CEL 62, CSS, Spring 2021
Lab 6: Firewall & Intrusion Detection System

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 Reequirment

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 ubantu 14.04 or Ubantu Long Term Support (LTS) versionor Kali linux image
- The ubantu 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.

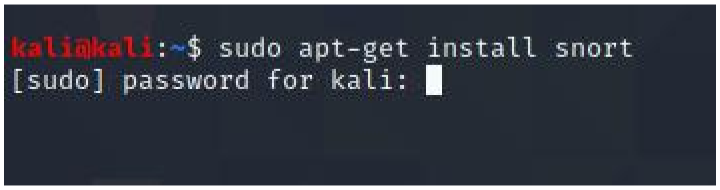
Login the Ubuntu image with username and password

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, you 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.

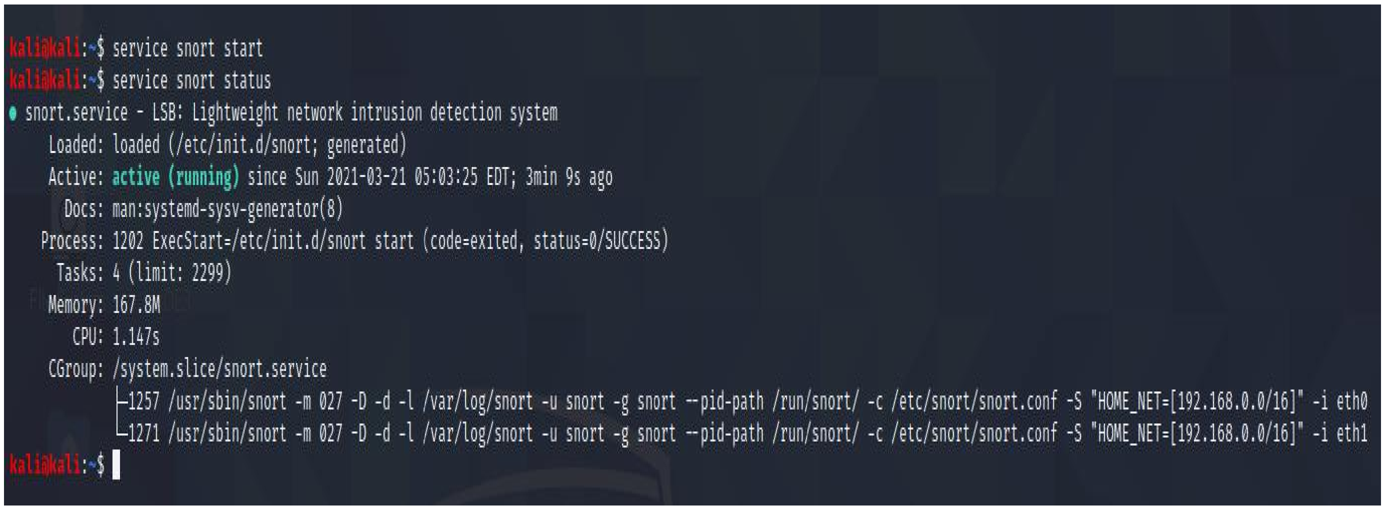
Configuring and Starting the Snort IDS

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

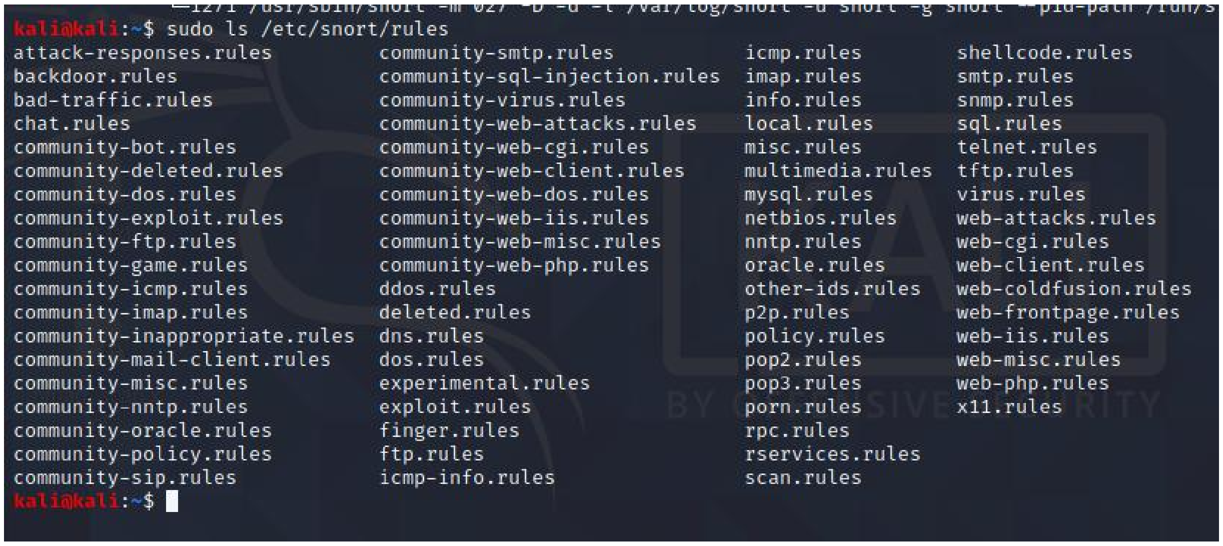


```
kali@kali:~$ service snort start
kali@kali:~$ service snort status
● snort.service - LSB: Lightweight network intrusion detection system
     Loaded: loaded (/etc/init.d/snort; generated)
     Active: active (running) since Sun 2021-03-21 05:03:25 EDT; 3min 9s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 1202 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
      Tasks: 4 (limit: 2299)
     Memory: 167.8M
        CPU: 1.147s
     CGroup: /system.slice/snort.service
             ├─1257 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort --pid-path /run/snort/ -c /etc/snort/snort.conf -S "HOME_NET=[192.168.0.0/16]" -i eth0
             └─1271 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort --pid-path /run/snort/ -c /etc/snort/snort.conf -S "HOME_NET=[192.168.0.0/16]" -i eth1
kali@kali:~$
```

