Department of Information Technology

Academic Year: 2022-2023 **Semester: V**

Class / Branch: TE IT

Subject: Security Lab

Subject Incharge: Prof. Apeksha Mohite

Experiment No. 06

1. **Aim: To study Intrusion Detection system SNORT and its log analysis.**
2. **Software Required** : Ubuntu 14.04 OS,
3. **Theory :**

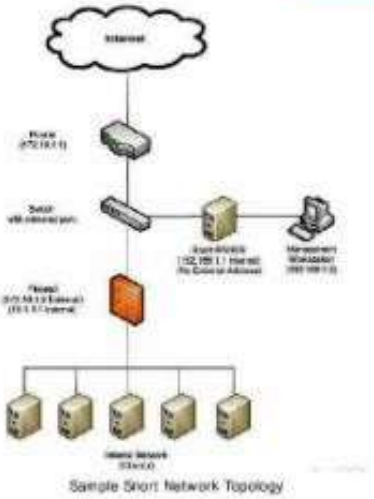
   Snort is a popular choice for running a network intrusion detection systems or NIDS. It monitors the package data sent and received through a specific network interface. NIDS can catch threats targeting your system vulnerabilities using signature-based detection and protocol analysis technologies. NIDS software, when installed and configured appropriately, can identify the latest attacks, malware infections, compromised systems, and network policy violations.

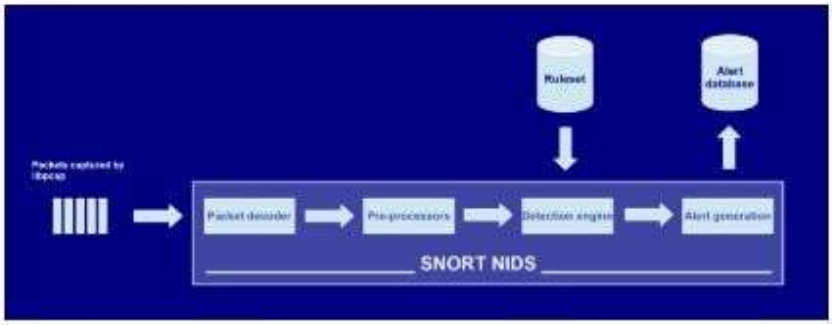
   Snort can run in two modes:

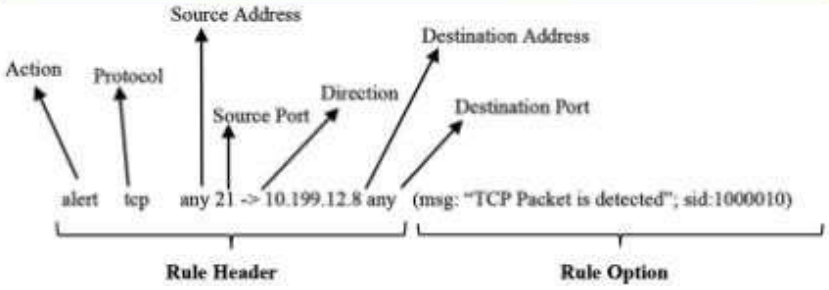
- Packet Sniffing
  - This mode have no special use, all you can do is just look at the traffic coming at the interface.
- Network Intrusion detection
  - This mode is the actual use of snort, in this mode snort monitor the traffic and block any unwanted traffic using the rules.

Snort

- Snort is an open source network-based intrusion detection system (NIDS)
  - It has the ability to perform real-time traffic analysis and packet logging on Internet protocol (IP) networks
  - It performs protocol analysis, content searching, and content matching
- Snort can be configured in three main modes:
  - Sniffer mode
  - Packet logger mode
  - Network intrusion detection system (NIDS)


Sample Snort Network Topology





Step 1: Prepare to install

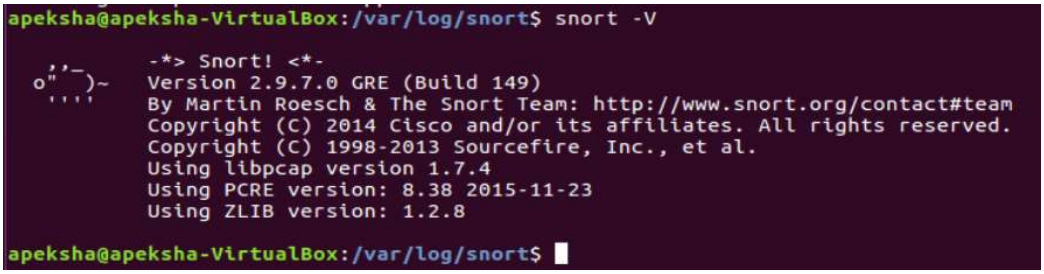
Before actually installing snort, their are some of its per-requisites, you can run following commands to install all the required per-requisites.

sudo apt-get update

sudo apt-get dist-upgrade

Step 2 : sudo apt-get install snort

Snort is now installed on your system, but you need to configure snort to make use of it. To make sure snort is installed on your system, run **`snort -V`** , if you see the following output, then you are on right track.



