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

AUTOMATED INTRUSION DETECTION USING SNORT PROGRAM

SLOT: F1

CSE 3501: INFORMATION SECURITY ANALYSIS AND AUDIT

SUBMITTED BY:

SASHANK RIJAL 19BCE2484,

PRASON POUDEL 19BCE2550,

MICKEY KUMAR ROUNIYAR 19BCE2520

SUBMITTED TO:

CHANDRA MOHAN B

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:

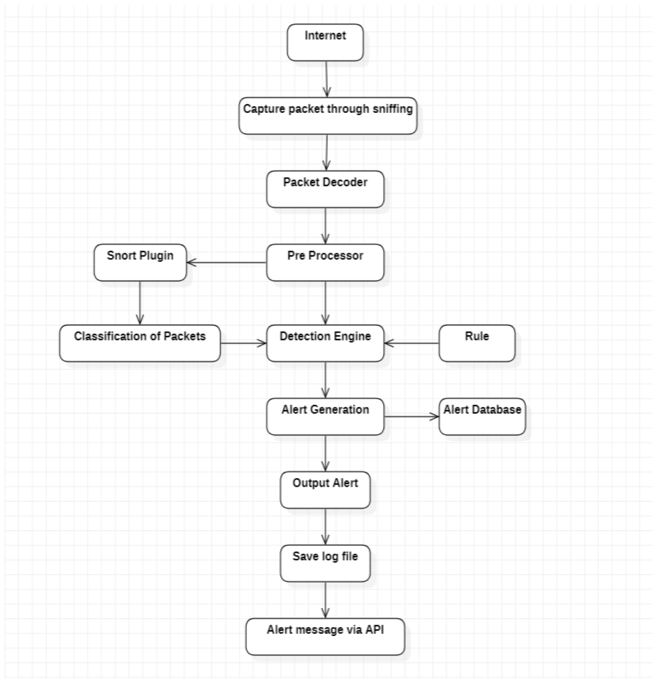
- Scalability: Snort can be successfully deployed on any network environment.
- Flexibility and Usability: Snort can run on various operating systems including Linux, Windows, and Mac OS X.
- Live and Real-Time: Snort can deliver real-time network traffic event information.

- 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:

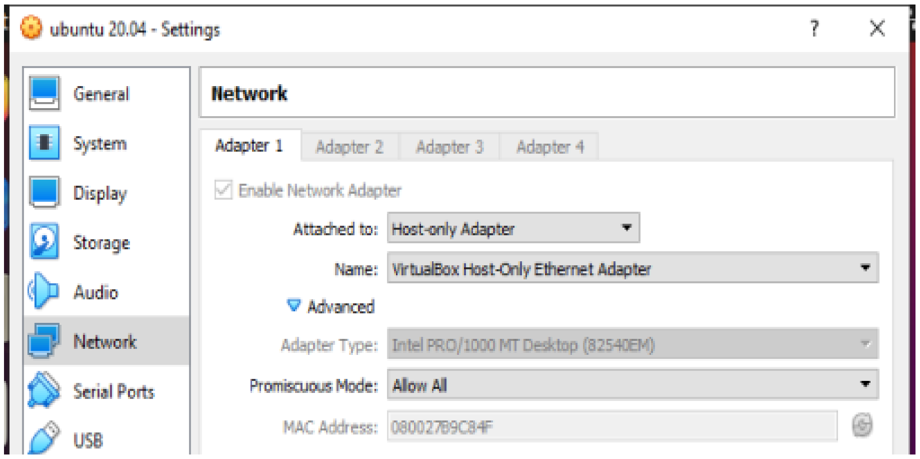
- Rule Based Detection uses a set of predefined rules to identify an intruder or attack.
- This category can be divided further into two subcategories; Anomaly detection and Penetration identification.
- Anomaly detection uses rules that are produced by looking at previous attack patterns or signatures 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:

- 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.



Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter

Advanced

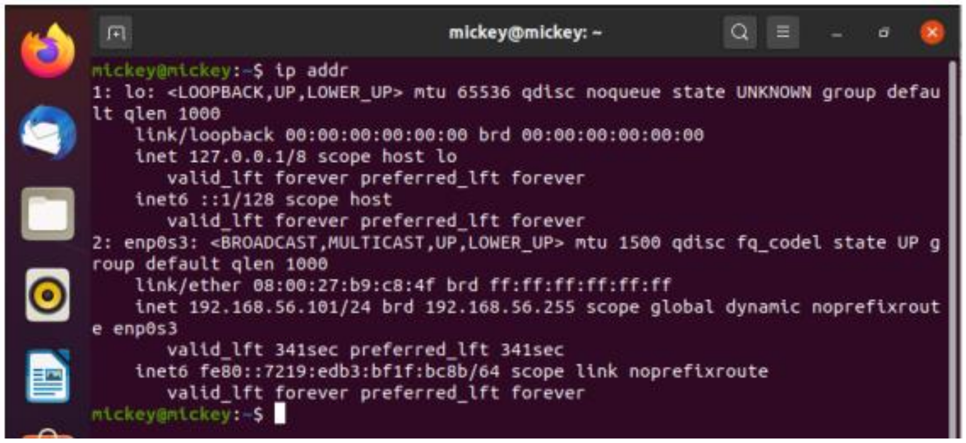
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 080027B9C84F

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.

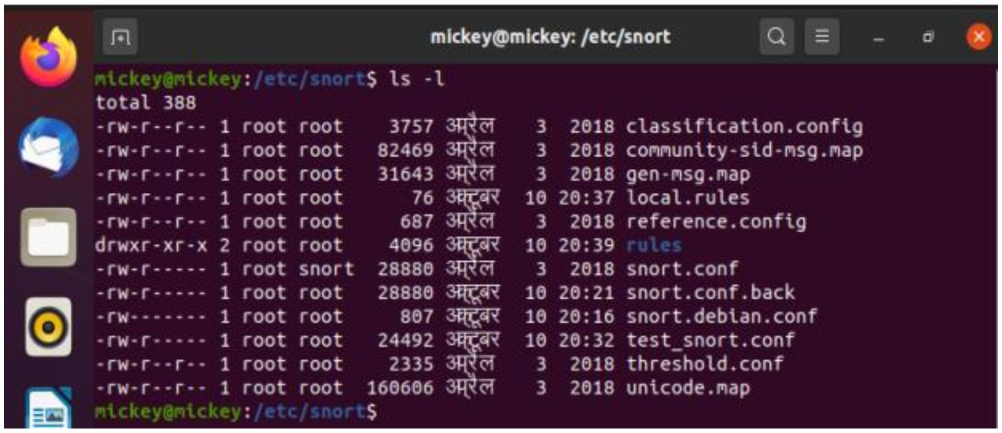


```
mickey@mickey:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:b9:c8:4f brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixrout
e enp0s3
       valid_lft 341sec preferred_lft 341sec
    inet6 fe80::7219:edb3:bf1f:bc8b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
mickey@mickey:~$ 
```