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.

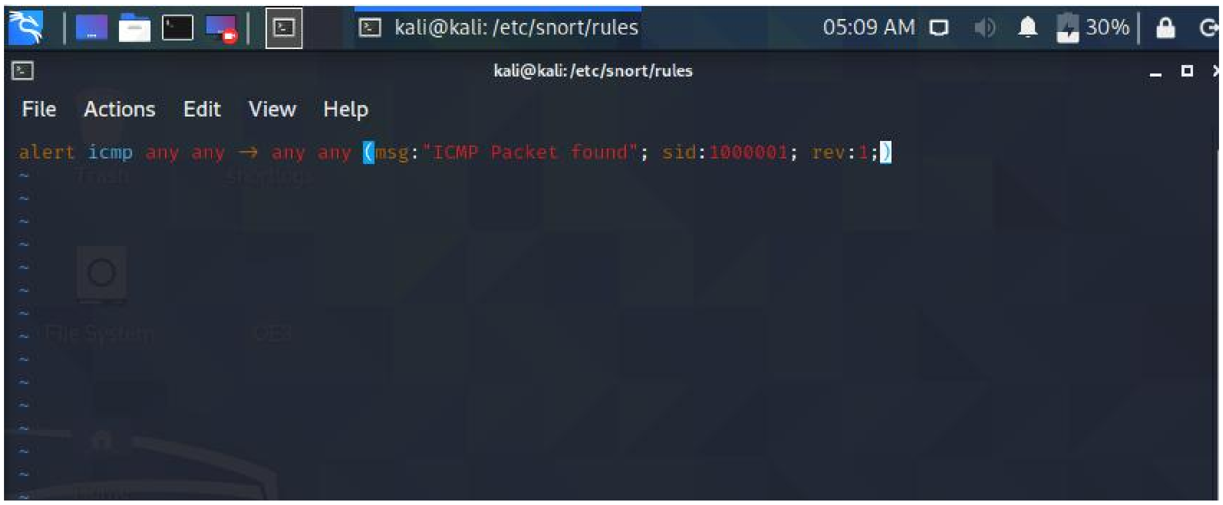
$ ls /etc/snort/rules



```
    └─1271 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort --pid-path /run/s
kali@kali:~$ sudo ls /etc/snort/rules
attack-responses.rules          community-smtp.rules            icmp.rules        shellcode.rules
backdoor.rules                  community-sql-injection.rules   imap.rules        smtp.rules
bad-traffic.rules               community-virus.rules           info.rules        snmp.rules
chat.rules                      community-web-attacks.rules     local.rules       sql.rules
community-bot.rules             community-web-cgi.rules         misc.rules        telnet.rules
community-deleted.rules         community-web-client.rules      multimedia.rules  tftp.rules
community-dos.rules             community-web-dos.rules         mysql.rules       virus.rules
community-exploit.rules         community-web-iis.rules         netbios.rules     web-attacks.rules
community-ftp.rules             community-web-misc.rules        nntp.rules        web-cgi.rules
community-game.rules            community-web-php.rules         oracle.rules      web-client.rules
community-icmp.rules            ddos.rules                      other-ids.rules   web-coldfusion.rules
community-imap.rules            deleted.rules                   p2p.rules         web-frontpage.rules
community-inappropriate.rules   dns.rules                       policy.rules      web-iis.rules
community-mail-client.rules     dos.rules                       pop2.rules        web-misc.rules
community-misc.rules            experimental.rules              pop3.rules        web-php.rules
community-nntp.rules            exploit.rules                   porn.rules        x11.rules
community-oracle.rules          finger.rules                    rpc.rules
community-policy.rules          ftp.rules                       rservices.rules
community-sip.rules             icmp-info.rules                 scan.rules
kali@kali:~$
```

