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

Name: Ameya Jangam
UID: 2019130025
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 Kali as our VM image.

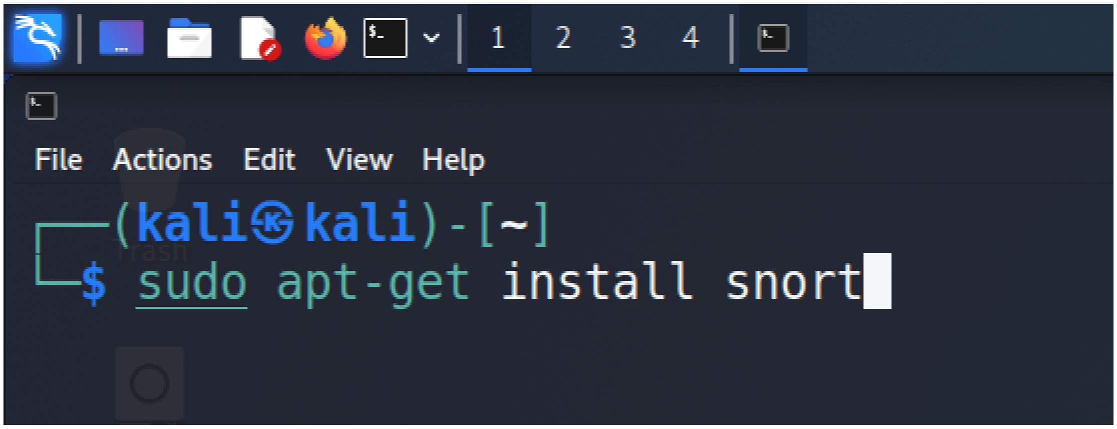
Login the Kali image with username and password

Installing Snort into the Operating System

To install the latest version of the snort, you can follow the installation instructions from the snort website. Note that installation instructions 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
```

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 for carrying 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 depending upon what has been identified.

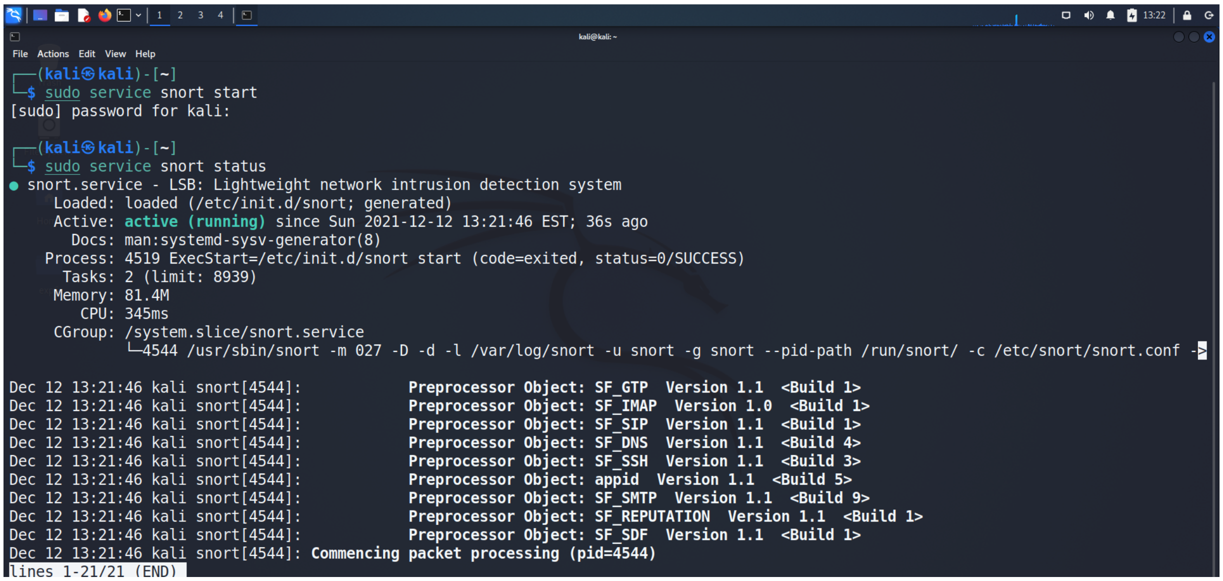
Configuring and Starting the Snort IDS

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
```

