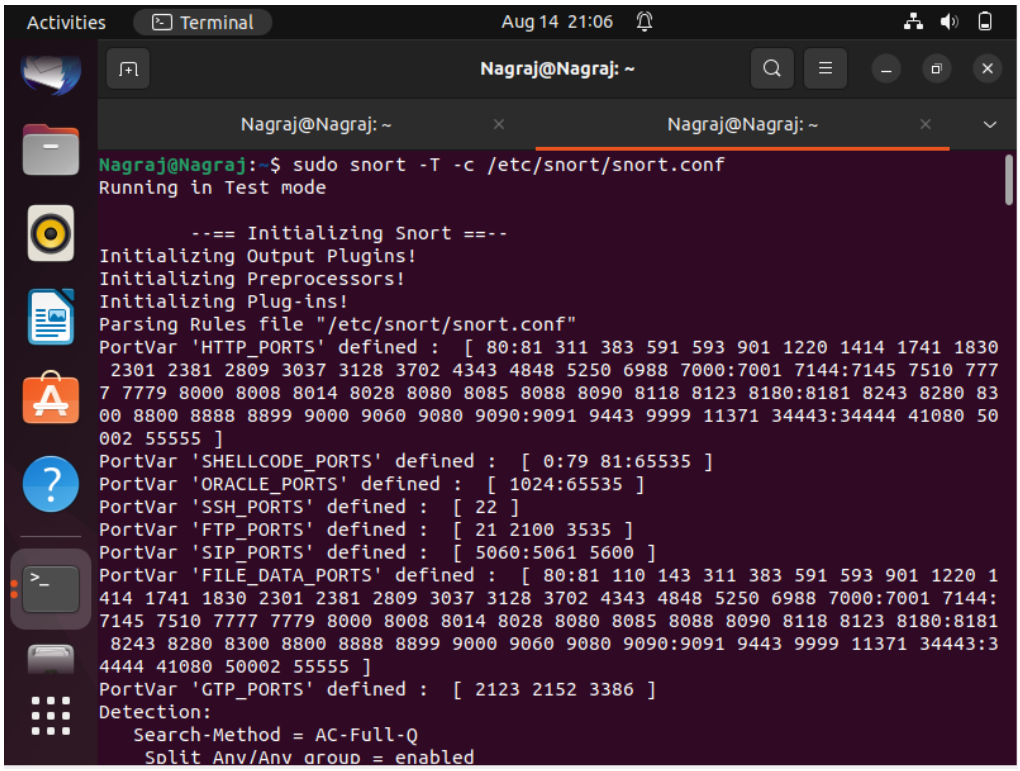
sudo nano /etc/snort/rules/local.rules

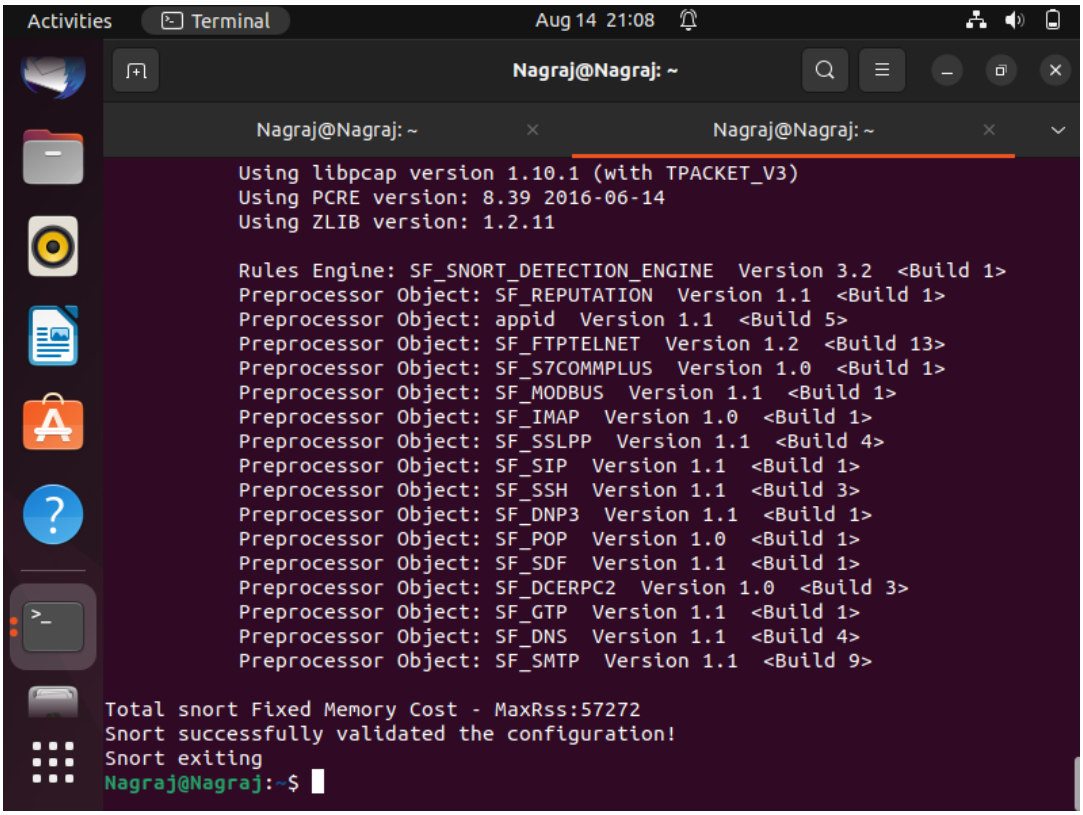
- We have created four rules.



- Initializing the Snort-IDS in test mode to know if it has been installed correctly.

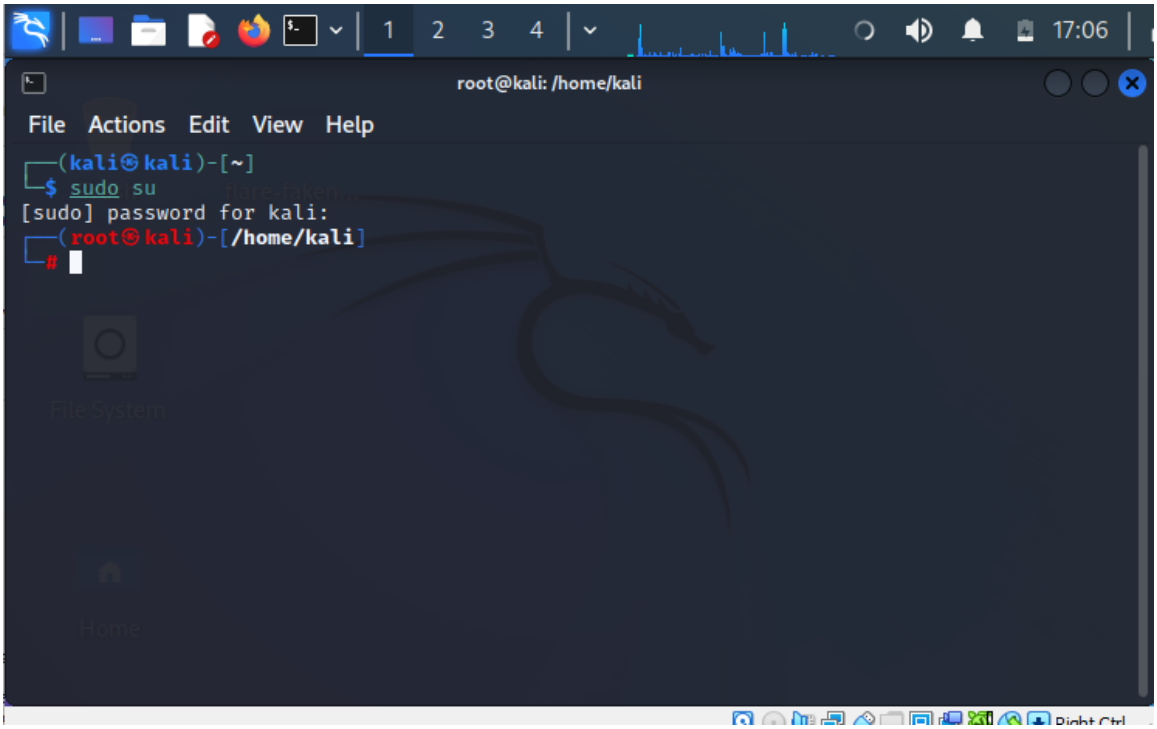
sudo snort -T -c /etc/snort/snort.conf



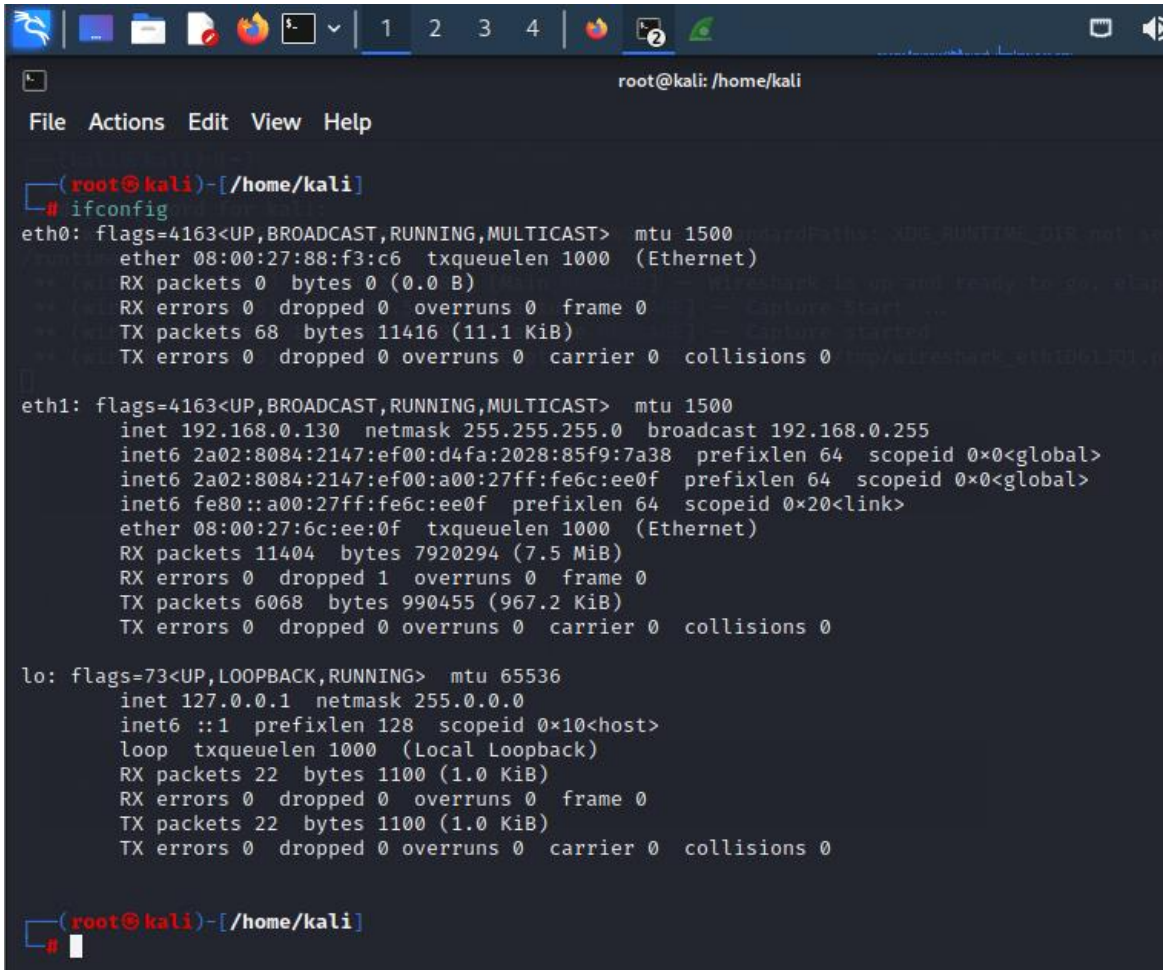


6 Evaluation

- 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.



