ET2595: Network and system security

Laboratory-3

Monica Gattupalli

moga20@student.bth.se

19991130-5002

Introduction

The aim of the lab is to achieve the snort intrusion detection system (IDS).

To perform the task IDS, virtual box appliance from lab1 is used which contains the server A, client A, server B, client B. In this lab only server A and Server B are used, and the environment is configured to achieve IDS.

Snort

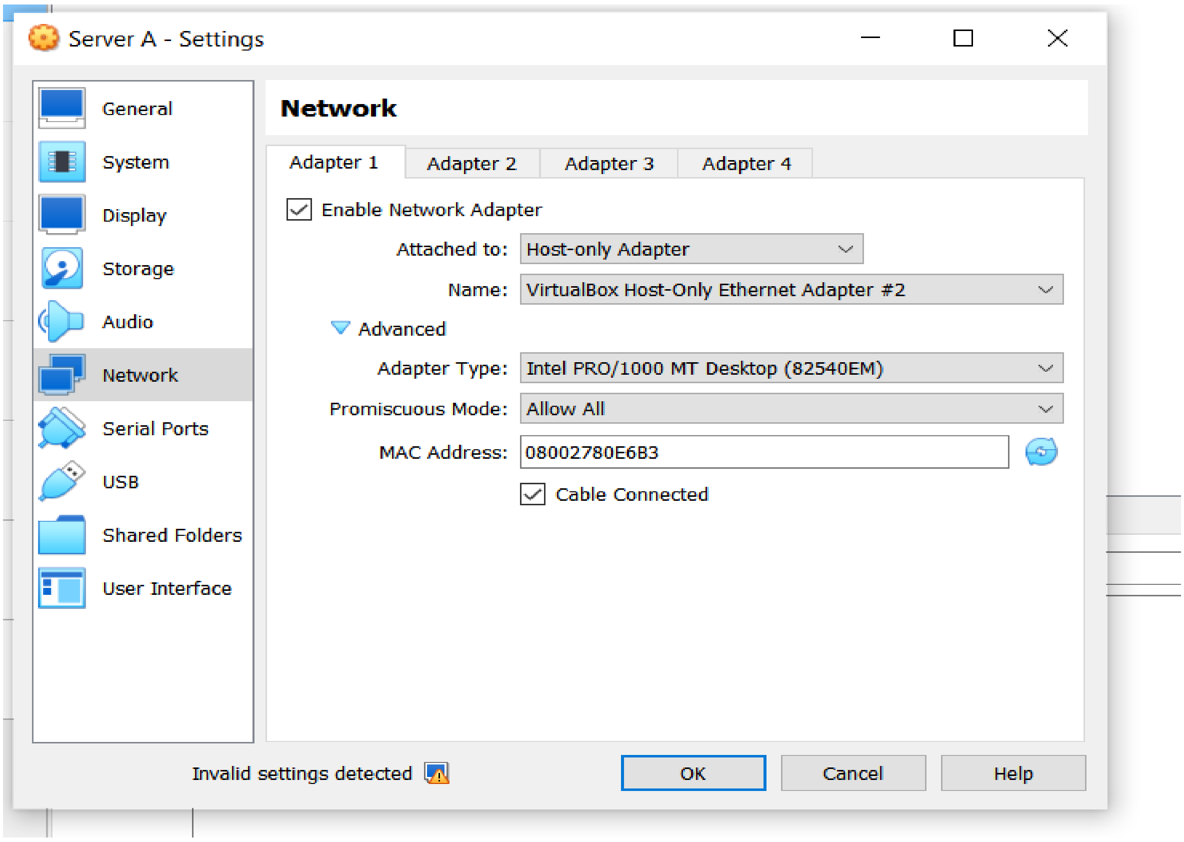
This is open-source IDS which needs to be installed in Server A by using “apt install snort”. Some snort rules are written in server A which helps in detecting the type of attack.

Metasploit

This is testing software which test the system security this is installed on server B. to start the Metasploit enter the “msfconsole”. Using this server B will be attacking Server A and the server A will detect them using snort and those are captured using wire shark.

Server A and Server B will be working under shared network “192.168.70.0/24”, both the server has the promiscuous as the “ALLOW All”.

Below two figures are the Server A-settings and Server B-settings where the promiscuous mode is set to “Allow all”.