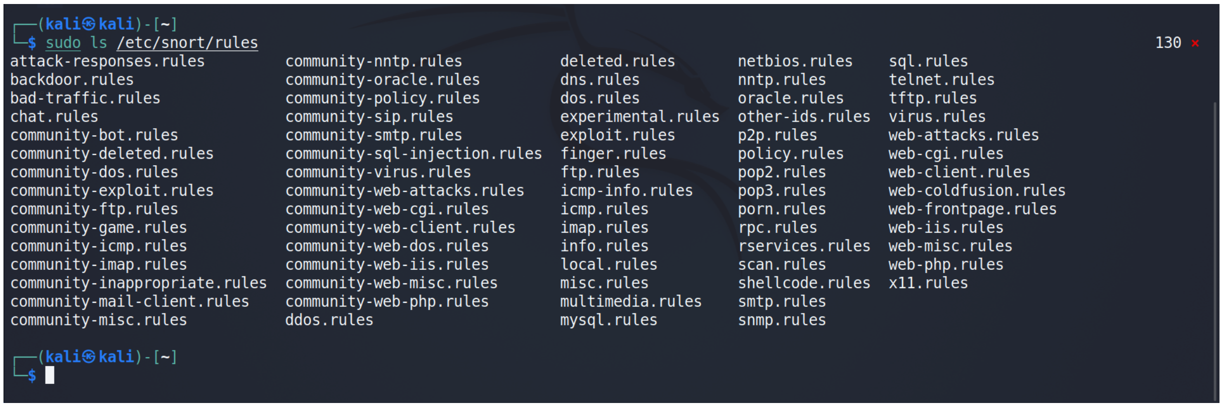


```
┌──(kali㉿kali)-[~]
└─$ sudo service snort start
[sudo] password for kali:

┌──(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 Sun 2021-12-12 13:21:46 EST; 36s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 4519 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
      Tasks: 2 (limit: 8939)
     Memory: 81.4M
        CPU: 345ms
     CGroup: /system.slice/snort.service
             └─4544 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort --pid-path /run/snort/ -c /etc/snort/snort.conf ->

Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: appid  Version 1.1  <Build 5>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
Dec 12 13:21:46 kali snort[4544]:           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
Dec 12 13:21:46 kali snort[4544]: Commencing packet processing (pid=4544)
lines 1-21/21 (END)
```

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

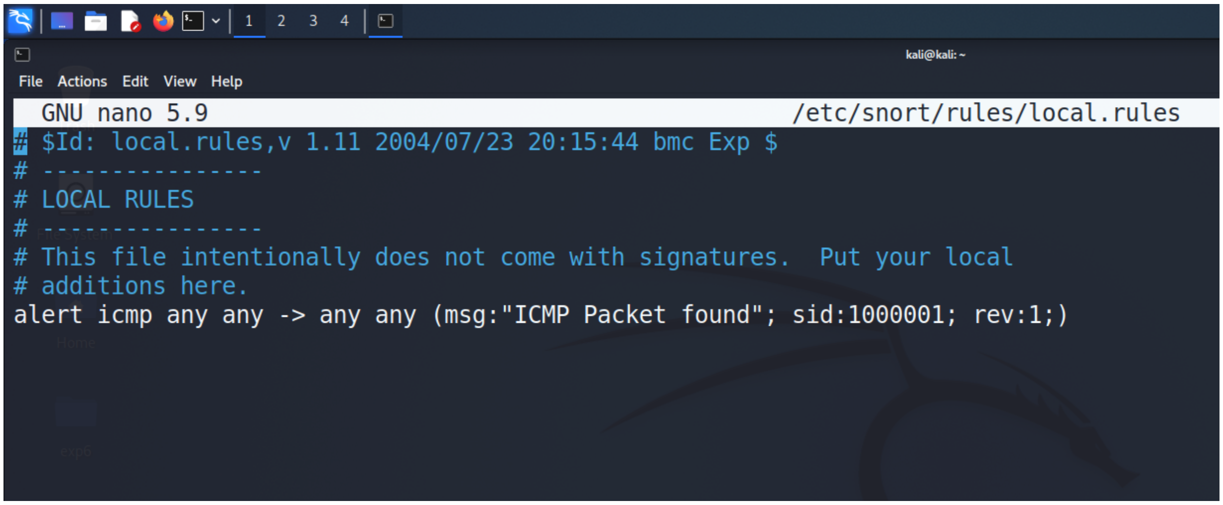


```
┌──(kali㉿kali)-[~]
└─$ sudo ls /etc/snort/rules                                                                        130 ×
attack-responses.rules        community-nntp.rules            deleted.rules       netbios.rules     sql.rules
backdoor.rules                community-oracle.rules          dns.rules           nntp.rules        telnet.rules
bad-traffic.rules             community-policy.rules          dos.rules           oracle.rules      tftp.rules
chat.rules                    community-sip.rules             experimental.rules  other-ids.rules   virus.rules
community-bot.rules           community-smtp.rules            exploit.rules       p2p.rules         web-attacks.rules
community-deleted.rules       community-sql-injection.rules   finger.rules        policy.rules      web-cgi.rules
community-dos.rules           community-virus.rules           ftp.rules           pop2.rules        web-client.rules
community-exploit.rules       community-web-attacks.rules     icmp-info.rules     pop3.rules        web-coldfusion.rules
community-ftp.rules           community-web-cgi.rules         icmp.rules          porn.rules        web-frontpage.rules
community-game.rules          community-web-client.rules      imap.rules          rpc.rules         web-iis.rules
community-icmp.rules          community-web-dos.rules         info.rules          rservices.rules   web-misc.rules
community-imap.rules          community-web-iis.rules         local.rules         scan.rules        web-php.rules
community-inappropriate.rules community-web-misc.rules        misc.rules          shellcode.rules   x11.rules
community-mail-client.rules   community-web-php.rules         multimedia.rules    smtp.rules
community-misc.rules          ddos.rules                      mysql.rules         snmp.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;)

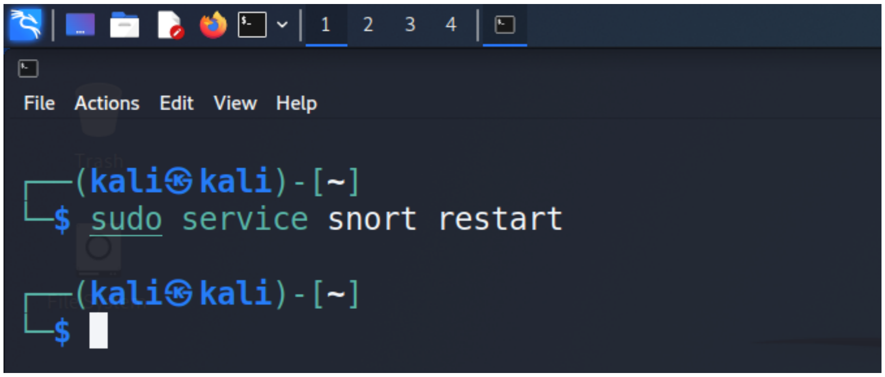