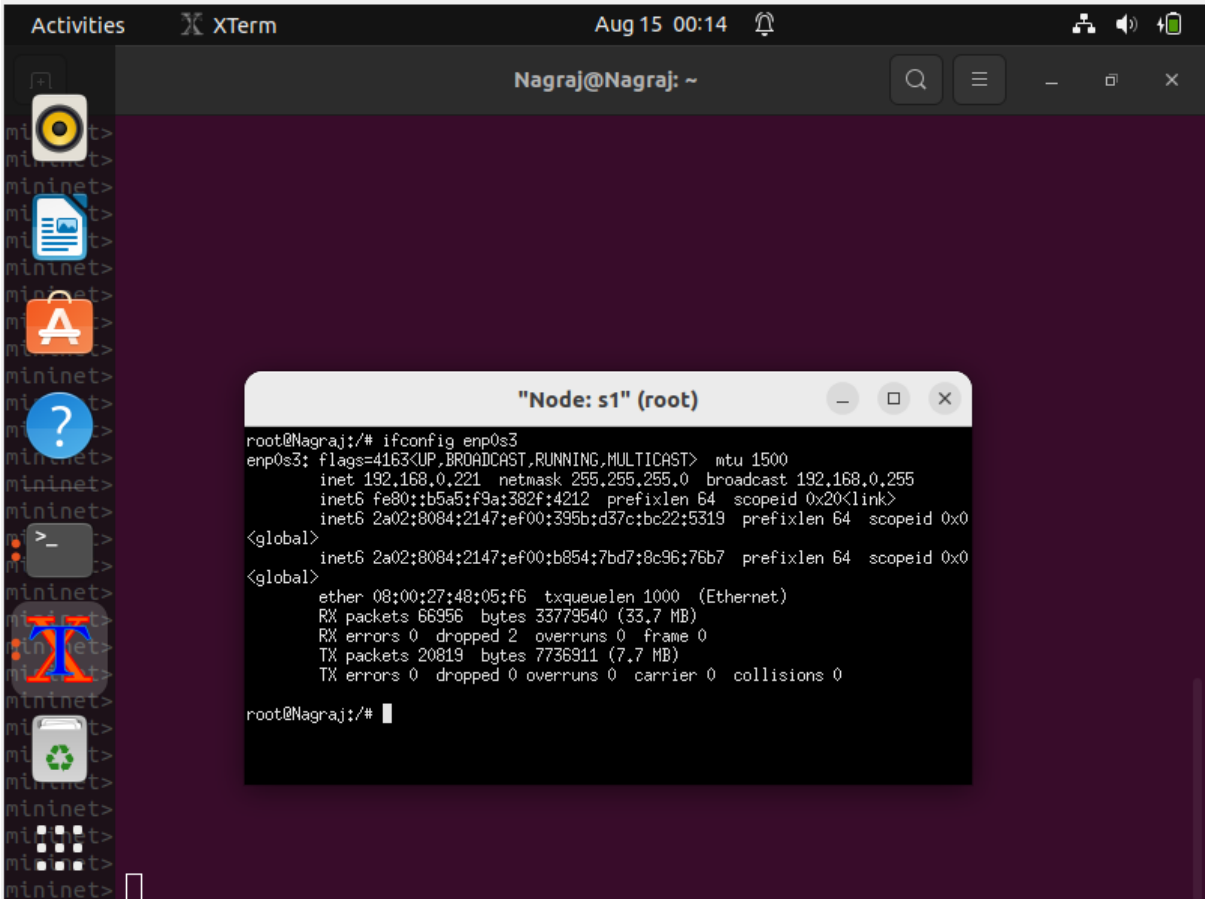
- Use command ifconfig to know the ip address of the machine.



- 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.

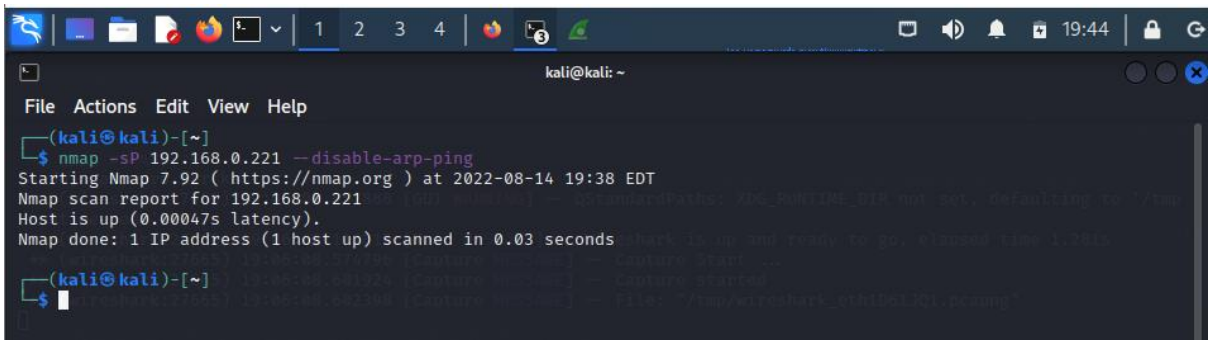
  xterm h1

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

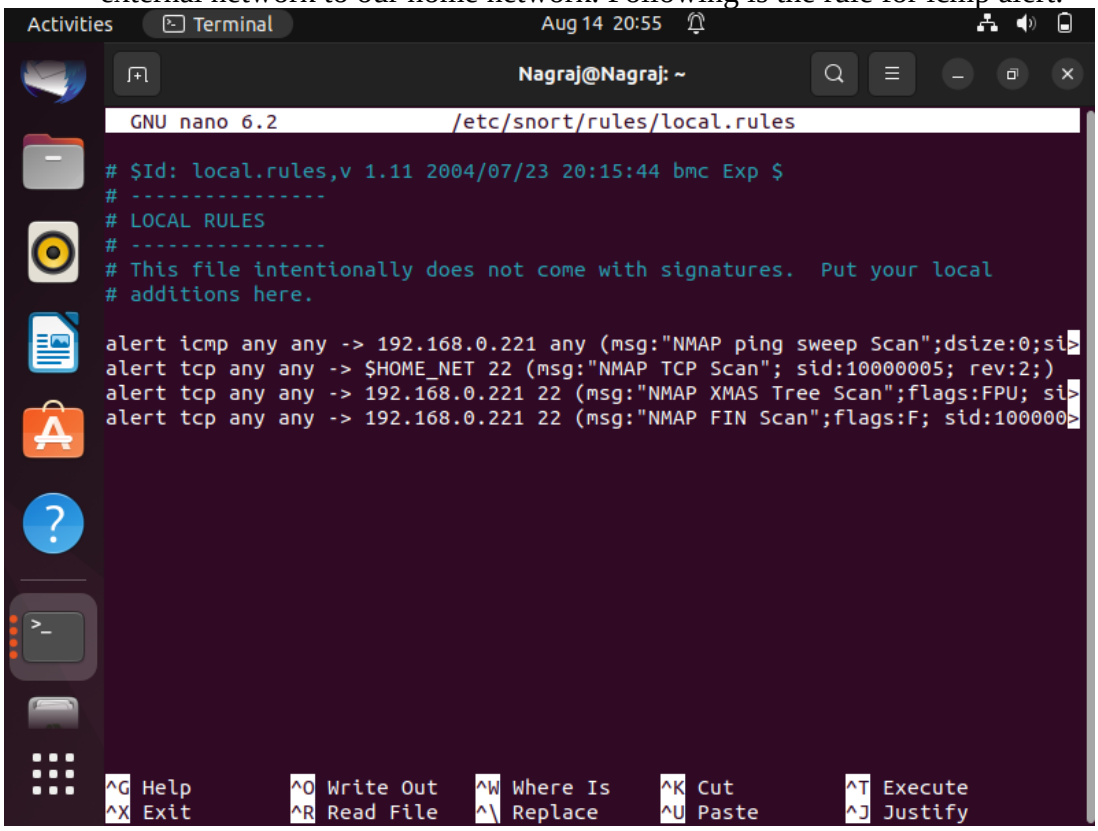


- We have set rules on Snort-IDS to alert the administrator, we will test the rules using nmap on kali linux.
- We will use below command to send icmp packages to host machine to know whether the host is up or not

  nmap -sP 192.168.0.221 –disable -arp-ping

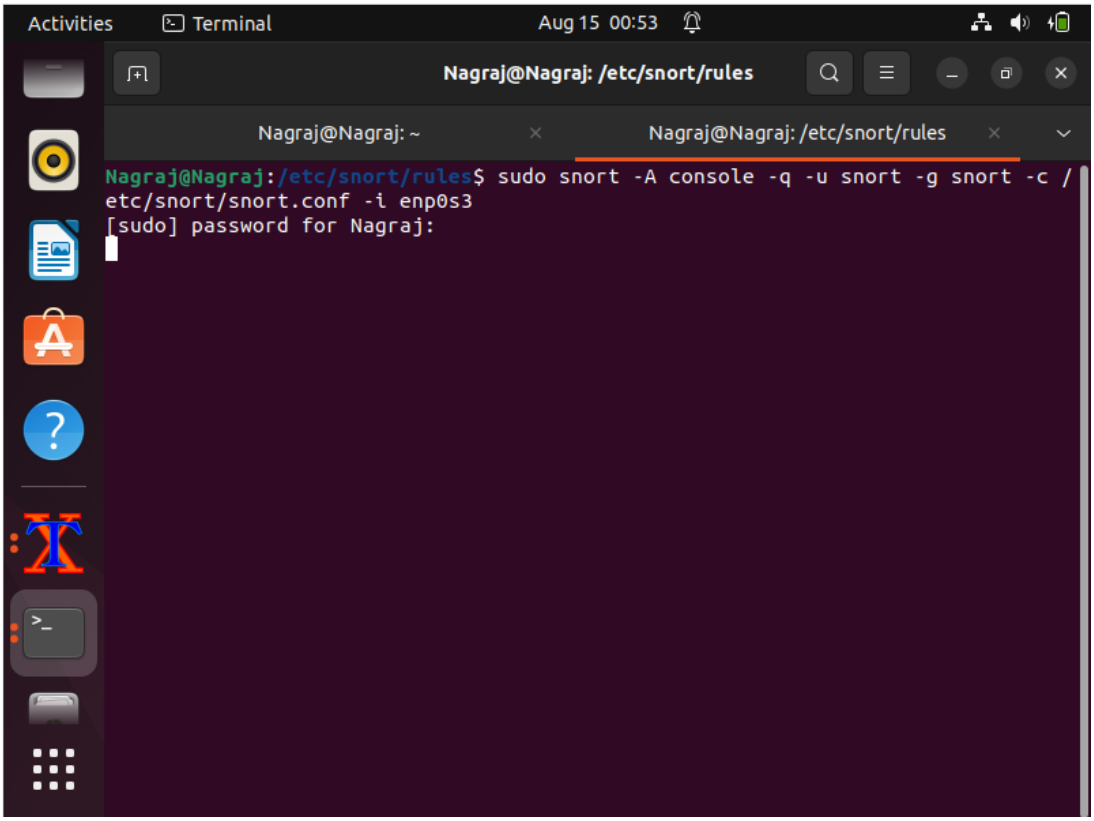


- We have set the rule for icmp packet to alert the administrator when coming from any external network to our home network. Following is the rule for icmp alert.