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

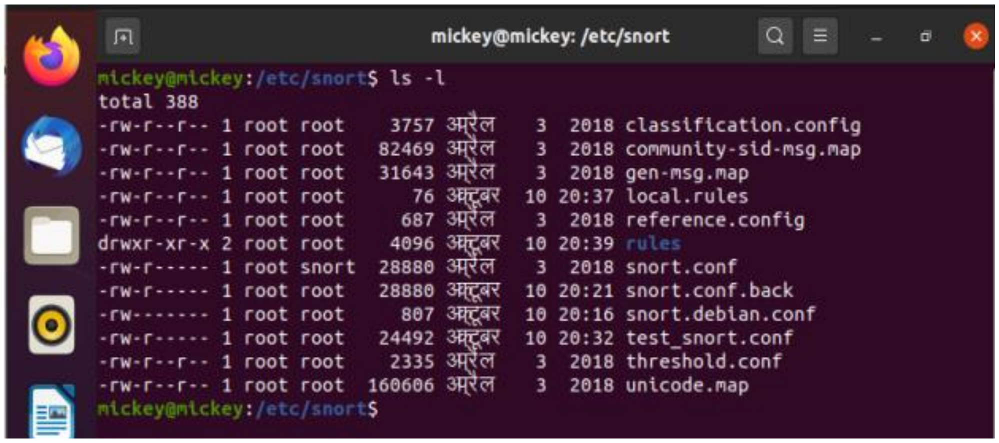
- file just execute the command under the snort directory “cd /etc/snort/” and then execute these
  - $ cp snort.conf snort.conf.back
  - $ sudo cp snort.conf snort.conf.back



```
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----- 1 root snort  28880 अप्रैल    3  2018 snort.conf
-rw-r----- 1 root root   28880 अक्टूबर  10 20:21 snort.conf.back
-rw------- 1 root root     807 अक्टूबर  10 20:16 snort.debian.conf
-rw-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$
```

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

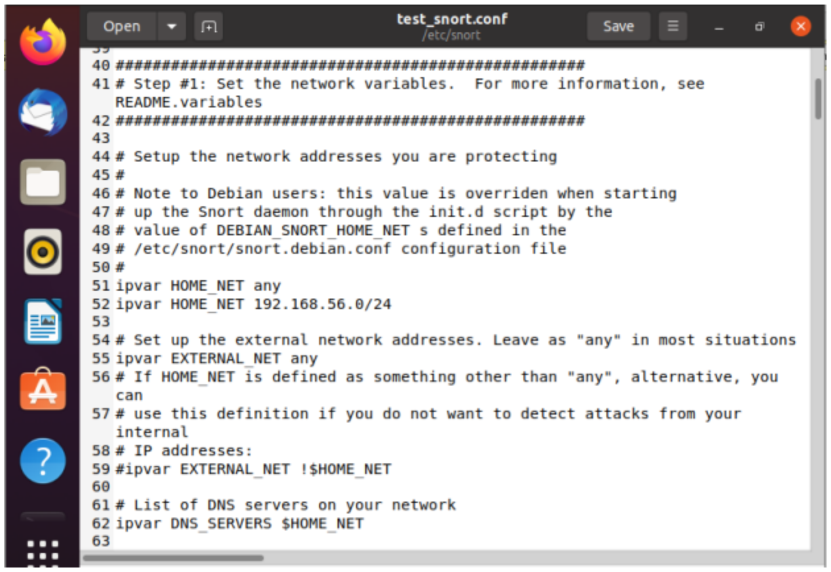


```
mickey@mickey: /etc/snort
mickey@mickey:/etc/snort$ ls -l
total 388
-rw-r--r-- 1 root root    3757 अप्रैल    3  2018 classification.config
-rw-r--r-- 1 root root   82469 अप्रैल    3  2018 community-sid-msg.map
-rw-r--r-- 1 root root   31643 अप्रैल    3  2018 gen-msg.map
-rw-r--r-- 1 root root      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----- 1 root snort  28880 अप्रैल    3  2018 snort.conf
-rw-r----- 1 root root   28880 अक्टूबर  10 20:21 snort.conf.back
-rw------- 1 root root     807 अक्टूबर  10 20:16 snort.debian.conf
-rw-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$
```

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

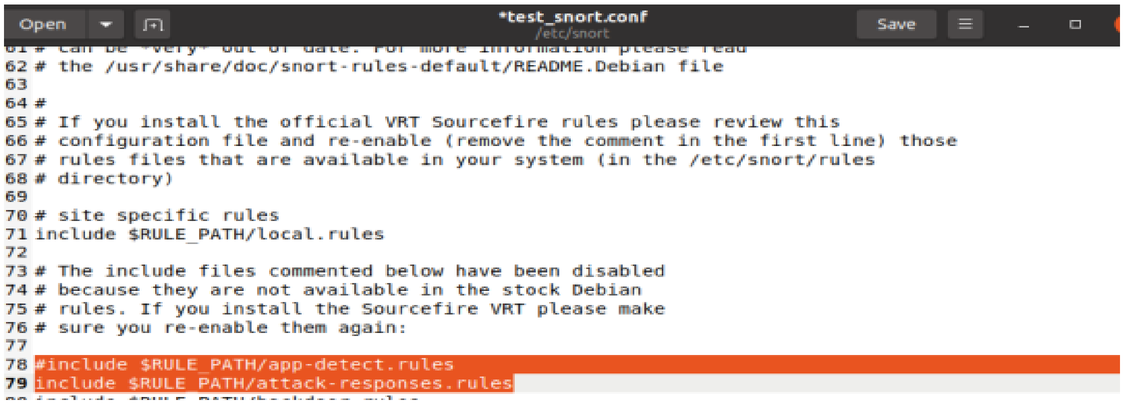
Since, for this purpose, we have an IP of 192.168.56.101/24, our home network's IP becomes 192.168.56.0/24. After knowing the IP address, just add this line: ipvar HOME_NET 192.168.56.0/24



```
###################################################
# Step #1: Set the network variables.  For more information, see
README.variables
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET any
ipvar HOME_NET 192.168.56.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you
can
# use this definition if you do not want to detect attacks from your
internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