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



```
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.

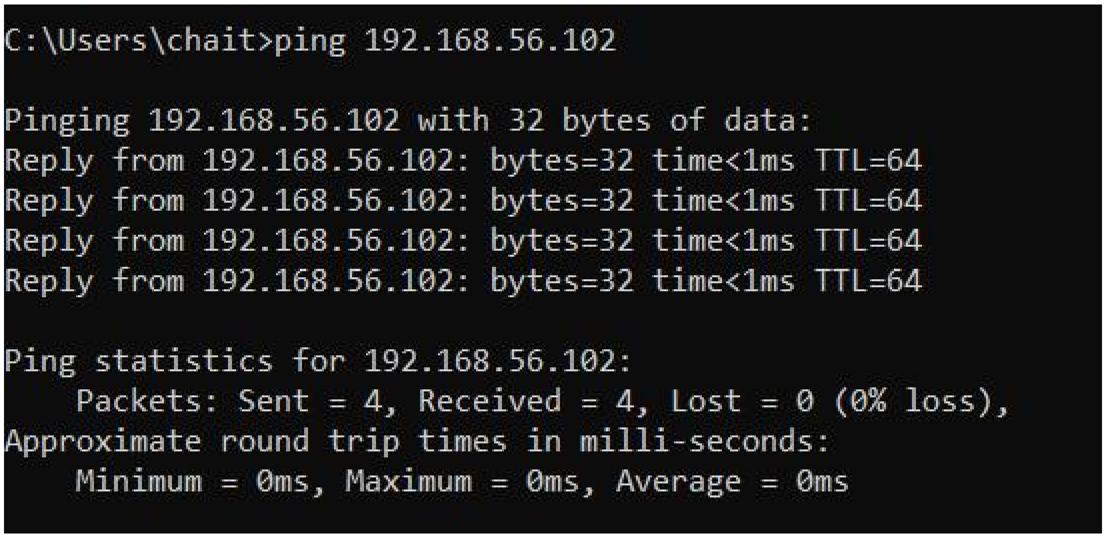
To make the rule become effective, you need to restart the snort service by typing the following command.

$ service snort restart

Triggering an Alert for the New Rule

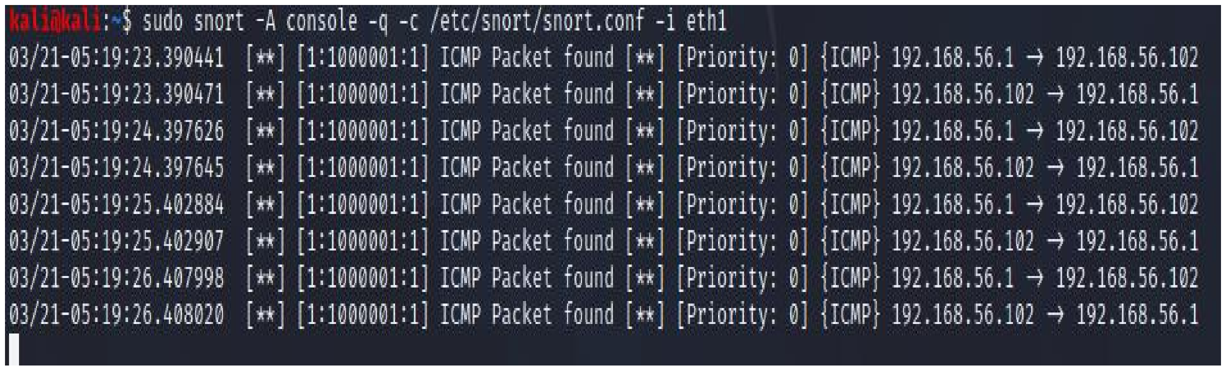
To trigger an alert for the new rule, you only need to send an ICMP message to the VM image where snort runs. First, you need to find the IP address of the VM by typing the following command.

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

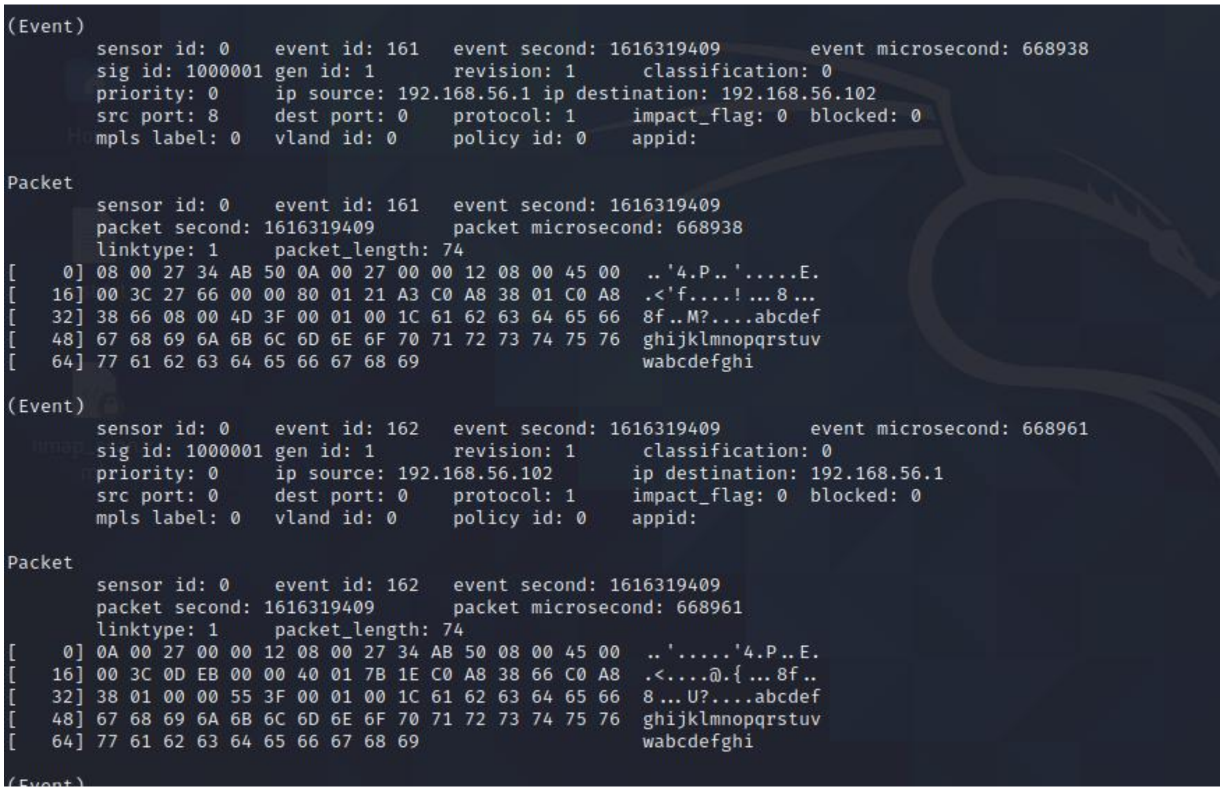


