

“AUTOMATED INTRUSION DETECTION SYSTEM USING SNORT AND DISCORD API”

Review Report

Submitted by:

Prason Poudel 19BCE2550

Sashank Rijal 19BCE2484

Anubhav Bhandary 19BCE2483

Mickey Kumar Rouniyar 19BCE2520

Prepared for:

CSE3502 - Information Security Management

Submitted to:

Prof. Vimala Devi k

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Contents

S.No	Topic	Page No
1.	Abstract	3
2.	Aim	3
3.	Snort IDS	4
4.	Rule based detection	5
5.	Demo Design	5
6.	Methodology	6 - 14
7.	Implementation & Results	15- 19
8.	Audit Report	20 - 26

ABSTRACT

Our lives have been recorded into digital clouds and stored on hard drives since the digital revolution took hold. In order to ensure privacy and security, protecting this data has become a top responsibility. One of the most pressing issues we face today is computer security. Furthermore, if the situation involves a networked computer, the problem becomes even worse. It's become critical to keep intruders from gaining access to our data over the internet.

In this project, we deal with various terms, strategies, and procedures connected to the Intrusion Detection and Prevention System (IDPS). There are numerous approaches to implementing IDPS that are based on a thorough examination of diverse research endeavors. We focus on the concept of an Intrusion Detection System (IDS) utilizing Snort, a famous network security tool. It is commonly regarded by corporate sectors as a means of securing their network. The document provides a thorough understanding of Snort, including its objective, associated modes, implementations, and applications.

AIM

- To detect threats and notify the user about the intrusion and apply the appropriate rules for it.
- To detect malicious traffic and attacks against a network
- To act as a quality control operation for security design and management of vulnerable system
- To give important information regarding actual intrusions, enabling for better detection, improvement, and repair of contributory causes.

SNORT

- Snort can be successfully deployed on any network environment.
- Snort can run on various operating systems including Linux, Windows, and Mac OS X.
- Snort can deliver real-time network traffic event information.
- There are thousands of ways that Snort can be deployed.
- Snort sensors are modular and can monitor multiple machines from one physical and logical location.

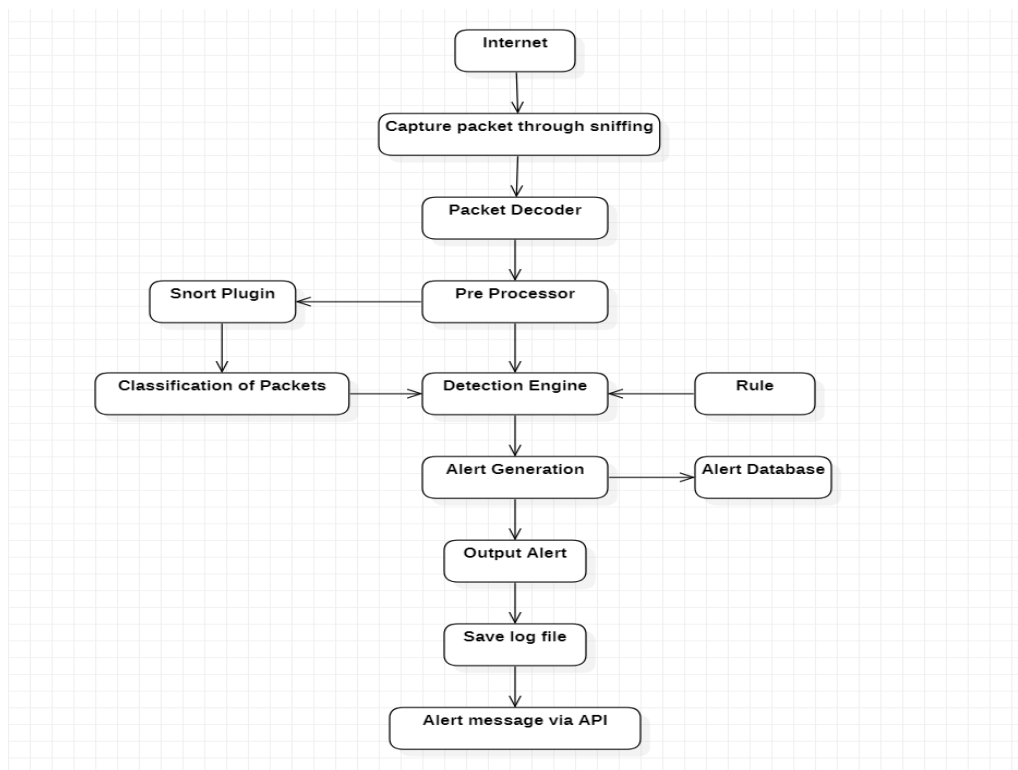
ADVANTAGES OF SNORT:

- **Scalability:** Snort can be successfully deployed on any network environment.
- **Flexibility and Usability:** Snort can run on various operating systems including Linux, Windows, and Mac OS X.
- **Live and Real-Time:** Snort can deliver real-time network traffic event information.
- **Flexibility in Deployment:** There are thousands of ways that Snort can be deployed and a myriad of databases, logging systems, and tools with which it can work.
- **Speed in Detecting and Responding to Security Threats:** Used in conjunction with a firewall and other layers of security infrastructure, Snort helps organizations detect and respond to system crackers, worms, network vulnerabilities, security threats, and policy abusers that aim to take down network and computer systems.
- **Modular Detection Engine:** Snort sensors are modular and can monitor multiple machines from one physical and logical location. Snort be placed in front of the firewall, behind the firewall, next to the firewall, and everywhere else to monitor an entire network. As a result, organizations use Snort as a security solution to find out if there are unauthorized attempts to hack in the network or if a hacker has gained unauthorized access into the network system.

RULE BASED DETECTION:

- Rule Based Detection uses a set of predefined rules to identify an intruder or attack.
- This category can be divided further into two subcategories; Anomaly detection and Penetration identification.
- Anomaly detection uses rules that are produced by looking at previous attack patterns or signatures of malicious traffic.
- Penetration identification uses an expert system containing rules written by security experts, that are used when searching for suspicious behavior in a network or on a host system

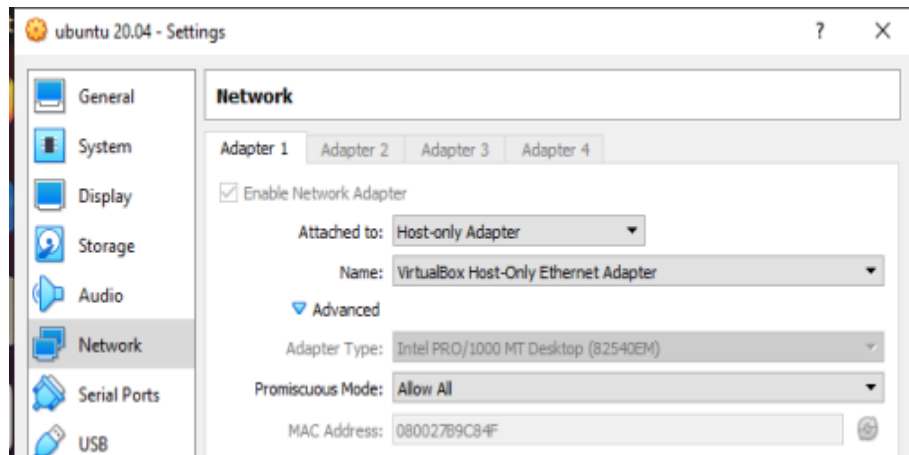
DEMO DESIGN:



PROPOSED WORK / METHODOLOGY:

1. SETTING UP THE VIRTUAL BOX ENVIRONMENT FOR SNORT

- select the option of Host-only-Adapter available under the Attached to: option.
- After this setting is selected, go to the advanced option on the same box and under promiscuous mode and select Allow all option.



