Implementation of Intrusion Detection System using Snort

Hamed Hilal Al Ghafri[1], Z. Zainal Abidin[2], Kamil Kurbonov[3] and Robiah Yusof[4]
[1,2,3,4]Centre for Advanced Computing Technology,
Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka, Malaysia.
Email [1]: m031610013@student.utem.edu.my

Abstract— Intrusion Detection System (IDS) is a vital network security tool for protecting the network systems that consists of software and hardware to monitor all the inbound and outbound network and system activities for malicious activities in the network traffic. The purpose of IDS is to assists the network administrator or the system by sending alerts and notifications when there are possible incidents, which violations of computer security policies exist. However, IDS causes a false alarm when attacker perform modifications at the rules settings. Therefore, this study performs experiment to detect anomaly incident and intruder in the network system. The implementation on snort development is provided and testing is executed in order to prove that snort capable to detect intruder. The findings showed that anomaly user can be detected based on port scanning, telnet to port to detect the unusual traffic and monitoring using NMAP to identify abnormal activities. As a result, the impact of Snort could bring an alternative solution on network monitoring in terms of continuous detection on unusual traffic movements, cost effective since Snort is an open source product and it can be customized to suit with the network environment.

Index Terms—Intrusion Detection System, Snort.

I. Introduction

Intrusion Detection System (IDS) is software, hardware or combination of both used to detect intruder activity [1]. IDS has different capabilities depending upon how complex and sophisticated the components are. IDS uses signatures, anomaly-based techniques or both [2]. There are so many types of open source IDS in the market and significantly these study explain about one of the open source IDS which is known as Snort. Snort is an IDS available to the general public. The first part of the study mainly about the a) Snort components b) Snort installation process and also involving the process of make sure the Snort is up and running. After that, it shows the c) Snort capabilities to identify the unusual traffic coming from the attackers. All the logs from the experiment is recorded and explained. Furthermore, the screen capture of evidence and the log evidence are also included in this study.

A. Intrusion Detection System

Intrusion Detection System (IDS) is a type of security system for computers. IDS helps to detects outer and inner attacks performed by attackers [3]. IDS collects information from various sources and analyzes information from many areas within a computer or a network to identify a potential security breaches from outside and inside attacks occurred at the organization [4].

The advantages of IDS are it tracks changes of behavior in a network, inspects normal and abnormal system activities and performed automated process for attacks or incidents detection [5]. In fact, IDS is a tool to early detection and mitigation process for Denial of Service (DoS) Attack [6], [7]. Meanwhile, the disadvantages of IDS are it sometimes give a false alarm [8], time consuming and not really safe from attacks.

B. Snort Components

Snort is a combination of components that work together to detect incidents as illustrated in Figure 1, and each of the component is elaborated as given information [9] and [10]:

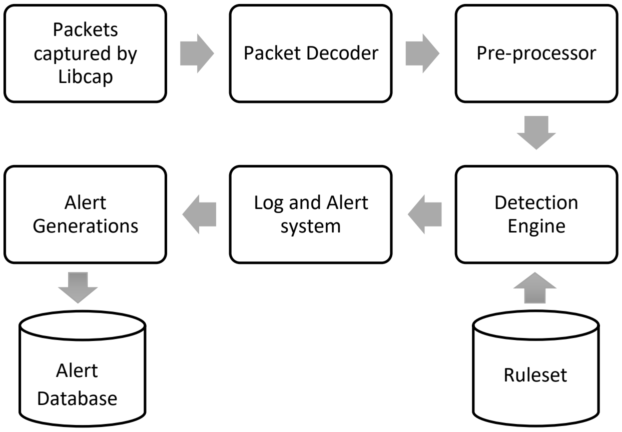


Fig. 1. Snort Components.

- Packet Decoder [11]

Packet decoder collects packets from various network interfaces and prepares the packet to be preprocessed or sent to detection engine. Ethernet, SLIP or PPP is example of network interface.

- Pre-processor [11]

The Pre-processor arranges the packet before detection engine applies some operation on the packet if the packet is corrupted. Sometimes, alert is generated if any anomalies are found in the packet. The intruder can bypass the IDS by changing or adding extra value at the sequence of string pattern. The major task perform by the pre-processor is the defragmentation. However, the attacker breaks the signature into two parts and sending them as two packets.

- Detection Engine

The detection engine takes the data from pre-processor and it's plug-in and that data is checked through a set of rules. If the rules match the data in the packet, they are sent to alert processor. Nonetheless, it takes some times to response since different packets represents different pattern and also depending on the machine that able to execute rules defined. This is because the load of detection engine depends on number of rules, computational of the machine, speed of internal bus used in Snort machine and number of load on network.

- Log and Alert System

Log and alert system is vital for packet generation and messages since the packet is useful for log activity or create alert. Logs are kept in simple text files or tcp-dump style files. The location of logs and alerts can be modified using *–l* command in the command prompt.