```
  GNU nano 5.9                                    /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> 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 than 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



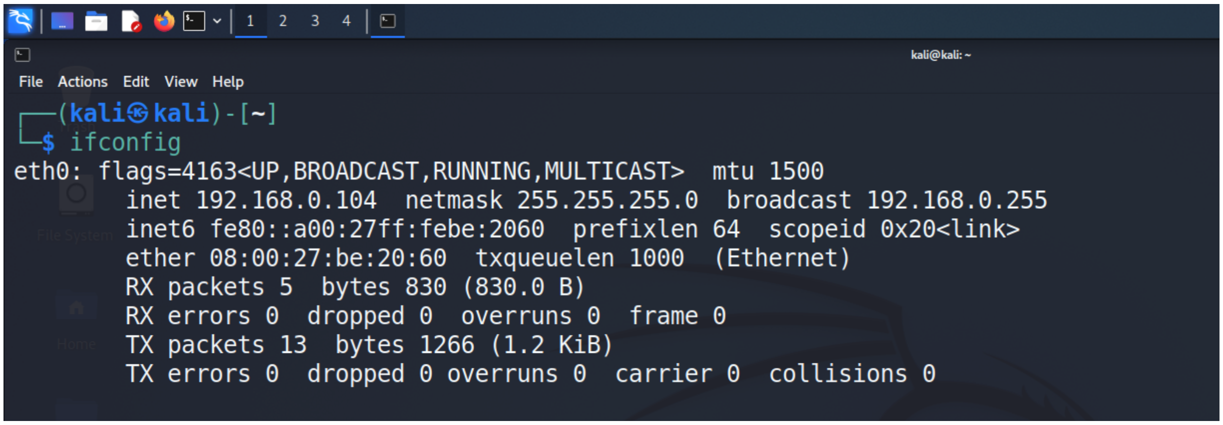
```
┌──(kali㉿kali)-[~]
└─$ sudo service snort restart

┌──(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.

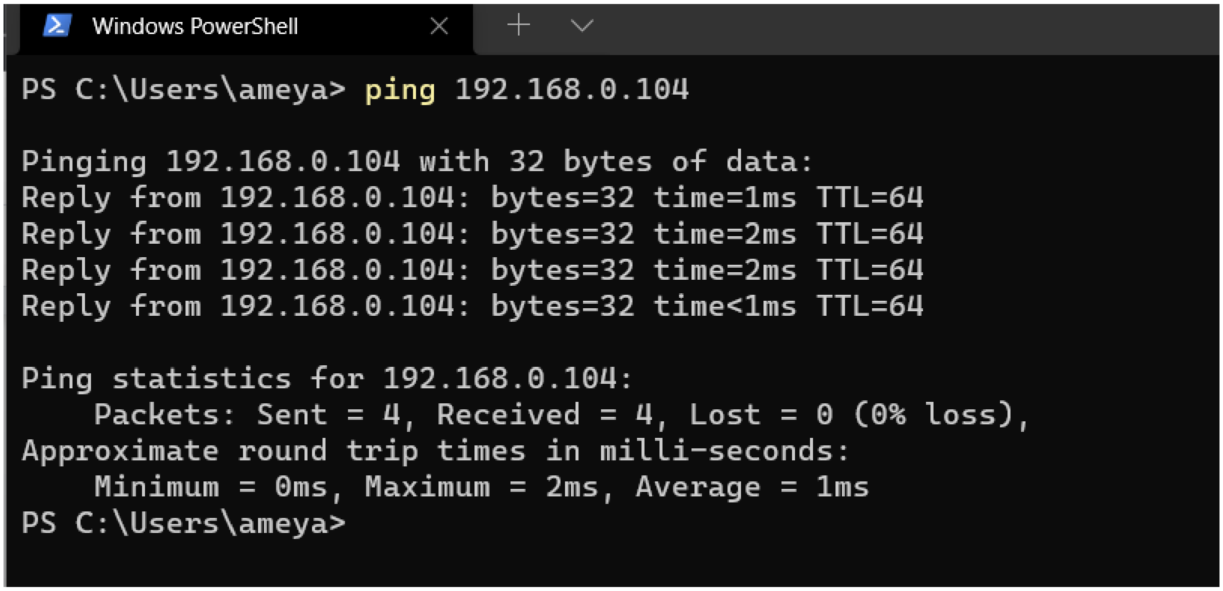
After you have a terminal, you can just type the following command to send ping messages to the VM.

