

**Name:** Shashank Sarma

**UID:** 2019130053

**Class:** TE Comps

## Objectives

In this lab students will explore the Snort Intrusion Detection Systems. The students will study Snort IDS, a signature based intrusion detection system used to detect network attacks. Snort can also be used as a simple packet logger. For the purpose of this lab the students will use snort as a packet sniffer and write their own IDS rules.

## Software Requirement

All required files are packed and configured in the provided virtual machine image.

- The VMWare Software - <http://apps.eng.wayne.edu/MPStudents/Dreamspark.aspx>
- The ubuntu 14.04 or Ubuntu Long Term Support (LTS) version or Kali linux image
- The ubuntu 14.04 or Ubuntu 14.04 Long Term Support (LTS) Version
- Snort: A signature-based Intrusion Detection System <https://www.snort.org/#get-started>

## Implementation

### Starting the Lab 1 Virtual Machine

In this lab, we use Ubuntu as our VM image.

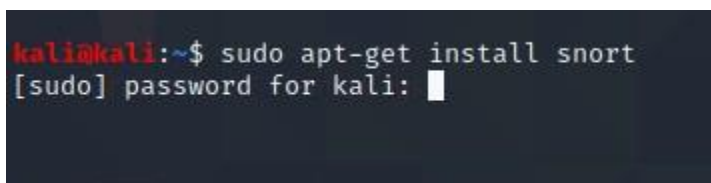
Login the Ubuntu image with username and password

### Installing Snort into the Operating System

To install the latest version of the snort, you can follow the installation instruction from the snort website. Note that installation instructions are vary from OSes. The instruction below shows how to install snort from its source code on Linux.

You can find more information here:

<https://www.snort.org/#get-started>



```
kali@kali:~$ sudo apt-get install snort
[sudo] password for kali: █
```

While you install the snort, your system may miss some libraries. You need to install the required libraries, too.

Snort is software created by Martin Roesch, which is widely used as Intrusion Prevention System [IPS] and Intrusion Detection System [IDS] in the network. It is separated into the five most important mechanisms for instance: Detection engine, Logging, and alerting system, a Packet decoder, Preprocessor, and Output modules.

The program is quite famous to carry out real-time traffic analysis, also used to detect query or attacks, packet logging on Internet Protocol networks, to detect malicious activity, denial of service attacks and port scans by monitoring network traffic, buffer overflows, server message block probes, and stealth port scans.

Snort can be configured in three main modes:

Sniffer mode: it will observe network packets and present them on the console.

Packet logger mode: it will record packets to the disk.

Intrusion detection mode: the program will monitor network traffic and analyze it against a rule set defined by the user.

After that, the application will execute a precise action depend upon what has been identified.

### **Configuring and Starting the Snort IDS**

After installing the Snort, we need to configure it. The configuration file of snort is stored at /etc/snort/snort.conf. The screenshot below shows the commands to configure the Snort. You need to switch to root to gain the permission to read the snort configurations file.

After configuring the Snort, you need to start the Snort. You can simply type the following command to start the service.

```
$ service snort start
```

```
snort start
```

```
kali@kali: ~  
File Actions Edit View Help  
kali@kali:~$ sudo service snort start  
kali@kali:~$ sudo service snort status  
● snort.service - LSB: Lightweight network intrusion detection system  
   Loaded: loaded (/etc/init.d/snort; generated)  
   Active: active (running) since Sat 2021-04-10 02:11:24 EDT; 5s ago  
     Docs: man:systemd-sysv-generator(8)  
  Process: 2560 ExecStart=/etc/init.d/snort start (code=exited, status=0>  
    Tasks: 4 (limit: 2319)  
   Memory: 170.4M  
      CPU: 1.175s  
   CGroup: /system.slice/snort.service  
           └─2609 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snor>  
             └─2623 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snor>  
  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_SSH >  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_SDF >  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: appid V>  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_IMAP >  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_DNS >  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_GTP >  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_FTPTE>  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_SSLPP>  
Apr 10 02:11:24 kali snort[2623]:      Preprocessor Object: SF_DCERP>  
Apr 10 02:11:24 kali snort[2623]: Commencing packet processing (pid=2623)  
kali@kali:~$ sudo ls /etc/snort/rules/  
attack-responses.rules      icmp-info.rules  
backdoor.rules              icmp.rules
```