```

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

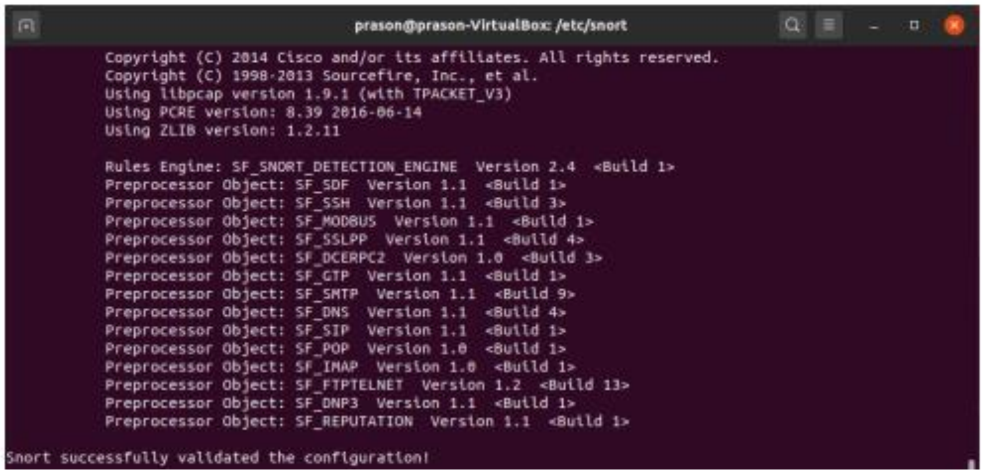


```
# can be *very* out of date. For more information please read
# the /usr/share/doc/snort-rules-default/README.Debian file

#
# If you install the official VRT Sourcefire rules please review this
# configuration file and re-enable (remove the comment in the first line) those
# rules files that are available in your system (in the /etc/snort/rules
# directory)

# site specific rules
include $RULE_PATH/local.rules

# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:

#include $RULE_PATH/app-detect.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
```

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

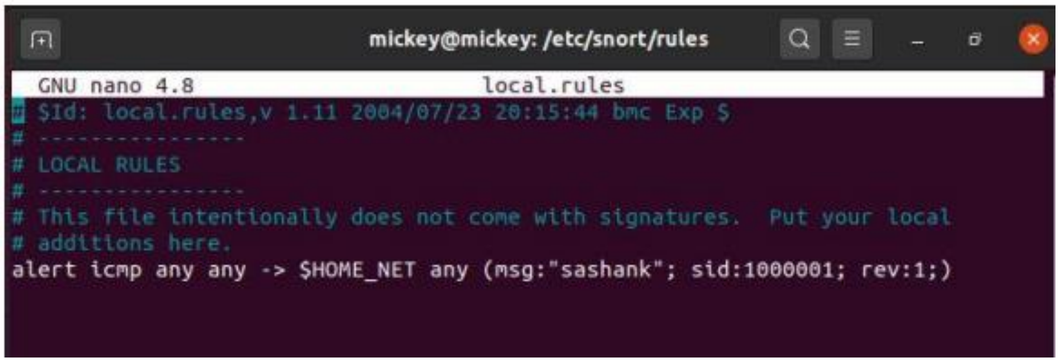


prason@prason-VirtualBox: /etc/snort

```
           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!
```

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 rules 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;)

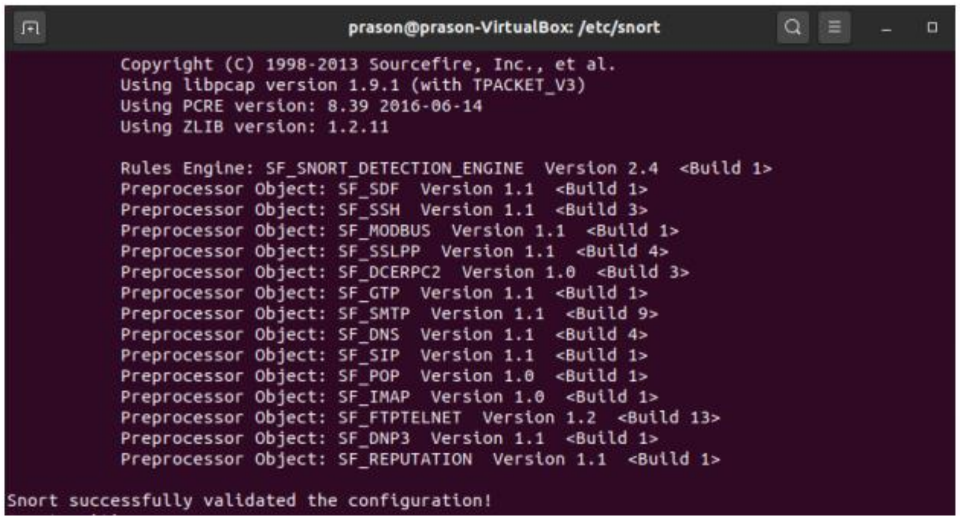


mickey@mickey: /etc/snort/rules

```
  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:"sashank"; sid:1000001; rev:1;)
```