$ ifconfig



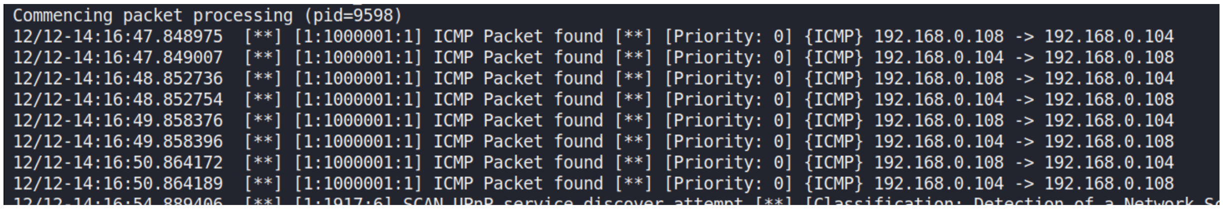
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.104  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:febe:2060  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:be:20:60  txqueuelen 1000  (Ethernet)
        RX packets 5  bytes 830 (830.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1266 (1.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

pinging from host machine

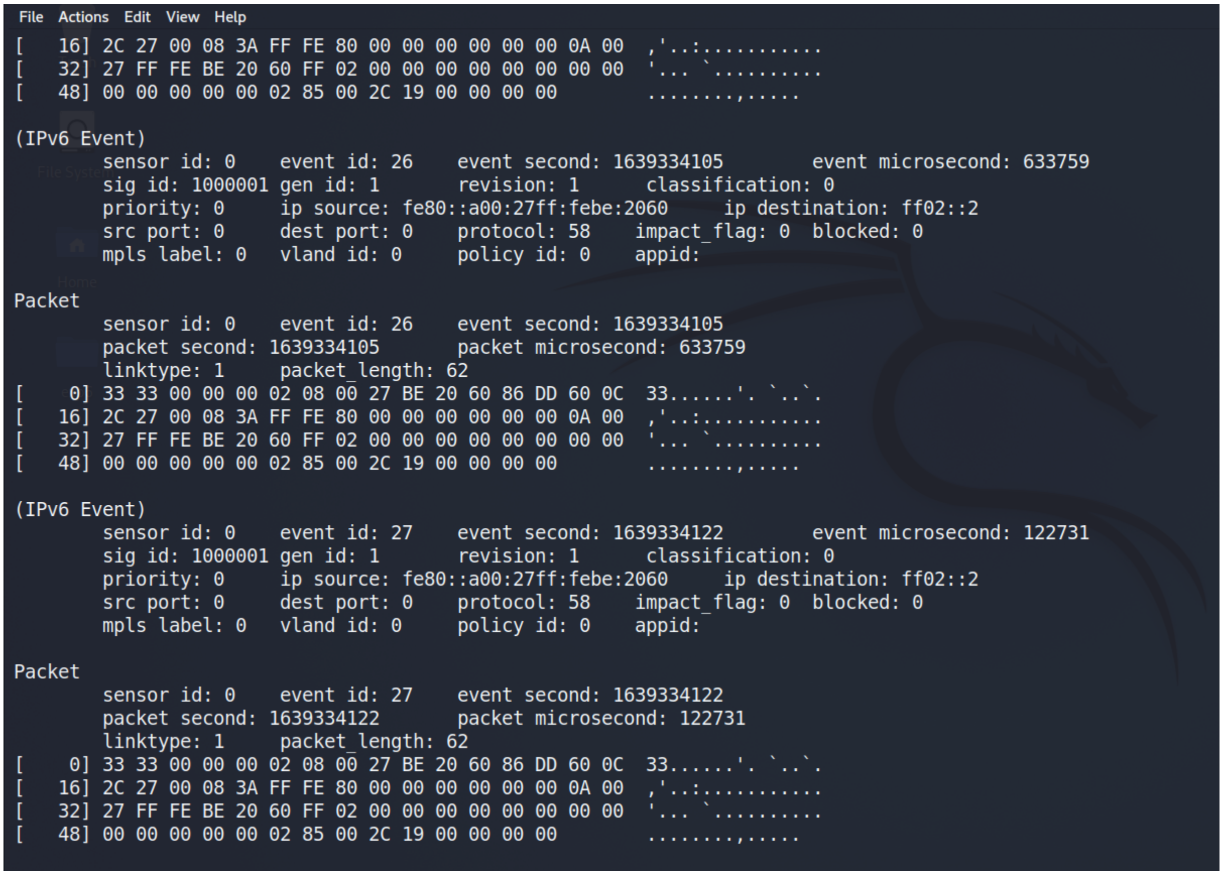