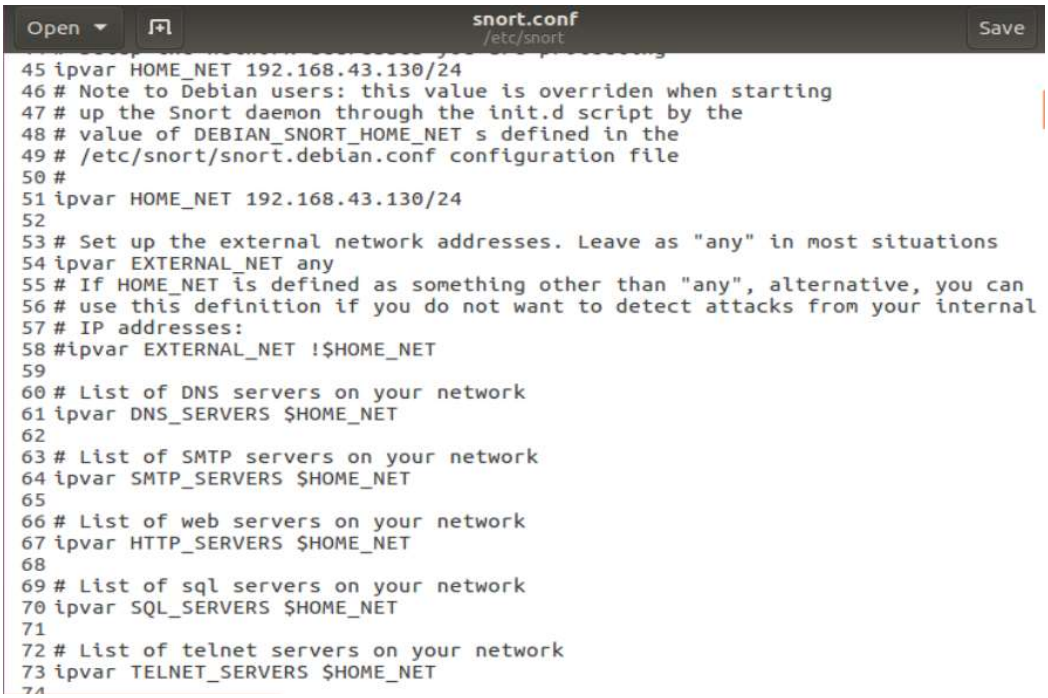
Step 4: Editing snort configuration files

Next, we need to configure our HOME_NET value: the network we will be protecting. First, enter ifconfig in your terminal shell to see the network configuration. Note the IP address and the network interface value. See the image below (your IP may be different).

This command will open the snort.conf file and move you to 45th line, make sure your following line look like this

```
sudo vi +45 /etc/snort/snort.conf
```

ipvar HOME_NET 192.168.43.130/24




```
sudo vi +104 /etc/snort/snort.conf
```

Following the line at 104, make sure your paths look like this.

var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
var WHITE_LIST_PATH /etc/snort/rules/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists

```
sudo vi +545 /etc/snort/snort.conf
```