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 the 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!
```

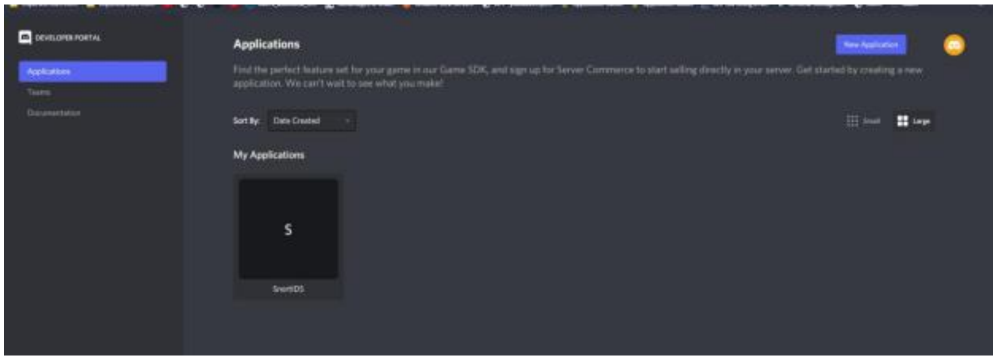
ENABLING THE SNORT SYSTEM TO SNIFF FOR ANY SORT OF INTRUSIONS:

- 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 in available in the system.
- The other mode is the fast mde. For this, we can use the command:
    - $ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf

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 the make a discord bot application for it.



Applications

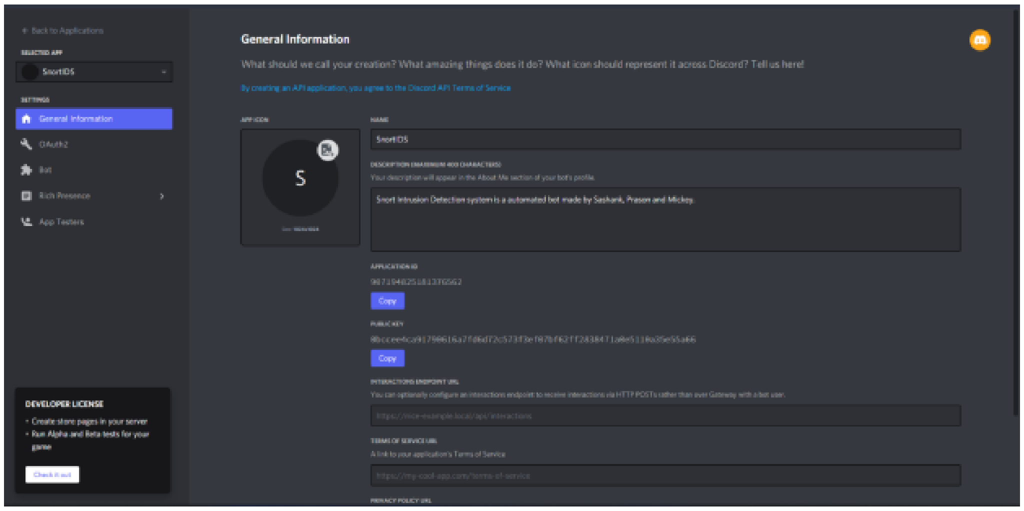
Find the perfect feature set for your game in our Game SDK, and sign up for Server Commerce to start selling directly in your server. Get started by creating a new application. We can't wait to see what you make!

Sort By: Date Created

My Applications

SnortIDS

After creating the developers portal, you can make a discord bot:



General Information

What should we call your creation? What amazing things does it do? What icon should represent it across Discord? Tell us here!

By creating an API application, you agree to the Discord API Terms of Service

SELECTED APP: SnortIDS

NAME: SnortIDS

DESCRIPTION (MAXIMUM 400 CHARACTERS)
Your description will appear in the About Me section of your bot's profile.
Snort Intrusion Detection system is a automated bot made by Sashank, Prasen and Mickey.

APPLICATION ID
987194825181376562

PUBLIC KEY
8bccee4ca91798616a7fd6d72c573f3ef87bf62ff283847 1a8e5118a35e55a66

INTERACTIONS ENDPOINT URL
You can optionally configure an interactions endpoint to receive interactions via HTTP POSTs rather than over Gateway with a bot user.
https://nice-example.local/api/interactions

TERMS OF SERVICE URL
A link to your application's Terms of Service
https://my-cool-app.com/terms-of-service

PRIVACY POLICY URL

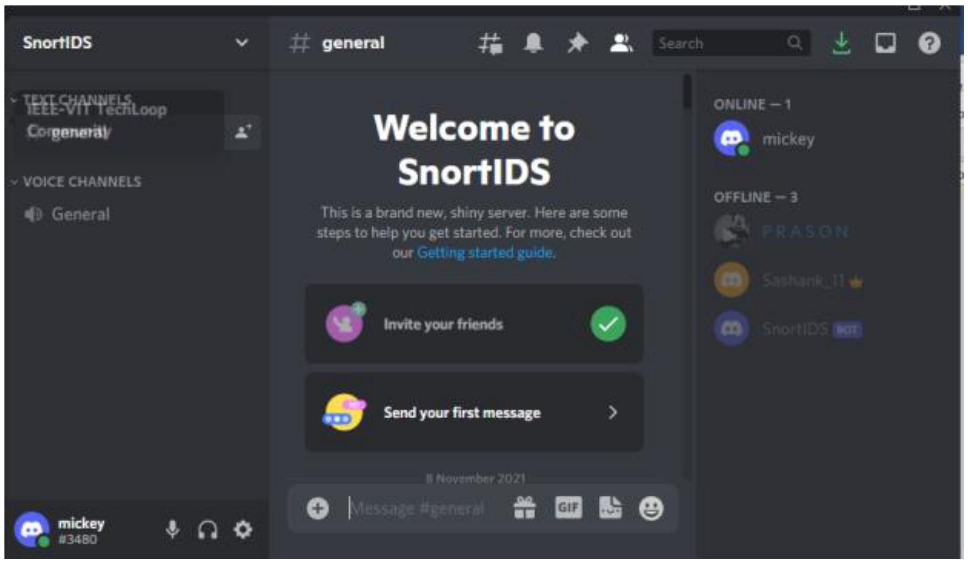
DEVELOPER LICENSE
- Create store pages in your server
- Run Alpha and Beta tests for your game

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)

▪ The discord bot named “SnortIDS” will be currently in offline status as it is not in used.



WRTING 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

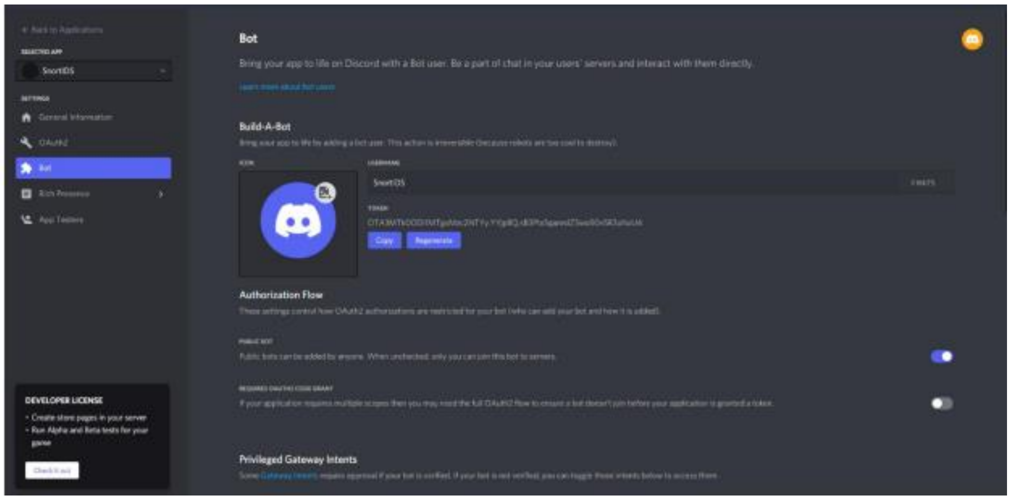
▪ We are using python here as python is pre-installed in the Linux environment. Apart from python, we

▪ 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
```



Bot

Bring your app to life on Discord with a Bot user. Be a part of chat in your users' servers and interact with them directly.