```
PS C:\Users\ameya> ping 192.168.0.104

Pinging 192.168.0.104 with 32 bytes of data:
Reply from 192.168.0.104: bytes=32 time=1ms TTL=64
Reply from 192.168.0.104: bytes=32 time=2ms TTL=64
Reply from 192.168.0.104: bytes=32 time=2ms TTL=64
Reply from 192.168.0.104: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms
PS C:\Users\ameya>
```

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.



```
Commencing packet processing (pid=9598)
12/12-14:16:47.848975  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.108 -> 192.168.0.104
12/12-14:16:47.849007  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.104 -> 192.168.0.108
12/12-14:16:48.852736  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.108 -> 192.168.0.104
12/12-14:16:48.852754  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.104 -> 192.168.0.108
12/12-14:16:49.858376  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.108 -> 192.168.0.104
12/12-14:16:49.858396  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.104 -> 192.168.0.108
12/12-14:16:50.864172  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.108 -> 192.168.0.104
12/12-14:16:50.864189  [**] [1:1000001:1] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.0.104 -> 192.168.0.108
12/12-14:16:54.889406  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network S
```



```
[   16] 2C 27 00 08 3A FF FE 80 00 00 00 00 00 00 0A 00  ,'..:..........
[   32] 27 FF FE BE 20 60 FF 02 00 00 00 00 00 00 00 00  '... `.........
[   48] 00 00 00 00 00 02 85 00 2C 19 00 00 00 00        .......,.....

(IPv6 Event)
        sensor id: 0    event id: 26    event second: 1639334105        event microsecond: 633759
        sig id: 1000001 gen id: 1       revision: 1      classification: 0
        priority: 0     ip source: fe80::a00:27ff:febe:2060     ip destination: ff02::2
        src port: 0     dest port: 0    protocol: 58    impact_flag: 0  blocked: 0
        mpls label: 0   vland id: 0     policy id: 0    appid:

Packet
        sensor id: 0    event id: 26    event second: 1639334105
        packet second: 1639334105       packet microsecond: 633759
        linktype: 1     packet_length: 62
[    0] 33 33 00 00 00 02 08 00 27 BE 20 60 86 DD 60 0C  33......'. `..`.
[   16] 2C 27 00 08 3A FF FE 80 00 00 00 00 00 00 0A 00  ,'..:..........
[   32] 27 FF FE BE 20 60 FF 02 00 00 00 00 00 00 00 00  '... `.........
[   48] 00 00 00 00 00 02 85 00 2C 19 00 00 00 00        .......,.....