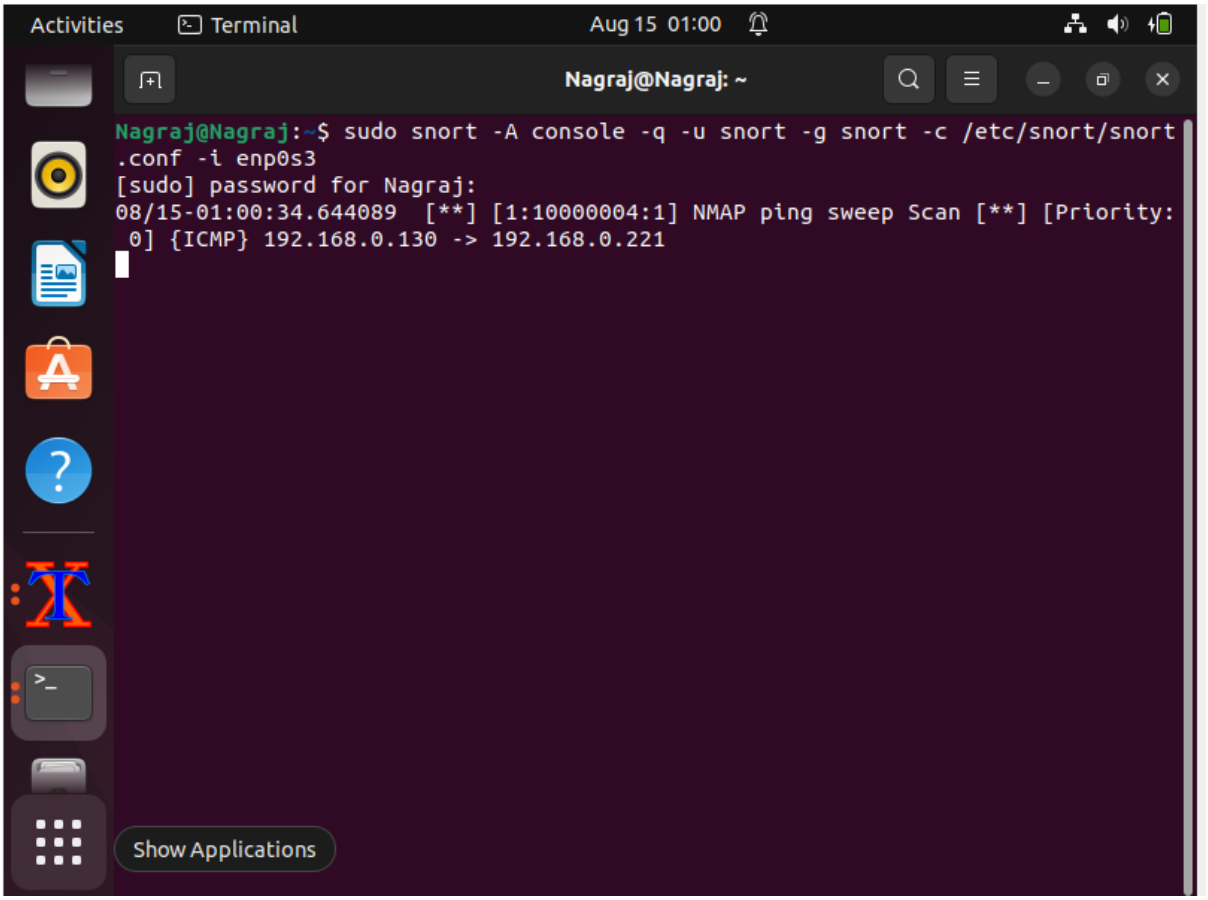


- Now using the below command initialize the Snort-IDS.

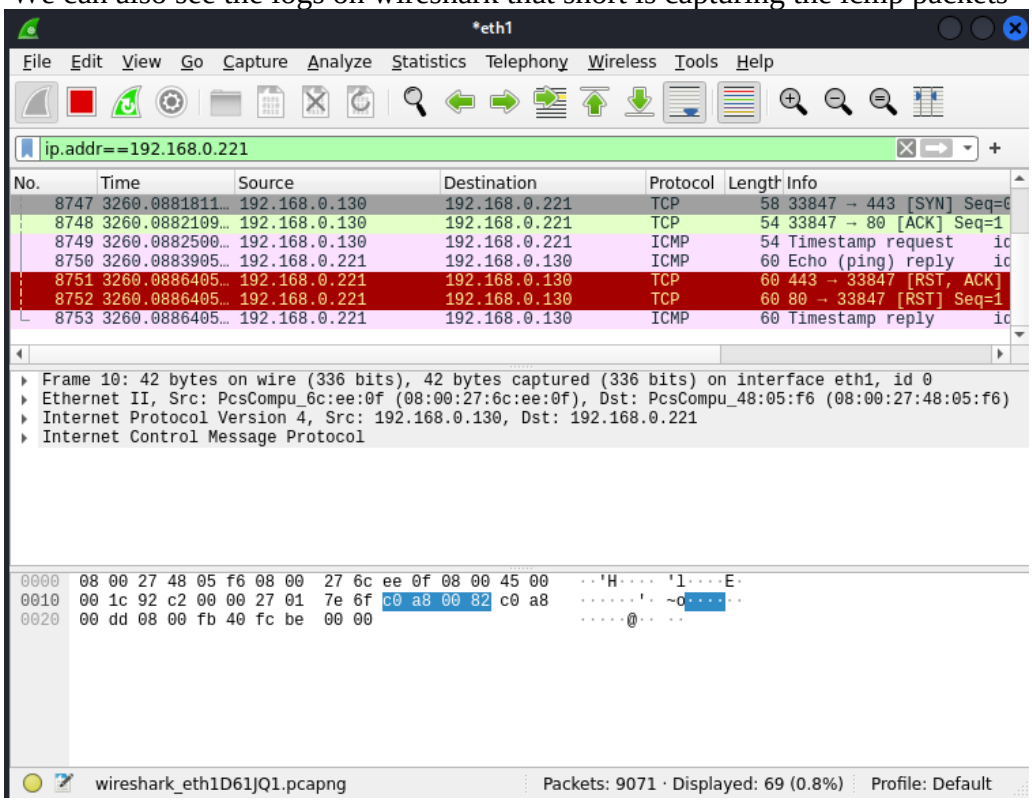
sudo snort -A console -q -u snort 0g snort -c /etc/snort/snort.conf -i enp0s3



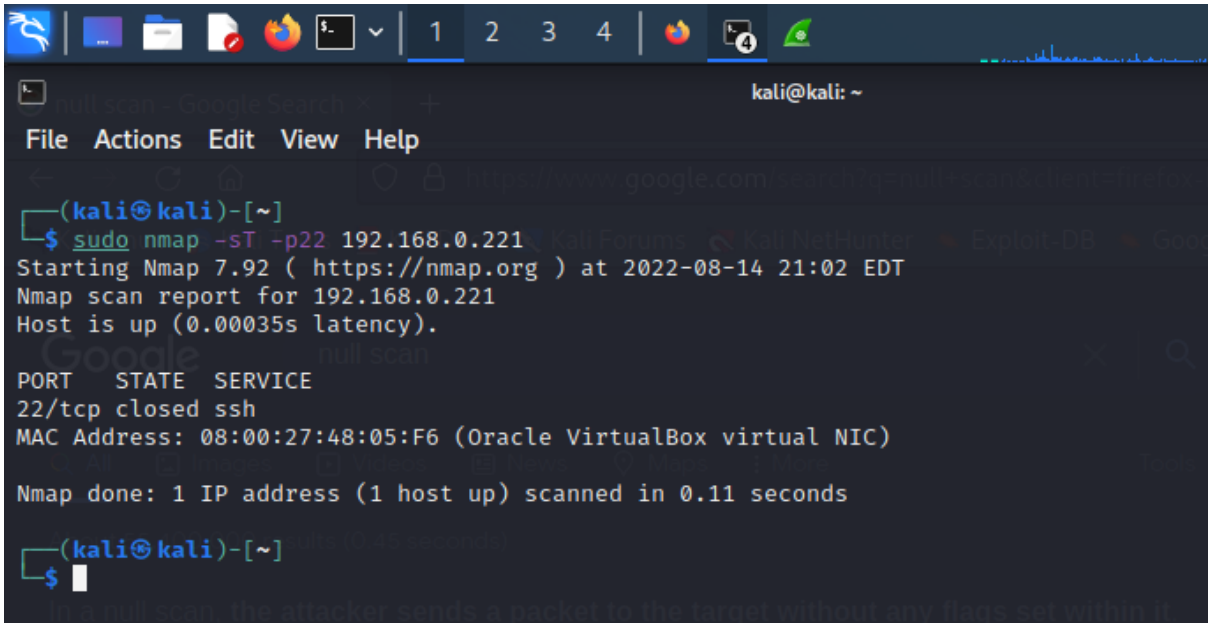
- Alert captured on Snort-IDS terminal



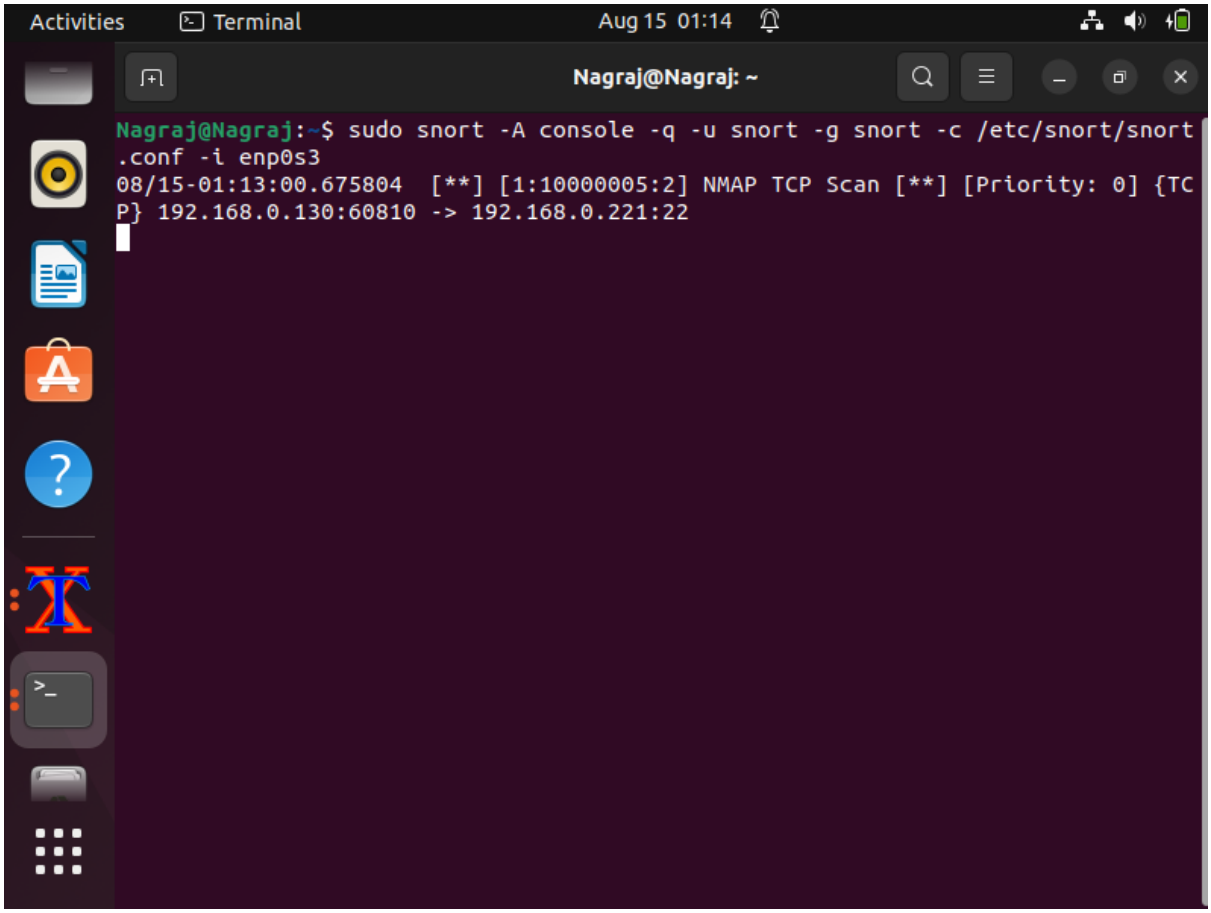
- We can also see the logs on wireshark that snort is capturing the icmp packets