Build-A-Bot

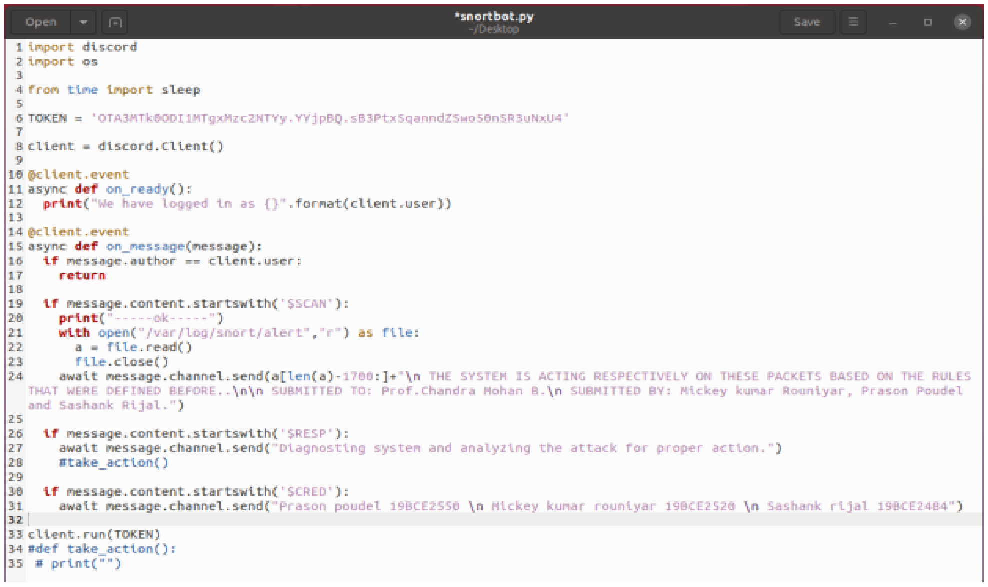
USERNAME: SnortIDS

Authorization Flow

These settings control how OAuth2 authorizations are restricted for your bot (who can add your bot and how it is added).

Privileged Gateway Intents

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



```python
import discord
import os

from time import sleep

TOKEN = 'OTA3MTk0ODI1MTgxMzc2NTYy.YYjpBQ.sB3PtxSqanndZSwo50nSR3uNxU4'

client = discord.Client()

@client.event
async def on_ready():
  print("We have logged in as {}".format(client.user))

@client.event
async def on_message(message):
  if message.author == client.user:
    return

  if message.content.startswith('$SCAN'):
    print("-----ok-----")
    with open("/var/log/snort/alert","r") as file:
      a = file.read()
      file.close()
    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.Chandra Mohan B.\n SUBMITTED BY: Mickey kumar Rouniyar, Prason Poudel and Sashank Rijal.")

  if message.content.startswith('$RESP'):
    await message.channel.send("Diagnosting system and analyzing the attack for proper action.")
    #take_action()

  if message.content.startswith('$CRED'):
    await message.channel.send("Prason poudel 19BCE2550 \n Mickey kumar rouniyar 19BCE2520 \n Sashank rijal 19BCE2484")

client.run(TOKEN)
#def take_action():
 # print("")
```

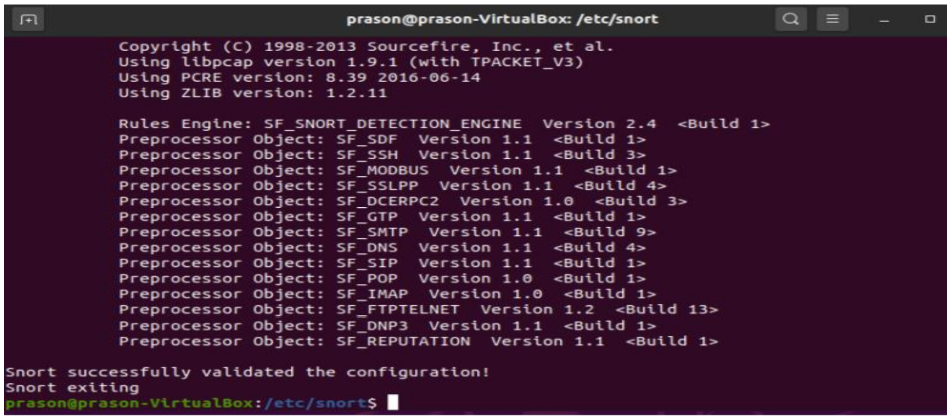
RESULTS:

- Detects all kinds of intrusion like ICMP FTP, SHH 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:

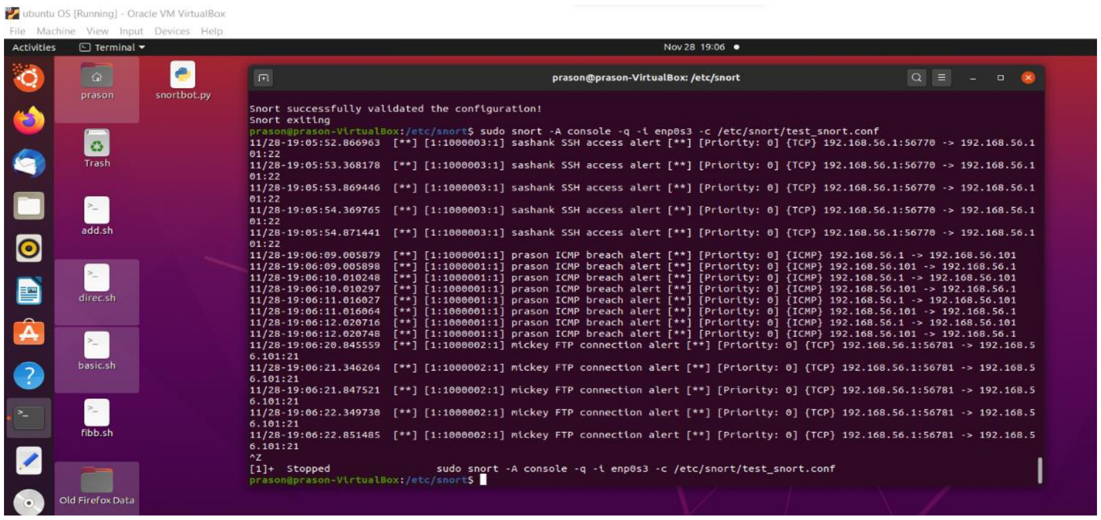
- $ sudo snort -T -i enp0s3 -c /etc/snort/test_snort.conf



```
           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:

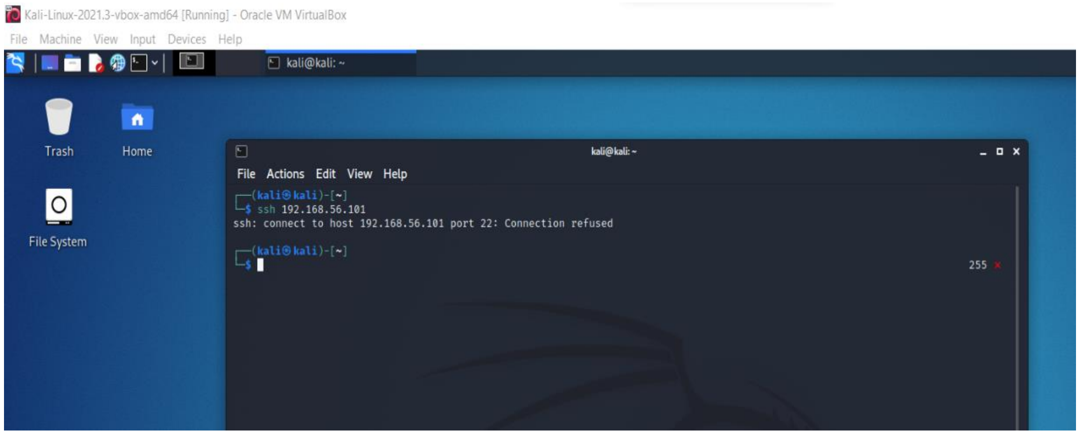
- $ sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf



```
Snort successfully validated the configuration!
Snort exiting
prason@prason-VirtualBox:/etc/snort$ sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf
11/28-19:05:52.866963  [**] [1:1000003:1] sashank SSH access alert [**] [Priority: 0] {TCP} 192.168.56.1:56770 -> 192.168.56.1
01:22
11/28-19:05:53.368178  [**] [1:1000003:1] sashank SSH access alert [**] [Priority: 0] {TCP} 192.168.56.1:56770 -> 192.168.56.1
01:22
11/28-19:05:53.869446  [**] [1:1000003:1] sashank SSH access alert [**] [Priority: 0] {TCP} 192.168.56.1:56770 -> 192.168.56.1
01:22
11/28-19:05:54.369765  [**] [1:1000003:1] sashank SSH access alert [**] [Priority: 0] {TCP} 192.168.56.1:56770 -> 192.168.56.1
01:22
11/28-19:05:54.871441  [**] [1:1000003:1] sashank SSH access alert [**] [Priority: 0] {TCP} 192.168.56.1:56770 -> 192.168.56.1
01:22
11/28-19:06:09.005879  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
11/28-19:06:09.005898  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/28-19:06:10.010248  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
11/28-19:06:10.010297  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/28-19:06:11.016027  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
11/28-19:06:11.016064  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/28-19:06:12.020716  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
11/28-19:06:12.020748  [**] [1:1000001:1] prason ICMP breach alert [**] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/28-19:06:20.845559  [**] [1:1000002:1] mickey FTP connection alert [**] [Priority: 0] {TCP} 192.168.56.1:56781 -> 192.168.5
6.101:21
11/28-19:06:21.346264  [**] [1:1000002:1] mickey FTP connection alert [**] [Priority: 0] {TCP} 192.168.56.1:56781 -> 192.168.5
6.101:21
11/28-19:06:21.847521  [**] [1:1000002:1] mickey FTP connection alert [**] [Priority: 0] {TCP} 192.168.56.1:56781 -> 192.168.5
6.101:21
11/28-19:06:22.349730  [**] [1:1000002:1] mickey FTP connection alert [**] [Priority: 0] {TCP} 192.168.56.1:56781 -> 192.168.5
6.101:21
11/28-19:06:22.851485  [**] [1:1000002:1] mickey FTP connection alert [**] [Priority: 0] {TCP} 192.168.56.1:56781 -> 192.168.5
6.101:21
^Z
[1]+  Stopped                 sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@prason-VirtualBox:/etc/snort$
```

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

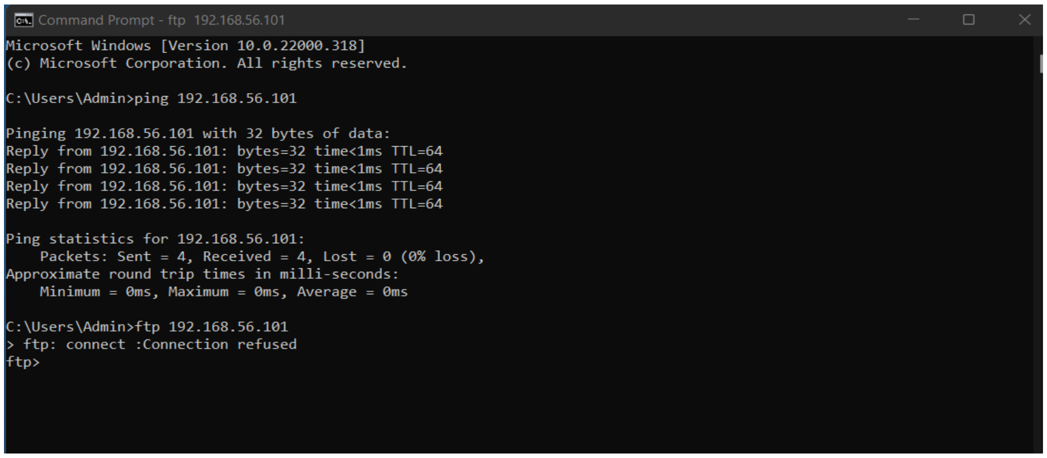
- $ssh 192.168.56.101


Kali-Linux-2021.3-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
Trash
Home
File System
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)-[~]
└─$                                                                255 ×
```