(IPv6 Event)
        sensor id: 0    event id: 27    event second: 1639334122        event microsecond: 122731
        sig id: 1000001 gen id: 1       revision: 1      classification: 0
        priority: 0     ip source: fe80::a00:27ff:febe:2060     ip destination: ff02::2
        src port: 0     dest port: 0    protocol: 58    impact_flag: 0  blocked: 0
        mpls label: 0   vland id: 0     policy id: 0    appid:

Packet
        sensor id: 0    event id: 27    event second: 1639334122
        packet second: 1639334122       packet microsecond: 122731
        linktype: 1     packet_length: 62
[    0] 33 33 00 00 00 02 08 00 27 BE 20 60 86 DD 60 0C  33......'. `..`.
[   16] 2C 27 00 08 3A FF FE 80 00 00 00 00 00 00 0A 00  ,'..:..........
[   32] 27 FF FE BE 20 60 FF 02 00 00 00 00 00 00 00 00  '... `.........
[   48] 00 00 00 00 00 02 85 00 2C 19 00 00 00 00        .......,.....
```

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?

- 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 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.

Examples of recent zero day attacks:

- **2020: Zoom**

  A vulnerability was found in the popular video conferencing platform. This zero-day attack example involved hackers accessing a user’s PC remotely if they were running an older version of Windows. If the target was an administrator, the hacker could completely take over their machine and access all their files.

- **2019: Microsoft Windows, Eastern Europe**

  This attack focused on local escalation privileges, a vulnerable part of Microsoft Windows, and targeted government institutions in Eastern Europe. The zero-day exploit abused a local privilege vulnerability in Microsoft Windows to run arbitrary code and install applications and view and change the data on compromised applications. Once the attack was identified and reported to the Microsoft Security Response Center, a patch was developed and rolled out.

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.

- I looked up online to see if snort can detect zero day attacks or not and found this research paper “Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?” and found out that the results from their study show that Snort clearly is able to detect zero-days (a mean of 17% 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 = 99.9% of 1000000 = 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

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

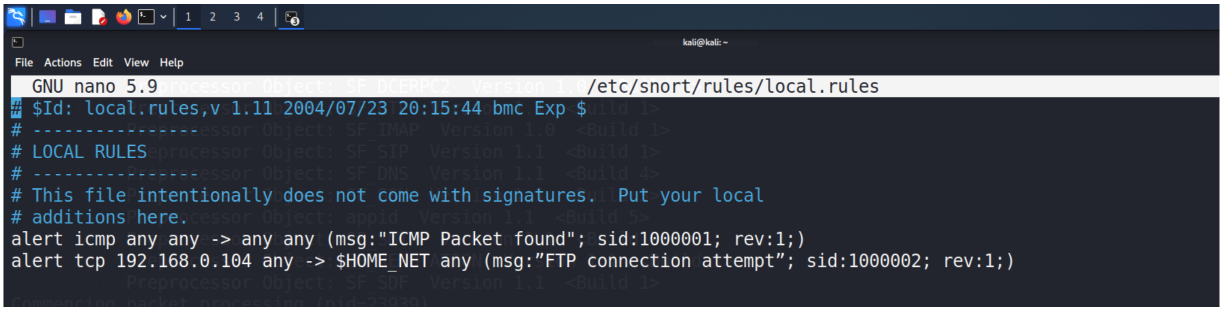
Therefore, Number of false alarms = 1000 – 950 = 50 alarms.

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

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

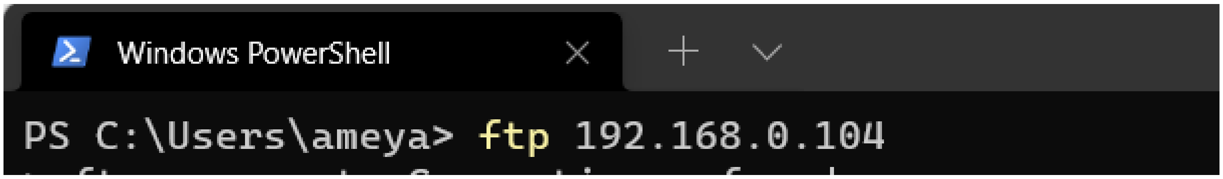
3.Write and add another snort rule and show me you trigger it.

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

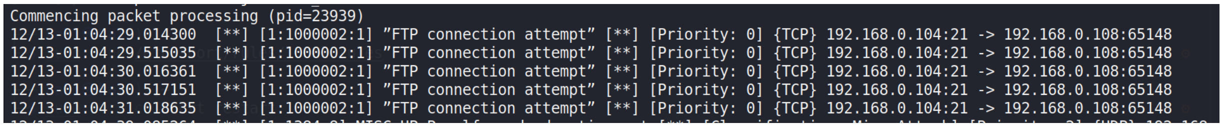