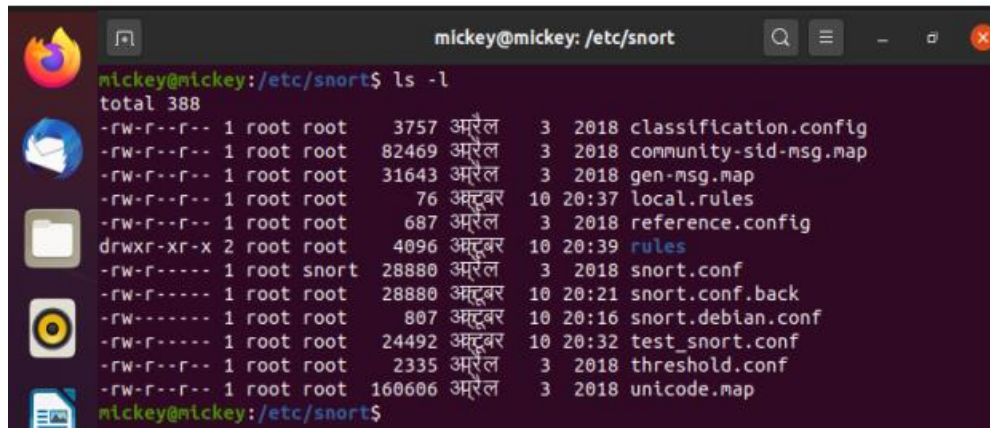
2. DOWNLOADING AND INSTALLING THE SNORT SYSTEM:

- To download and install snort, just execute the following command.
 - `sudo apt-get install snort`
- After this, the system will ask for the interface for which snort system should listen to.
- Just type the interface we selected before this step under “ip addr” option. For this case, we select `enp0s3` and then press tab and then enter.

```
Apr 23 20:31 •
prason@ubuntu: ~
prason@ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:a7:10:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
        valid_lft 595sec preferred_lft 595sec
    inet6 fe80::ff80:2e60:975c:b375/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
prason@ubuntu:~$
```

3. CREATING A BACKUP FILE FOR SNORT AND FUTURE PURPOSE:

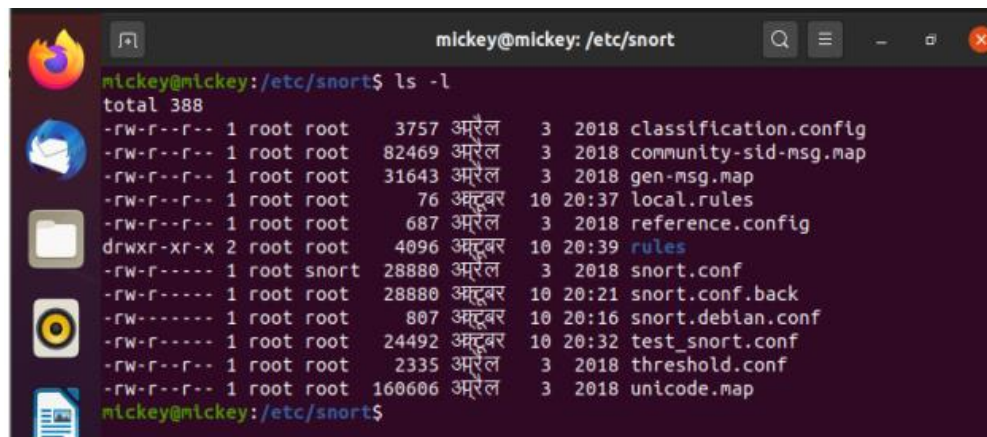
- This step is optional but might result as a very important decision in future purpose. To create a backup
- file just execute the command under the snort directory “cd /etc/snort/” and then execute these
 - \$ cp snort.conf snort.conf. back
 - \$ sudo cp snort.conf snort.conf. back



```
mickey@mickey: /etc/snort
mickey@mickey:/etc/snort$ ls -l
total 388
-rw-r--r-- 1 root root 3757 अप्रैल 3 2018 classification.config
-rw-r--r-- 1 root root 82469 अप्रैल 3 2018 community-sid-msg.map
-rw-r--r-- 1 root root 31643 अप्रैल 3 2018 gen-msg.map
-rw-r--r-- 1 root root 76 अक्टूबर 10 20:37 local.rules
-rw-r--r-- 1 root root 687 अप्रैल 3 2018 reference.config
drwxr-xr-x 2 root root 4096 अक्टूबर 10 20:39 rules
-rw-r--r-- 1 root snort 28880 अप्रैल 3 2018 snort.conf
-rw-r--r-- 1 root root 28880 अक्टूबर 10 20:21 snort.conf.back
-rw-r--r-- 1 root root 807 अक्टूबर 10 20:16 snort.debian.conf
-rw-r--r-- 1 root root 24492 अक्टूबर 10 20:32 test_snort.conf
-rw-r--r-- 1 root root 2335 अप्रैल 3 2018 threshold.conf
-rw-r--r-- 1 root root 160606 अप्रैल 3 2018 unicode.map
mickey@mickey:/etc/snort$
```

4. MAKING A TESTING FILE TO WORK WITH RULES AND TESTING THEM:

- To create a test configuration file, just execute the following command:
 - \$ sudo cp snort.conf test_snort.conf

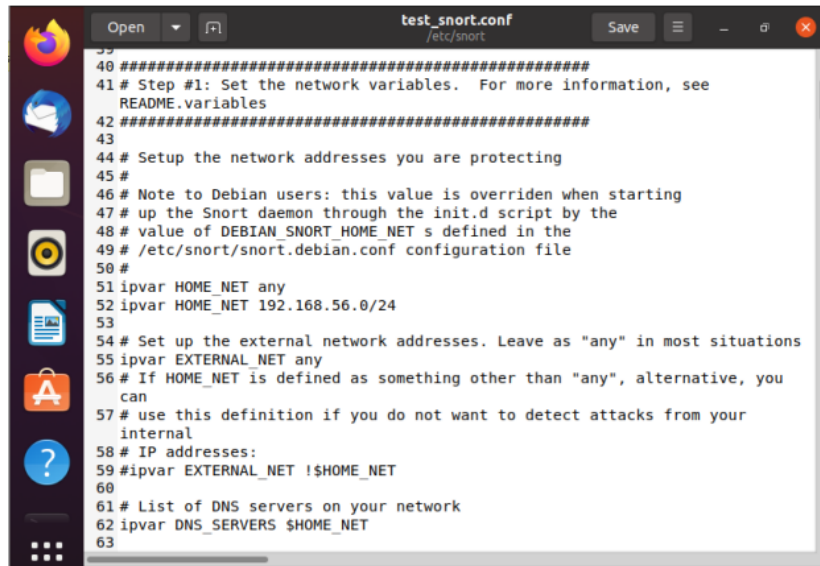


```
mickey@mickey: /etc/snort
mickey@mickey:/etc/snort$ ls -l
total 388
-rw-r--r-- 1 root root 3757 अप्रैल 3 2018 classification.config
-rw-r--r-- 1 root root 82469 अप्रैल 3 2018 community-sid-msg.map
-rw-r--r-- 1 root root 31643 अप्रैल 3 2018 gen-msg.map
-rw-r--r-- 1 root root 76 अक्टूबर 10 20:37 local.rules
-rw-r--r-- 1 root root 687 अप्रैल 3 2018 reference.config
drwxr-xr-x 2 root root 4096 अक्टूबर 10 20:39 rules
-rw-r--r-- 1 root snort 28880 अप्रैल 3 2018 snort.conf
-rw-r--r-- 1 root root 28880 अक्टूबर 10 20:21 snort.conf.back
-rw-r--r-- 1 root root 807 अक्टूबर 10 20:16 snort.debian.conf
-rw-r--r-- 1 root root 24492 अक्टूबर 10 20:32 test_snort.conf
-rw-r--r-- 1 root root 2335 अप्रैल 3 2018 threshold.conf
-rw-r--r-- 1 root root 160606 अप्रैल 3 2018 unicode.map
mickey@mickey:/etc/snort$
```

5. SETTING UP THE HOME NETWORK:

- Here, we first set the configuration for our home network.
- To do this, we first open the test_snort.conf file by following command:
 - `$ sudo gedit test_snort.conf`

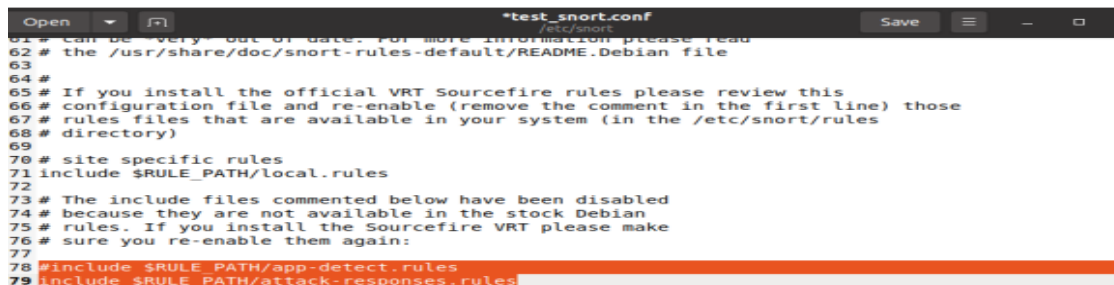
Since, for this purpose, we have an IP of 192.168.56.101/24, our home network's IP becomes 192.168.56.0/24. After knowing the IP address, just add this line: `ipvar HOME_NET 192.168.56.0/24`

A screenshot of a text editor window titled 'test_snort.conf' with the path '/etc/snort' shown below the title. The editor contains configuration text for Snort. Line 52 has been modified to 'ipvar HOME_NET 192.168.56.0/24'. The text includes comments about setting network variables, setup of network addresses, and definitions for HOME_NET, EXTERNAL_NET, and DNS_SERVERS.

```
40 #####
41 # Step #1: Set the network variables. For more information, see
  README.variables
42 #####
43
44 # Setup the network addresses you are protecting
45 #
46 # Note to Debian users: this value is overridden when starting
47 # up the Snort daemon through the init.d script by the
48 # value of DEBIAN_SNORT_HOME_NET s defined in the
49 # /etc/snort/snort.debian.conf configuration file
50 #
51 ipvar HOME_NET any
52 ipvar HOME_NET 192.168.56.0/24
53
54 # Set up the external network addresses. Leave as "any" in most situations
55 ipvar EXTERNAL_NET any
56 # If HOME_NET is defined as something other than "any", alternative, you
  can
57 # use this definition if you do not want to detect attacks from your
  internal
58 # IP addresses:
59 #ipvar EXTERNAL_NET !$HOME_NET
60
61 # List of DNS servers on your network
62 ipvar DNS_SERVERS $HOME_NET
63
```

6. REMOVING THE DEFINED RULES ALREADY PRESENT IN THE SNORT FOR ADDING OUR NEW RULES:

- For this, open the same file in the previous step where we added our own network.
 - `$ sudo gedit test_snort.conf`
- After this, go in line 579 where you can see a statement starting with a rule of
 - `#include $RULE_PATH/app-detect.rules`

A screenshot of a text editor window titled '*test_snort.conf' with the path '/etc/snort' shown below the title. The editor shows the bottom part of the configuration file, specifically the 'include' statements. Lines 78 and 79 are highlighted in orange. Line 78 is '#include \$RULE_PATH/app-detect.rules' and line 79 is '#include \$RULE_PATH/attack-responses.rules'.