UN-comment the 545th line and make it look like this

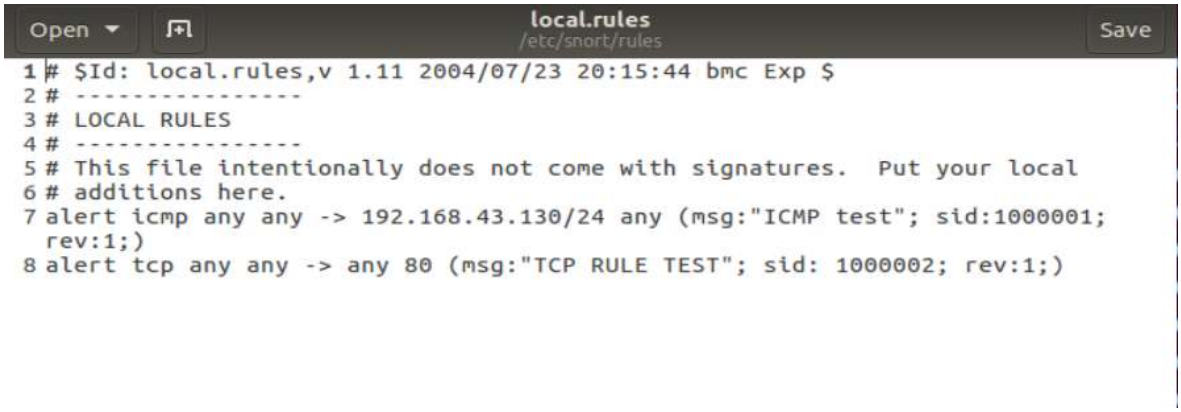
include $RULE_PATH/local.rules

Let's create our first simple test rule. This rule will generate an alert whenever Snort detects an ICMP Echo request (ping) or Echo reply message. Open the local.rules file in a text editor as root with the following command:

```
sudo gedit /etc/snort/rules/local.rules
```

You should see that the file is empty. Add the following rule (as one string of code, no line breaks):

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



Let's walk through the syntax of this rule:

Rule Header

`alert` – Rule action. Snort will generate an alert when the set condition is met.

`any` – Source IP. Snort will look at all sources.

`any` – Source port. Snort will look at all ports.

`->` – Direction. From source to destination.

`$HOME_NET` – Destination IP. We are using the HOME_NET value from the snort.conf file.

`any` – Destination port. Snort will look at all ports on the protected network.

Rule Options

`msg:"ICMP test"` – Snort will include this message with the alert.

`sid:1000001` – Snort rule ID. Remember all numbers < 1,000,000 are reserved, this iswhy we are starting with 1000001 (you may use any number, as long as it's greater than 1,000,000).

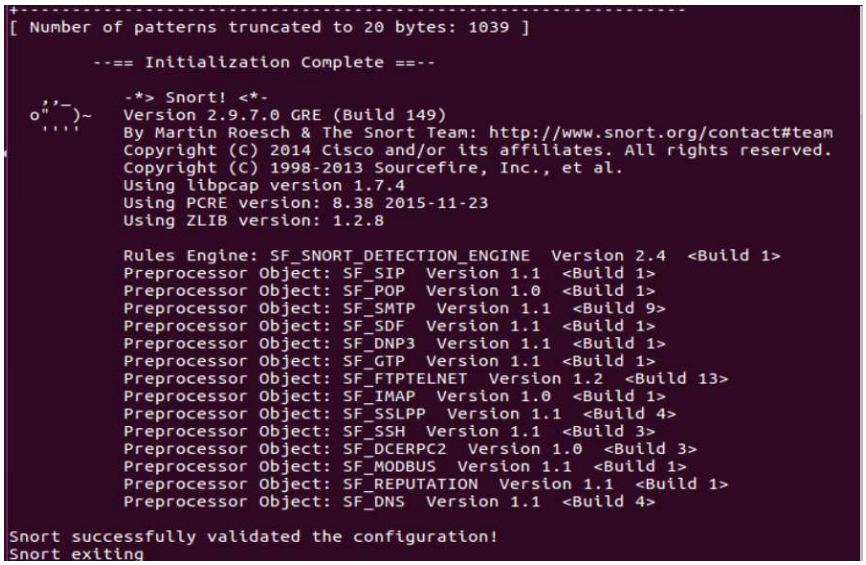
`rev:1` – Revision number. This option allows for easier rule maintenance.

`classtype:icmp-event` – Categorizes the rule as an "icmp-event", one of the predefined Snort categories. This option helps with rule organization.