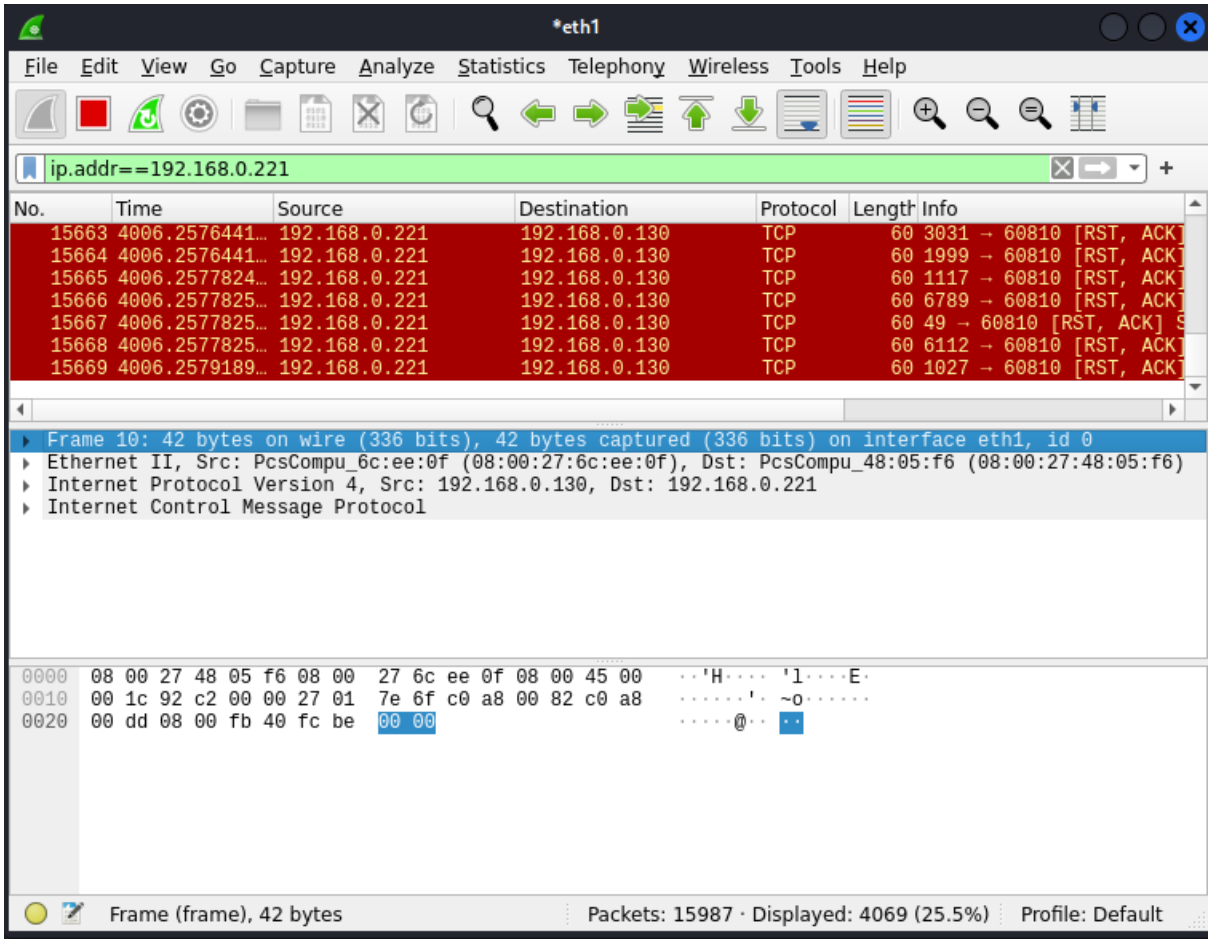
- Following screenshots capture the alert for tcp packets alert coming from any external network using nmap tcp scan



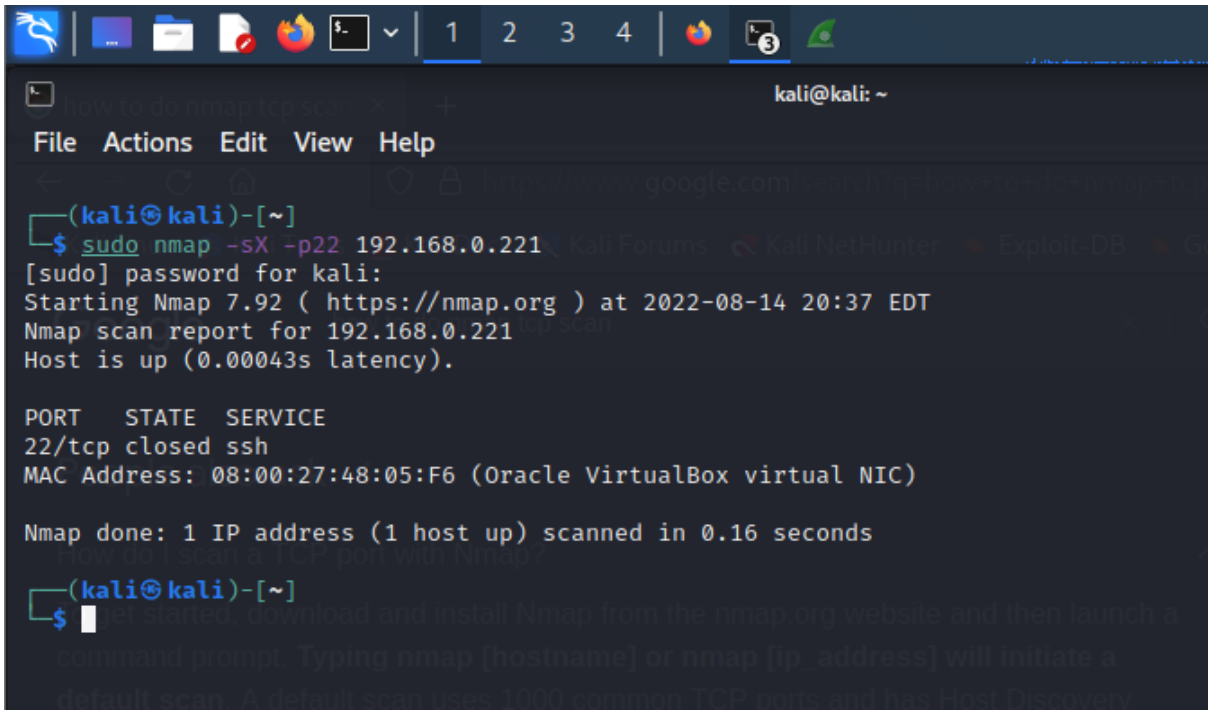
- We will see the TCP packet alert coming on Snort-IDS terminal in below screenshot:



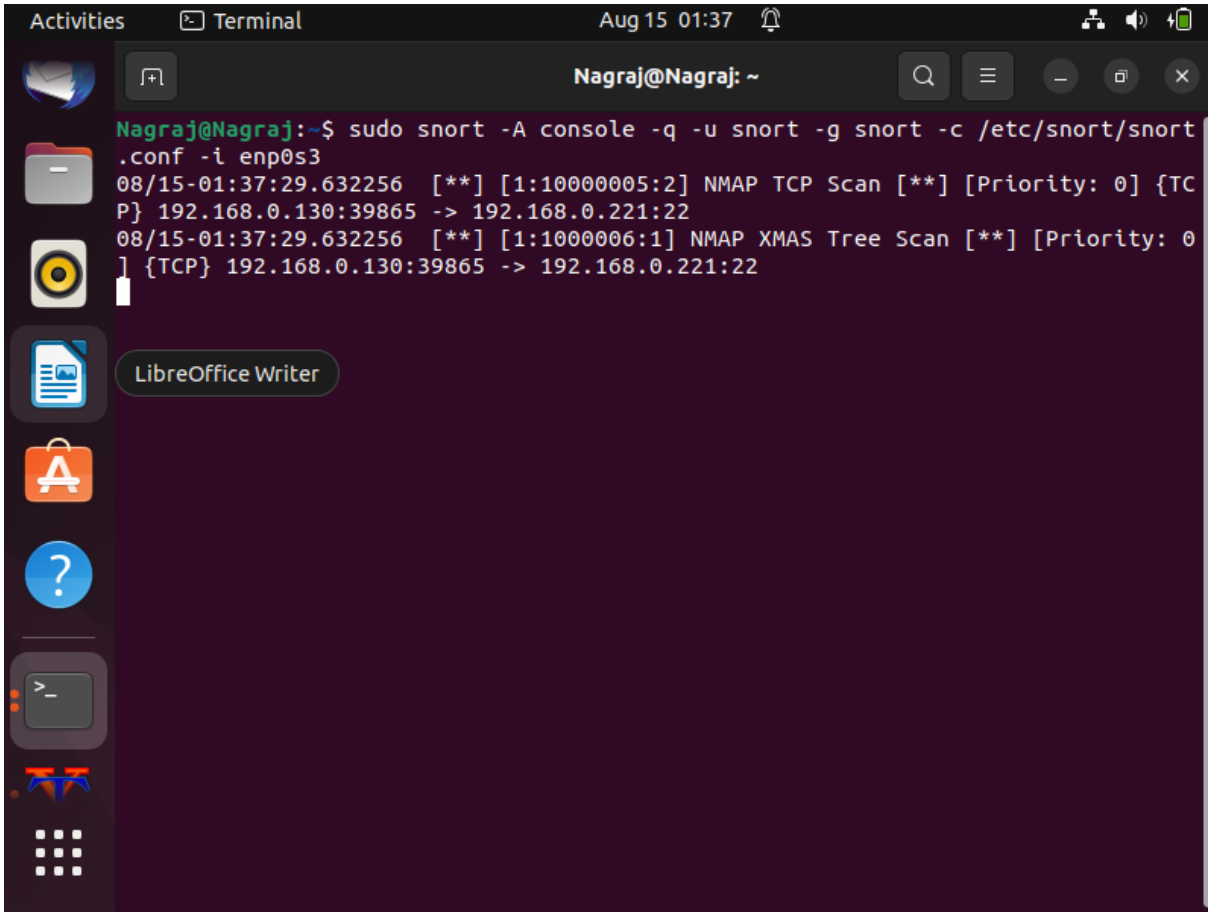
- Using Wireshark for validating the TCP packets.



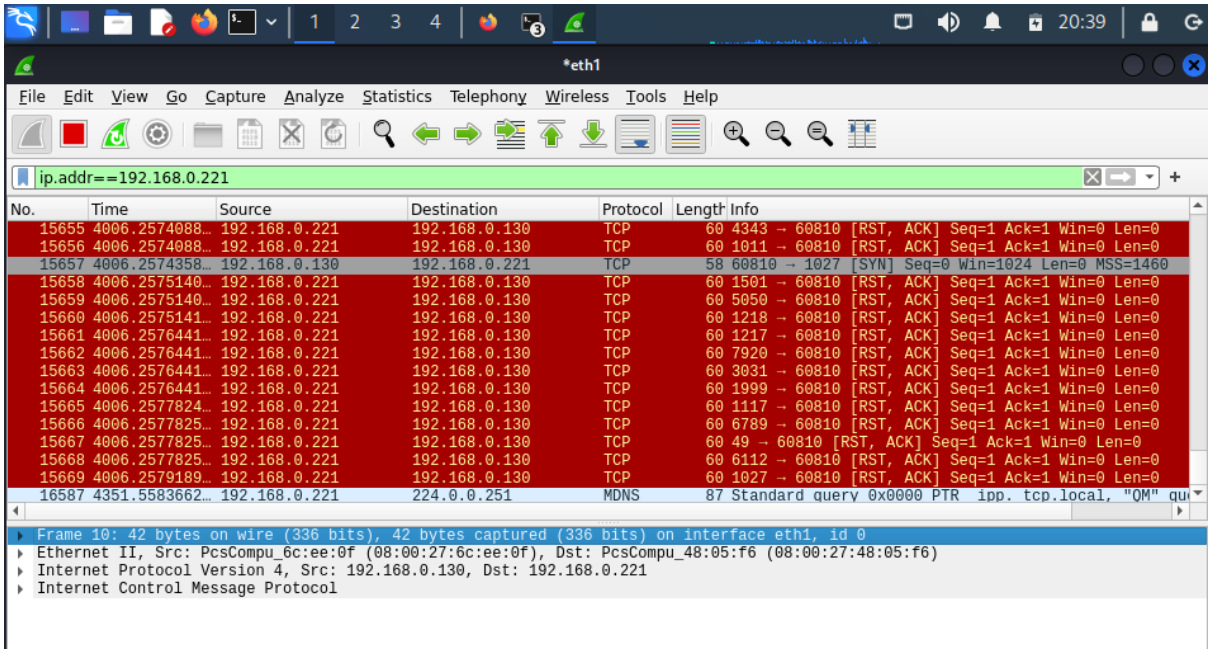
- Below screenshot captures alert for XMAS Scan in which the attacker manipulates the TCP header