```
  GNU nano 5.9                                /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
alert tcp 192.168.0.104 any -> $HOME_NET any (msg:”FTP connection attempt”; sid:1000002; rev:1;)
```

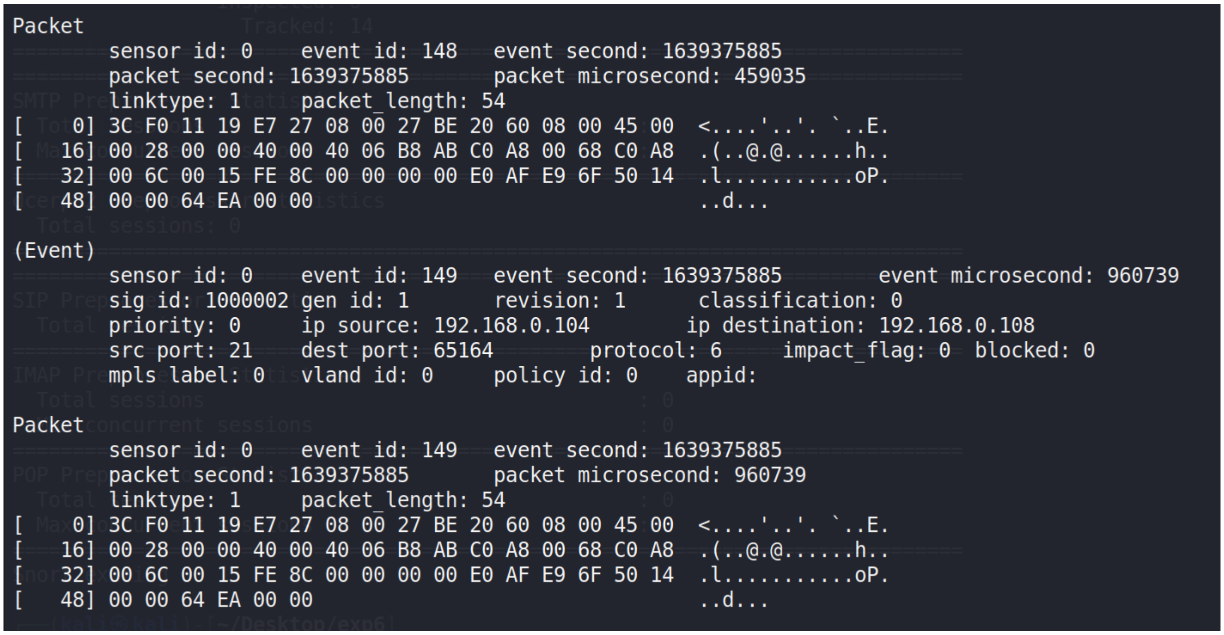
b. A description of how you triggered the alert. The alert itself from the log file (after converting it to readable text)



```
PS C:\Users\ameya> ftp 192.168.0.104
```

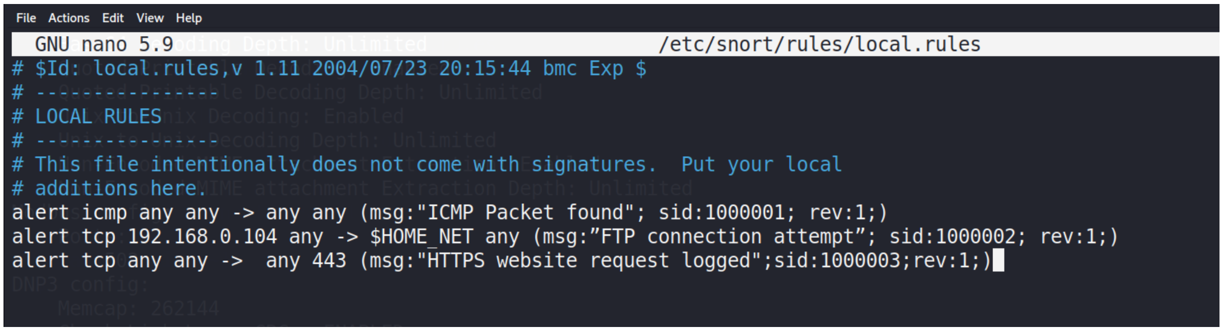