```
C:\Users\chait>ping 192.168.56.102

Pinging 192.168.56.102 with 32 bytes of data:
Reply from 192.168.56.102: bytes=32 time<1ms TTL=64
Reply from 192.168.56.102: bytes=32 time<1ms TTL=64
Reply from 192.168.56.102: bytes=32 time<1ms TTL=64
Reply from 192.168.56.102: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.102:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

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.



```
kali@kali:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i eth1
03/21-05:19:23.390441  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.102
03/21-05:19:23.390471  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.102 -> 192.168.56.1
03/21-05:19:24.397626  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.102
03/21-05:19:24.397645  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.102 -> 192.168.56.1
03/21-05:19:25.402884  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.102
03/21-05:19:25.402907  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.102 -> 192.168.56.1
03/21-05:19:26.407998  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.1 -> 192.168.56.102
03/21-05:19:26.408020  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.56.102 -> 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: 192.168.56.102
        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: 192.168.56.102       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
```

Assignments for Lab 1

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?

- If 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 vulnerabilities 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, SQL injection, buffer overflows, missing authorizations, broken algorithms, URL redirects, bugs, or problems with password security.
- This makes zero day vulnerabilities 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.

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

- No, snort cannot catch zero-day attack. As 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,

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 thatan 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 = 99.9% of 1000000 = 999000 events

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

Therefore, Number of true alarms = 950 alarms

Since 95% of the total alarms are attacks, Number of total alarms = (100 * 950) / 95 = 1000 alarms