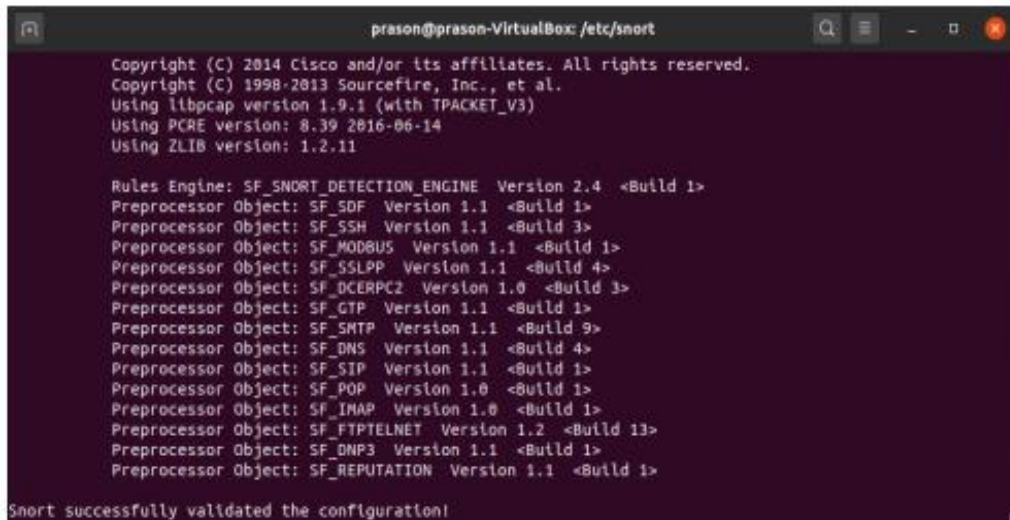
```
62 # the /usr/share/doc/snort-rules-default/README.Debian file
63
64 #
65 # If you install the official VRT Sourcefire rules please review this
66 # configuration file and re-enable (remove the comment in the first line) those
67 # rules files that are available in your system (in the /etc/snort/rules
68 # directory)
69
70 # site specific rules
71 include $RULE_PATH/local.rules
72
73 # The include files commented below have been disabled
74 # because they are not available in the stock Debian
75 # rules. If you install the Sourcefire VRT please make
76 # sure you re-enable them again:
77
78 #include $RULE_PATH/app-detect.rules
79 #include $RULE_PATH/attack-responses.rules

```


7. CONFIGURING THE SYSTEM AFTER REMOVING ALL THE PREDEFINED RULES:

After removing all the rules, we need to configure the system after removing the rules. For this, run this command

- `$ sudo snort -T -i enp0s3 -c /etc/snort/test_snort.conf`

A terminal window titled 'prason@prason-VirtualBox: /etc/snort' showing the output of the 'snort -T' command. The output lists copyright information, library versions (libpcap 1.9.1, PCRE 8.39, ZLIB 1.2.11), and a detailed list of preprocessor objects and their versions. At the bottom, it states 'Snort successfully validated the configuration!'.

```
prason@prason-VirtualBox: /etc/snort
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>

Snort successfully validated the configuration!
```

8. WRITING OUR CUSTOM RULES:

- Since all the rules are written in the custom rules section, we have to first go to the directory and open
- the local.rules named file. To go to the rule's directory, write the given command:

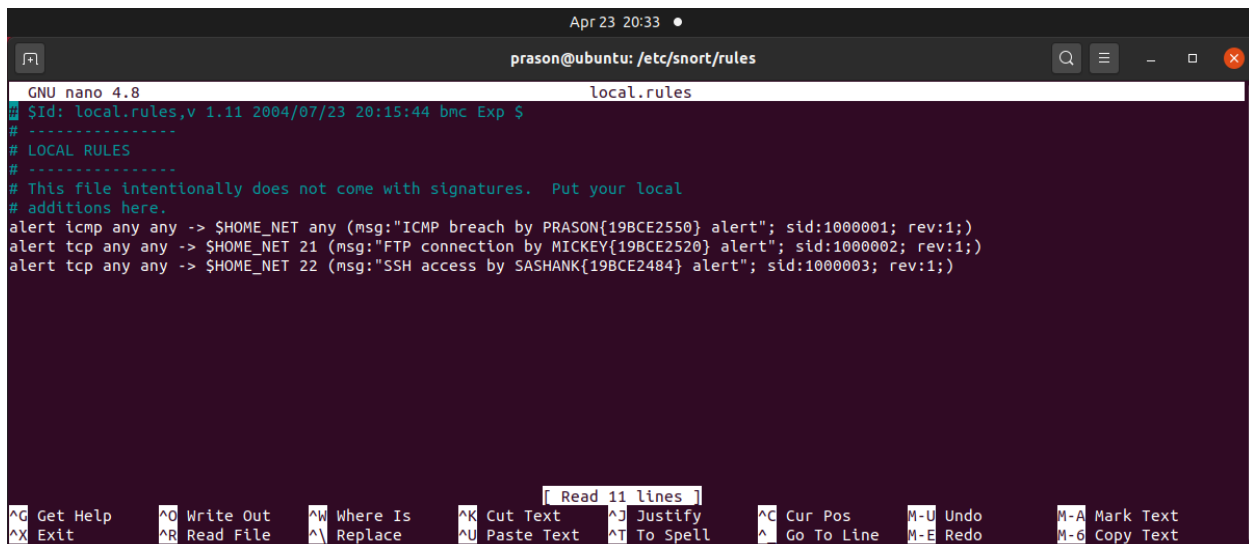
- `$ cd /etc/snort/rules`

- After getting to the required path, open the local.rules files to add any sort of manual or custom rules.
- To open the local.rules, type the following command:

- `$ sudo nano local.rules`

- One example of custom rule is:

```
alert icmp any any -> $HOME_NET any ( msg:"MICKEY ALERT"; sid: 1000001; rev:1;)
```

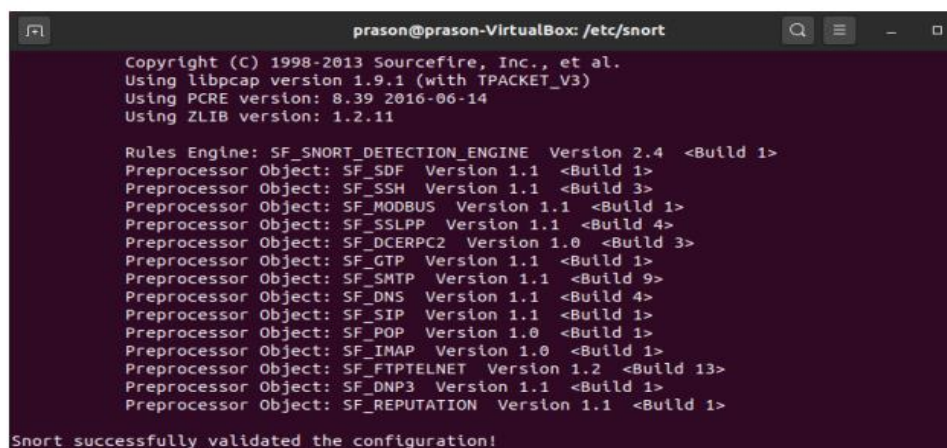


```
prason@ubuntu: /etc/snort/rules
GNU nano 4.8 local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> $HOME_NET any (msg:"ICMP breach by PRASON{19BCE2550} alert"; sid:1000001; rev:1;)
alert tcp any any -> $HOME_NET 21 (msg:"FTP connection by MICKEY{19BCE2520} alert"; sid:1000002; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH access by SASHANK{19BCE2484} alert"; sid:1000003; rev:1;)

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo        M-A Mark Text
^X Exit          ^R Read File    ^_ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line   M-E Redo        M-G Copy Text
```