Server A - Settings

Network

Adapter 1

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #2

Advanced

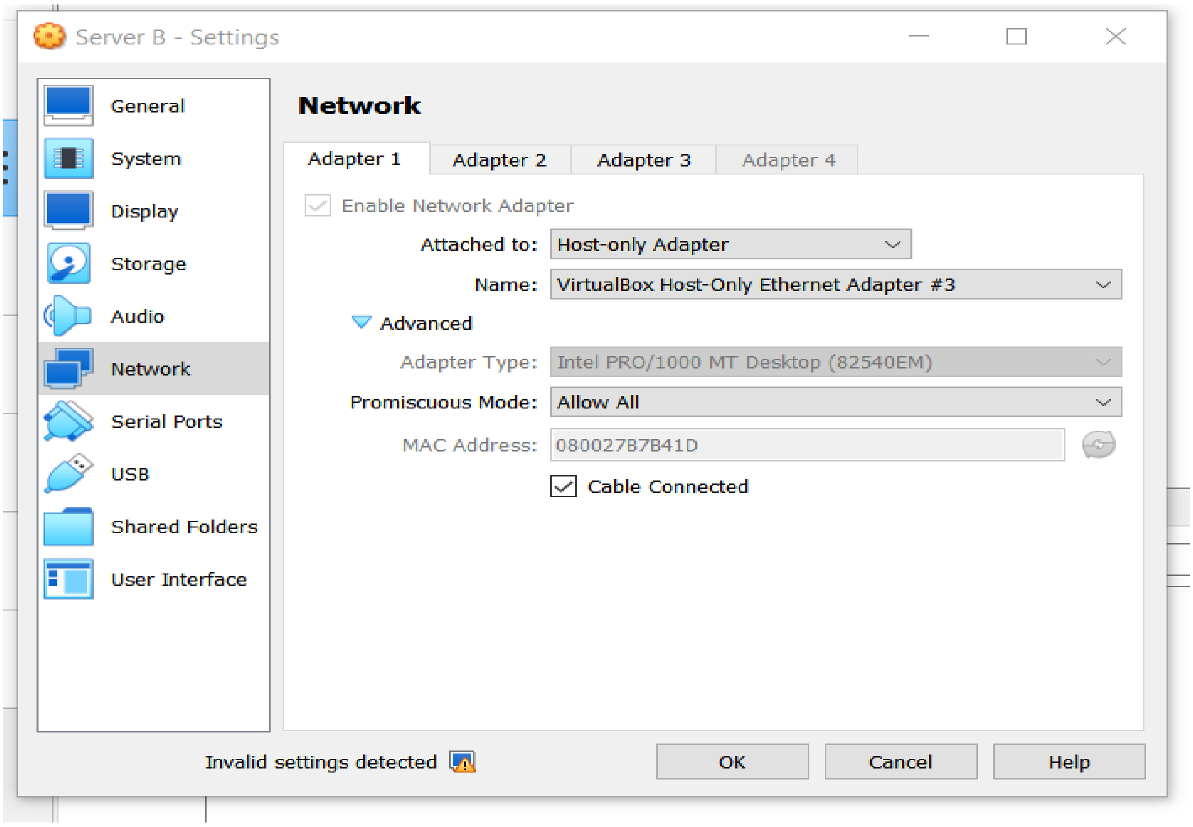
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 08002780E6B3

☑ Cable Connected

Invalid settings detected



☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #3

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 080027B7B41D

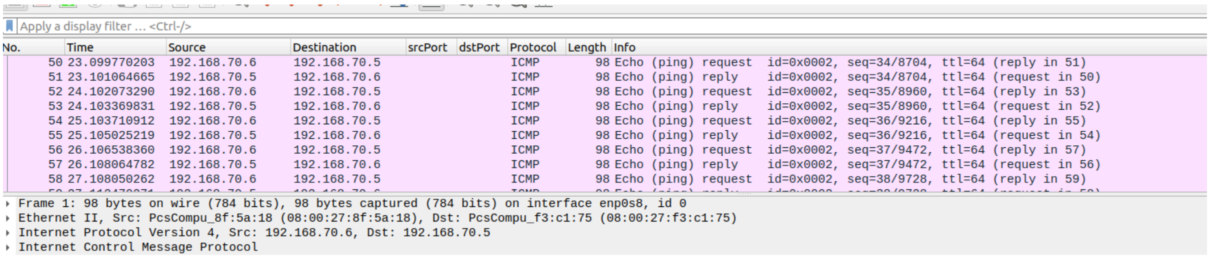
☑ Cable Connected

Invalid settings detected

Task-1 Check Connectivity

In this task we need to check the connectivity between the server A and server B to do that we will ping the IP address of the server A “192.168.70.5” in the server B and open the wire shark in Server A and select the enp0s8 to see the ICMP echo and reply to packets. If we can see echo and reply, then we can understand that there is the connection between server A and server B.