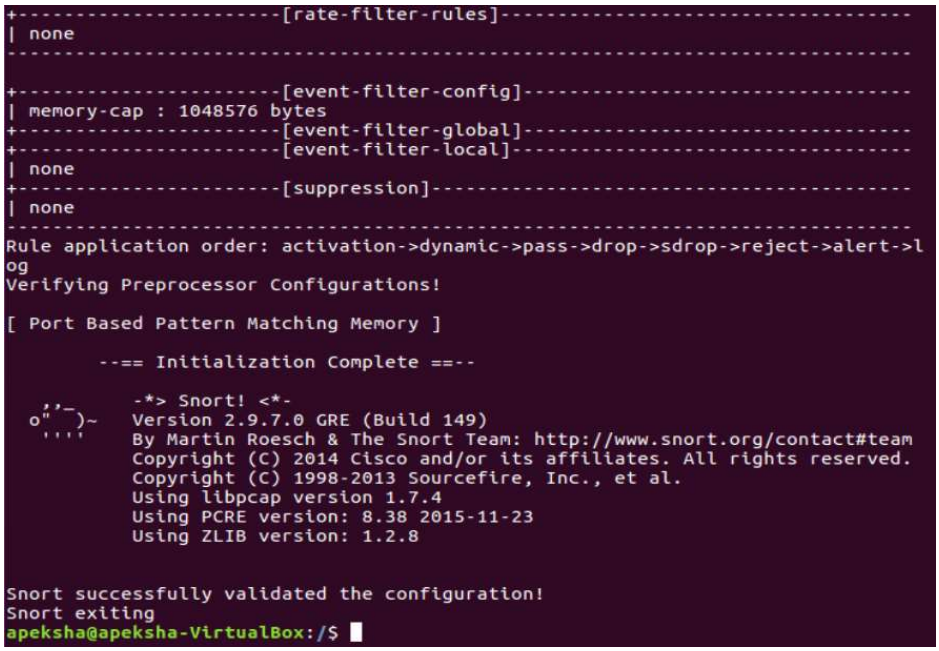
Click Save and close the file. Now let's run the Snort configuration test command again:

Test Snort :

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



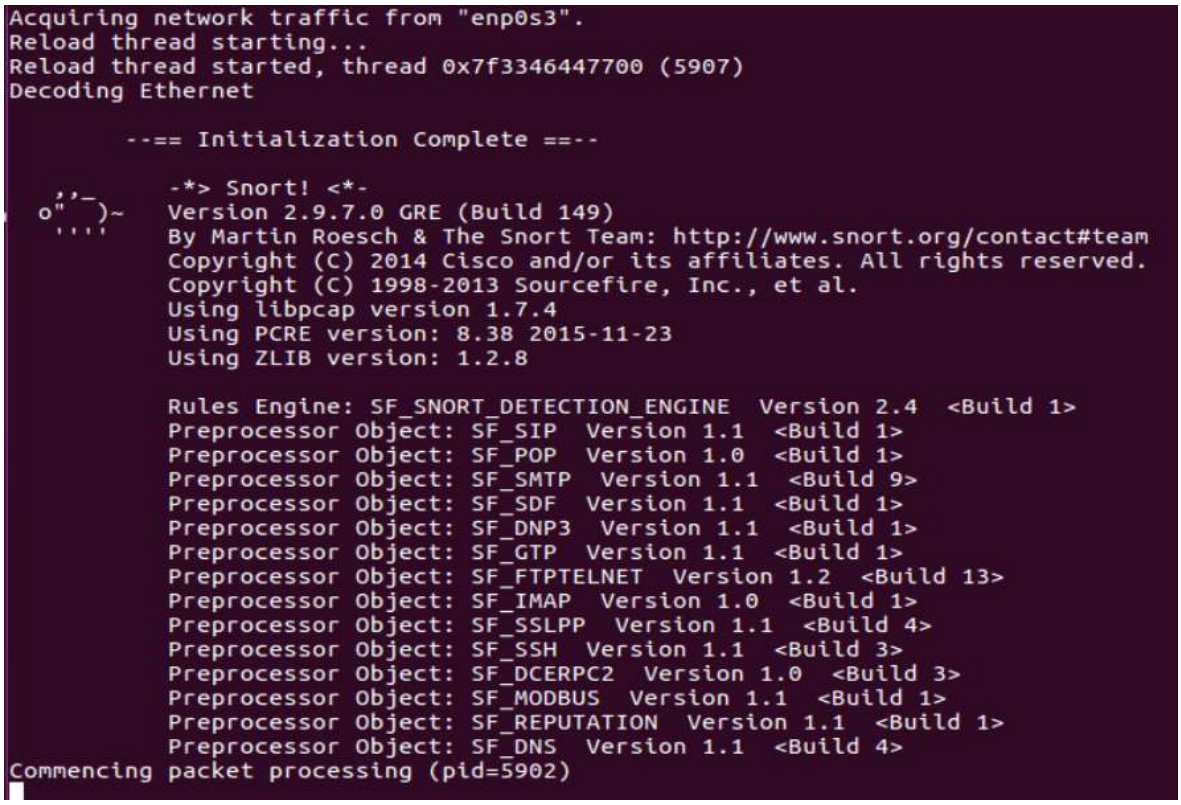
Test Snort :
sudo snort -T -c /etc/snort/rules/local.rules



Now, let's start Snort in IDS mode and tell it to display alerts to the console:

Now in order to work snort as IDS first we need to keep snort in listening mode so that it will get the alerts which we have set in local.rules file

sudo snort -A console -c /etc/snort/snort.conf



While snort in listening mode ping it from other system in our case 192.168.43.24

Here we are getting ICMP alert messages as "ICMP Testing Rule" , when another machine tries to ping the snort configured machine.

```
           Using ZLIB version: 1.2.8

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.4  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Commencing packet processing (pid=5902)
10/10-11:54:16.008427  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc
activity] [Priority: 3] {ICMP} 192.168.43.24 -> 192.168.43.130
10/10-11:54:16.008427  [**] [1:1000001:1] ICMP test [**] [Priority: 0] {ICMP} 19
2.168.43.24 -> 192.168.43.130
10/10-11:54:16.008427  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
ity] [Priority: 3] {ICMP} 192.168.43.24 -> 192.168.43.130
10/10-11:54:16.008474  [**] [1:1000001:1] ICMP test [**] [Priority: 0] {ICMP} 19
2.168.43.130 -> 192.168.43.24
10/10-11:54:16.008474  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
 activity] [Priority: 3] {ICMP} 192.168.43.130 -> 192.168.43.24
10/10-11:54:17.008813  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc
activity] [Priority: 3] {ICMP} 192.168.43.24 -> 192.168.43.130
10/10-11:54:17.008813  [**] [1:1000001:1] ICMP test [**] [Priority: 0] {ICMP} 19
2.168.43.24 -> 192.168.43.130
10/10-11:54:17.008813  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
```

While snort in listening mode perform a scan on the system from other system in our case 192.168.43.24

```
10/10-11:55:43.936916  [**] [1:1000002:1] TCP RULE TEST [**] [Priority: 0] {TCP}
 192.168.43.24:45768 -> 192.168.43.130:80
10/10-11:55:44.039264  [**] [1:1000002:1] TCP RULE TEST [**] [Priority: 0] {TCP}
 192.168.43.24:45814 -> 192.168.43.130:80
10/10-11:55:44.085212  [**] [1:1418:11] SNMP request tcp [**] [Classification: A
ttempted Information Leak] [Priority: 2] {TCP} 192.168.43.24:59838 -> 192.168.43
.130:161
10/10-11:55:44.100870  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classifica
tion: Attempted Information Leak] [Priority: 2] {TCP} 192.168.43.24:33120 -> 192
.168.43.130:705
```

As soon as the alert gets generated snort also creates log file of all the activity. Which can be seen in /var/log/snort path.

```
apeksha@apeksha-VirtualBox:/$ cd /var/log/snort
apeksha@apeksha-VirtualBox:/var/log/snort$ ls
alert           snort.log.1665381307  snort.log.1665381687  snort.log.1665383016
archived_logs   snort.log.1665381601  snort.log.1665382116
apeksha@apeksha-VirtualBox:/var/log/snort$
```

```
apeksha@apeksha-VirtualBox:/var/log/snort$ cat alert
[**] [1:1000001:1] ICMP test [**]
[Priority: 0]
10/10-11:25:26.339631 192.168.43.24 -> 192.168.43.130
ICMP TTL:64 TOS:0x0 ID:42860 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:3240   Seq:1  ECHO

[**] [1:1000001:1] ICMP test [**]
[Priority: 0]
10/10-11:25:27.340885 192.168.43.24 -> 192.168.43.130
ICMP TTL:64 TOS:0x0 ID:42923 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:3240   Seq:2  ECHO

[**] [1:1000001:1] ICMP test [**]
[Priority: 0]
10/10-11:25:28.340139 192.168.43.24 -> 192.168.43.130
ICMP TTL:64 TOS:0x0 ID:43167 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:3240   Seq:3  ECHO

[**] [1:1000001:1] ICMP test [**]
[Priority: 0]
```

The log file can be read by using command mentioned in the following screenshot

```
apeksha@apeksha-VirtualBox:/var/log/snort$ sudo tcpdump -r snort.log.1665381601
reading from file snort.log.1665381601, link-type EN10MB (Ethernet)
11:30:09.625700 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 1, length 64
11:30:10.626688 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 2, length 64
11:30:11.627072 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 3, length 64
11:30:12.628298 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 4, length 64
11:30:13.630351 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 5, length 64
11:30:14.631738 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 6, length 64
11:30:15.633914 IP 192.168.43.24 > 192.168.43.130: ICMP echo request, id 3252, s
eq 7, length 64
apeksha@apeksha-VirtualBox:/var/log/snort$
```

4. **Conclusion:** Hence we have successfully studied Snort which is network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be usedto detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Also we have done analysis of log generated by snort.