- Output Modules

The output module saves the output generated by the log and alert system of Snort into the real time alerts and other parameters for network administrator to evaluate the network performance of the network in an organization. Functions of output modules are following:

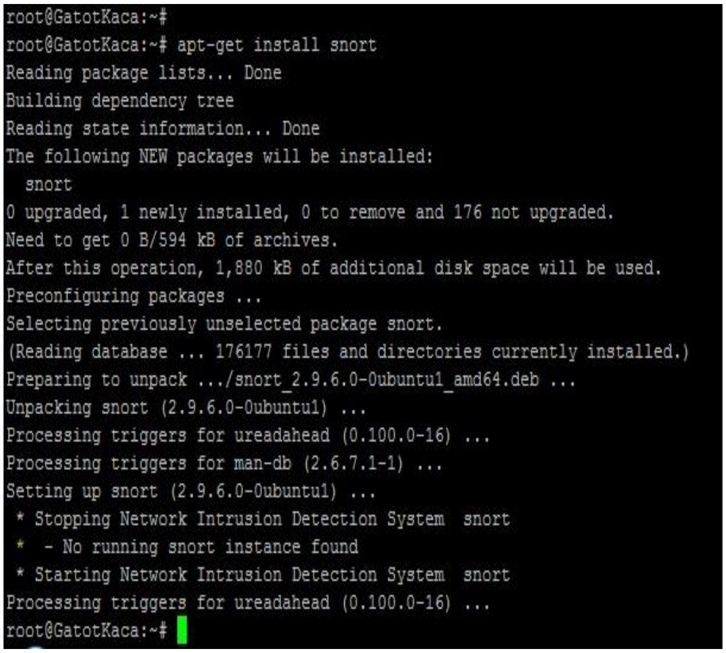
- Logging to /var/log/snort/alerts file or some other file
- Sending SNMP Traps
- Sending messages to syslog facility
- Logging to a database like MySQL or Oracle
- Generate extensible markup language (XML) output
- Modifying configuration on routers and firewalls
- Sending Server Message Block (SMB) messages to machines.

II. Snort Implementation on Debian

A. Snort Installation

Snort installation is able to be deployed on Linux Operating System or on Windows. The concept of installation and how to use is almost similar. After Linux OS ready to use and repository already pointing to correct and available Linux index repo, simply type following command from terminal/console (do it as root /super user). This command is for Debian Family, as illustrated in Figure 2.

- apt-get install Snort



```
root@GatotKaca:~#
root@GatotKaca:~# apt-get install snort
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  snort
0 upgraded, 1 newly installed, 0 to remove and 176 not upgraded.
Need to get 0 B/594 kB of archives.
After this operation, 1,880 kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously unselected package snort.
(Reading database ... 176177 files and directories currently installed.)
Preparing to unpack .../snort_2.9.6.0-0ubuntu1_amd64.deb ...
Unpacking snort (2.9.6.0-0ubuntu1) ...
Processing triggers for ureadahead (0.100.0-16) ...
Processing triggers for man-db (2.6.7.1-1) ...
Setting up snort (2.9.6.0-0ubuntu1) ...
 * Stopping Network Intrusion Detection System  snort
 *  - No running snort instance found
 * Starting Network Intrusion Detection System  snort
Processing triggers for ureadahead (0.100.0-16) ...
root@GatotKaca:~#
```

Fig. 2. Debian SNORT Installation.

After the installation process, in order to run Snort manually, the following command could be used to make sure the status of execution is enabled and stated as "OK".

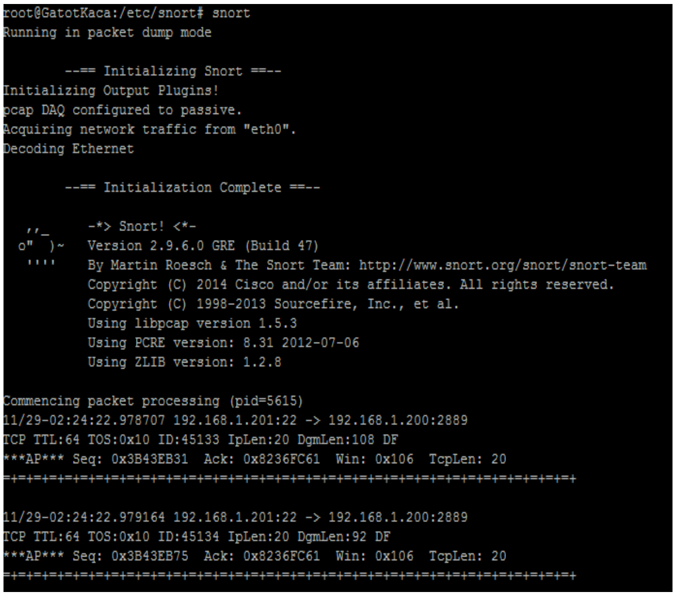
```
/etc/init.d/Snort {strat|stop|restart}

root@GatotKaca:~# /etc/init.d/Snort restart

*Stopping Network Intrusion Detection System Snort
[OK]

*Starting Network Intrusion Detection System Snort
[OK]
```

To test if the installation process is correct and Snort able to collect and capture traffic properly, performed the following command:

•"Snort" to set Snort on foreground state, and display to captured traffic on console immediately,

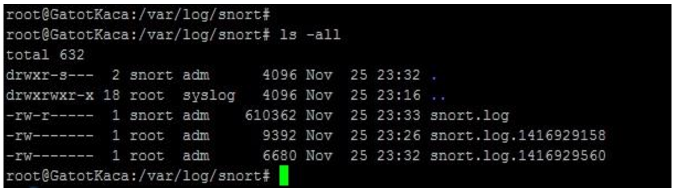
•Snort –D to set Snort on daemon state, output of captured traffic can see on Snort log folder /var/log/Snort