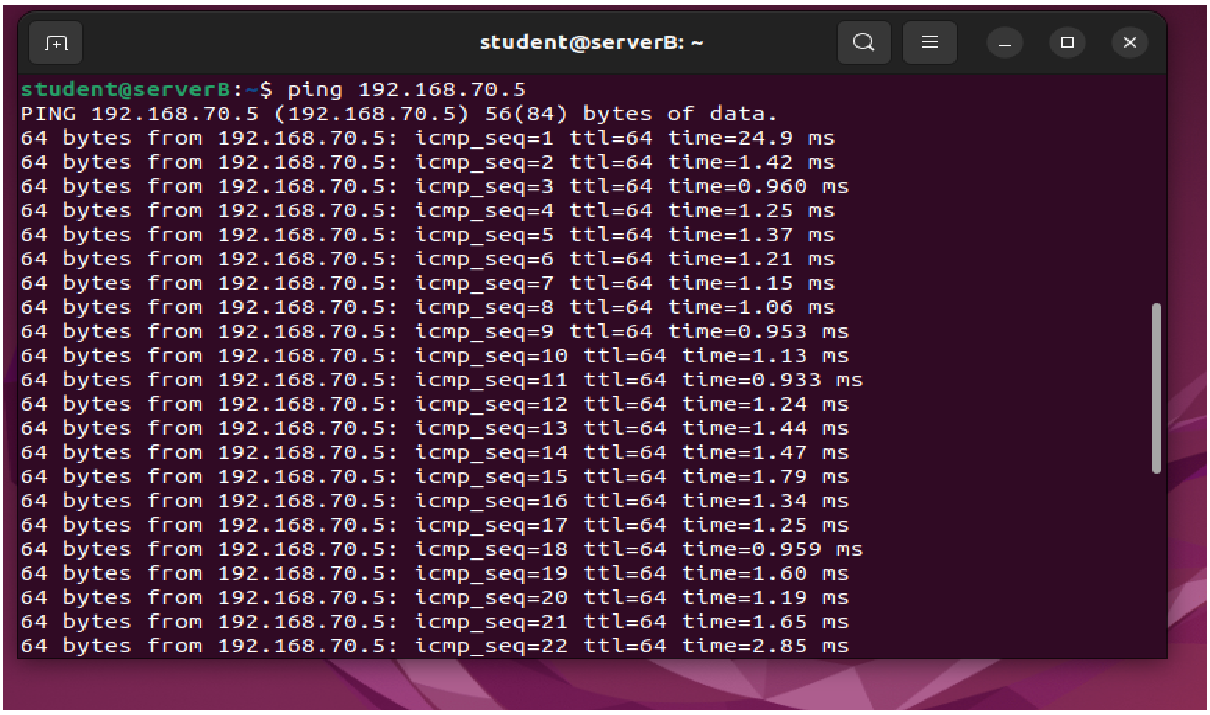
In the below figure is the capture of Wireshark, we can see that ICMP echo and reply to packets transmissions between the Server A and Server B.



```
No.       Time          Source         Destination     srcPort dstPort Protocol Length Info
      50 23.099770203  192.168.70.6   192.168.70.5                     ICMP         98 Echo (ping) request  id=0x0002, seq=34/8704, ttl=64 (reply in 51)
      51 23.101064665  192.168.70.5   192.168.70.6                     ICMP         98 Echo (ping) reply    id=0x0002, seq=34/8704, ttl=64 (request in 50)
      52 24.102073290  192.168.70.6   192.168.70.5                     ICMP         98 Echo (ping) request  id=0x0002, seq=35/8960, ttl=64 (reply in 53)
      53 24.103369831  192.168.70.5   192.168.70.6                     ICMP         98 Echo (ping) reply    id=0x0002, seq=35/8960, ttl=64 (request in 52)
      54 25.103710912  192.168.70.6   192.168.70.5                     ICMP         98 Echo (ping) request  id=0x0002, seq=36/9216, ttl=64 (reply in 55)
      55 25.105025219  192.168.70.5   192.168.70.6                     ICMP         98 Echo (ping) reply    id=0x0002, seq=36/9216, ttl=64 (request in 54)
      56 26.106538360  192.168.70.6   192.168.70.5                     ICMP         98 Echo (ping) request  id=0x0002, seq=37/9472, ttl=64 (reply in 57)
      57 26.108064782  192.168.70.5   192.168.70.6                     ICMP         98 Echo (ping) reply    id=0x0002, seq=37/9472, ttl=64 (request in 56)
      58 27.108050262  192.168.70.6   192.168.70.5                     ICMP         98 Echo (ping) request  id=0x0002, seq=38/9728, ttl=64 (reply in 59)
› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s8, id 0
› Ethernet II, Src: PcsCompu_8f:5a:18 (08:00:27:8f:5a:18), Dst: PcsCompu_f3:c1:75 (08:00:27:f3:c1:75)
› Internet Protocol Version 4, Src: 192.168.70.6, Dst: 192.168.70.5
› Internet Control Message Protocol
```

Figure 1: Wireshark for connection

In below picture we can see that “192.168.70.5” is ping in server B which is the address of the server A.



```
student@serverB:~$ ping 192.168.70.5
PING 192.168.70.5 (192.168.70.5) 56(84) bytes of data.
64 bytes from 192.168.70.5: icmp_seq=1 ttl=64 time=24.9 ms
64 bytes from 192.168.70.5: icmp_seq=2 ttl=64 time=1.42 ms
64 bytes from 192.168.70.5: icmp_seq=3 ttl=64 time=0.960 ms
64 bytes from 192.168.70.5: icmp_seq=4 ttl=64 time=1.25 ms
64 bytes from 192.168.70.5: icmp_seq=5 ttl=64 time=1.37 ms
64 bytes from 192.168.70.5: icmp_seq=6 ttl=64 time=1.21 ms
64 bytes from 192.168.70.5: icmp_seq=7 ttl=64 time=1.15 ms
64 bytes from 192.168.70.5: icmp_seq=8 ttl=64 time=1.06 ms
64 bytes from 192.168.70.5: icmp_seq=9 ttl=64 time=0.953 ms
64 bytes from 192.168.70.5: icmp_seq=10 ttl=64 time=1.13 ms
64 bytes from 192.168.70.5: icmp_seq=11 ttl=64 time=0.933 ms
64 bytes from 192.168.70.5: icmp_seq=12 ttl=64 time=1.24 ms
64 bytes from 192.168.70.5: icmp_seq=13 ttl=64 time=1.44 ms
64 bytes from 192.168.70.5: icmp_seq=14 ttl=64 time=1.47 ms
64 bytes from 192.168.70.5: icmp_seq=15 ttl=64 time=1.79 ms
64 bytes from 192.168.70.5: icmp_seq=16 ttl=64 time=1.34 ms
64 bytes from 192.168.70.5: icmp_seq=17 ttl=64 time=1.25 ms
64 bytes from 192.168.70.5: icmp_seq=18 ttl=64 time=0.959 ms
64 bytes from 192.168.70.5: icmp_seq=19 ttl=64 time=1.60 ms
64 bytes from 192.168.70.5: icmp_seq=20 ttl=64 time=1.19 ms
64 bytes from 192.168.70.5: icmp_seq=21 ttl=64 time=1.65 ms
64 bytes from 192.168.70.5: icmp_seq=22 ttl=64 time=2.85 ms
```