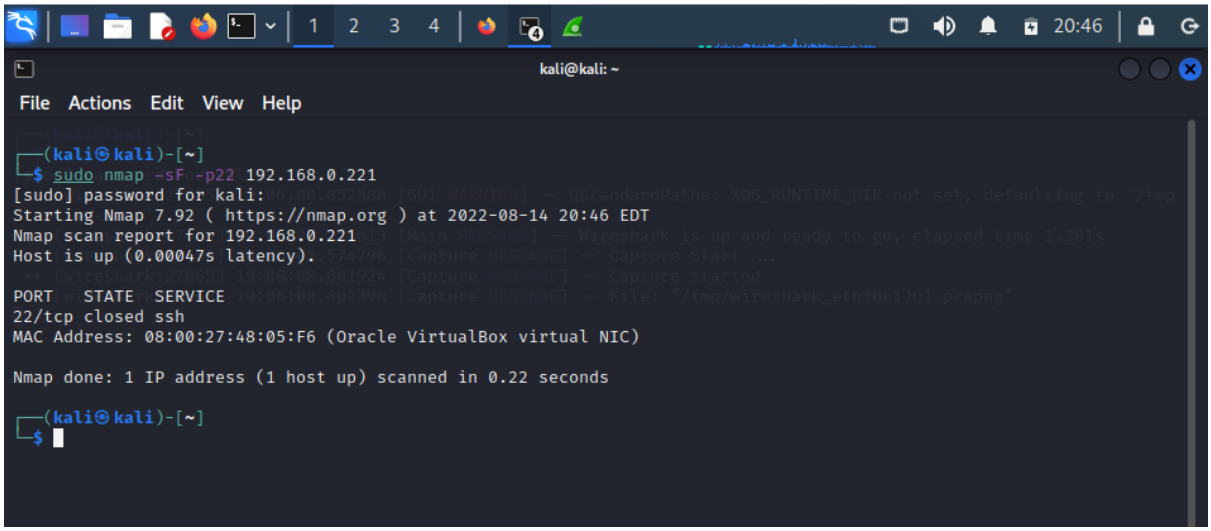
- Refer below screenshot for Snort-IDS terminal capturing the XMAS Scan:



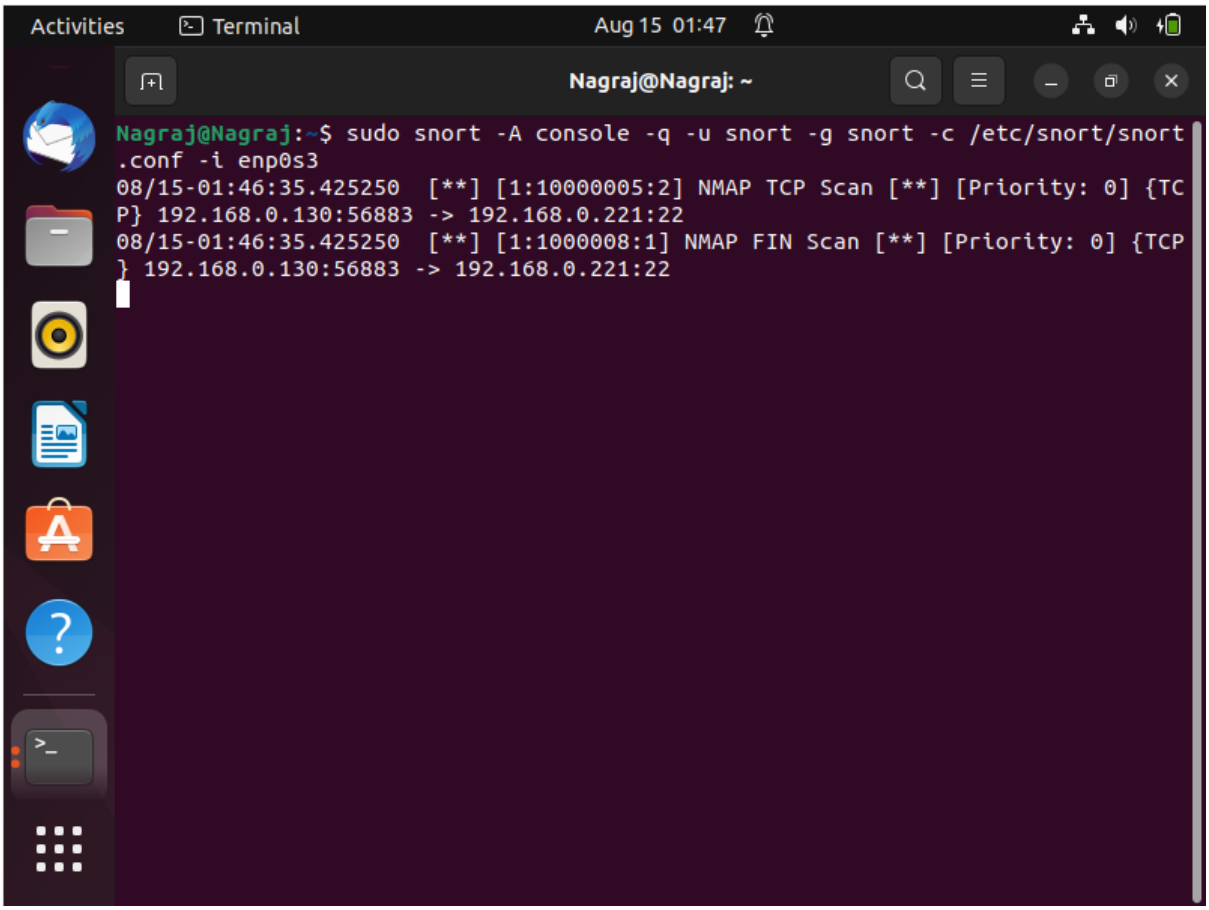
- Analyzing the wireshark output:



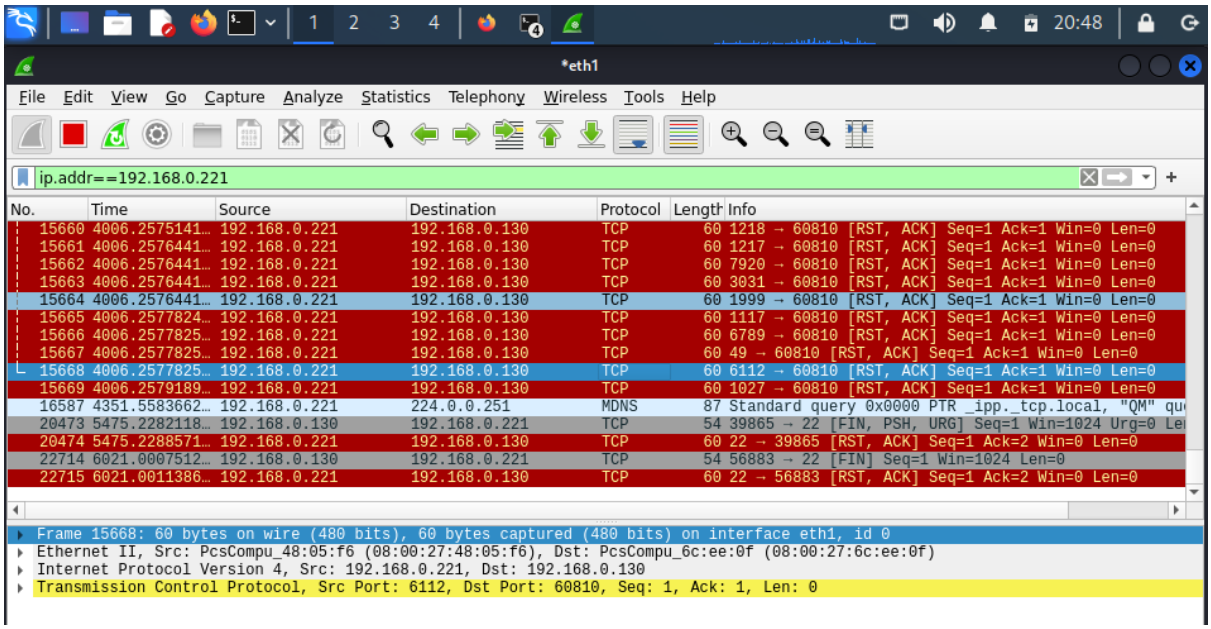
- Getting alert for FIN Scan which is used to terminate TCP connection after completion of the data transfer.