## Snort Rules

Snort is a signature-based IDS, and it defines rules to detect the intrusions. All rules of Snort are stored under /etc/snort/rules directory. The screenshot below shows the files that contain rules of Snort.

```
$ ls /etc/snort/rules
```

```
kali@kali: ~  
File Actions Edit View Help  
community-inappropriate.rules  p2p.rules  
community-mail-client.rules    policy.rules  
community-misc.rules           pop2.rules  
community-nntp.rules           pop3.rules  
community-oracle.rules         porn.rules  
community-policy.rules         rpc.rules  
community-sip.rules            rservices.rules  
community-smtp.rules           scan.rules  
community-sql-injection.rules  shellcode.rules  
community-virus.rules          smtp.rules  
community-web-attacks.rules    snmp.rules  
community-web-cgi.rules        sql.rules  
community-web-client.rules     telnet.rules  
community-web-dos.rules        tftp.rules  
community-web-iis.rules        virus.rules  
community-web-misc.rules       web-attacks.rules  
community-web-php.rules        web-cgi.rules  
ddos.rules                    web-client.rules  
deleted.rules                 web-coldfusion.rules  
dns.rules                    web-frontpage.rules  
dos.rules                    web-iis.rules  
experimental.rules           web-misc.rules  
exploit.rules                web-php.rules  
finger.rules                 x11.rules  
ftp.rules
```

## Writing and Adding a Snort Rule

Next, we are going to add a simple snort rule. You should add your own rules at /etc/snort/rules/local.rules. Add the following line into the local.rules file

```
/etc/snort/rules/local.rules - Sublime Text (UNREGISTERED)  
File Edit Selection Find View Goto Tools Project Preferences Help  
local.rules x  
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $  
2 # -----  
3 # LOCAL RULES  
4 # -----  
5 # This file intentionally does not come with signatures. Put your local  
6 # additions here.  
7 alert icmp any any -> any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)]
```

Basically, this rule defines that an alert will be logged if an ICMP packet is found. The ICMP packet could be from any IP address and the rule ID is 1000001. e.g. Make sure to pick a SID greater 1000000 for your own rules.

To make the rule become effective, you need to restart the snort service by typing the following command.

```
$ service snort restart
```

```
kali@kali:~$ sudo subl /etc/snort/rules/local.rules  
kali@kali:~$ sudo service snort restart  
kali@kali:~$
```

## Triggering an Alert for the New Rule

To trigger an alert for the new rule, you only need to send an ICMP message to the VM image where snort runs. First, you need to find the IP address of the VM by typing the following command.

After you have a terminal, you can just type the following command to send ping messages to the VM.

```
C:\Users\91932>ping 192.168.56.103

Pinging 192.168.56.103 with 32 bytes of data:
Reply from 192.168.56.103: bytes=32 time=1ms TTL=64
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64
Reply from 192.168.56.103: bytes=32 time=2ms TTL=64

Ping statistics for 192.168.56.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\Users\91932>_
```