Figure 2: ping of server A

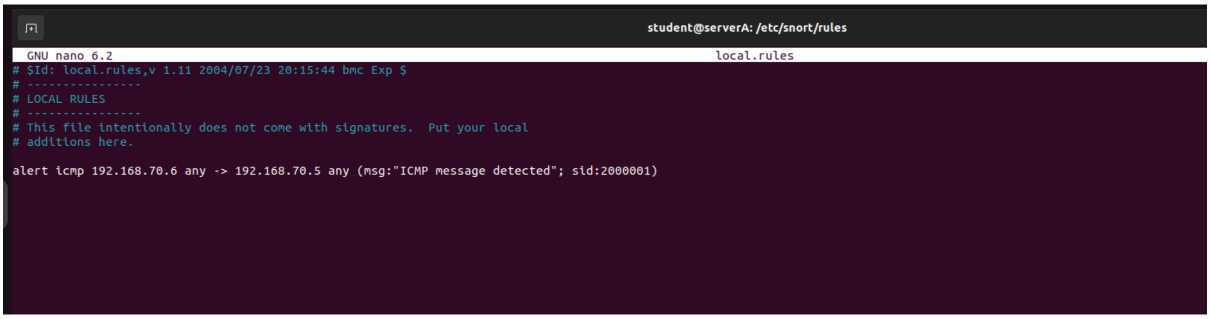
Task-2: Detect incoming pings.

In this task we need to edit the snort.conf file on server A and we need to comment the line **“include $RULE_PATH/icmp-info.rules”** along with that, in “local.rules” file we need to add a snort rule for ICMP detection.

“alert icmp 192.168.70.6 any -> 192.168.70.5 any (msg: “ICMP message detected”; sid:2000001)”.

alert	It generates the alert
ICMP	It is the protocol name
192.168.70.6, 192.168.70.5	Source and destination IP address
->	Direction operation which takes care about the direction of the traffic
msg	The message that needs to be displayed.
sid	Unique number to identify the snort rule

The below figure we can see the snort rule is added in the local.rules file of server A.

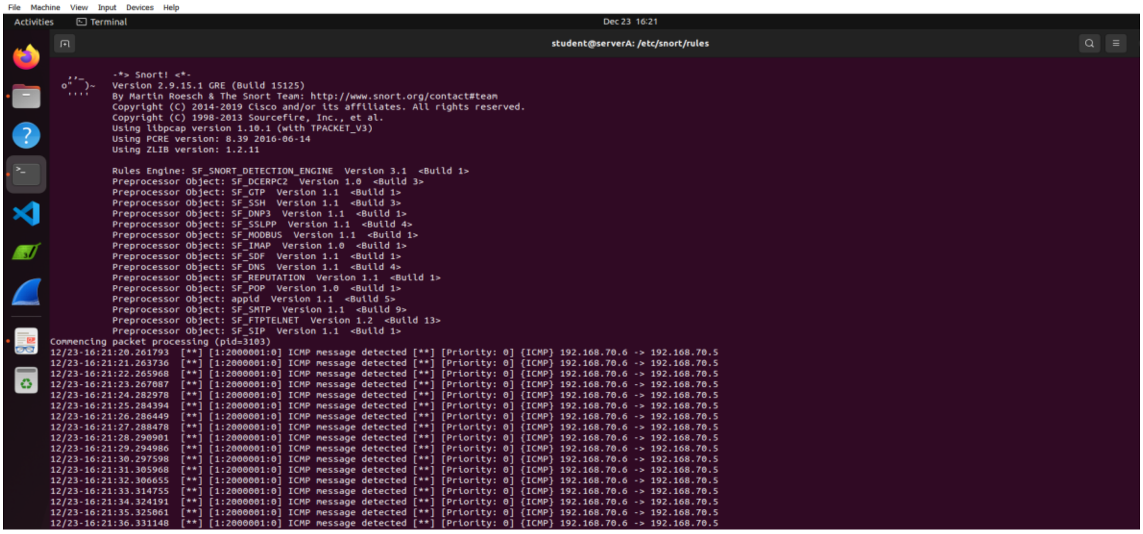


```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp 192.168.70.6 any -> 192.168.70.5 any (msg:"ICMP message detected"; sid:2000001)
```

Figure-3: local rules files

After adding the rule to files save and run the file using the command **“sudo snort -i enp0s8 -A console -c/etc/snort/snort.conf”.**

After executing the command, In the below figure we can observe the server A have the matching alerts this indicates snort rule is successful.