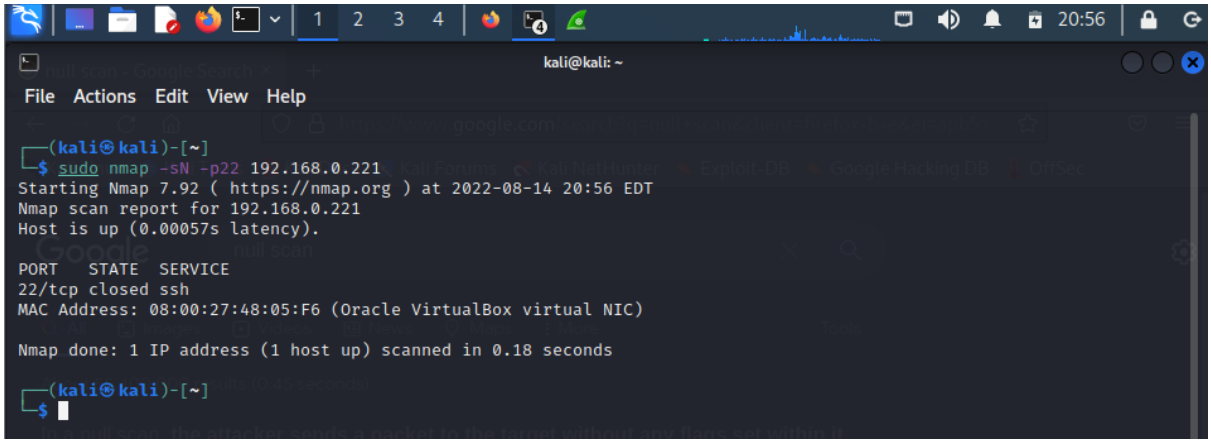
- Getting the alerts for FIN Scan



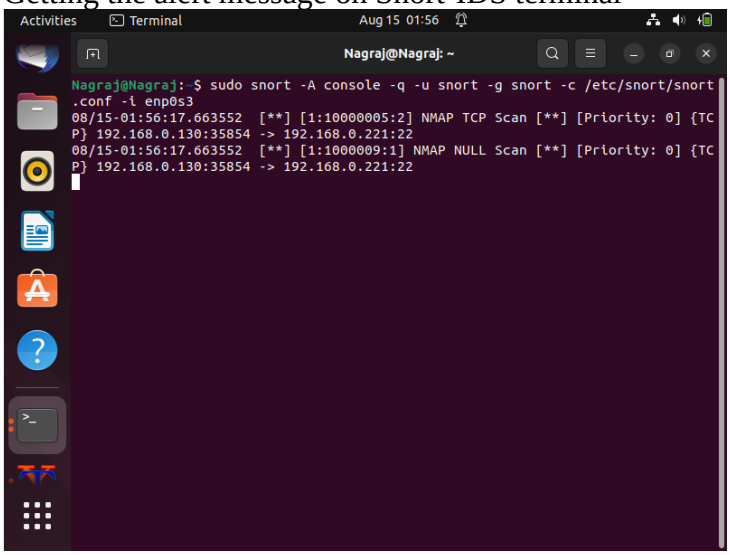
- Validating using Wireshark



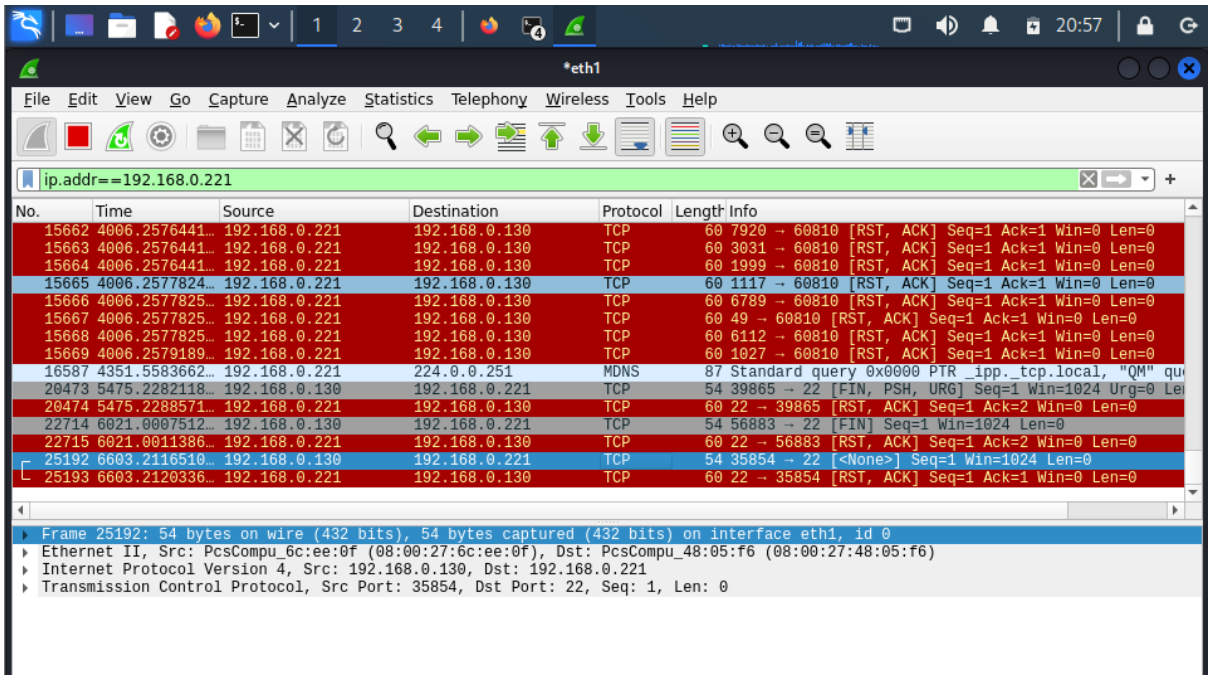
- Capturing NULL Scan in which the packets forwarded by the attacker are without flags



- Getting the alert message on Snort-IDS terminal

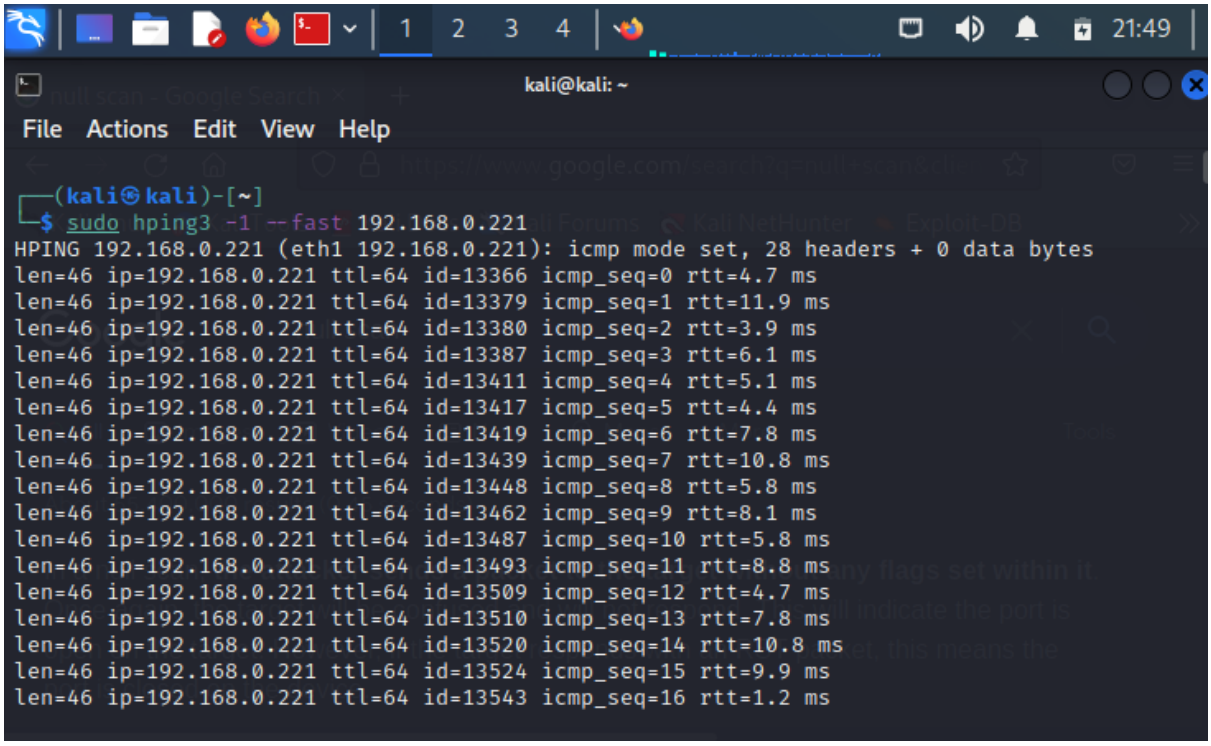


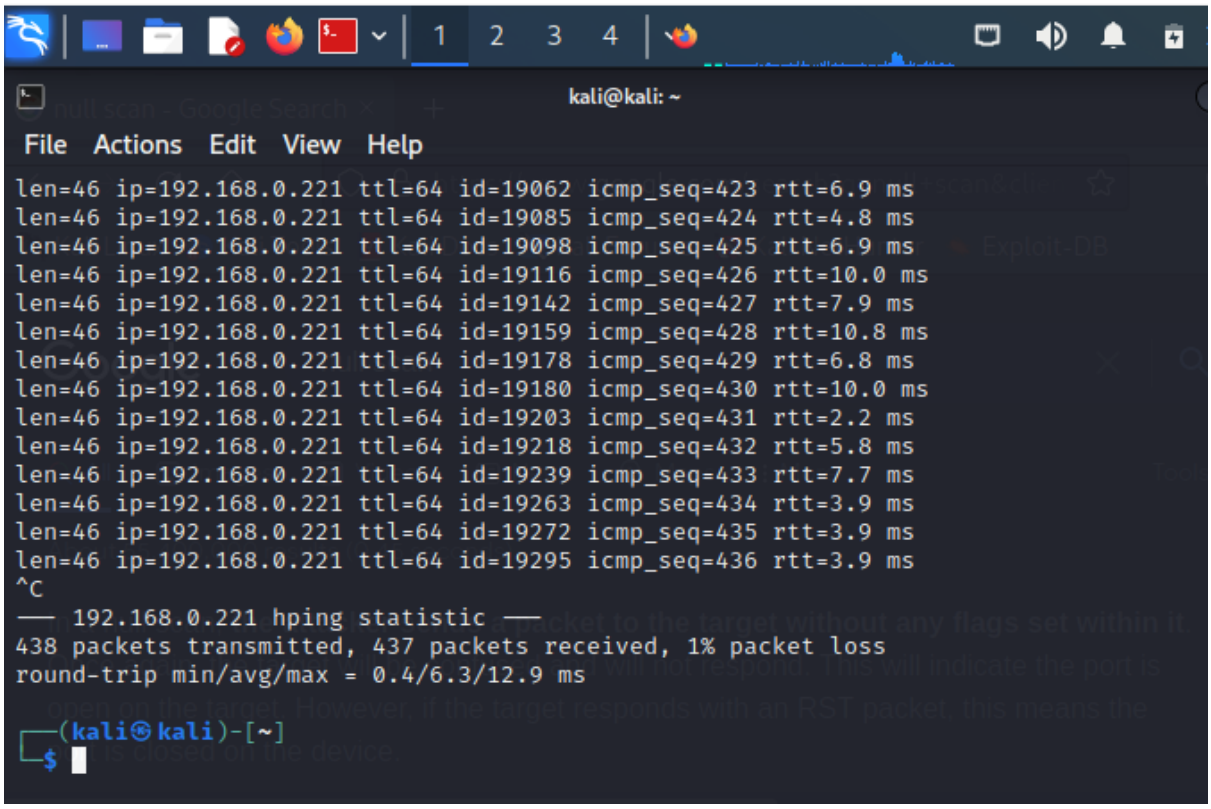
- The line highlighted in the blue is a FIN packet



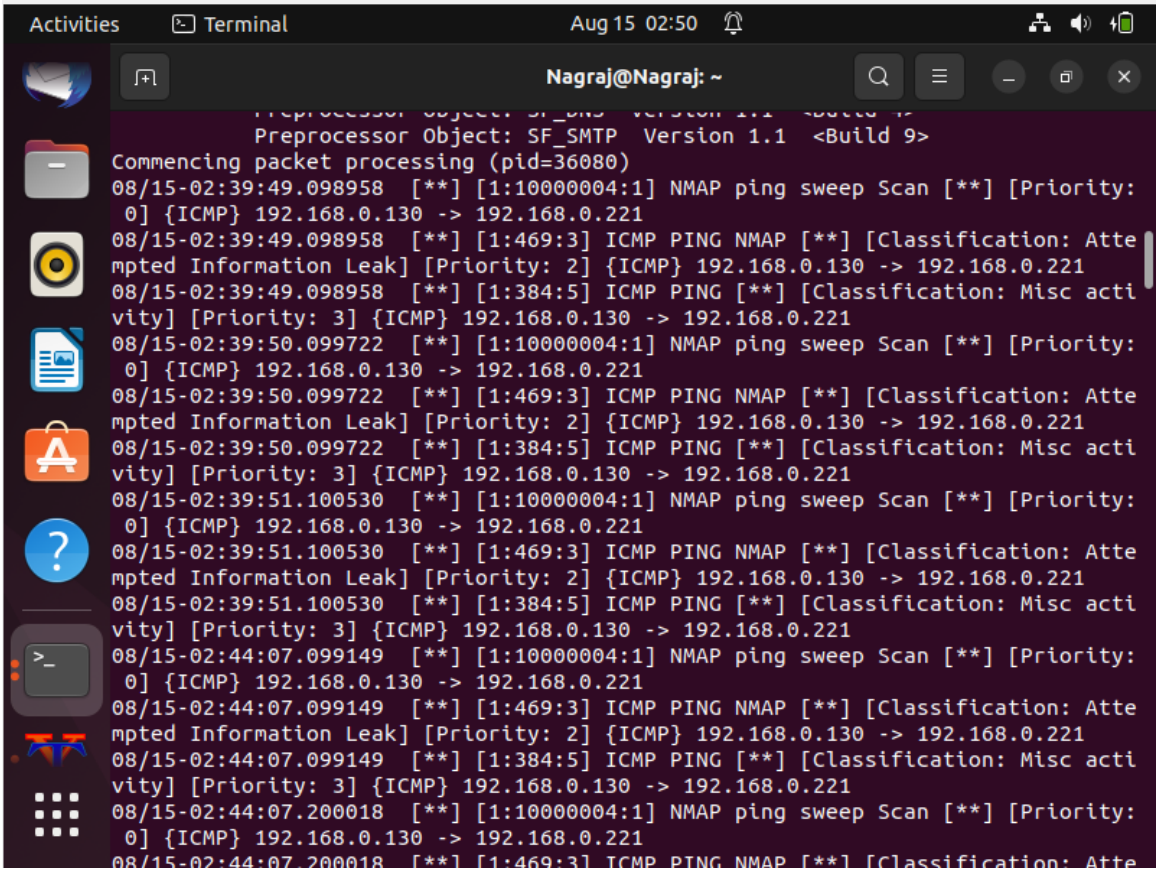
- Simulating DoS attack and replication a simple botnet attack: Getting alert for DOS Ping-Flood attack. For this, first using following command on attacker machine to send packets to victim machine:

  sudo hping3 -1 –fast 192.168.0,221

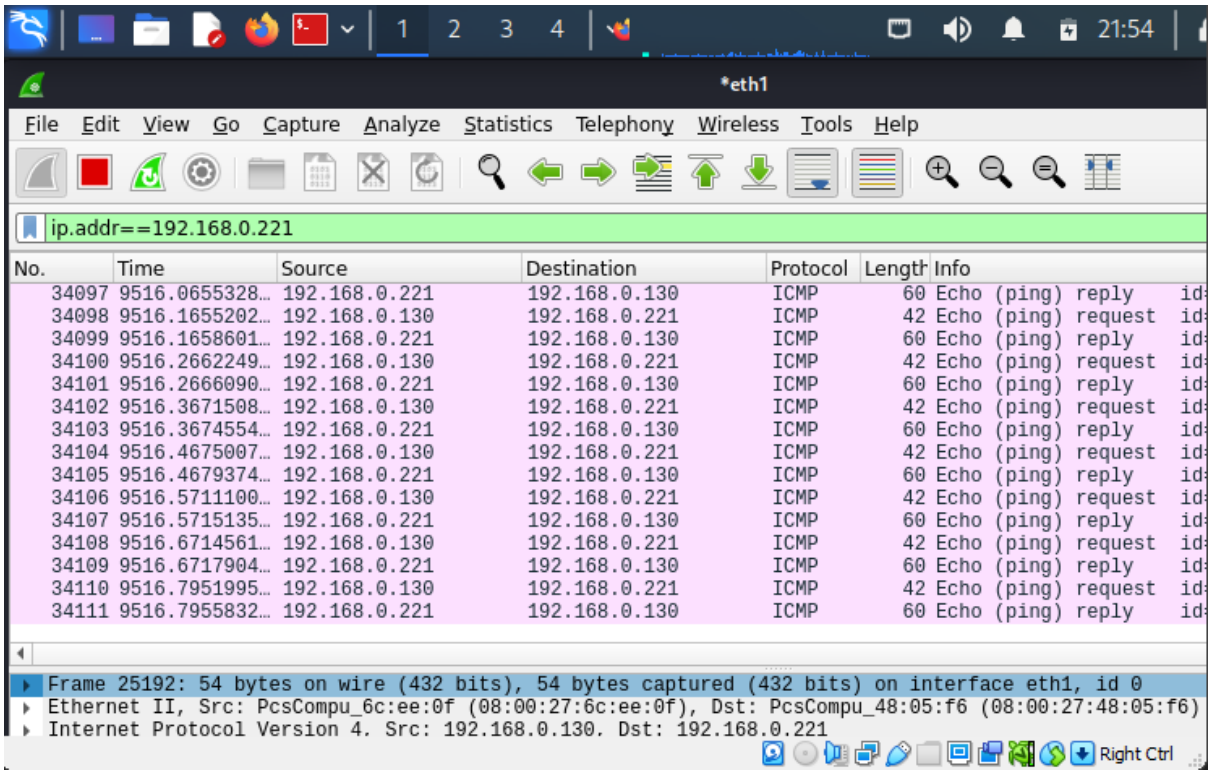

1 2 3 4
21:49
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ sudo hping3 -1 --fast 192.168.0.221
HPING 192.168.0.221 (eth1 192.168.0.221): icmp mode set, 28 headers + 0 data bytes
len=46 ip=192.168.0.221 ttl=64 id=13366 icmp_seq=0 rtt=4.7 ms
len=46 ip=192.168.0.221 ttl=64 id=13379 icmp_seq=1 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


1 2 3 4
kali@kali: ~
File Actions Edit View Help
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



- Using Wireshark for capturing and observing the packet transfer:

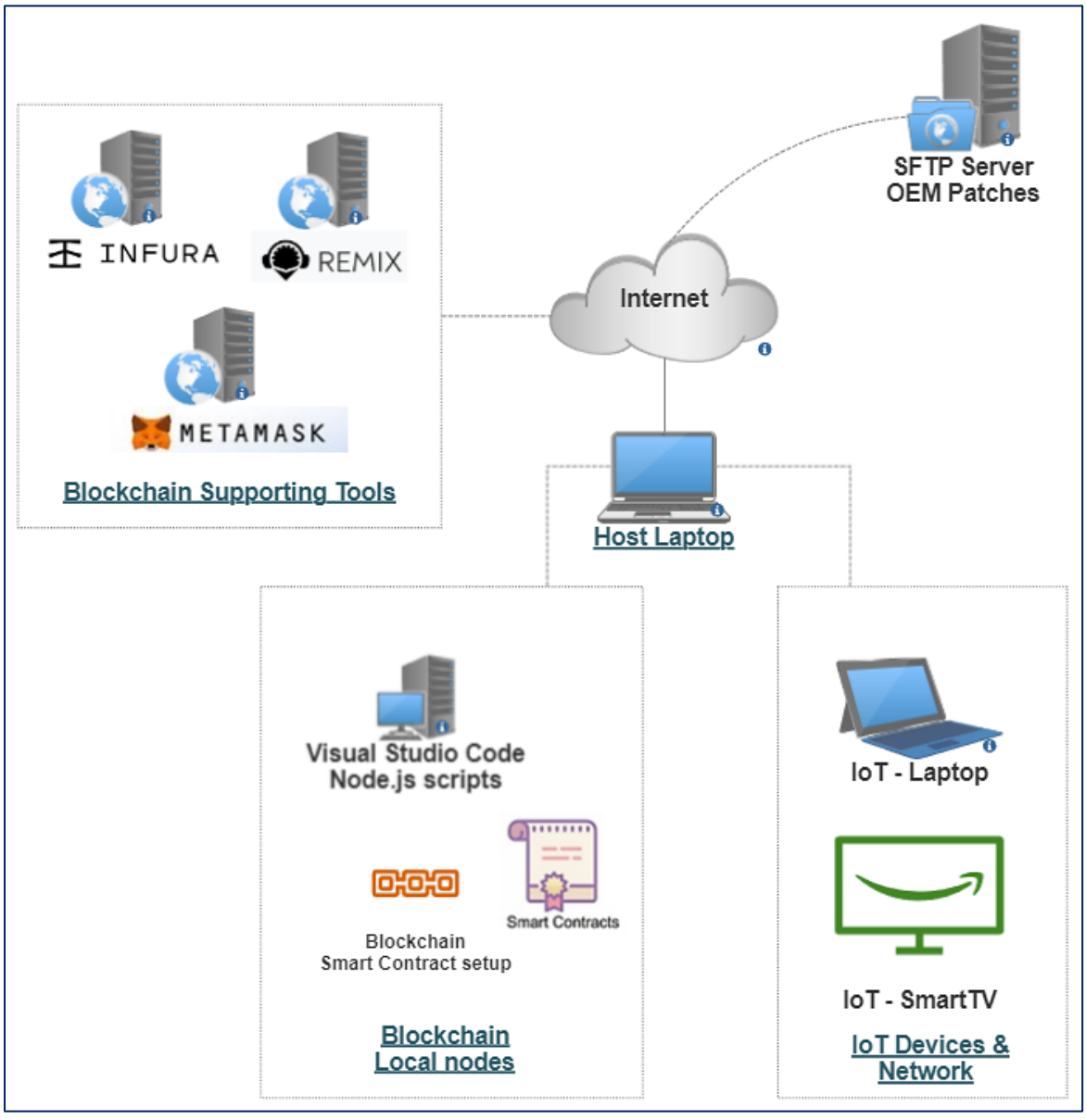


Blockchain Implementation Steps:

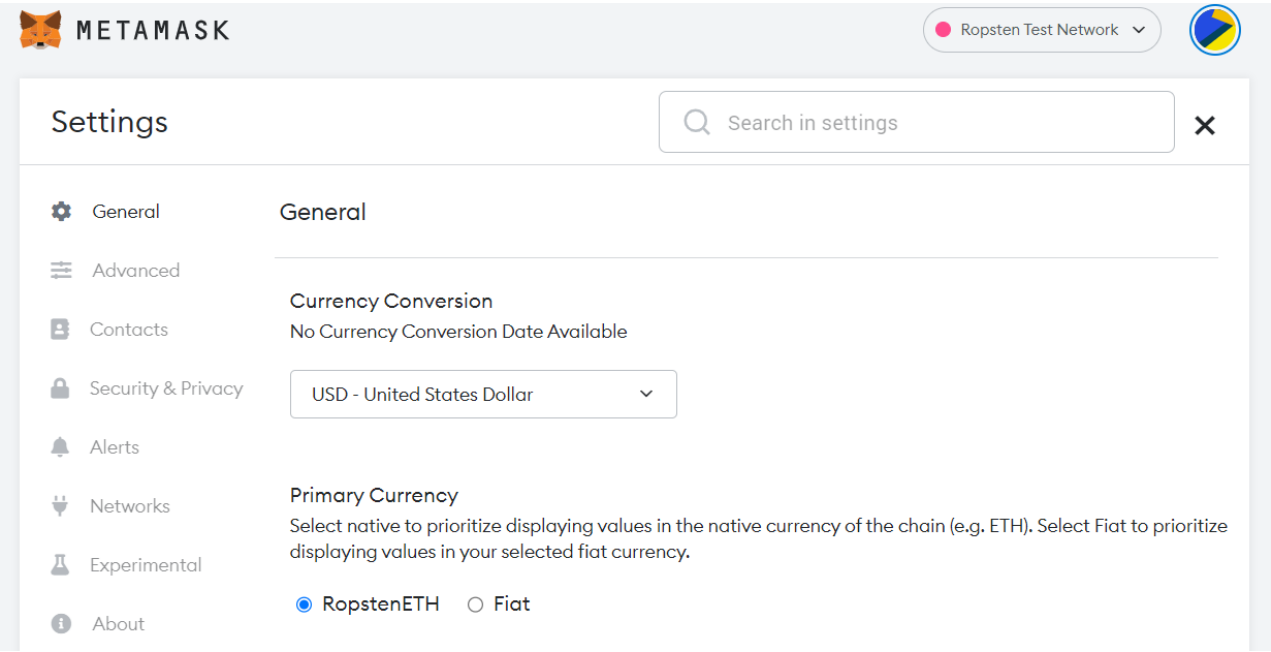
Below are the steps used for implementation of Blockchain setup for this Project, this will be followed with some relevant screenshots and inputs:

- 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

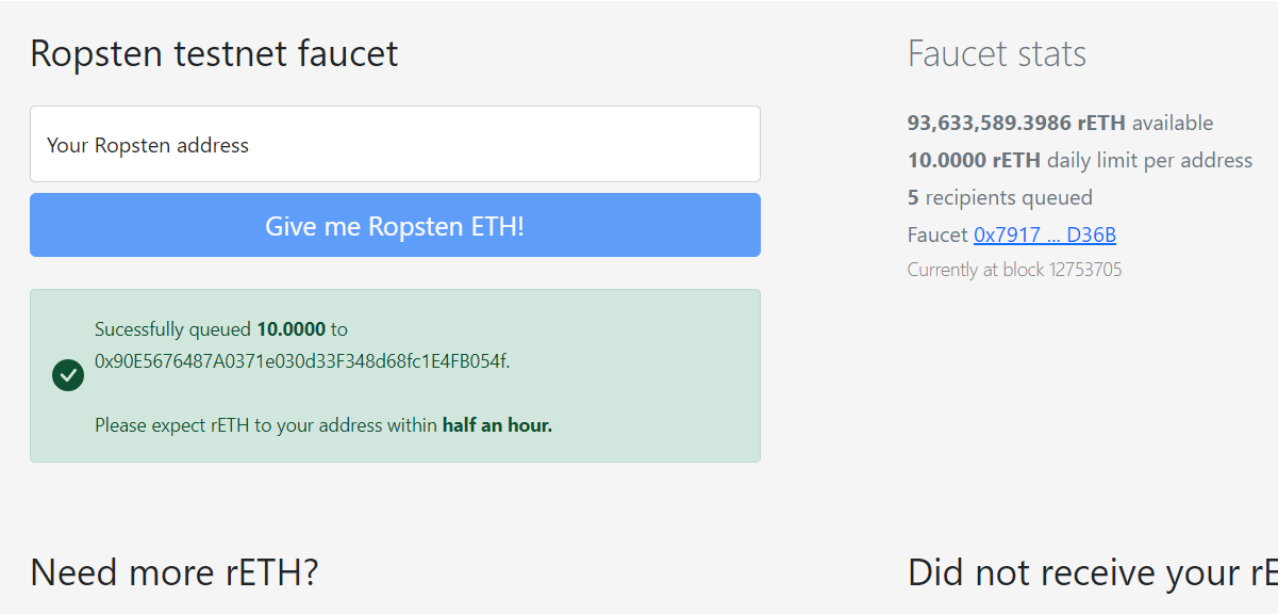
Blockchain setup architecture diagram:



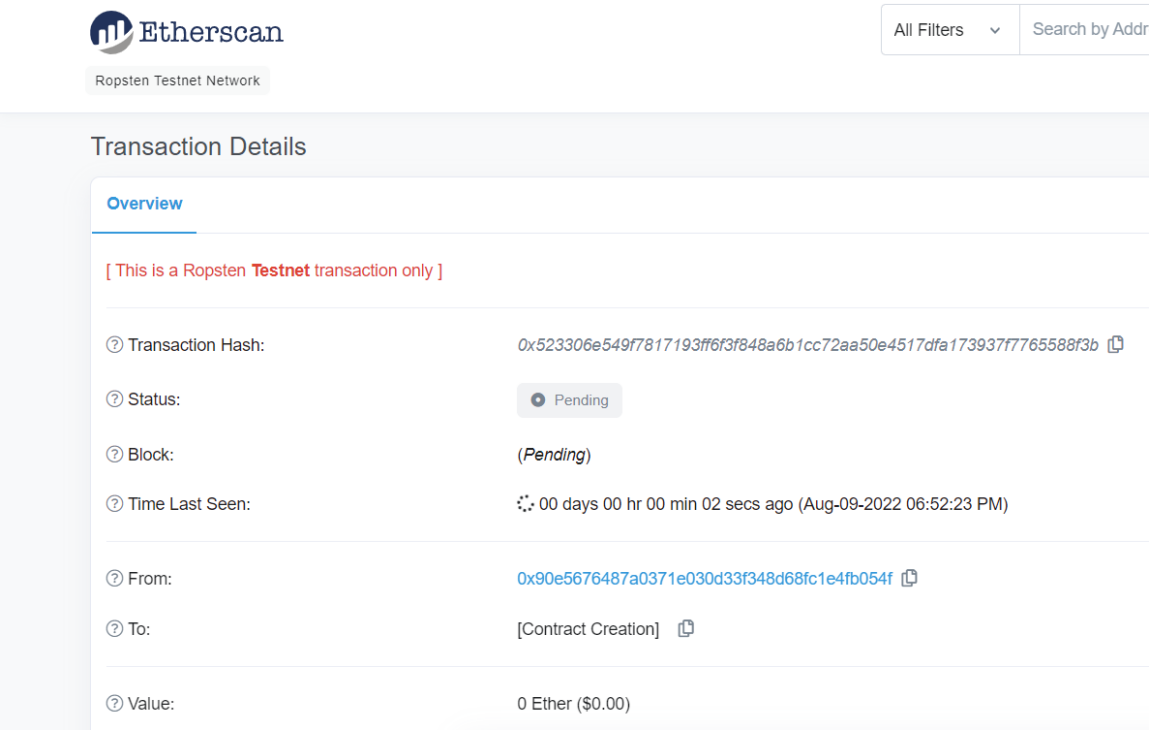
1. Creation of Metamask account and selecting Ropsten network



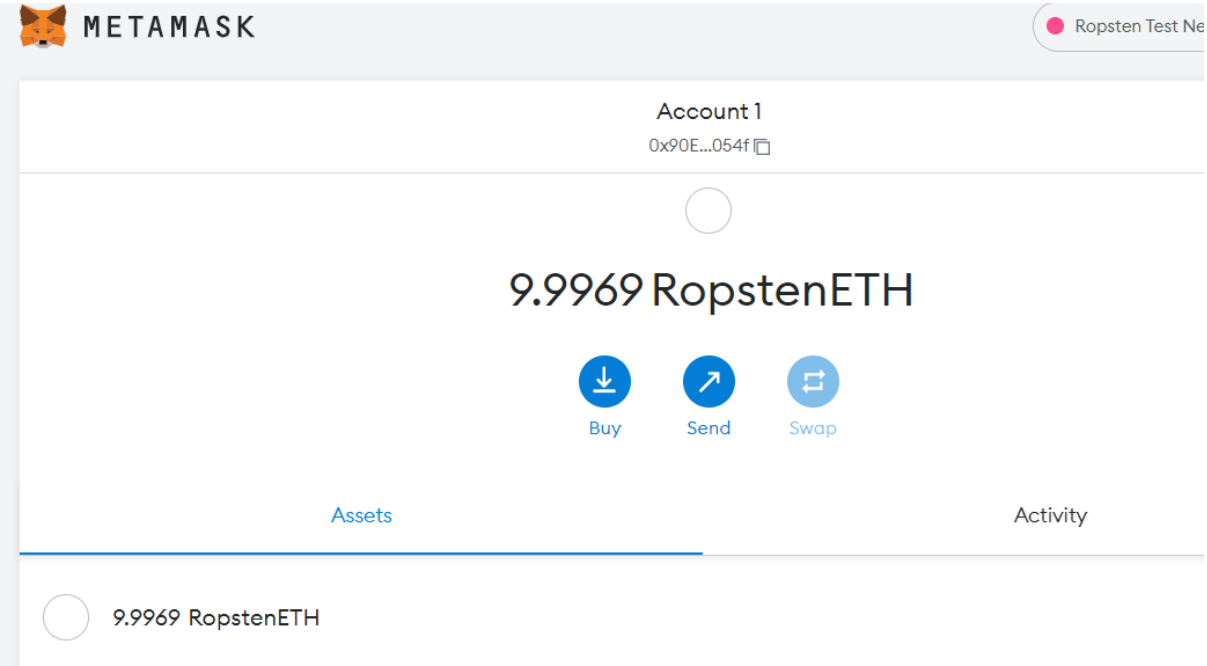
2. Adding Free Faucet Tokens – required to execute smart contracts and simulate tests.



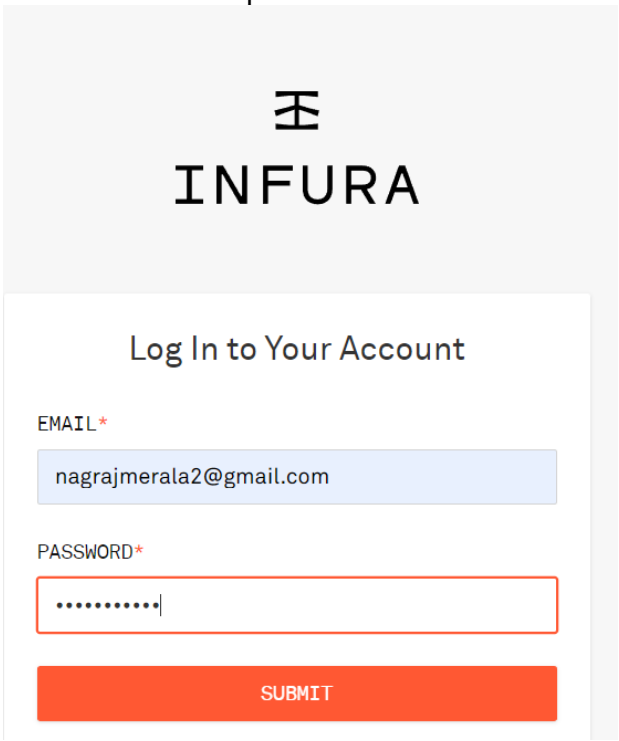
3. Etherscan for validation



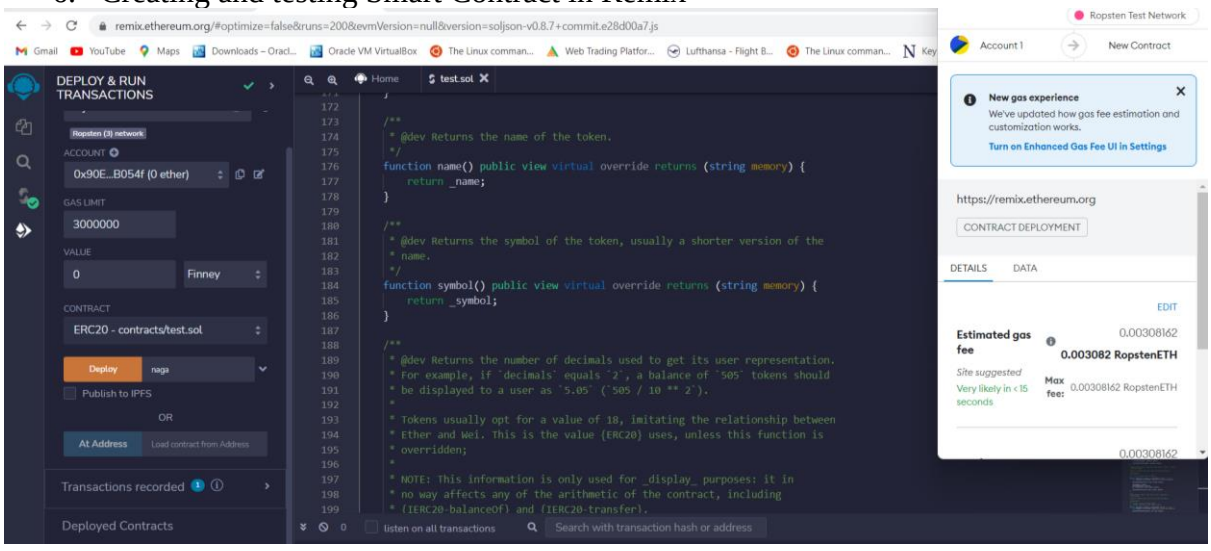
4. Receipt of Tokens on Metamask



5. Creation of INFURA account - required for automation



6. Creating and testing Smart Contract in Remix

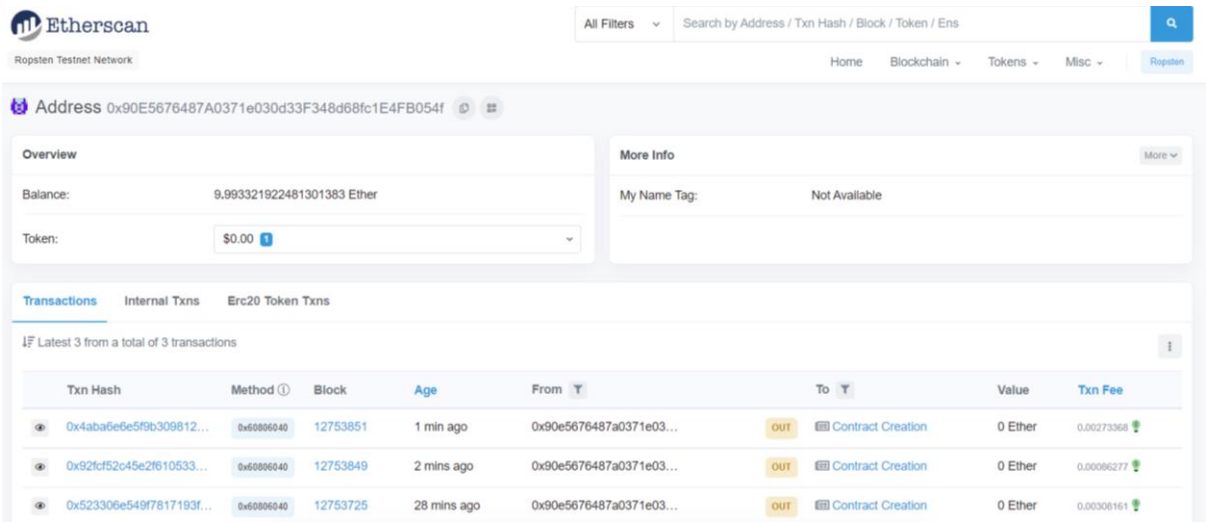


7. Using Node.js for automated script on Smart Contract creation

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:



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.