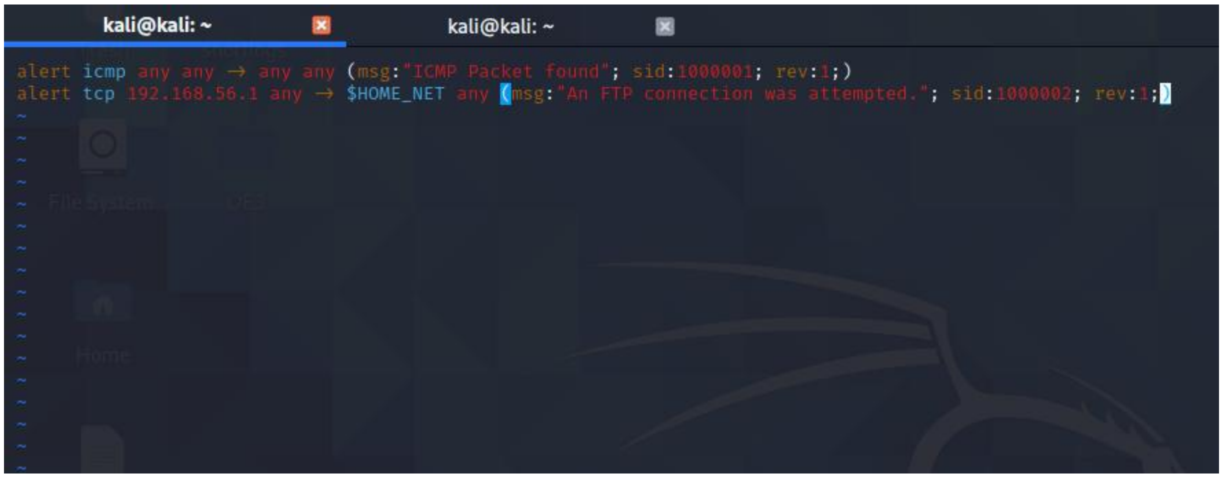
Therefore, Number of false alarms = 1000 – 950 = 50 alarms.

Therefore, False Alarm Rate = (Number of false alarms / Total Benign Events) * 100

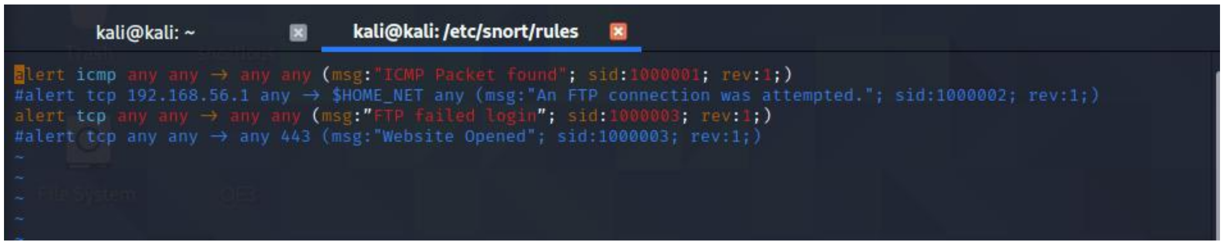
= (50 / 999000) * 100 = **0.005%**

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

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

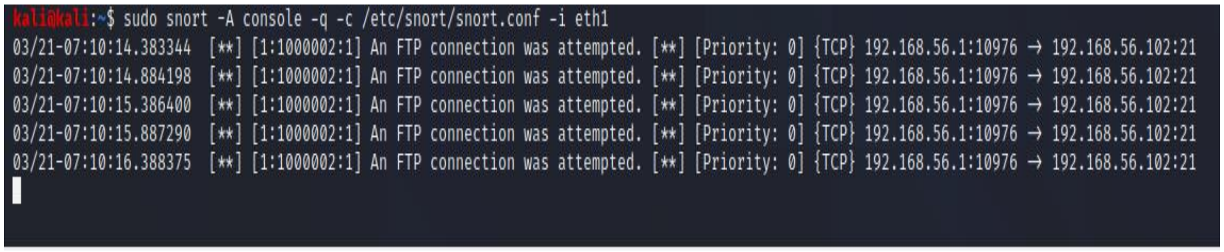


```
alert icmp any any → any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
alert tcp 192.168.56.1 any → $HOME_NET any (msg:"An FTP connection was attempted."; sid:1000002; rev:1;)
~
```