After you send the ping messages, the alerts should be triggered and you can find the log messages in /var/log/snort/snort.log. However, the snort.log file will be binary format. You need to use a tool, called u2spewfoo, to read it. Observer terminal on screen with log where you can see that the SID is 1000001, and the alerts are generated by the ICMP messages.

```
04/10-02:44:47.489819  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 → 192.168.56.103
04/10-02:44:47.489832  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.103 → 192.168.56.1
04/10-02:44:48.492111  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 → 192.168.56.103
04/10-02:44:48.492132  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.103 → 192.168.56.1
04/10-02:44:49.496694  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 → 192.168.56.103
04/10-02:44:49.496717  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.103 → 192.168.56.1
04/10-02:44:50.501882  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 → 192.168.56.103
04/10-02:44:50.502479  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.103 → 192.168.56.1
```



```

(Event)
  sensor id: 0      event id: 161      event second: 1616319409      event microsecond: 668938
  sig id: 1000001  gen id: 1          revision: 1      classification: 0
  priority: 0      ip source: 192.168.56.1 ip destination: 192.168.56.103
  src port: 8      dest port: 0        protocol: 1      impact_flag: 0 blocked: 0
  mpls label: 0    vland id: 0        policy id: 0     appid:

Packet
  sensor id: 0      event id: 161      event second: 1616319409
  packet second: 1616319409 packet microsecond: 668938
  linktype: 1      packet_length: 74
[  0] 08 00 27 34 AB 50 0A 00 27 00 00 12 08 00 45 00  .. '4.P..'.....E.
[ 16] 00 3C 27 66 00 00 80 01 21 A3 C0 A8 38 01 C0 A8  .<'f....! ... 8...
[ 32] 38 66 08 00 4D 3F 00 01 00 1C 61 62 63 64 65 66  8f..M?...abcdef
[ 48] 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76  ghijklmnopqrstuv
[ 64] 77 61 62 63 64 65 66 67 68 69                    wabcdefghi

(Event)
  sensor id: 0      event id: 162      event second: 1616319409      event microsecond: 668961
  sig id: 1000001  gen id: 1          revision: 1      classification: 0
  priority: 0      ip source: 192.168.56.103 ip destination: 192.168.56.1
  src port: 0      dest port: 0        protocol: 1      impact_flag: 0 blocked: 0
  mpls label: 0    vland id: 0        policy id: 0     appid:

Packet
  sensor id: 0      event id: 162      event second: 1616319409
  packet second: 1616319409 packet microsecond: 668961
  linktype: 1      packet_length: 74
[  0] 0A 00 27 00 00 12 08 00 27 34 AB 50 08 00 45 00  .. '.....'4.P..E.
[ 16] 00 3C 0D EB 00 00 40 01 7B 1E C0 A8 38 66 C0 A8  .<....@.{ ... 8f..
[ 32] 38 01 00 00 55 3F 00 01 00 1C 61 62 63 64 65 66  8 ... U?...abcdef
[ 48] 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76  ghijklmnopqrstuv
[ 64] 77 61 62 63 64 65 66 67 68 69                    wabcdefghi
(Event)

```

## Assignments for Lab

1. Read the lab instructions above and finish all the tasks.
2. Answer the questions and justify your answers. Simple yes or no answer will not get any credits.
  - a. What is a zero-day attack?
    - When a hacker manages to exploit the vulnerability before software developers can find a fix, that exploit becomes known as a zero-day attack.
    - Zero day attack can take almost any form, because they can manifest as any type of broader software vulnerability. For example, they could take the form of missing data encryption, missing authorizations, broken algorithms, URL redirects, bugs, or problems with password security.
    - This makes zero day attack difficult to proactively find—which in some ways is good news, because it also means hackers will have a hard time finding them. But it also means it's difficult to guard against these vulnerabilities effectively.
    - Hulu is a streaming platform which was the victim of a zero day attack in recent times.

b. Can Snort catch zero-day network attacks? If not, why not? If yes, how?

- Since snort checks with the predefined rules for prevention of attacks and zero-day attacks are unknown to the developers, so without the rules it cannot be prevented, so, snort can't catch zero-day network attacks.
- The results from a study show that Snort clearly is able to detect zero-days' (a mean of 14% detection). The detection rate is however on overall greater for theoretically known attacks (a mean of 54% detection).

c. Given a network that has 1 million connections daily where 0.1% (not 10%) are attacks. If the IDS has a true positive rate of 95%, and the probability that an alarm is an attack is 95%. What is the false alarm rate?

Number of attacks on the network = 0.1% of 1000000 = 1000 attacks

Number of benign events = 1000000 - 1000 = 999000 events

IDS has a true positive rate of 95% means that out of 1000 attacks, only 950 will set off alarms.

Therefore, Number of true alarms = 950 alarms (actual attacks)

Since 95% of the total alarms are attacks

Number of total alarms =  $(100 * 950) / 95 = 1000$  alarms

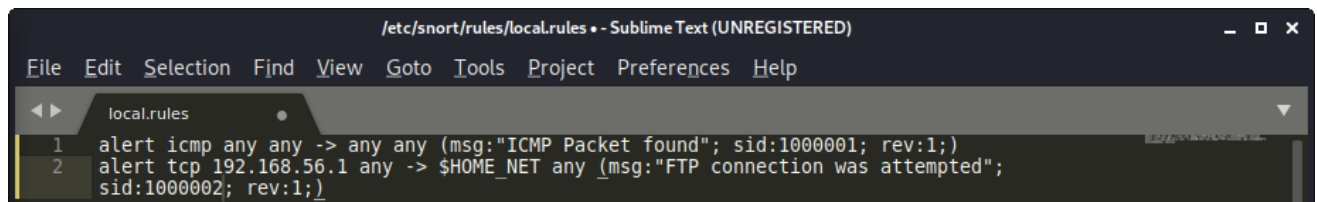
Therefore, Number of false alarms =  $1000 - 950 = 50$  alarms.