9. CONFIGURING THE SYSTEM AFTER ADDING THE CUSTOM RULE.

- After writing the custom rule, we need to configure the snort system for the rules that are added. For this, go to the path
 - \$ cd /etc/snort/
- And then run the following command:
 - \$ sudo snort -T -i enp0s3 -c /etc/snort/test_snort.conf
- After running this, it takes some time to configure the system. After configuring, you might see the message like



```
prason@prason-VirtualBox: /etc/snort
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>

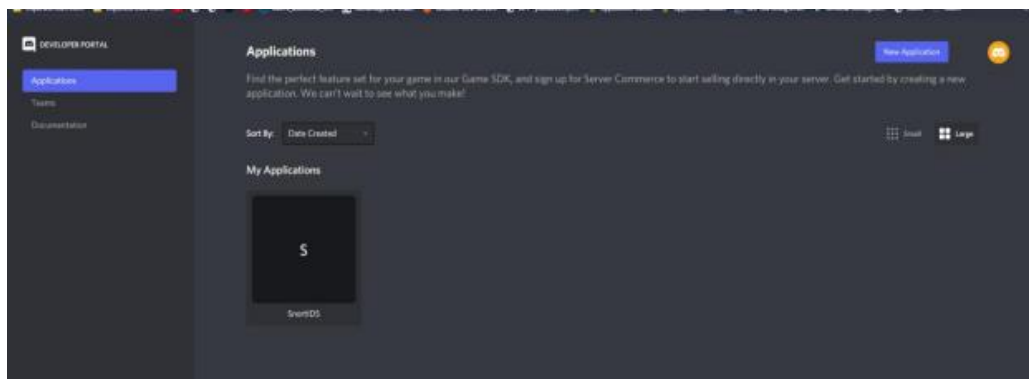
Snort successfully validated the configuration!
```

10. ENABLING THE SNORT SYSTEM TO SNIFF FOR ANY SORT OF INTRUSIONS:

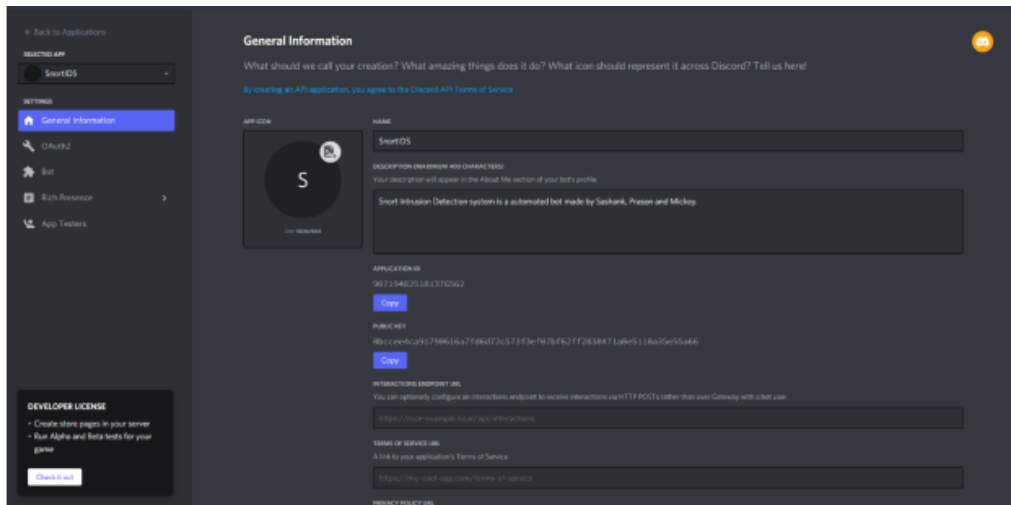
- Now, after the written rules are configured, we can enable the snort system for detecting any sort of intrusions.
- For this, run the command
 - `$ sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf`
- Description of the command:
 - `-c`: specifies the config file
 - `-i` : specifies the interface mode , if a loopback address is running then “lo” will be written , for Ethernet
- “eth0” or “eth1” will be written. `-A`: It will print the output to the console
- We can enable the snort using two modes. One is the console mode and other is the fast mode. Console mode is used for printing and displaying any alerts in the terminal itself. This can be used when the user is available in the system.
- The other mode is the fast mode. For this, we can use the command:
 - `$ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf`

11. SETTING UP THE DISCORD DEVELOPERS ACCOUNT AND CREATING THE DISCORD BOT:

- For interacting the snort system with the discord, we first need to create a developers account in the discord and then make a discord bot application for it.

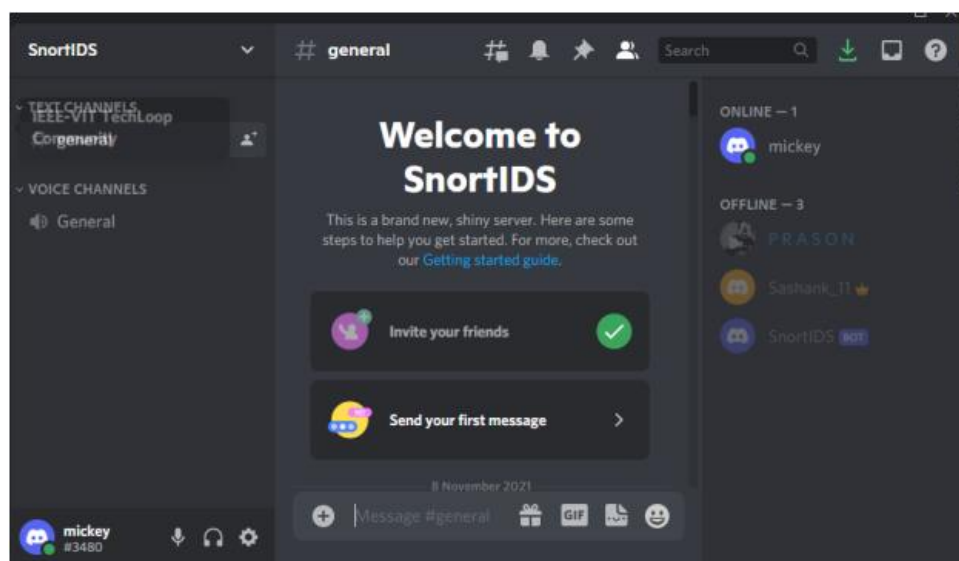


After creating the developer's portal, you can make a discord bot:



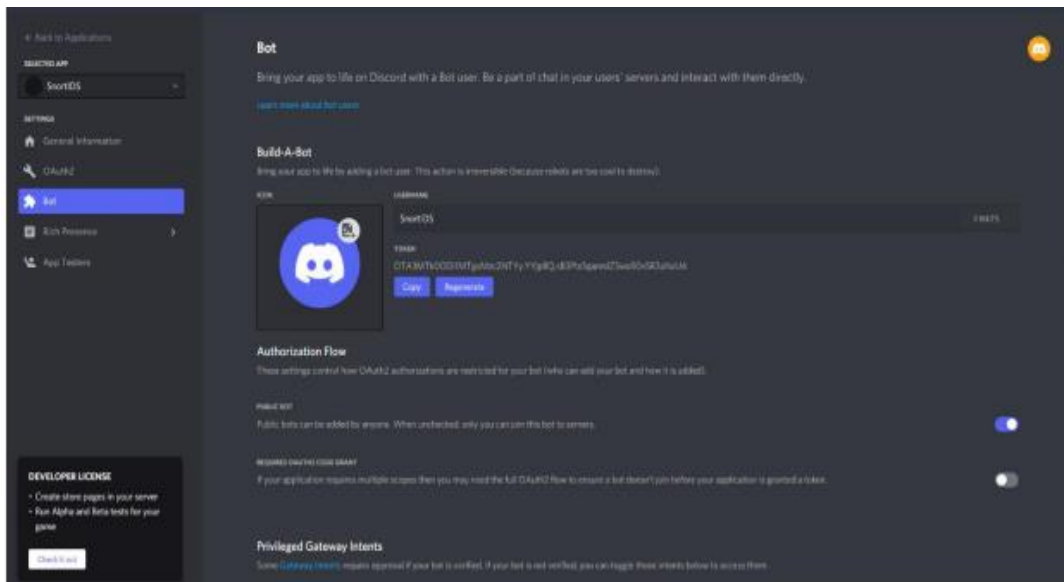
12. SETTING UP THE DISCORD BOT AND ADDING IT TO OUR OWN SERVER:

- After creating the discord bot, we need a server to deploy it so that the user can check his system from anywhere. For this, after creating the discord bot, create a new server with any name in the discord application and add the bot to it along with other members associated with the project. For this, the members for our server are:
 - ID: mickey (MICKEY KUMAR ROUNIYAR)
 - ID: P R A S O N (PRASON POUDEL)
 - ID: Sashank_11 (SASHANK RIJAL)
 - ID: Xperience (ANUBHAV BHANDARY)
- The discord bot named “SnortIDS” will be currently in offline status as it is not in used.