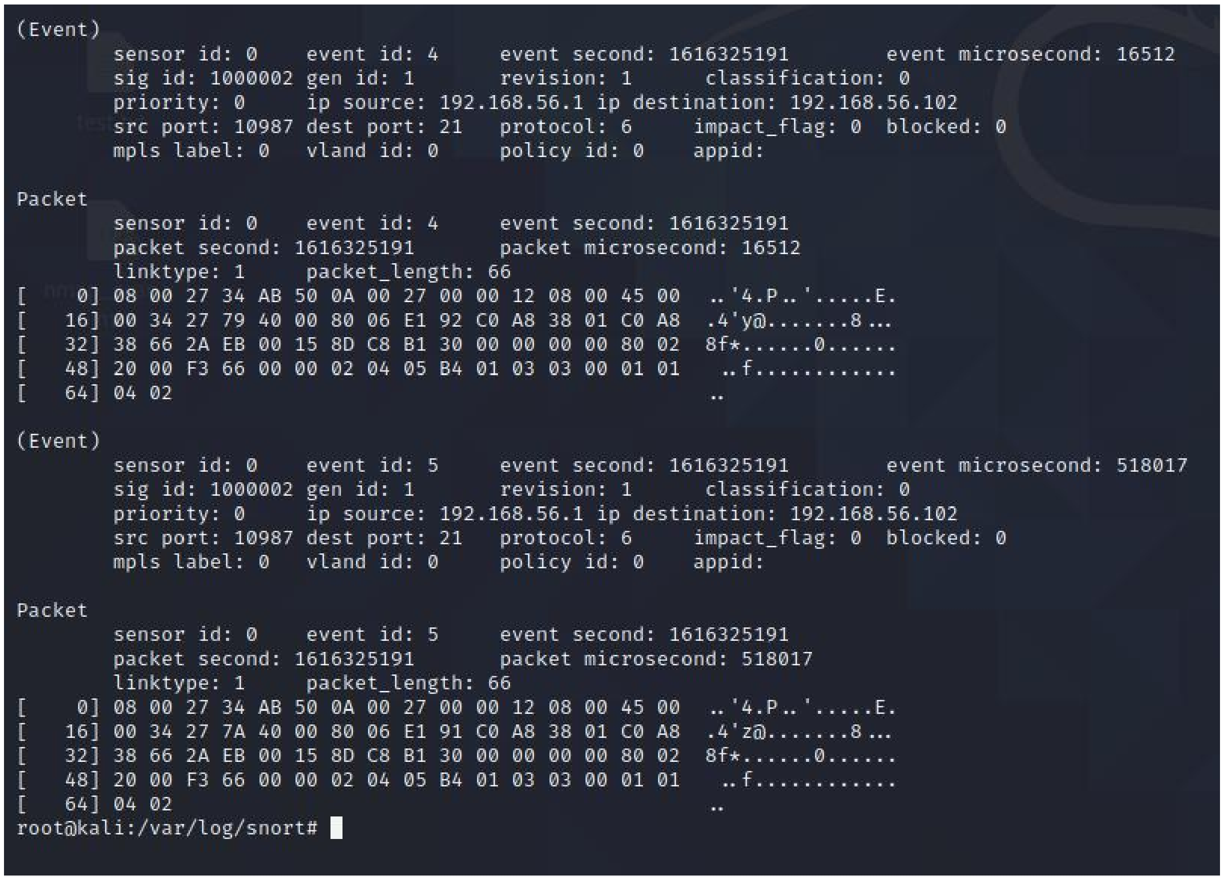


```
alert icmp any any → any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
#alert tcp 192.168.56.1 any → $HOME_NET any (msg:"An FTP connection was attempted."; sid:1000002; rev:1;)
alert tcp any any → any any (msg:"FTP failed login"; sid:1000003; rev:1;)
#alert tcp any any → any 443 (msg:"Website Opened"; sid:1000003; rev:1;)
~
```

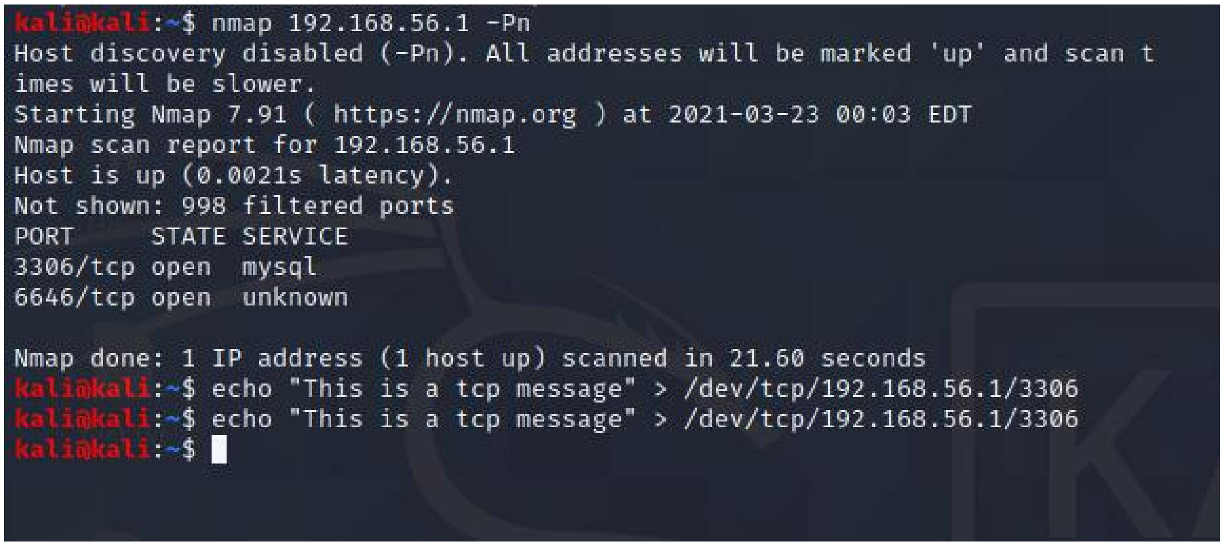
b. A description of how you triggered the alertc.The alert itself from the log file (after converting it to readable text)