```
Commencing packet processing (pid=23939)
12/13-01:04:29.014300  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:21 -> 192.168.0.108:65148
12/13-01:04:29.515035  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:21 -> 192.168.0.108:65148
12/13-01:04:30.016361  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:21 -> 192.168.0.108:65148
12/13-01:04:30.517151  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:21 -> 192.168.0.108:65148
12/13-01:04:31.018635  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:21 -> 192.168.0.108:65148
```

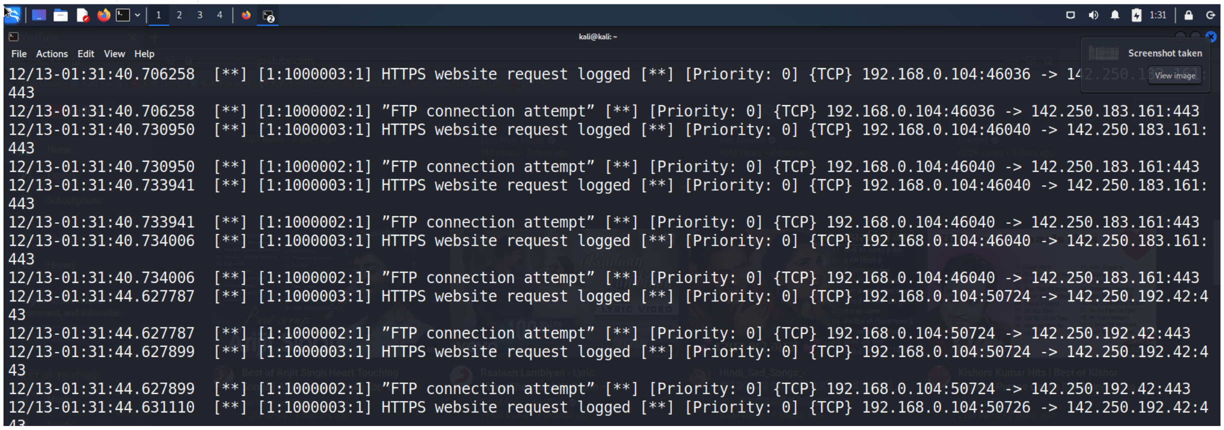


```
Packet
        sensor id: 0     event id: 148    event second: 1639375885
        packet second: 1639375885         packet microsecond: 459035
        linktype: 1      packet_length: 54
[    0] 3C F0 11 19 E7 27 08 00 27 BE 20 60 08 00 45 00  <....'..'. `..E.
[   16] 00 28 00 00 40 00 40 06 B8 AB C0 A8 00 68 C0 A8  .(..@.@......h..
[   32] 00 6C 00 15 FE 8C 00 00 00 00 E0 AF E9 6F 50 14  .l...........oP.
[   48] 00 00 64 EA 00 00                                ..d...
(Event)
        sensor id: 0     event id: 149    event second: 1639375885        event microsecond: 960739
        sig id: 1000002 gen id: 1         revision: 1      classification: 0
        priority: 0      ip source: 192.168.0.104        ip destination: 192.168.0.108
        src port: 21     dest port: 65164         protocol: 6      impact_flag: 0  blocked: 0
        mpls label: 0    vland id: 0      policy id: 0     appid:
Packet
        sensor id: 0     event id: 149    event second: 1639375885
        packet second: 1639375885         packet microsecond: 960739
        linktype: 1      packet_length: 54
[    0] 3C F0 11 19 E7 27 08 00 27 BE 20 60 08 00 45 00  <....'..'. `..E.
[   16] 00 28 00 00 40 00 40 06 B8 AB C0 A8 00 68 C0 A8  .(..@.@......h..
[   32] 00 6C 00 15 FE 8C 00 00 00 00 E0 AF E9 6F 50 14  .l...........oP.
[   48] 00 00 64 EA 00 00                                ..d...
```

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.



```
  GNU nano 5.9                                         /etc/snort/rules/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 -> any any (msg:"ICMP Packet found"; sid:1000001; rev:1;)
alert tcp 192.168.0.104 any -> $HOME_NET any (msg:”FTP connection attempt”; sid:1000002; rev:1;)
alert tcp any any ->  any 443 (msg:"HTTPS website request logged";sid:1000003;rev:1;)
```



```
12/13-01:31:40.706258  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:46036 -> 14
443
12/13-01:31:40.706258  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:46036 -> 142.250.183.161:443
12/13-01:31:40.730950  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:
443
12/13-01:31:40.730950  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:443
12/13-01:31:40.733941  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:
443
12/13-01:31:40.733941  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:443
12/13-01:31:40.734006  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:
443
12/13-01:31:40.734006  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:46040 -> 142.250.183.161:443
12/13-01:31:44.627787  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:50724 -> 142.250.192.42:4
43
12/13-01:31:44.627787  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:50724 -> 142.250.192.42:443
12/13-01:31:44.627899  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:50724 -> 142.250.192.42:4
43
12/13-01:31:44.627899  [**] [1:1000002:1] ”FTP connection attempt” [**] [Priority: 0] {TCP} 192.168.0.104:50724 -> 142.250.192.42:443
12/13-01:31:44.631110  [**] [1:1000003:1] HTTPS website request logged [**] [Priority: 0] {TCP} 192.168.0.104:50726 -> 142.250.192.42:4
43
```

Github Link : https://github.com/ameyajangam22/CSS-LAB-2019130025