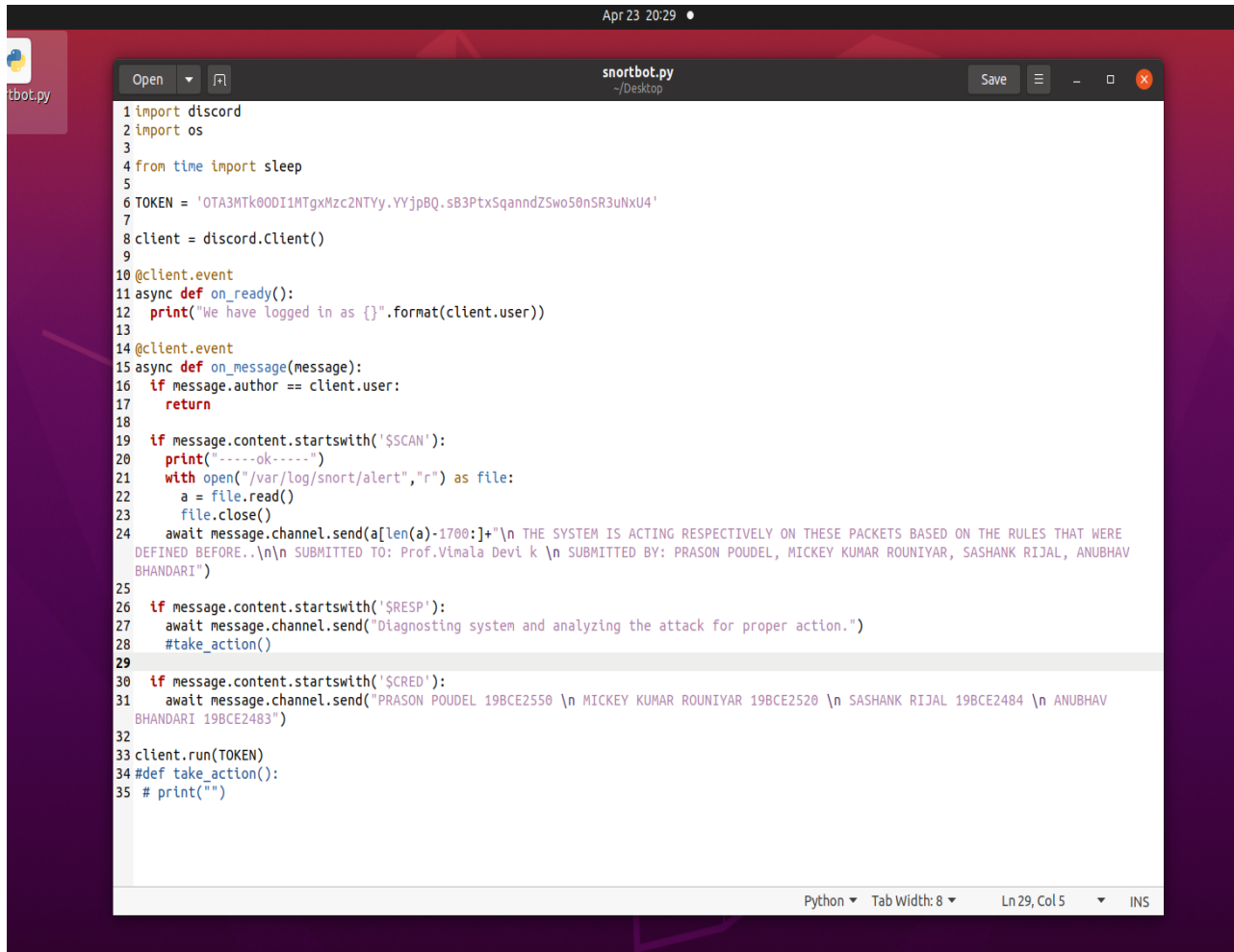


13. WRITING THE PYTHON SCRIPT FOR THE CONNECTION OF OUR SYSTEM WITH THE DISCORD API:

- After the server is created and all the members along with the discord bot is added to it, we then need
- to make connection between our snort system and the discord. For this, we will be writing a python code.
- Before writing the python code, we need to know the token number of our bot for interaction.
- From here, copy the token number which will be used further in the script of the python.
 - TOKEN NUMBER:
OTA3MTk0ODI1MTgxMzc2NTYy.YYjpBQ.sB3PtxSqanndZSwo50nSR3uNxU4
- We are using python here as python is pre-installed in the Linux environment. Apart from python, we
- need to download some other libraries before running the script. The additional libraries can be
- downloaded by running the following command like:
 - \$ sudo apt install python3-pip3
 - \$ sudo pip3 install discord.py



- After this, write the following code and save it as file in the desktop or other known location with the extension .py (python file).



The screenshot shows a code editor window titled 'snortbot.py' with a file icon and a 'Save' button. The code is a Python script for a Discord bot named 'snortbot.py'. It imports 'discord' and 'os', and uses 'time.sleep' for delays. A Discord token is defined, and a client is created. The bot has two main event handlers: 'on_ready' which prints a login message, and 'on_message' which checks for specific keywords in messages. If a message starts with '\$SCAN', it reads a file, processes its content, and sends a detailed alert message to a Discord channel. If a message starts with '\$RESP', it sends a diagnostic message. If a message starts with '\$CRED', it sends a list of credentials. The script also includes a 'run' method for the client and a 'take_action' function that is currently commented out.

```

1 import discord
2 import os
3
4 from time import sleep
5
6 TOKEN = 'OTA3MTk0ODI1MTgxMzc2NTYy.YYjpBQ.SB3PtXsqandZSwo50nSR3uNxU4'
7
8 client = discord.Client()
9
10 @client.event
11 async def on_ready():
12     print("We have logged in as {}".format(client.user))
13
14 @client.event
15 async def on_message(message):
16     if message.author == client.user:
17         return
18
19     if message.content.startswith('$SCAN'):
20         print("-----ok-----")
21         with open("/var/log/snort/alert", "r") as file:
22             a = file.read()
23             file.close()
24         await message.channel.send(a[len(a)-1700:]+ "\n THE SYSTEM IS ACTING RESPECTIVELY ON THESE PACKETS BASED ON THE RULES THAT WERE
DEFINED BEFORE..\n\n SUBMITTED TO: Prof.Vimala Devi k \n SUBMITTED BY: PRASON POUDEL, MICKEY KUMAR ROUNIYAR, SASHANK RIJAL, ANUBHAV
BHANDARI")
25
26     if message.content.startswith('$RESP'):
27         await message.channel.send("Diagnostics system and analyzing the attack for proper action.")
28         #take_action()
29
30     if message.content.startswith('$CRED'):
31         await message.channel.send("PRASON POUDEL 19BCE2550 \n MICKEY KUMAR ROUNIYAR 19BCE2520 \n SASHANK RIJAL 19BCE2484 \n ANUBHAV
BHANDARI 19BCE2483")
32
33 client.run(TOKEN)
34 #def take_action():
35 # print("")

```

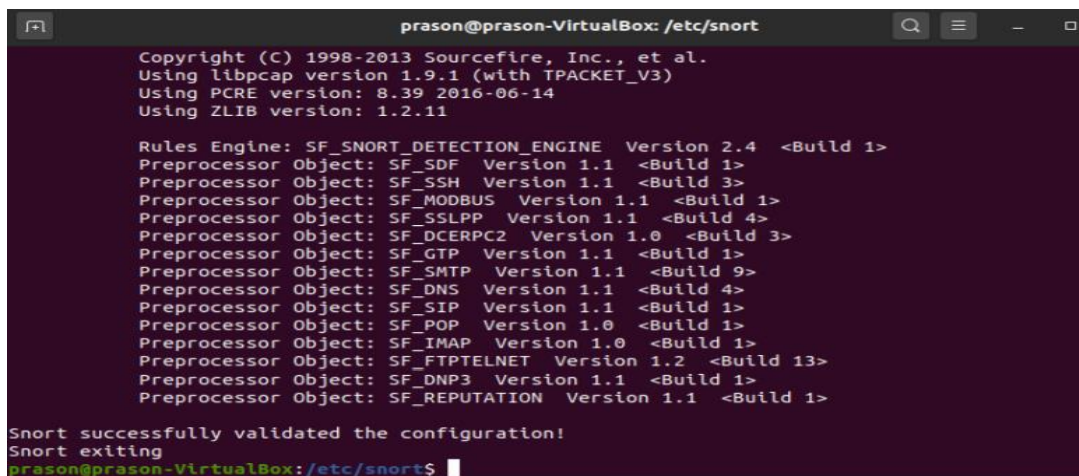
Python Tab Width: 8 Ln 29, Col 5 INS

RESULTS:

- Detects all kinds of intrusion like ICMP FTP, SSH and generates alert for specific kind on intrusion
- It logs it to the input packet into a snort.config file.
- Saves the log file into the system and sends the alert message to the user using the communication medium to discord bot API
- Moderates and processes the intrusion and takes specific action based on the predefined rules or the rules (icmp rules, ftp rules) written by the user.