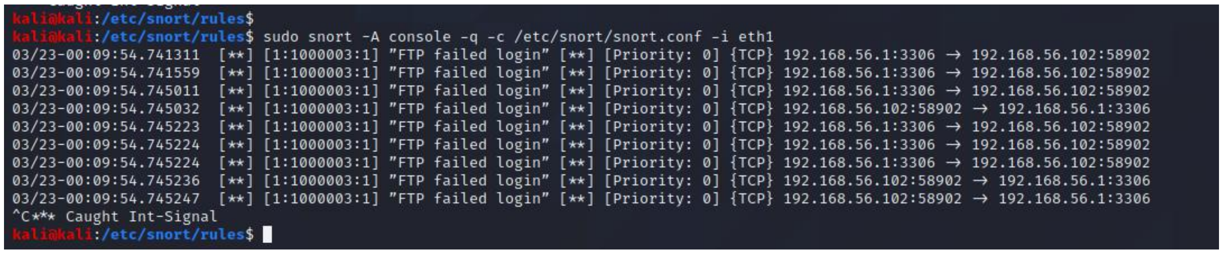
```
kali@kali:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i eth1
03/21-07:10:14.383344  [**] [1:1000002:1] An FTP connection was attempted. [**] [Priority: 0] {TCP} 192.168.56.1:10976 → 192.168.56.102:21
03/21-07:10:14.884198  [**] [1:1000002:1] An FTP connection was attempted. [**] [Priority: 0] {TCP} 192.168.56.1:10976 → 192.168.56.102:21
03/21-07:10:15.386400  [**] [1:1000002:1] An FTP connection was attempted. [**] [Priority: 0] {TCP} 192.168.56.1:10976 → 192.168.56.102:21
03/21-07:10:15.887290  [**] [1:1000002:1] An FTP connection was attempted. [**] [Priority: 0] {TCP} 192.168.56.1:10976 → 192.168.56.102:21
03/21-07:10:16.388375  [**] [1:1000002:1] An FTP connection was attempted. [**] [Priority: 0] {TCP} 192.168.56.1:10976 → 192.168.56.102:21
```



```
(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: 192.168.56.102
        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 B4 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: 192.168.56.102
        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 B4 01 03 03 00 01 01   ..f............
[   64] 04 02                                            ..
root@kali:/var/log/snort#
```

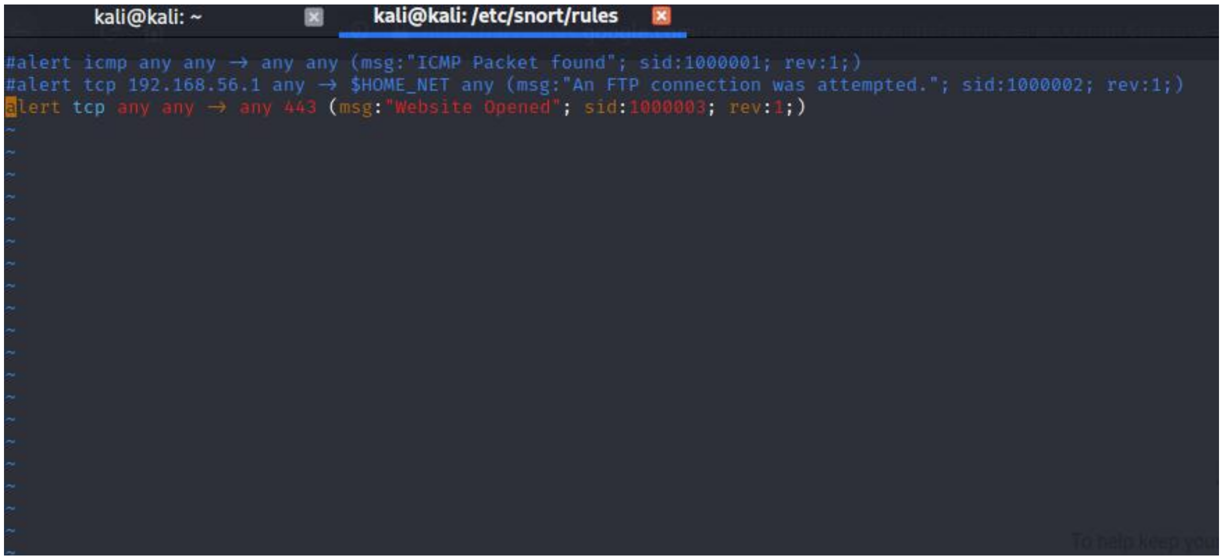


```
kali@kali:~$ nmap 192.168.56.1 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan t
imes will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-23 00:03 EDT
Nmap scan report for 192.168.56.1
Host is up (0.0021s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
3306/tcp open  mysql
6646/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 21.60 seconds
kali@kali:~$ echo "This is a tcp message" > /dev/tcp/192.168.56.1/3306
kali@kali:~$ echo "This is a tcp message" > /dev/tcp/192.168.56.1/3306
kali@kali:~$
```