File Machine View Input Devices Help
Activities
Terminal
Dec 23 16:21
student@serverA: /etc/snort/rules

```
   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Commencing packet processing (pid=3103)
12/23-16:21:20.261793  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:21.263736  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:22.265968  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:23.267087  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:24.282978  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:25.284394  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:26.286449  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:27.288478  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:28.290901  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:29.294986  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:30.297598  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:31.305968  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:32.306655  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:33.314755  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:34.324191  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:35.325061  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
12/23-16:21:36.331148  [**] [1:2000001:0] ICMP message detected [**] [Priority: 0] {ICMP} 192.168.70.6 -> 192.168.70.5
```

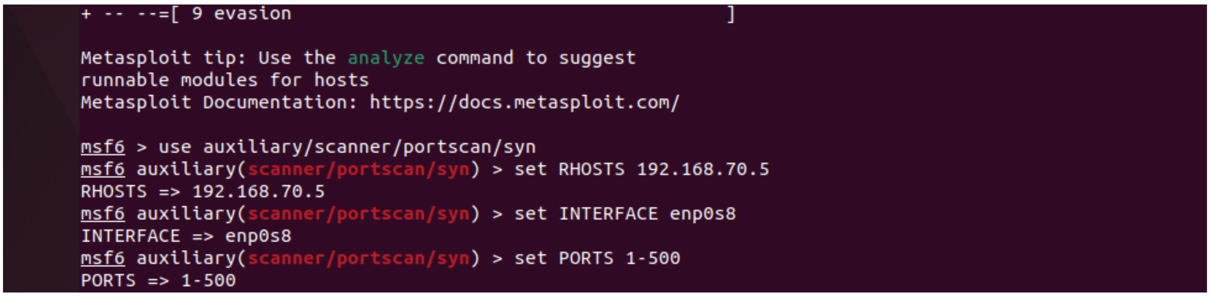
Figure -4: ping of the server A in server B

Task-3 Detect TCP Port Scanning

In this task we need to create the new snort rule on our own. In serverB metasploit is started by using **“sudo msf console”** after that execution you will redirected to the msf promt.In msf6 promt enter the following path **“use auxiliary/scanner/portscan/syn”.**Now set the hosts, interface and ports as 192.168.70.5, enp0s8 and 1-500 respectively.

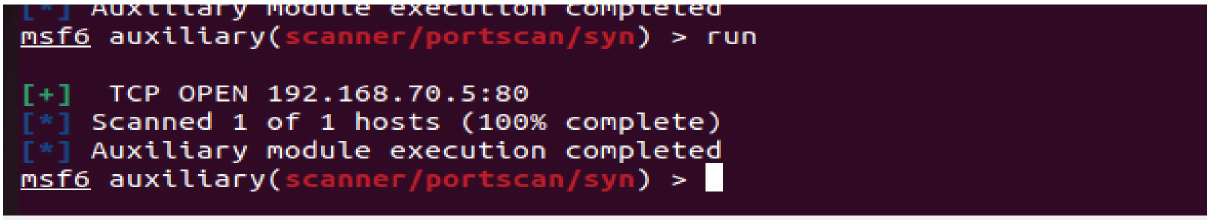
By following commands

1. **set RHOSTS 192.168.70.5**
2. **set INTERFACE enp0s8**
3. **set PORTS 1-500**



```
+ -- --=[ 9 evasion                                          ]

Metasploit tip: Use the analyze command to suggest
runnable modules for hosts
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use auxiliary/scanner/portscan/syn
msf6 auxiliary(scanner/portscan/syn) > set RHOSTS 192.168.70.5
RHOSTS => 192.168.70.5
msf6 auxiliary(scanner/portscan/syn) > set INTERFACE enp0s8
INTERFACE => enp0s8
msf6 auxiliary(scanner/portscan/syn) > set PORTS 1-500
PORTS => 1-500
```