At first, we configured the rules in the terminal by using the command:

- `$ sudo snort -T -i enp0s3 -c /etc/snort/test_snort.conf`

A terminal window titled 'prason@prason-VirtualBox: /etc/snort' showing the output of the 'snort -T' command. The output lists various components and their versions, including libpcap, PCRE, ZLIB, and the Rules Engine. It also lists several Preprocessor Objects and their versions. At the bottom, it states 'Snort successfully validated the configuration!' and 'Snort exiting'.

```
prason@prason-VirtualBox: /etc/snort
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>

Snort successfully validated the configuration!
Snort exiting
prason@prason-VirtualBox: /etc/snort$
```

After the written rules are configured, we enabled the snort system for detecting any sort of intrusions in the console by running the command:

- `$ sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf`


```
Apr 23 20:35 •
prason@ubuntu: /etc/snort/rules

Snort successfully validated the configuration!
Snort exiting
prason@ubuntu:/etc/snort/rules$ sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf
04/23-20:34:32.188977  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
04/23-20:34:32.188992  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
04/23-20:34:33.191588  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
04/23-20:34:33.191625  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
04/23-20:34:34.206678  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
04/23-20:34:34.206703  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
04/23-20:34:35.211179  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
04/23-20:34:35.211192  [**] [1:1000001:1] ICMP breach by PRASON{19BCE2550} alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
04/23-20:34:38.836168  [**] [1:1000003:1] SSH access by SASHANK{19BCE2484} alert [**] [Priority: 0] {TCP} 192.168.56.1:60533 -> 192.168.56.101:22
04/23-20:34:39.337395  [**] [1:1000003:1] SSH access by SASHANK{19BCE2484} alert [**] [Priority: 0] {TCP} 192.168.56.1:60533 -> 192.168.56.101:22
04/23-20:34:39.838261  [**] [1:1000003:1] SSH access by SASHANK{19BCE2484} alert [**] [Priority: 0] {TCP} 192.168.56.1:60533 -> 192.168.56.101:22
04/23-20:34:40.339544  [**] [1:1000003:1] SSH access by SASHANK{19BCE2484} alert [**] [Priority: 0] {TCP} 192.168.56.1:60533 -> 192.168.56.101:22
04/23-20:34:40.840635  [**] [1:1000003:1] SSH access by SASHANK{19BCE2484} alert [**] [Priority: 0] {TCP} 192.168.56.1:60533 -> 192.168.56.101:22
04/23-20:34:58.850954  [**] [1:1000002:1] FTP connection by MICKEY{19BCE2520} alert [**] [Priority: 0] {TCP} 192.168.56.1:60535 -> 192.168.56.101:21
04/23-20:34:59.351992  [**] [1:1000002:1] FTP connection by MICKEY{19BCE2520} alert [**] [Priority: 0] {TCP} 192.168.56.1:60535 -> 192.168.56.101:21
04/23-20:34:59.853811  [**] [1:1000002:1] FTP connection by MICKEY{19BCE2520} alert [**] [Priority: 0] {TCP} 192.168.56.1:60535 -> 192.168.56.101:21
04/23-20:35:00.354617  [**] [1:1000002:1] FTP connection by MICKEY{19BCE2520} alert [**] [Priority: 0] {TCP} 192.168.56.1:60535 -> 192.168.56.101:21
04/23-20:35:00.855940  [**] [1:1000002:1] FTP connection by MICKEY{19BCE2520} alert [**] [Priority: 0] {TCP} 192.168.56.1:60535 -> 192.168.56.101:21
```

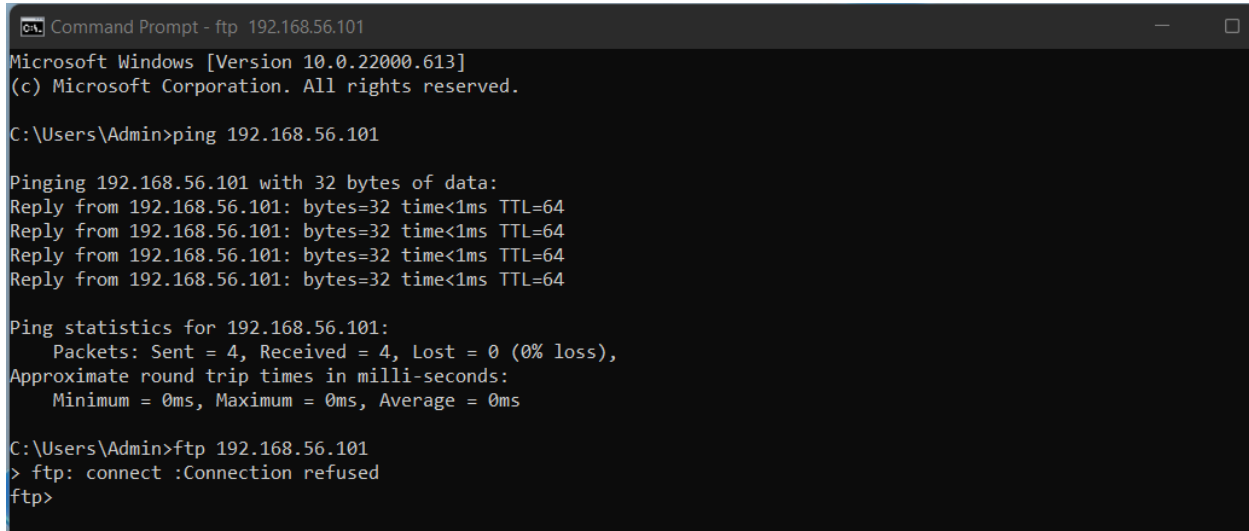
Alongside, we perform ssh attack from kali-Linux terminal to the host by using the command:

- `$ssh 192.168.56.101`

```
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ ssh 192.168.56.101
ssh: connect to host 192.168.56.101 port 22: Connection refused
(kali@kali)-[~]
$
```


Also we perform TCP and ICMP attacks from the windows terminal to the host by using the commands:

- Ping 192.168.56.101 and [ftp 192.168.56.101](#).



```
Command Prompt - ftp 192.168.56.101
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>ping 192.168.56.101

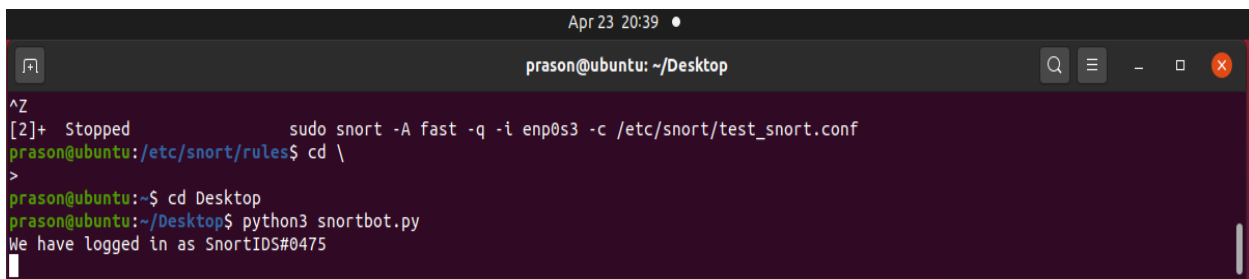
Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Admin>ftp 192.168.56.101
> ftp: connect :Connection refused
ftp>
```

The other mode is the fast mode. For this, we can use the command:

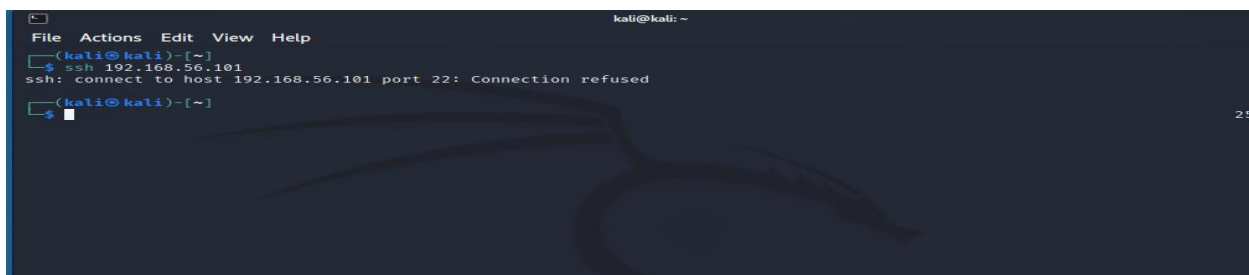
- `$ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf`



```
prason@ubuntu: ~/Desktop
^Z
[2]+  Stopped                  sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@ubuntu:/etc/snort/rules$ cd \
>
prason@ubuntu:~$ cd Desktop
prason@ubuntu:~/Desktop$ python3 snortbot.py
We have logged in as SnortIDS#0475
```

Alongside, we perform ssh attack from kali-Linux terminal to the host by using the command:

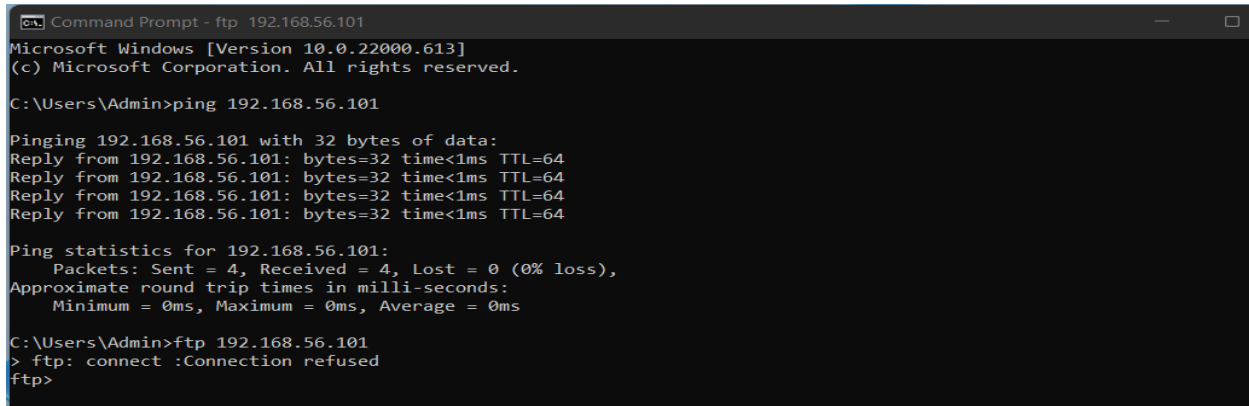
- `$ssh 192.168.56.101`



```
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ ssh 192.168.56.101
ssh: connect to host 192.168.56.101 port 22: Connection refused
(kali@kali)-[~]
$
```

Also we perform TCP and ICMP attacks from the windows terminal to the host by using the commands:

- Ping 192.168.56.101 and [ftp 192.168.56.101](#).



```
Command Prompt - ftp 192.168.56.101
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>ping 192.168.56.101

Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64

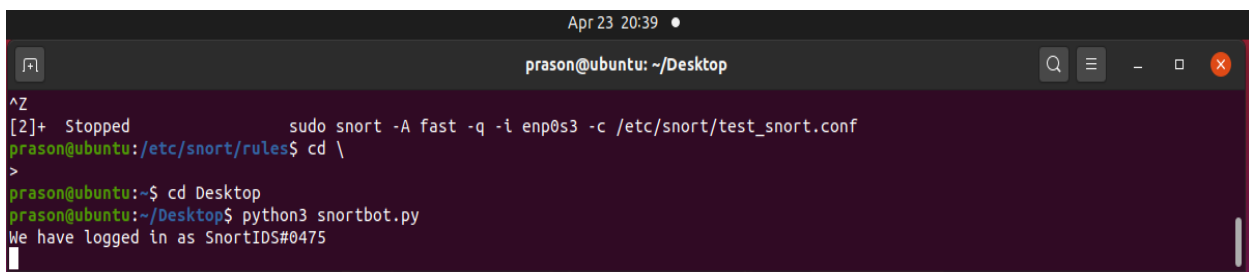
Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Admin>ftp 192.168.56.101
> ftp: connect :Connection refused
ftp>
```

Then after running the fast mode command we need to run the snortbot.py file to fetch the information to the discord bot using the command:

- `$ python3 snortbot.py`

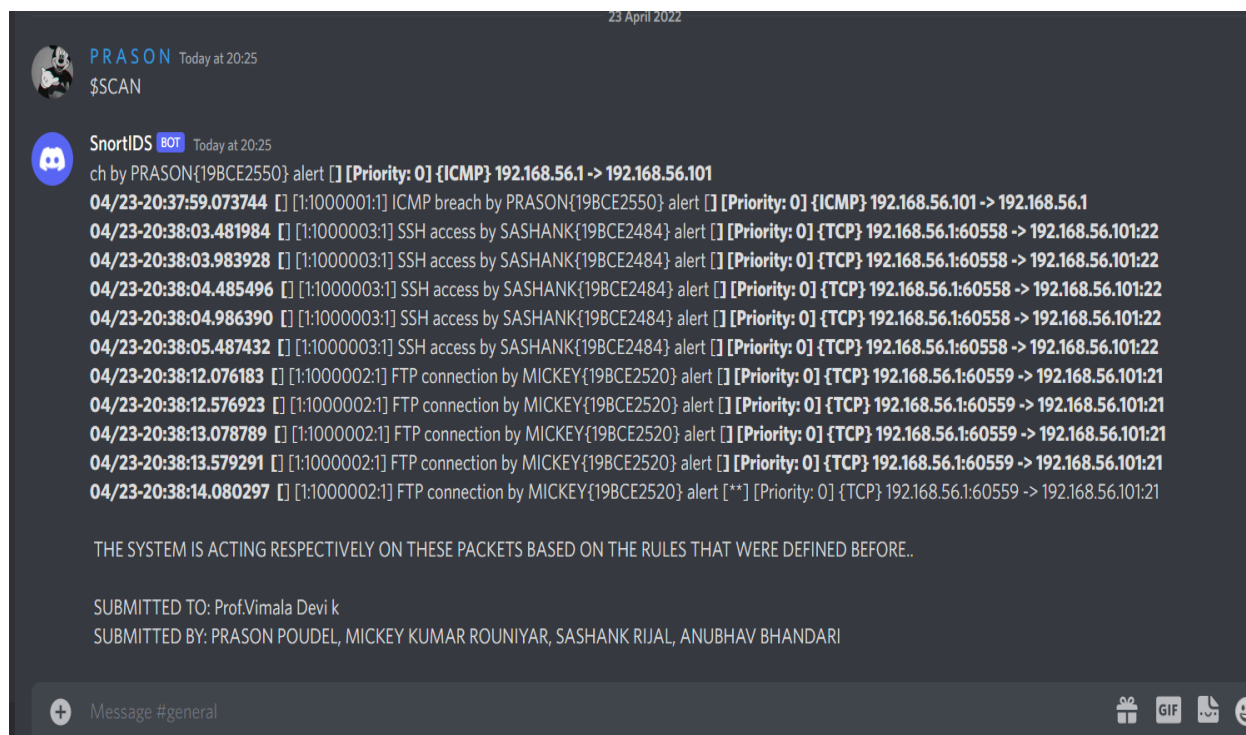
After running the python file we logged in to our snortIDS where we can get alert about the intrusions.



```
Apr 23 20:39
prason@ubuntu: ~/Desktop
^Z
[2]+  Stopped                  sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@ubuntu:/etc/snort/rules$ cd \
>
prason@ubuntu:~$ cd Desktop
prason@ubuntu:~/Desktop$ python3 snortbot.py
We have logged in as SnortIDS#0475
```

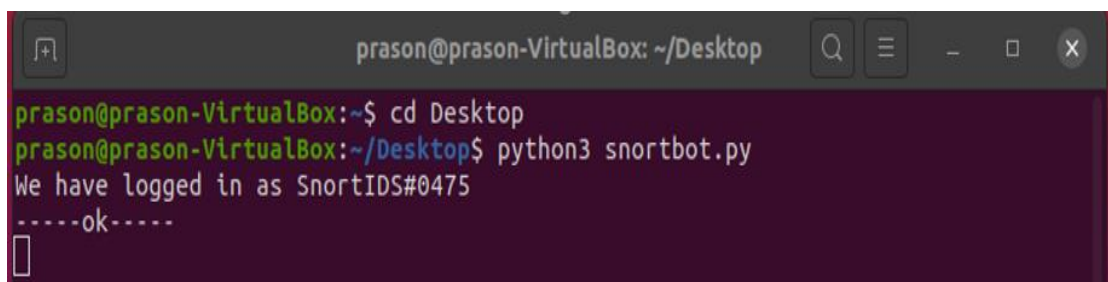
After we logged in to the snortIDS, we can get all the alert information in our discord server by using the commands:

- \$SCAN → for all the intrusion alerts
-



After we use the command in the discord server it will display -----ok-----

for each time in the terminal. It depends on how many times you use the command to check alert. If you use the command multiple times then it will appear multiple times.



AUDIT REPORT

1.1 PURPOSE AND SCOPE

The main aim and scope of this document is to explore and find various vulnerabilities in the network after the snort system is manually configured and enabled. It also deals with the detection of malicious traffic, and helps to identify if there is any abnormal behavior in the network and to verify if the configured snort system implemented and tested in the host machine is working properly and acting according to the rules written by the behavior.

1.2 OBJECTIVE

- To validate and ensure that the configured rules are written in accordance to the snort documentation rule writing format.
- To ensure that all the packets are filtered and checked with the manually written rules in the host machine and are properly classified for abnormal behavior.
- To ensure that all the rules configured in the system is tested and implemented in live sniffing phase of intrusion detection.
- To ensure the secured and efficient alert database for storing the log data of the host machine during the implementation of the configured snort system in the running phase.
- To ensure that the alert messages are securely transferred to the discord server without any delay with all the alerts present in the log file and the alert database.