The output is shown in Figure 3. On the other hand, Figure 4 shows the Log Folder based on the administrator, power user and permission for reading or writing or viewing the files in the folder.



```
root@GatotKaca:/etc/snort# snort
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.6.0 GRE (Build 47)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.5.3
           Using PCRE version: 8.31 2012-07-06
           Using ZLIB version: 1.2.8

Commencing packet processing (pid=5615)
11/29-02:24:22.978707 192.168.1.201:22 -> 192.168.1.200:2889
TCP TTL:64 TOS:0x10 ID:45133 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0x3B43EB31  Ack: 0x8236FC61  Win: 0x106  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/29-02:24:22.979164 192.168.1.201:22 -> 192.168.1.200:2889
TCP TTL:64 TOS:0x10 ID:45134 IpLen:20 DgmLen:92 DF
***AP*** Seq: 0x3B43EB75  Ack: 0x8236FC61  Win: 0x106  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Fig. 3. Debian Snort Execution Test.

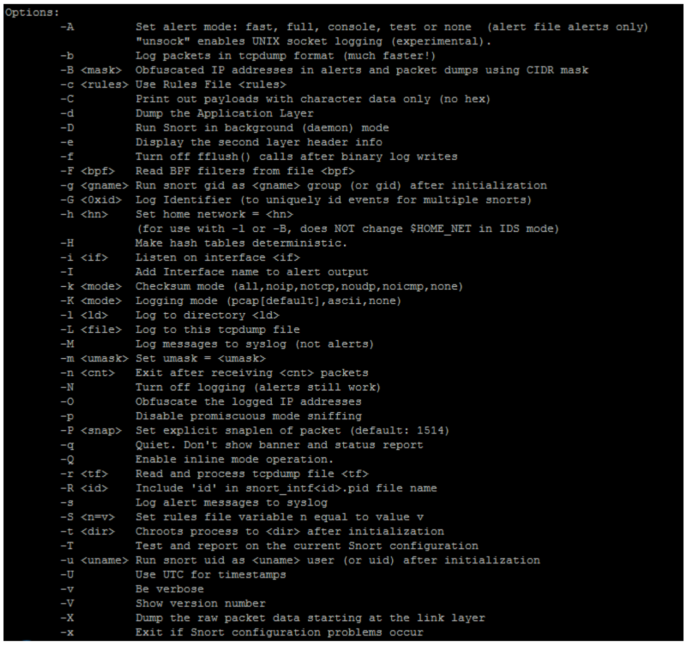


```
root@GatotKaca:/var/log/snort#
root@GatotKaca:/var/log/snort# ls -all
total 632
drwxr-s---  2 snort adm      4096 Nov  25 23:32 .
drwxrwxr-x 18 root  syslog   4096 Nov  25 23:16 ..
-rw-r-----  1 snort adm    610362 Nov  25 23:33 snort.log
-rw-------  1 root  adm      9392 Nov  25 23:26 snort.log.1416929158
-rw-------  1 root  adm      6680 Nov  25 23:32 snort.log.1416929560
root@GatotKaca:/var/log/snort#
```

Fig. 4. Debian Snort Log Folder.

B. SNORT Command Line

Figure 5 shows an options that can be added when using Snort, this simple option that usually only used for simple SNORT configuration (option taken from Snort help menu).



```
Options:
        -A         Set alert mode: fast, full, console, test or none  (alert file alerts only)
                   "unsock" enables UNIX socket logging (experimental).
        -b         Log packets in tcpdump format (much faster!)
        -B <mask>  Obfuscated IP addresses in alerts and packet dumps using CIDR mask
        -c <rules> Use Rules File <rules>
        -C         Print out payloads with character data only (no hex)
        -d         Dump the Application Layer
        -D         Run Snort in background (daemon) mode
        -e         Display the second layer header info
        -f         Turn off fflush() calls after binary log writes
        -F <bpf>   Read BPF filters from file <bpf>
        -g <gname> Run snort gid as <gname> group (or gid) after initialization
        -G <0xid>  Log Identifier (to uniquely id events for multiple snorts)
        -h <hn>    Set home network = <hn>
                   (for use with -l or -B, does NOT change $HOME_NET in IDS mode)
        -H         Make hash tables deterministic.
        -i <if>    Listen on interface <if>
        -I         Add Interface name to alert output
        -k <mode>  Checksum mode (all,noip,notcp,noudp,noicmp,none)
        -K <mode>  Logging mode (pcap[default],ascii,none)
        -l <ld>    Log to directory <ld>
        -L <file>  Log to this tcpdump file
        -M         Log messages to syslog (not alerts)
        -m <umask> Set umask = <umask>
        -n <cnt>   Exit after receiving <cnt> packets
        -N         Turn off logging (alerts still work)
        -O         Obfuscate the logged IP addresses
        -p         Disable promiscuous mode sniffing
        -P <snap>  Set explicit snaplen of packet (default: 1514)
        -q         Quiet. Don't show banner and status report
        -Q         Enable inline mode operation.
        -r <tf>    Read and process tcpdump file <tf>
        -R <id>    Include 'id' in snort_intf<id>.pid file name
        -s         Log alert messages to syslog
        -S <n=v>   Set rules file variable n equal to value v
        -t <dir>   Chroots process to <dir> after initialization
        -T         Test and report on the current Snort configuration
        -u <uname> Run snort uid as <uname> user (or uid) after initialization
        -U         Use UTC for timestamps
        -v         Be verbose
        -V         Show version number
        -X         Dump the raw packet data starting at the link layer
        -x         Exit if Snort configuration problems occur
```

Fig. 5. Debian SNORT Command Line Option.

III. Results and Findings

Snort has the capability to detect unusual incoming packet traffic in a network. Thus, based on the experiments conducted, several steps were implemented in obtaining the results. Firstly, sniffer mode, packet logger mode and network intrusion detection mode, which applied the components of IDS of packet decoder, pre-processor and detection engine. Secondly, the Log and Alert System; and Output Modules components are implemented in this experiment, which developed in Snort Rules, NMAP port scanning, setting alerts and monitoring phases.

A. Snort Sniffer Mode

In this mode, Snort acts commonly TCP dump program, it can display traffic in detail. Example of this mode can produce by following commands:

Snort –v

output show only TCP/IP packet header to the screen console.