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

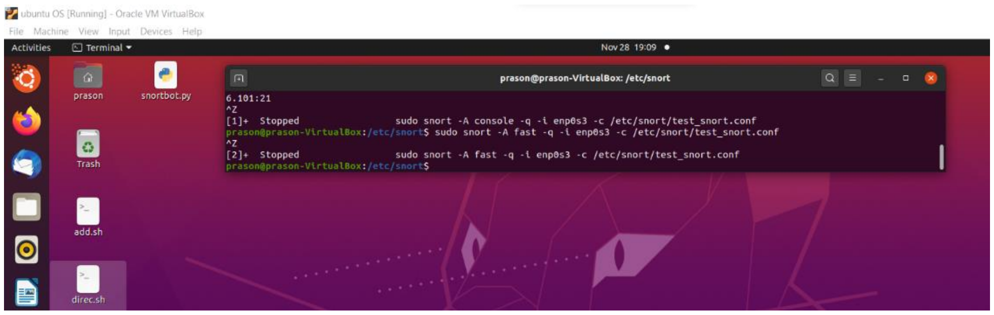
- Ping 192.168.56.101 and ftp 192.168.56.101.



```
Microsoft Windows [Version 10.0.22000.318]
(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:

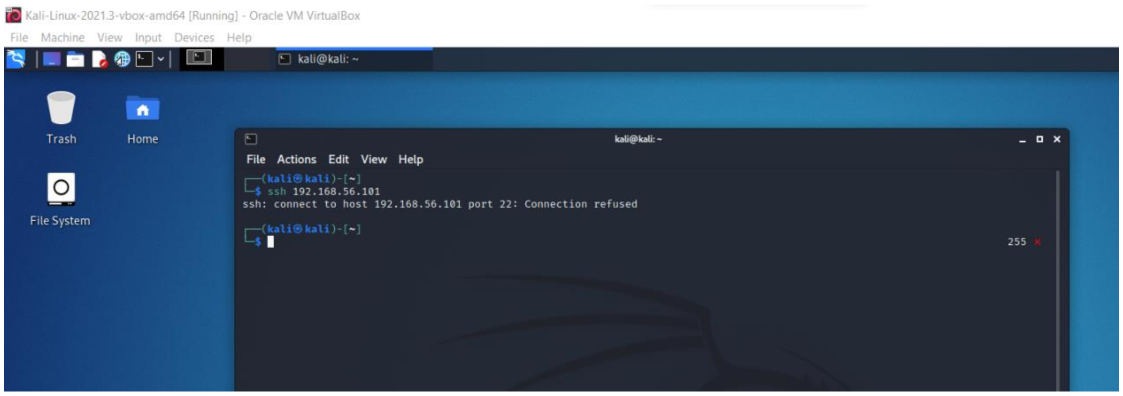
- $ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf


ubuntu OS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 28 19:09
prason
snortbot.py
Trash
add.sh
direc.sh
prason@prason-VirtualBox: /etc/snort

```
6.101:21
^Z
[1]+  Stopped                 sudo snort -A console -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@prason-VirtualBox:/etc/snort$ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
^Z
[2]+  Stopped                 sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@prason-VirtualBox:/etc/snort$
```

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

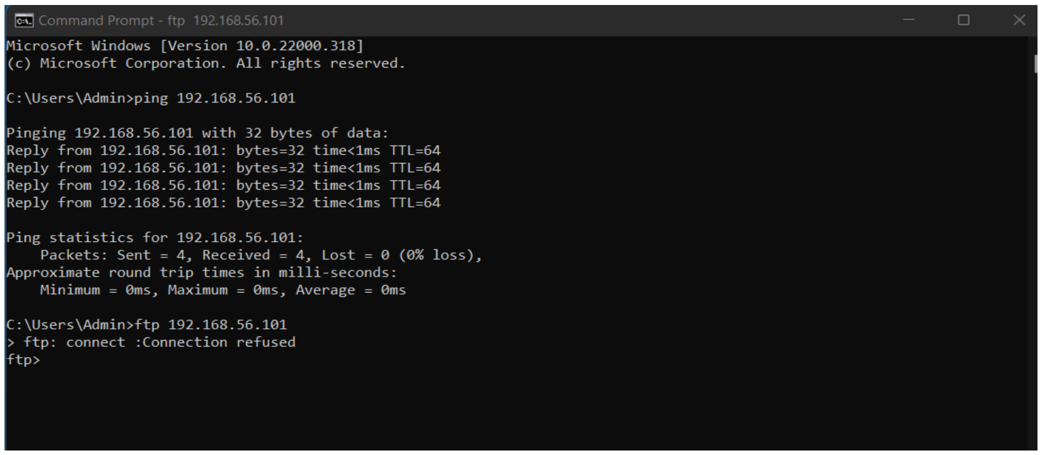
- $ssh 192.168.56.101


Kali-Linux-2021.3-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
Trash
Home
File System
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)-[~]
└─$                                                          255 ×
```

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.