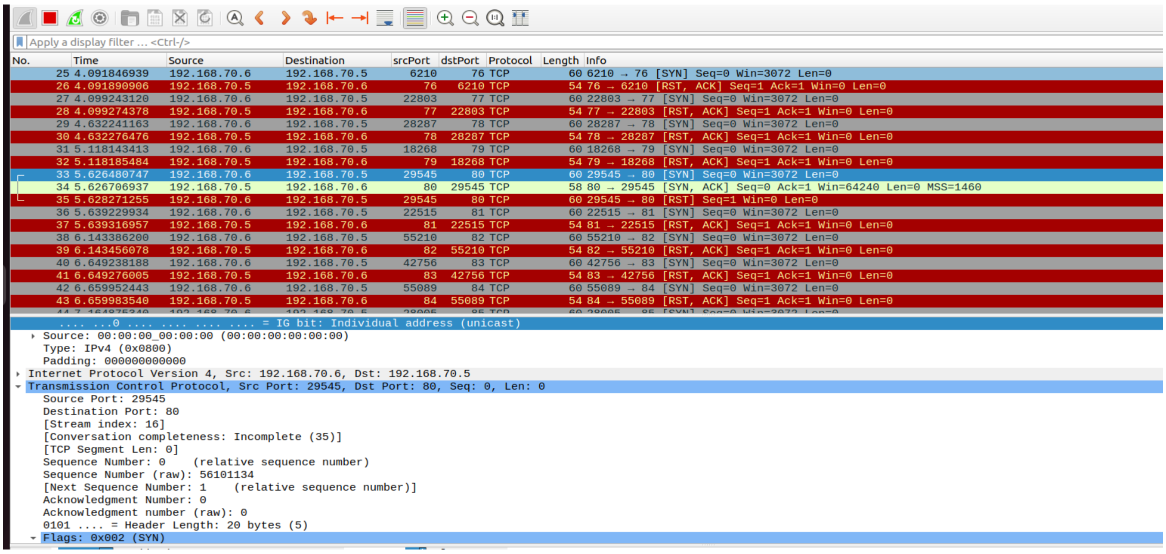
Enter the command run to see the open ports and from the below figure we can see that 80 is the open port.



```
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/syn) > run

[+]  TCP OPEN 192.168.70.5:80
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/syn) > 
```

Figure-5: msfconsole

The below figure-6 is giving 2 observations, when we observe the ports like 77, 78 ,79 ports as the communication is closed, they have [RST, ACK] message, when observing port 80 as it is now open port and allows the communication it has the [SYN, ACK] (synchronize-acknowledge) message.



Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	srcPort	dstPort	Protocol	Length	Info
25	4.091846939	192.168.70.6	192.168.70.5	6210	76	TCP	60	6210 → 76 [SYN] Seq=0 Win=3072 Len=0
26	4.091890906	192.168.70.5	192.168.70.6	76	6210	TCP	54	76 → 6210 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	4.099243120	192.168.70.6	192.168.70.5	22803	77	TCP	60	22803 → 77 [SYN] Seq=0 Win=3072 Len=0
28	4.099274378	192.168.70.5	192.168.70.6	77	22803	TCP	54	77 → 22803 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	4.632241163	192.168.70.6	192.168.70.5	28287	78	TCP	60	28287 → 78 [SYN] Seq=0 Win=3072 Len=0
30	4.632276476	192.168.70.5	192.168.70.6	78	28287	TCP	54	78 → 28287 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	5.118143413	192.168.70.6	192.168.70.5	18268	79	TCP	60	18268 → 79 [SYN] Seq=0 Win=3072 Len=0
32	5.118185484	192.168.70.5	192.168.70.6	79	18268	TCP	54	79 → 18268 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
33	5.626480747	192.168.70.6	192.168.70.5	29545	80	TCP	60	29545 → 80 [SYN] Seq=0 Win=3072 Len=0
34	5.626706937	192.168.70.5	192.168.70.6	80	29545	TCP	58	80 → 29545 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
35	5.628271255	192.168.70.6	192.168.70.5	29545	80	TCP	60	29545 → 80 [RST] Seq=1 Win=0 Len=0
36	5.639229934	192.168.70.6	192.168.70.5	22515	81	TCP	60	22515 → 81 [SYN] Seq=0 Win=3072 Len=0
37	5.639316957	192.168.70.5	192.168.70.6	81	22515	TCP	54	81 → 22515 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
38	6.143386200	192.168.70.6	192.168.70.5	55210	82	TCP	60	55210 → 82 [SYN] Seq=0 Win=3072 Len=0
39	6.143456078	192.168.70.5	192.168.70.6	82	55210	TCP	54	82 → 55210 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40	6.649238188	192.168.70.6	192.168.70.5	42756	83	TCP	60	42756 → 83 [SYN] Seq=0 Win=3072 Len=0
41	6.649276005	192.168.70.5	192.168.70.6	83	42756	TCP	54	83 → 42756 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42	6.659952443	192.168.70.6	192.168.70.5	55089	84	TCP	60	55089 → 84 [SYN] Seq=0 Win=3072 Len=0
43	6.659983540	192.168.70.5	192.168.70.6	84	55089	TCP	54	84 → 55089 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
44	7.164875340	192.168.70.6	192.168.70.5	28005	85	TCP	60	28005 → 85 [SYN] Seq=0 Win=3072 Len=0

```
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Type: IPv4 (0x0800)
    Padding: 000000000000
Internet Protocol Version 4, Src: 192.168.70.6, Dst: 192.168.70.5
Transmission Control Protocol, Src Port: 29545, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 29545
    Destination Port: 80
    [Stream index: 16]
    [Conversation completeness: Incomplete (35)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 56101134
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x002 (SYN)
```