```
11/29-03:00:12.783040 192.168.1.200:2889 -> 192.168.1.201:22 TCP
TTL:128 TOS:0x0 ID:1766 IpLen:20 DgmLen:40 DF ***A**** Seq:
0x8237039D Ack: 0x3B443DA9 Win: 0x3EC5 TcpLen: 20
```

Snort –dv

output show IP Address, TCP/UPD/ICMP headers.

```
11/29-03:00:46.864897      192.168.1.201:22      ->
192.168.1.200:2889  TCP  TTL:64  TOS:0x10  ID:45313
IpLen:20  DgmLen:172  DF  ***AP***  Seq:  0x3B447AC5
Ack: 0x82370661  Win: 0x106  TcpLen: 20 C5 CD 85 4A
43 72 A7 ED CC 75 9F 61 EC B4 EF 58  ...JCr...u.a...X
3C 3A 26 A6 BC 71 A8 E1 93 E8 E7 D4 E4 16 3A 9E
<:&..q.........:. 3F 74 37 F4 0D C0 15 EC 83 29 06 6F
44 F2 99 61  ?t7......).oD..a B3 3F F5 FD 18 72 AD
35 1C 95 9D 64 D8 1B C4 63  .?...r.5...d...c CA 15
B9 9B FF B7 02 11 31 E5 12 D2 5D FC 36 42
.........1...].6B F1 33 B8 53 20 C3 49 D5 8B BA 5D 10
76 53 E5 8C  .3.S .I...].vS.. CD 52 38 50 A5 C5 8C
A9 06 A3 06 90 B2 77 2A 34  .R8P.........w*4 8F F0
FC 28 4B 10 5F D1 FF 38 92 A2 84 47 B1 10
...(K._..8...G.. 05 7D B8 84                .}..
```

Snort –dev

output show detail information of the headers include data link layer headers.

```
11/29-03:01:10.014120      00:0C:29:2B:EE:C3      ->
00:50:56:C0:00:01          type:0x800          len:0x9A
192.168.1.201:22 -> 192.168.1.200:2889 TCP TTL:64
TOS:0x10 ID:45373 IpLen:20 DgmLen:140 DF ***AP***
Seq: 0x3B44CF69 Ack: 0x8237098D Win: 0x106 TcpLen:
20 F3 DC D6 8F D4 E9 B7 2C E6 91 DF 56 F3 1A AA 23
.......,...V...# 8E A9 64 96 F5 CD A1 7A C8 28 69 8D
F7 CB 1B E8  ..d....z.(i..... 8E B8 57 CF 94 9E DD
8A CB D9 E0 AE 43 ED BE 79  ..W..........C..y FB 7D
9D E0 8C 9F 2E 24 FA 6C B5 75 B9 48 DC EA
.}.....$.l.u.H.. EF A7 D9 59 26 47 0D C0 FE 26 5D 2B
1E 0A 24 08  ...Y&G...&]+..$. F9 05 E2 9B D4 40 1D
```

```
5F 13 B4 56 F6 25 D8 AA B0  .....@._..V.%... 25 B4
57 BB %.W.
```

B. Packet Logger Mode

In this mode all traffic that captured by Snort is stored on log folder. Before using this mode, make sure the folder is exists.

```
Snort –dev –L /var/log/Snort/Snort.log
Snort –dev –l /var/log/Snort/
```

C. Network Intrusion Detection System Mode (NIDS)

Snort captures and display all traffic packets and save them to the log file. In this mode, Snort applies all rules on every captured packet. If match with rules, Snort makes decision just by displaying it on the log or generate an alert. If packet does not match with any rules, it drops and Snort does not create any log. This command could be used to start snort on NIDS mode.

Snort –c /etc/Snort/Snort.conf

That command loads every line of **Snort.conf** and apply it as IDS like rules, ports, connecting folder and many more. Every log on every captured traffic that matched with Snort rules, execute -L command to pointing to the log folder.

Snort –L/var/log/Snort/Snort.log –c /etc/Snort/Snort.conf

D. Snort Rules

To analyze intruder or malicious activity, it is foremost in understanding the malicious pattern. The malicious pattern is a basic information to generate Snort rules. Snort detection is based on the Snort rules, which matched with the Snort rules in the database. Snort rules is a simple rules and easy to understand, because most of Snort rules write on single line and describes about basic structure of rule and how to use it. Snort rules are built from two logical parts, which are **Rule Header** and **Rule Option** [12]. Rule Header contains action that rule must be taken, while Rule Option comprises alert message and content criteria for matching with packet traffic. Table 1 is the basic structure of Rule Header and Rule Option.

TABLE I
Structure of Snort Rule Header and Rule Option

Rule Header	Rule Option

Inside the rule header consists of several rules in one data type or detecting multiples intrusion signatures as in Table 2.

TABLE II
Structure of Snort Rule Header

Action	Protocol	Address	Port	Direction	Message	Port

The first part of Snort Rule Header is action, which shows what action will Snort take when traffic data match with rules or when the condition on that rule is true. Following action that can be made by Snort:

- Pass – This action make Snort ignores traffic
- Log – This action used to Log traffic
- Alert – This action used to generate an alert when traffic matched with rule condition
- Activate – This action used to generate an alert and then activate another rule to check traffic (Nested rule)
- Dynamic – This rule related with activate rule, this rule can be activated waiting "activate" action on other rule (passive rule)
- User Defined Action – User can define their own action like saved as XML, send log via SNMP or send log to syslog.

The second part is the protocol that describes what kind of traffic type is compared with rule. The protocol type that understand by SNORT are IP, ICMP, TCP and UDP.

The third part is address. There are two part of address on Snort Rule which are Source Address and Destination Address. This address can be a single address or network address or any address (all address). The standard to write address for Snort rules is CIDR (Classless Inter Domain Routing). Snort supports to exclude addresses by using "!" symbol when define a Source Address or Destination Address. Example of address on Snort Rules.

- 192.168.1.0/24 define a network class C with 192.168.1.0 to 1.255 address range
- 192.168.1.100/32 define a single host with IP 192.168.1.100
- ![192.168.1.0/24] define network address not from 192.168.1.0/24 network address
- [192.168.1.0/24,192.168.2.0/24] define their rule is for 2 different network address

A few example of Snort rules:

```
•   alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET not on console"; flow:from_server,established; content:"not on system console"; nocase; reference:arachnids,365; classtype:bad-unknown; sid:717; rev:6;)
•   alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP"; dsize:0; itype:8; reference:arachnids,162; classtype:attempted-recon; sid:469; rev:3;)
```

The fourth part of the rule header is port. Port part is useful for incoming packet or outgoing packet to the particular port or range ports. However, port part only useful for TCP and UDP type of packet. If type of packet is IP or ICMP, the port part just ignore them. Following is the example of port part :

- Set to port 23 to capture only packet originate to Telnet Server
- Set to 80 to capture HTTP packet only
- Set to 1024:2018 for capture range ports
- Set to !53 for capture all packets except from/to port 53

The fifth part is the direction. Direction part shows the flow of source and destination address/port. There are three type of direction:

- -> this symbol show that part on the left this direction is source address and port whereas on the right side is destination address and port.
- <- this symbol show that the packets are traveling from right side to the left side.
- <-> this symbol show that packets are traveling on booth direction.

The sixth part is the Message or Content (Rule Option). This message part activates when all content on rule option is on true state. Table 3 shows the type of content that used by SNORT:

TABLE III
General Rule Snort (Manual)

Keyword	Description
msg	The msg tells the logging and alerting engine the message to print with the packet dump or alert.
reference	The reference allows rule to include references to external attack identification systems.
gid	The gid (generator id) is used to identify what part of Snort generates the event when a particular rule fires.
sid	The sid is used to identify Snort rules.
rev	The rev is used to identify revisions of Snort rules.
classtype	The classtype is used to categorize a rule as detecting an attack that is part of a more general type of attack class.
priority	The priority keyword assigns a severity level to rules.
metadata	The metadata allows a rule writer to embed additional information about the rule, which is in a key-value format.

E. Monitoring Alert

To monitor an alert, two things that must be accomplished, firstly the rule type and secondly, how to read Snort Log. In this section described how to read and analyze the log generated by Snort. The alert described that the source and destination port contains the same value that is 22 that communicate in the same interface. The source IP address is 192.168.1.2 that indicates an AP sending TCP packet to a host IP address of 192.168.1.200. The following logs shows the detail alert that has been monitored previously.

```
11/29-04:49:57.953171 192.168.1.200:2889 -> 192.168.1.201:22
TCP TTL:128 TOS:0x0 ID:4568 IpLen:20 DgmLen:92 DF
***AP*** Seq: 0x8237779D  Ack: 0x3B4A04A9  Win:  TcpLen: 20
```

- Date and time the packet was captured.
- Source IP address is 192.168.1.2.
- Source port number is 22.
- Destination IP address is 192.168.1.200.
- Destination port is 22.
- Transport layer protocol used in this packet is TCP.
- Time To Live or TTL value in the IP header part is 128.
- Type of Service or TOS value is 0x01.
- Packet ID is 4568.
- Length of IP header is 20.
- IP payload is 92 bytes long.
- Don't Fragment or DF bit is set in IP header.
- Two TCP flags A and P are on.
- TCP sequence number is 0x8237779D.
- Acknowledgement number in TCP header is 0x3B4A04A9.
- TCP Window field is 0x4029.
- TCP header length is 20.

F. Tuning IDS

Tuning an IDS can be done by using a signature that already created by SNORT community or user can build it themselves using their knowledge. This two method have different advantages and disadvantages. Before start to create the rule of Snort, placement of Snort sensor must considered too. Location of that sensor effects the rule of Snort, which information used to make decision which location is the best location to locate Snort sensor. In fact, decision need to be made on type of IDS to be used, which is NIDS or HIDS or both of them.

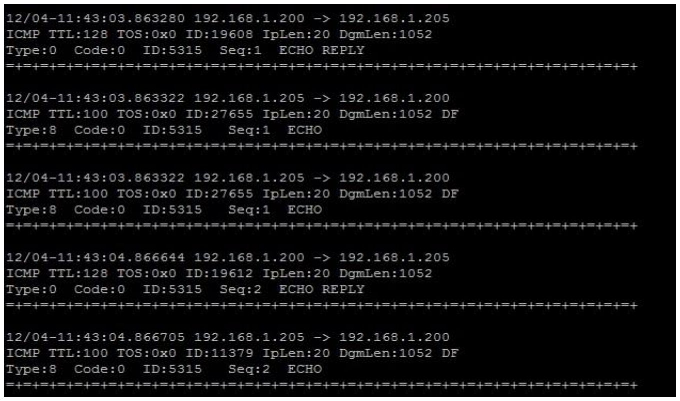
G. Unusual Traffic and Log Evidence Identification

- ICMP Unusual Packet

This scenario is to test an ICMP signature to detect unusual ICMP packet. By sending ICMP packet that have a big size and have not default Time to Life value. First we create the signature to handle that traffic.

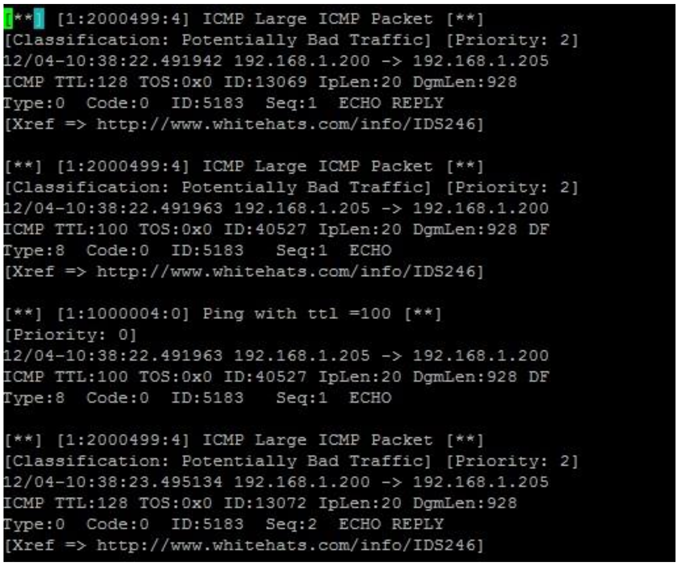
```
alert icmp any any -> $HOME_NET any (msg:"Ping
with ttl =100";ttl:=100;sid:1000004;)
alert icmp any any -> $HOME_NET any (msg:"ICMP
Large ICMP Packet"; dsize:>800; \
reference:arachnids,246; classtype:bad-unknown;
sid:2000499; rev:4;)
```

The signature inform that SNORT detects any ICMP packet from any address and any port that come to Home network with any port that have number of TTL 100 and have size more than 800 bytes. Figure 6 show the output log of SNORT and Figure 7 shows the alert log.