1.3 CONSTRAINTS

- Time constraints
- Third party access constraints
- Business operations continuity constraints
- Scope of audit engagement
- Technology tools constraints

1.4 COMPONENTS AND DEFINITIONS

This report comprises of various terminologies and terms that will be used further in this report such as IDS, remote access servers, vulnerabilities, security testing's, etc. some basic information about these terms discussed below in this section.

SECURITY TESTING:

Security testing is a sort of software testing that identifies vulnerabilities, hazards, and dangers in a software programmer and guards against intruder attacks. The goal of security tests is to find any possible flaws and weaknesses in the software system that could lead to a loss of data, revenue, or reputation at the hands of workers or outsiders.

IDS (INTRUSION DETECTION SYSTEM):

An Intrusion Detection System (IDS) is a network traffic monitoring system that detects suspicious behavior and sends out notifications when it is found. It's a piece of software that searches a network or system for malicious activities or policy violations. Any harmful activity or violation is usually reported to an administrator or collected centrally using a SIEM system. A SIEM system combines data from many sources and employs alert filtering techniques to distinguish between malicious and false alarms.

REMOTE ACCESS SERVERS:

A remote access server (RAS) is a server that delivers a set of services to users who are connected to a network or the Internet from a distance. It links remote users to an organization's internal local area network as a remote gateway or central server (LAN).

VULNERABILITY:

Any weakness in an organization's information system, system procedures, or internal controls is referred to as a cyber security vulnerability. These flaws are targets for lurking cybercriminals, and they can be exploited via the points of vulnerability.

1.5 AUDITING PHASES

The various phases and approaches that are followed for auditing are listed below:

1. Pre-audit agreement stage.
2. Initiation and Planning stage.
3. Data collection and fieldwork.
4. Analysis
5. Reporting
6. Follow-through

1.6 AUDITING TASKS

For the auditing, various tasks and procedures are followed for auditing of the automated snort system using the discord API.

LOG REVIEW:

Logs provides various sort of information to support audits and to analyze and study the effectiveness of the security policies and also is the foremost thing that should be done after any sort of security breaches or intrusions in the network. Log reports are beneficial for forensic analysis as well as it is made as a legal requirement in many of the companies to mitigate risks and to provide ways and analysis for vulnerabilities or incidents.

NETWORK SCANNING:

It helps to recognize the target hosts' accessible UDP and TCP network services, user's and the targeted hosts' filtering systems. This helps to monitor the network and look for any sort of disruption or track any unusual behaviors and identify other network elements for attacks and other such risks.

VULNERABILITY SCANNING:

This not only helps us to find the loopholes and other such flaws in the security of the system and network, but also is used to understand and monitor as well as secure the system by reducing the overall vulnerabilities of the system. This is a major component of the audit as it gives us insights about the security of the network by exploiting the loopholes and breaches in the network and minimize the probabilities of attacks and other such actions.

1.7 AUDITING METHODS:

RISK ASSESSMENT:

It helps in the identification of the hazards that could affect the snort system and can disrupt the protocols and other systems in the network. It generally begins with identifying the loopholes, and then determining what could be harmed and effected. After this, evaluation of the risks is done and control measures are developed. After this step, the findings are recorded and is reviews and updated on the general basis from then.

POLICY ASSESSMENTS:

Here, after all of the rules of the snort were removed and was manually written by the user and was configured, all the security policies of the snort system are assessed based on the compliance with various security standards. Here, the rule writing format and relevance is checked based on the standards mentioned in the snort system

INTERVIEWS:

Since every person working in the snort IDS and integration with the discord will be having different perspectives and procedures for working in the team. So, personal interviews can be done to gain insights from the system configured and developed and can be used to increase the security and compliance of snort.

SECURITY DESIGN REVIEWS:

Since there are various security designs that are made and every working team have their own way of designing and implementing security at various levels based on the requirements. Since intrusion detection systems are used for the security of the host machine in this case, all the IDS related files and rules should be updated and periodically reviewed to ensure the security for the network of the machine.

DOCUMENT REVIEW:

All of the related information and other such important details related to security, management, policies are recorded and those documents should be reviewed thoroughly. Apart from this, log reports that are obtained in general regular basis or after any security breach or incidence should also be reviews and collected. From this review, any new pattern or unusual behavior that are analyzed should be reported to minimize the risks and attacks associated with it in coming days.

1.8 AUDIT REPORT

Based on the auditing tasks that are listed above, various activities were implemented and the outcomes associated with the activities and validation are listed below in the report.

LOG REVIEW:

CHECKS IMPLEMENTED	OUTCOMES / FINDINGS
Check if all the activities happening in the system is well recorded and is present in the form of log file with proper information in the alert database.	All the data packet information flowing in the system and all abnormal packet information were present in the form of log file present in the snort system of the host machine.
All the log information and alert databases are secured from unauthorized access.	The log files are saved in the host machine. All the log related files were secured from unauthorized access and were tamper-free.
Check for indication of false positives by the snort IDS while filtering and checking the packets and verification of false alerts while configuration.	Few false positives were found while testing and configuration phase of the filtering of the snort system but the false alerts were found to be minimum and were manageable.

DOCUMENT REVIEW:

CHECKS IMPLEMENTED	OUTCOMES / FINDINGS
Verification if all the related data is present in the document related to procedures, policies, log reports, etc.	The presence of all such details was ensured for the snort IDS and the procedures for manually configuring the system and integrating it with the discord was written in standard format.
Verification if the log file present in the host machine and log alerts received in the console mode as well as discord real time alerts were integrated with proper connection and format for log analysis in the event of security breach or attack.	The details and information present in the core snort-based system and console mode along with the discord real time mode were found to be same, with some minor false packet information or repetitive information for different time synchronization issue. All the related details were well in format and did not miss any critical information prevailing in the system.

NETWORK SCANNING:

CHECKS IMPLEMENTED	OUTCOMES / FINDINGS
Verification if the snort IDS is working correctly and is active for scanning all the incoming data packet traffic into the system using either the terminal mode (console) or fast mode.	Proper validation is ensured with efficient working and filtering the packets coming into the system with the Snort IDS.
Verification if the incoming packets into the host system is classified in accordance with the rules written and configured by the user.	All the rules written by the user were properly configured into the system and all the packets were classified and saved into the database of the snort.
Verification of the connectivity of the Developers Discord Bot API of the discord server created for the alert file through the script from the host machine and ensuring proper and efficient transmission.	The connection with the discord bot of the server was successful and instant with the confirmation displayed in the terminal itself and the script followed standard procedures with proper commands in it for user access through API.
Verification if the discord timeline feed received all the alerts with appropriate information like packet number, preference, priority, date, time, packet size, IP address, rev id, sig, id, etc. in real time with the list of all the log related details happening in the system.	All the details mentioned in the log file present in the host machine snort folder were correctly read and was transferred into the discord system without any failure or missing information.

VULNERABILITY SCANNING:

CHECKS IMPLEMENTED	OUTCOMES / FINDINGS
Verification of the snort IDS tool to ensure the possibility of some common attacks such as XML attacks, SQL injection attacks.	No loopholes were found in the database and information was secured and was well detected by the snort IDS and was alerted and logged properly.
Verification of all the data packets incoming to the system were filtered and checked with the rules written before logging them into the host machine.	Some of the data packets passing through the filtering rules were misclassified for false alerts and were configured logged into the system but the possibility of this happening was very less and can be ignored.

1.9 RECOMMENDED CONTROL CHANGES FOR THE COMPANY AFTER MERGING:

POLICIES AND PROCEDURE RECOMMENDATIONS:

All the rules written should follow the standard of the snort which can be found in the formal documentation of the snort IDS and IPS systems. Also, while viewing the predefined rules of the local rules section, some of the important commands and rules should be taken into consideration while modifying them and also should be modified with proper procedures and format.

SECURE CONFIGURATION:

Implementation for secure configuration of all the devices and machines where snort-based IDS systems are used for security and filtering.

NETWORK SECURITY:

Regular vulnerability scanning and monitoring should be done by the use of snort IDS and also IPS can be configured and deployed to further make the system robust and enhance the security of the machine and network.

SECURITY MONITORING

All sort of monitoring as well as use of tools such as IDS, IPS, Firewalls, etc. for security purposes.

RISK MANAGEMENT RECOMMENDATIONS

Conduct various risk management strategies and assessments in regular basis to ensure security at every step.

CONCLUSION

Hence, in this audit report, all the information related to the system developed with the manual configuration, implementation, testing and real time deployment of the snort-based intrusion detection system along with the discord bot implementation and integration were shown. All the procedures, phases and tasks related to the system were properly defined and related terminologies for this document were discussed in this report. Apart from that, auditing tasks important for the audit and checks were defined along with the audit report based on the checks implemented and outcomes received and verified. Some of the outcomes based on the scanning and document review were found to be contrasting and inefficient giving an area for improvement and security of that particular area. Also, recommendations were provided based on the outcome received in the audit report and future enhancement of security and efficiency of Snort based Intrusion Detection System with the integration of the discord bot API