Therefore, False Alarm Rate =  $(\text{Number of false alarms} / \text{Total Benign Events}) * 100$

$= (50 / 999000) * 100 = \mathbf{0.005\%}$

3. Write and add another snort rule and show me you trigger it.

a. The rule you added (from the rules file)



```
File Edit Selection Find View Goto Tools Project Preferences Help
local.rules
1 alert icmp any any -> any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
2 alert tcp 192.168.56.1 any -> $HOME_NET any (msg:"FTP connection was attempted";
sid:1000002; rev:1;)
```

b. A description of how you triggered the alert. The alert itself from the log file (after converting it to readable text)

```
C:\Users\91932>ftp 192.168.56.103
```

```
Commencing packet processing (pid=3447)
04/10-03:06:38.080466  [**] [1:1000002:1] FTP connection was attempted [**] [Priority: 0] {TCP} 192.168.56.1:5
1369 → 192.168.56.103:21
04/10-03:06:38.569389  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a
Network Scan] [Priority: 3] {UDP} 192.168.56.1:53591 → 239.255.255.250:1900
04/10-03:06:38.580713  [**] [1:1000002:1] FTP connection was attempted [**] [Priority: 0] {TCP} 192.168.56.1:5
1369 → 192.168.56.103:21
04/10-03:06:39.082481  [**] [1:1000002:1] FTP connection was attempted [**] [Priority: 0] {TCP} 192.168.56.1:5
1369 → 192.168.56.103:21
```

(Event)

```
sensor id: 0      event id: 4      event second: 1616325191      event microsecond: 16512
sig id: 1000002  gen id: 1      revision: 1      classification: 0
priority: 0      ip source: 192.168.56.1 ip destination: 192.168.56.103
src port: 10987  dest port: 21  protocol: 6      impact_flag: 0  blocked: 0
mpls label: 0    vland id: 0      policy id: 0      appid:
```

Packet

```
sensor id: 0      event id: 4      event second: 1616325191
packet second: 1616325191      packet microsecond: 16512
linktype: 1      packet_length: 66
```

```
[  0] 08 00 27 34 AB 50 0A 00 27 00 00 12 08 00 45 00  ..'4.P..'.....E.
[ 16] 00 34 27 79 40 00 80 06 E1 92 C0 A8 38 01 C0 A8  .4'y@.....8 ...
[ 32] 38 66 2A EB 00 15 8D C8 B1 30 00 00 00 00 80 02  8f*.....0.....
[ 48] 20 00 F3 66 00 00 02 04 05 84 01 03 03 00 01 01  ..f.....
[ 64] 04 02  ..
```

(Event)

```
sensor id: 0      event id: 5      event second: 1616325191      event microsecond: 518017
sig id: 1000002  gen id: 1      revision: 1      classification: 0
priority: 0      ip source: 192.168.56.1 ip destination: 192.168.56.103
src port: 10987  dest port: 21  protocol: 6      impact_flag: 0  blocked: 0
mpls label: 0    vland id: 0      policy id: 0      appid:
```

Packet

```
sensor id: 0      event id: 5      event second: 1616325191
packet second: 1616325191      packet microsecond: 518017
linktype: 1      packet_length: 66
```

```
[  0] 08 00 27 34 AB 50 0A 00 27 00 00 12 08 00 45 00  ..'4.P..'.....E.
[ 16] 00 34 27 7A 40 00 80 06 E1 91 C0 A8 38 01 C0 A8  .4'z@.....8 ...
[ 32] 38 66 2A EB 00 15 8D C8 B1 30 00 00 00 00 80 02  8f*.....0.....
[ 48] 20 00 F3 66 00 00 02 04 05 84 01 03 03 00 01 01  ..f.....
[ 64] 04 02  ..
```

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



## CONCLUSION

- I learned what Intrusion Detection Systems (IDS) are and how are they used to detect packets received by a terminal and set rules for them.
- I learned about Snort IDS, how it works and how to send alerts to the terminal on setting a particular rule.
- I learned about zero-day attacks and the effect of zero-day attacks on Intrusion detection system efficiency.