```
12/04-11:43:03.863280 192.168.1.200 -> 192.168.1.205
ICMP TTL:128 TOS:0x0 ID:19608 IpLen:20 DgmLen:1052
Type:0  Code:0  ID:5315  Seq:1  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-11:43:03.863322 192.168.1.205 -> 192.168.1.200
ICMP TTL:100 TOS:0x0 ID:27655 IpLen:20 DgmLen:1052 DF
Type:8  Code:0  ID:5315   Seq:1  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-11:43:03.863322 192.168.1.205 -> 192.168.1.200
ICMP TTL:100 TOS:0x0 ID:27655 IpLen:20 DgmLen:1052 DF
Type:8  Code:0  ID:5315   Seq:1  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-11:43:04.866644 192.168.1.200 -> 192.168.1.205
ICMP TTL:128 TOS:0x0 ID:19612 IpLen:20 DgmLen:1052
Type:0  Code:0  ID:5315  Seq:2  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-11:43:04.866705 192.168.1.205 -> 192.168.1.200
ICMP TTL:100 TOS:0x0 ID:11379 IpLen:20 DgmLen:1052 DF
Type:8  Code:0  ID:5315   Seq:2  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Fig. 6. Output SNORT Log.



```
[**] [1:2000499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-10:38:22.491942 192.168.1.200 -> 192.168.1.205
ICMP TTL:128 TOS:0x0 ID:13069 IpLen:20 DgmLen:928
Type:0  Code:0  ID:5183  Seq:1  ECHO REPLY
[Xref => http://www.whitehats.com/info/IDS246]

[**] [1:2000499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-10:38:22.491963 192.168.1.205 -> 192.168.1.200
ICMP TTL:100 TOS:0x0 ID:40527 IpLen:20 DgmLen:928 DF
Type:8  Code:0  ID:5183   Seq:1  ECHO
[Xref => http://www.whitehats.com/info/IDS246]

[**] [1:1000004:0] Ping with ttl =100 [**]
[Priority: 0]
12/04-10:38:22.491963 192.168.1.205 -> 192.168.1.200
ICMP TTL:100 TOS:0x0 ID:40527 IpLen:20 DgmLen:928 DF
Type:8  Code:0  ID:5183   Seq:1  ECHO

[**] [1:2000499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-10:38:23.495134 192.168.1.200 -> 192.168.1.205
ICMP TTL:128 TOS:0x0 ID:13072 IpLen:20 DgmLen:928
Type:0  Code:0  ID:5183  Seq:2  ECHO REPLY
[Xref => http://www.whitehats.com/info/IDS246]
```

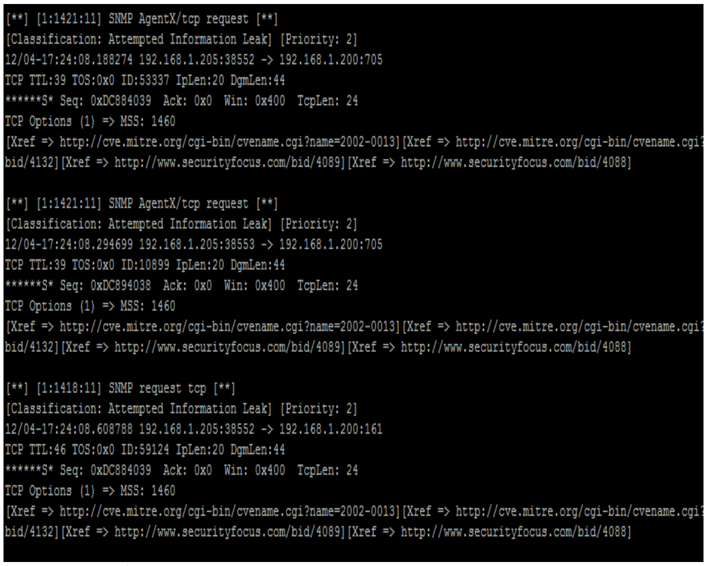
Fig. 7. Output SNORT Alert.