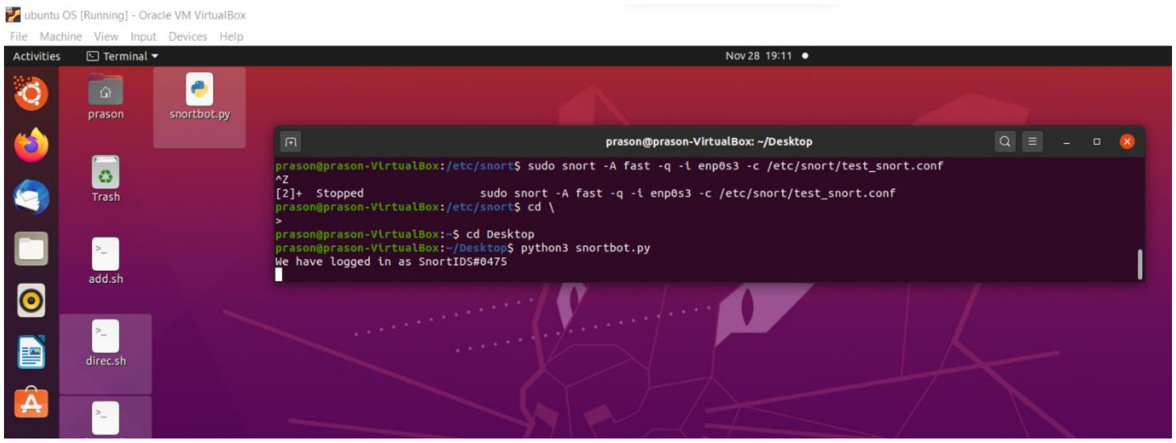


```
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

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

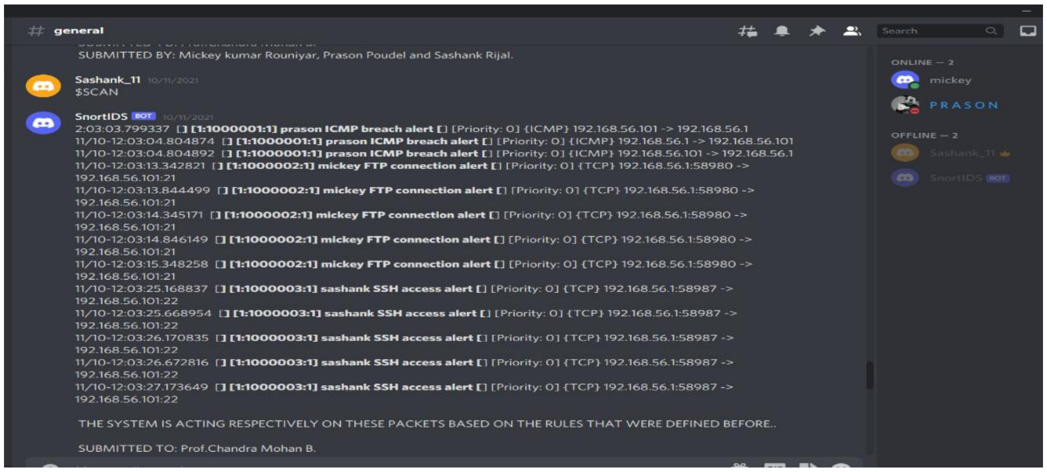
After running the python file we logged in to our snortIDS where we can get alert about the intrusions.



```
prason@prason-VirtualBox:/etc/snort$ sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
^Z
[2]+  Stopped                 sudo snort -A fast -q -i enp0s3 -c /etc/snort/test_snort.conf
prason@prason-VirtualBox:/etc/snort$ cd \
>
prason@prason-VirtualBox:~$ cd Desktop
prason@prason-VirtualBox:~/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

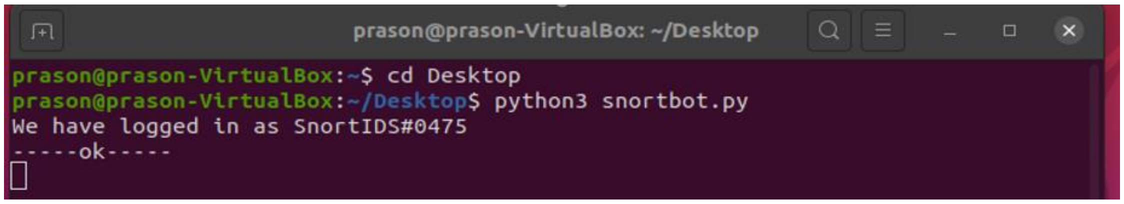


```
SUBMITTED BY: Mickey kumar Rouniyar, Prason Poudel and Sashank Rijal.

Sashank_11 10/11/2021
$SCAN

SnortIDS BOT 10/11/2021
2:03:03.799337  [] [1:1000001:1] prason ICMP breach alert [] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/10-12:03:04.804874  [] [1:1000001:1] prason ICMP breach alert [] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.101
11/10-12:03:04.804892  [] [1:1000001:1] prason ICMP breach alert [] [Priority: 0] {ICMP} 192.168.56.101 -> 192.168.56.1
11/10-12:03:13.342821  [] [1:1000002:1] mickey FTP connection alert [] [Priority: 0] {TCP} 192.168.56.1:58980 ->
192.168.56.101:21
11/10-12:03:13.844499  [] [1:1000002:1] mickey FTP connection alert [] [Priority: 0] {TCP} 192.168.56.1:58980 ->
192.168.56.101:21
11/10-12:03:14.345171  [] [1:1000002:1] mickey FTP connection alert [] [Priority: 0] {TCP} 192.168.56.1:58980 ->
192.168.56.101:21
11/10-12:03:14.846149  [] [1:1000002:1] mickey FTP connection alert [] [Priority: 0] {TCP} 192.168.56.1:58980 ->
192.168.56.101:21
11/10-12:03:15.348258  [] [1:1000002:1] mickey FTP connection alert [] [Priority: 0] {TCP} 192.168.56.1:58980 ->
192.168.56.101:21
11/10-12:03:25.168837  [] [1:1000003:1] sashank SSH access alert [] [Priority: 0] {TCP} 192.168.56.1:58987 ->
192.168.56.101:22
11/10-12:03:25.668954  [] [1:1000003:1] sashank SSH access alert [] [Priority: 0] {TCP} 192.168.56.1:58987 ->
192.168.56.101:22
11/10-12:03:26.170835  [] [1:1000003:1] sashank SSH access alert [] [Priority: 0] {TCP} 192.168.56.1:58987 ->
192.168.56.101:22
11/10-12:03:26.672816  [] [1:1000003:1] sashank SSH access alert [] [Priority: 0] {TCP} 192.168.56.1:58987 ->
192.168.56.101:22
11/10-12:03:27.173649  [] [1:1000003:1] sashank SSH access alert [] [Priority: 0] {TCP} 192.168.56.1:58987 ->
192.168.56.101:22

THE SYSTEM IS ACTING RESPECTIVELY ON THESE PACKETS BASED ON THE RULES THAT WERE DEFINED BEFORE..

SUBMITTED TO: Prof.Chandra Mohan B.
```

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.



```
prason@prason-VirtualBox:~$ cd Desktop
prason@prason-VirtualBox:~/Desktop$ python3 snortbot.py
We have logged in as SnortIDS#0475
-----ok-----
```