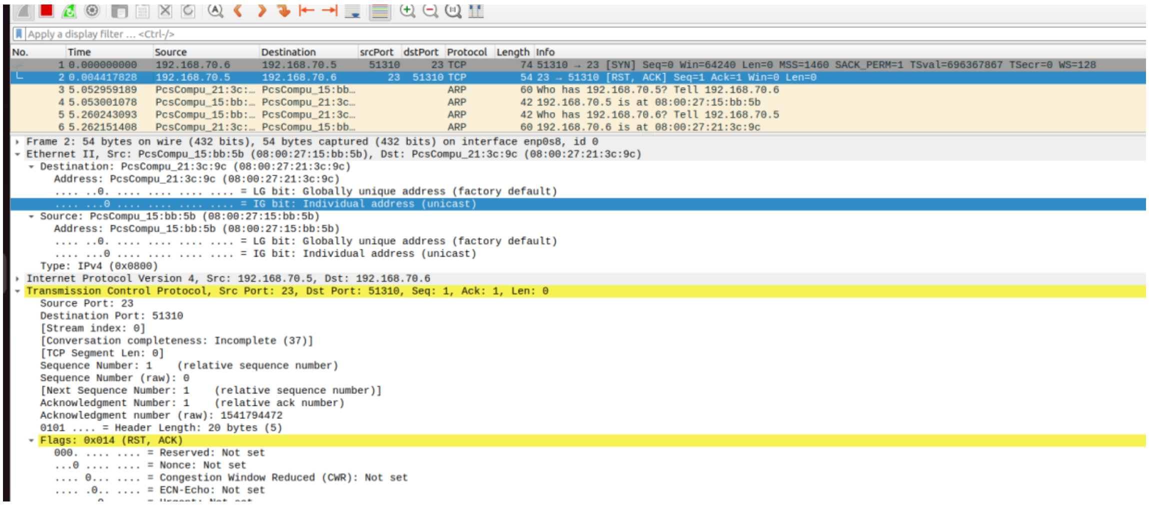
Figure 6: need to Metasploit wire shark picture.

Telnet connection

In general telnet is connected to port 23 by default if you did not give any port number in command, but in our case, it is different because we need to connect to the 80 only because it is the open port in our case.

In the below figure we can see that the telnet is not connected to the default port which is 23. So , we see the message as [RST,ACK] which says that there is no connection/ communication cannot be performed from this port.

Command used **“telnet 192.168.70.5”.**



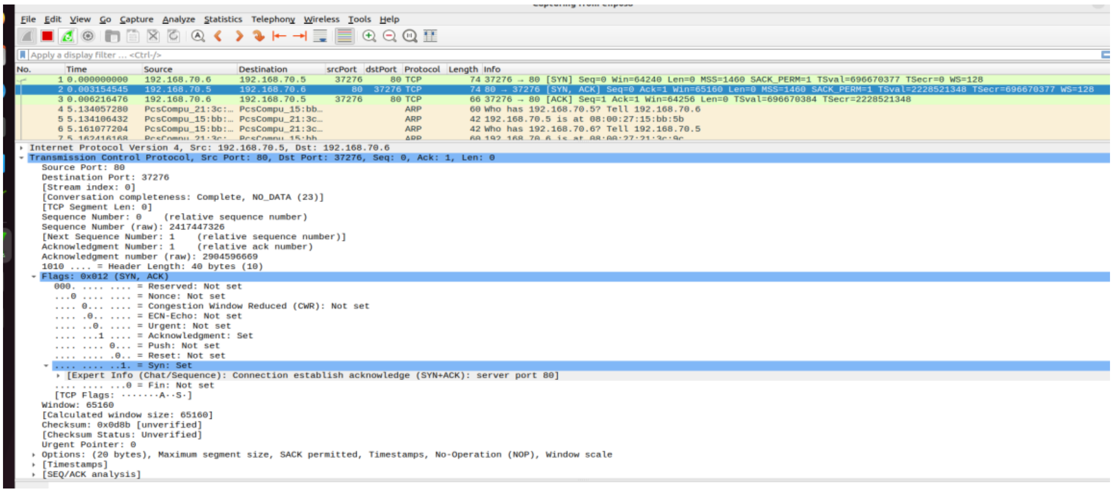
```
No.  Time           Source              Destination         srcPort  dstPort  Protocol  Length  Info
   1 0.000000000    192.168.70.6        192.168.70.5        51310       23    TCP       74 51310 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=696367867 TSecr=0 WS=128
   2 0.004417828    192.168.70.5        192.168.70.6           23    51310    TCP       54 23 → 51310 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   3 5.052959189    PcsCompu_21:3c:…    PcsCompu_15:bb…                       ARP       60 Who has 192.168.70.5? Tell 192.168.70.6
   4 5.053001078    PcsCompu_15:bb:…    PcsCompu_21:3c…                       ARP       42 192.168.70.5 is at 08:00:27:15:bb:5b
   5 5.260243093    PcsCompu_15:bb:…    PcsCompu_21:3c…                       ARP       42 Who has 192.168.70.6? Tell 192.168.70.5
   6 5.262151408    PcsCompu_21:3c:…    PcsCompu_15:bb…                       ARP       60 192.168.70.6 is at 08:00:27:21:3c:9c
```

```
› Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s8, id 0
⌄ Ethernet II, Src: PcsCompu_15:bb:5b (08:00:27:15:bb:5b), Dst: PcsCompu_21:3c:9c (08:00:27:21:3c:9c)
  ⌄ Destination: PcsCompu_21:3c:9c (08:00:27:21:3c:9c)
      Address: PcsCompu_21:3c:9c (08:00:27:21:3c:9c)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ⌄ Source: PcsCompu_15:bb:5b (08:00:27:15:bb:5b)
      Address: PcsCompu_15:bb:5b (08:00:27:15:bb:5b)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
› Internet Protocol Version 4, Src: 192.168.70.5, Dst: 192.168.70.6
⌄ Transmission Control Protocol, Src Port: 23, Dst Port: 51310, Seq: 1, Ack: 1, Len: 0
    Source Port: 23
    Destination Port: 51310
    [Stream index: 0]
    [Conversation completeness: Incomplete (37)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 0
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1541794472
    0101 .... = Header Length: 20 bytes (5)
  ⌄ Flags: 0x014 (RST, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
```

Figure7: telnet not connected without port.