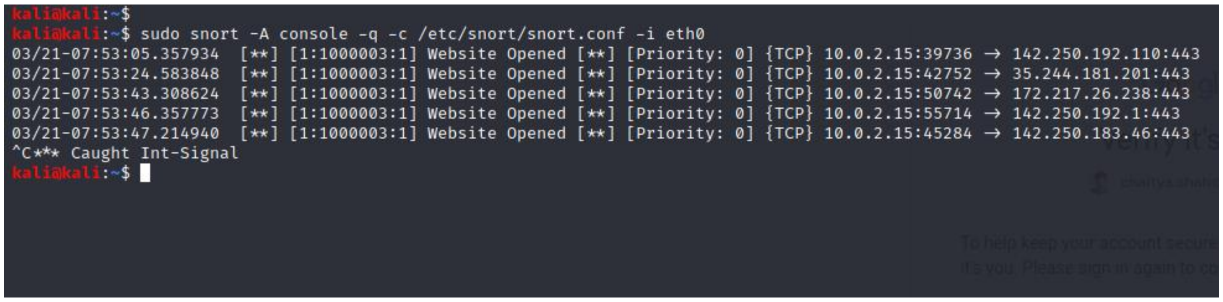


```
kali@kali:/etc/snort/rules$
kali@kali:/etc/snort/rules$ sudo snort -A console -q -c /etc/snort/snort.conf -i eth1
03/23-00:09:54.741311  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.741559  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.745011  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.745032  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.102:58902 → 192.168.56.1:3306
03/23-00:09:54.745223  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.745224  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.745224  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.1:3306 → 192.168.56.102:58902
03/23-00:09:54.745236  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.102:58902 → 192.168.56.1:3306
03/23-00:09:54.745247  [**] [1:1000003:1] "FTP failed login" [**] [Priority: 0] {TCP} 192.168.56.102:58902 → 192.168.56.1:3306
^C*** Caught Int-Signal
kali@kali:/etc/snort/rules$
```

Extra Credit (10pt):Write a rule that will fire when you browse to any site from the machine Snort is running on; it should look for any outbound TCP request to the site you have considered and alert on it.



```
#alert icmp any any → any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
#alert tcp 192.168.56.1 any → $HOME_NET any (msg:"An FTP connection was attempted."; sid:1000002; rev:1;)
alert tcp any any → any 443 (msg:"Website Opened"; sid:1000003; rev:1;)
~
~
~
```



```
kali@kali:~$
kali@kali:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
03/21-07:53:05.357934  [**] [1:1000003:1] Website Opened [**] [Priority: 0] {TCP} 10.0.2.15:39736 → 142.250.192.110:443
03/21-07:53:24.583848  [**] [1:1000003:1] Website Opened [**] [Priority: 0] {TCP} 10.0.2.15:42752 → 35.244.181.201:443
03/21-07:53:43.308624  [**] [1:1000003:1] Website Opened [**] [Priority: 0] {TCP} 10.0.2.15:50742 → 172.217.26.238:443
03/21-07:53:46.357773  [**] [1:1000003:1] Website Opened [**] [Priority: 0] {TCP} 10.0.2.15:55714 → 142.250.192.1:443
03/21-07:53:47.214940  [**] [1:1000003:1] Website Opened [**] [Priority: 0] {TCP} 10.0.2.15:45284 → 142.250.183.46:443
^C*** Caught Int-Signal
kali@kali:~$
```

Conclusion:

1. Zero day attacks cannot be prevented using snort.

2. We can easily analyze incoming traffic using snort IDS and prevent known attacks from happening.

3. I analyzed incoming and outgoing icmp and tcp packets. For sending tcp packets first the tcp ports were scanned using nmap and then an tcp echo request was send.