- Port Scanning using NMAP

Nmap is one of application that used by many attacker to scan the whole possible open port on targeted host. The method is by send SYN packet to all targeted host ports and wait for any ACK which inform attacker about the port status.

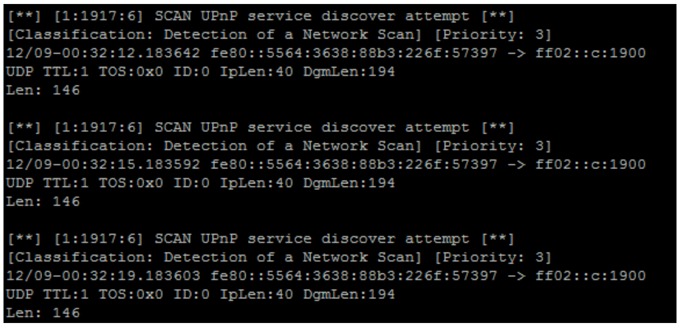
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 705
(msg:"SNMP AgentX/tcp request";
flow:stateless;flags:S+;\reference:bugtraq,4088;
reference:bugtraq,4089; reference:bugtraq,4132;
reference:cve,2002-0012;\
reference:cve,2002-0013; classtype:attempted-recon;
sid:1421; rev:11;)

alert udp $EXTERNAL_NET any -> $HOME_NET 1900
(msg:"SCAN UPnP service discover attempt";\content:"M-
SEARCH "; depth:9; content:"ssdp|3A|discover";
classtype:network-scan; sid:1917; rev:6;)
```

The first alert reads any traffic from external network to home network port 705 (protocol to communicate depending on the application), that have a stateless flow and a flag S packet as shown in Figure 8. Second alert detects every traffic that have M-Search content and on depth 9 that has ssdp or 3A or discover content as in Figure 9.



```
[**] [1:1421:11] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/04-17:24:08.188274 192.168.1.205:38552 -> 192.168.1.200:705
TCP TTL:39 TOS:0x0 ID:53337 IpLen:20 DgmLen:44
******S* Seq: 0xDC884039  Ack: 0x0  Win: 0x400  TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?
bid/4132][Xref => http://www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/4088]

[**] [1:1421:11] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/04-17:24:08.294699 192.168.1.205:38553 -> 192.168.1.200:705
TCP TTL:39 TOS:0x0 ID:10899 IpLen:20 DgmLen:44
******S* Seq: 0xDC894038  Ack: 0x0  Win: 0x400  TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?
bid/4132][Xref => http://www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/4088]

[**] [1:1418:11] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/04-17:24:08.608788 192.168.1.205:38552 -> 192.168.1.200:161
TCP TTL:46 TOS:0x0 ID:59124 IpLen:20 DgmLen:44
******S* Seq: 0xDC884039  Ack: 0x0  Win: 0x400  TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?
bid/4132][Xref => http://www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/4088]
```

Fig. 8. Alert of NMAP Port scanning.



```
[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/09-00:32:12.183642 fe80::5564:3638:88b3:226f:57397 -> ff02::c:1900
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:194
Len: 146

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/09-00:32:15.183592 fe80::5564:3638:88b3:226f:57397 -> ff02::c:1900
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:194
Len: 146

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/09-00:32:19.183603 fe80::5564:3638:88b3:226f:57397 -> ff02::c:1900
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:194
Len: 146
```

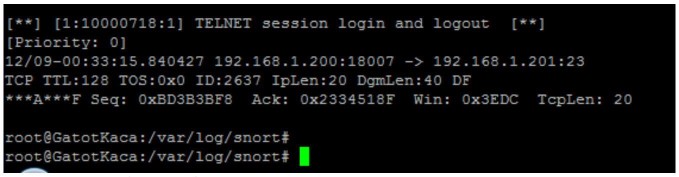
Fig. 9. Alert of NMAP Port scanning attempt.

- Telnet Session Login and Logout

Another example for unusual or incidents detection at any session login, comes from port 23 using Telnet log in session for instance:

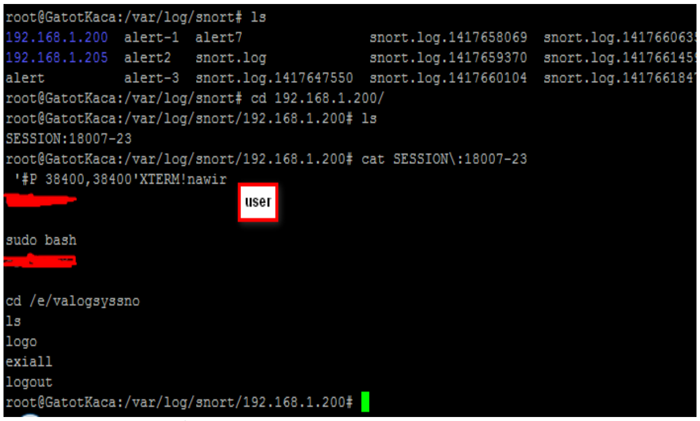
```
alert tcp any any - > $HOME_NET 23 (msg:"TELNET session login
and logout "; flags: A+F+; sid:10000718;\
rev:1;)
log tcp any any < > any 23 (session:printable; sid:2000001;)
```

This rule detects any traffic to home network port 23 that consist of flag ACK and FIN as in Figure 10, which means telnet session has been started and finished. Log out any session information traffic through port 23 as illustrated in Figure 11.



```
[**] [1:10000718:1] TELNET session login and logout  [**]
[Priority: 0]
12/09-00:33:15.840427 192.168.1.200:18007 -> 192.168.1.201:23
TCP TTL:128 TOS:0x0 ID:2637 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xBD3B3BF8  Ack: 0x2334518F  Win: 0x3EDC  TcpLen: 20

root@GatotKaca:/var/log/snort#
root@GatotKaca:/var/log/snort#
```

Fig. 10. Alert for any finished Telnet session.



```
root@GatotKaca:/var/log/snort# ls
192.168.1.200  alert-1  alert7             snort.log.1417658069  snort.log.141766063
192.168.1.205  alert2   snort.log          snort.log.1417659370  snort.log.141766145
alert          alert-3  snort.log.1417647550  snort.log.1417660104  snort.log.141766184
root@GatotKaca:/var/log/snort# cd 192.168.1.200/
root@GatotKaca:/var/log/snort/192.168.1.200# ls
SESSION:18007-23
root@GatotKaca:/var/log/snort/192.168.1.200# cat SESSION\:18007-23
 '#P 38400,38400'XTERM!nawir
[redacted]

sudo bash
[redacted]

cd /e/valogsyssno
ls
logo
exiall
logout
root@GatotKaca:/var/log/snort/192.168.1.200#
```

Fig. 11. Log session from Telnet session.

Based on the experiment results, shown that a user identity is detected as malicious incident since user typically need to access into the system. However, Figure 11 illustrated that the user is neither not log in nor log out. As a result, Snort is capable to detect malicious activity or anomaly incidents in the network system.

IV. Conclusion

As conclusion IDS is used to help the administrator to monitor the network and the host activity inside the organization. There are so many activity could be perform in the IDS to maintain the network security and alerts the administrator so that they could take an immediate actions if there is attacks from the intruders. On the other hand, the administrator also could fine-tuning IDS so that they could make their job easier and reduce the false negative alerts. The determining, tuning the IDS and understand the pattern would be easier when the administrator gain more knowledge and experience when applying the IDS in their environment.

Acknowledgment

The authors would like to thank the Center of Applied Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi (FTMK) and Universiti Teknikal Malaysia Melaka (UTeM).

References

[1] M.Chandu Jagan Sekhar, K.Tulasi, V.V.Amulya, D.Ravi Teja and M.Santhosh Kumar, “Implementation of IDS using Snort on Bayesian Network” International Journal of Computer Science and Mobile Computing, vol 4, no. 4, pp. 790-795, April 2015.

[2] A. Alazab, M. Hobbs, J. Abawajy, A. Khraisat, M. Alazb, “Using response action with intelligent intrusion detection and prevention system against web application malware”, Information Management and Computer Security, vol. 22, no.5, pp. 431-449, 2014.

[3] S. k. Kang, D. Lindskog and H. Samuel, "An Implementation of Hierarchical Intrusion Detection Systems using Snort and Federated Databases," 2018 17th IEEE International Conference On Trust, Security and Privacy in Computing and Communications 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, 2018, pp. 1521-1525. doi: 10.1109/TrustCom/BigDataSE.2018.00216.

[4] S. Kumawat, A. Kumawat and A.K. Sharma, “Intrusion Detection System and Prevention System in Cloud Computing using Snort” International Journal of Computer Engineering and Information Technology Research, vol. 5, no. 6, pp. 31-40, Dec 2015.

[5] A. Mikail and B. Pranggono, “Securing Infrastructure-as-a-Service Public Clouds using Security Onion”, Applied System Innovation, MDPI, vol. 2. No. 6, pp. 1-17, 2019.

[6] P. Manso, J. Moura and C. Serrão, “SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks”, Information, MDPI, vol. 10, no. 106, pp. 1-17, 2019.

[7] R. M. Almuttairi, M. K. Al-Anni and D. A. Aljburi, “Implementing Secure Cluster using Hadoop and Snort for Intrusion Detection” Journal of Engineering and Applied Sciences, vol. 13, no. 22, pp. 9789-9799, 2018.

[8] S. A. Raza Shah and B. Issac, “Performance comparison of intrusion detection systems and application of machine learning to Snort system” Future Generation Computer Systems, Elsevier, vol. 80, pp. 157-170, 2018.

[9] Goel and A. Vasishtha, “The implementation and assessment of snort capabilities”, International Journal of Computer Applications, vol. 167, pp. 15-23, June 2017.

[10] P. Agarwal and S. Satapathy, “Implementation of Signature-based Fetection System using Snort in Windows”, International Journal of Innovations and Advancement in Computer Science, vol. 3, no.3, pp. 120-127, May 2014.

[11] G. Ahmed, M. N. Ahmed Khan and M. S. Bashir, “A Linus-based IDPS using Snort” Computer Fraud and Security, vol 2015, no 8, pp. 13-18, 2015.

[12] N. Khamphakdee, N. Benjamas and S. Saiyod, “Improving Intrusion Detection System based on Snort Rules for Network Proble Attacks detection with Association rules technique of Data Mining”, journal of ICT Research and Application, Vol. 8, No.3, pp. 234-250, 2015.