To make the connection successful we need to give the open port in the command **“telnet 192.168.70.5 80”**.

we can see the connection (syn, syn/ack packets) which says the communication can be done through that port from the Wireshark in the below figure.



```
No.  Time           Source              Destination         srcPort  dstPort  Protocol  Length  Info
   1 0.000000000    192.168.70.6        192.168.70.5        37276       80    TCP       74 37276 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=696670377 TSecr=0 WS=128
   2 0.003154545    192.168.70.5        192.168.70.6           80    37276    TCP       74 80 → 37276 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2228521348 TSecr=696670377 WS=128
   3 0.006216476    192.168.70.6        192.168.70.5        37276       80    TCP       66 37276 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=696670384 TSecr=2228521348
   4 5.134057280    PcsCompu_21:3c:…    PcsCompu_15:bb…                       ARP       60 Who has 192.168.70.5? Tell 192.168.70.6
   5 5.134106432    PcsCompu_15:bb:…    PcsCompu_21:3c…                       ARP       42 192.168.70.5 is at 08:00:27:15:bb:5b
   6 5.161077204    PcsCompu_15:bb:…    PcsCompu_21:3c…                       ARP       42 Who has 192.168.70.6? Tell 192.168.70.5
   7 5.162416168    PcsCompu_21:3c:…    PcsCompu_15:bb…                       ARP       60 192.168.70.6 is at 08:00:27:21:3c:9c
```

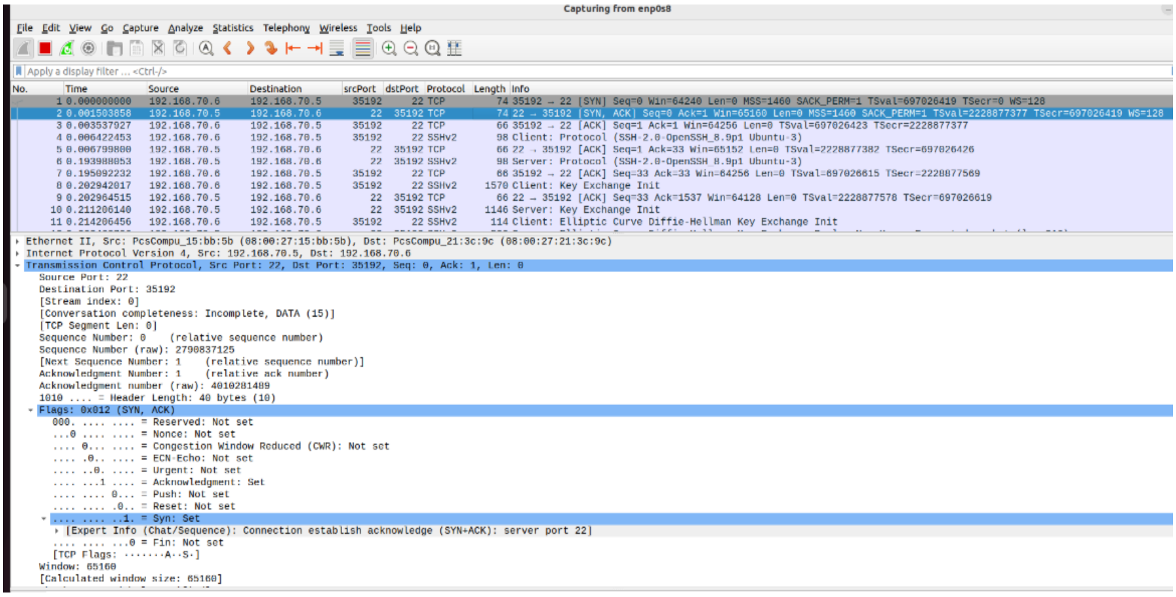
```
› Internet Protocol Version 4, Src: 192.168.70.5, Dst: 192.168.70.6
⌄ Transmission Control Protocol, Src Port: 80, Dst Port: 37276, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 37276
    [Stream index: 0]
    [Conversation completeness: Complete, NO_DATA (23)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2417447326
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2904596669
    1010 .... = Header Length: 40 bytes (10)
  ⌄ Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ⌄ .... .... ..1. = Syn: Set
      › [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 80]
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A··S·]
    Window: 65160
    [Calculated window size: 65160]
    Checksum: 0x0d8b [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  › Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  › [Timestamps]
  › [SEQ/ACK analysis]
```

Figure 8: telnet connected to with port 443.

SSH Connection

Now the patterns hold by running the ssh command can be seen from the Wireshark and below is the figure. It is connected to port 22 as it is open port we can see [SYN/ACK] .

The command used is “**ssh 192.168.70.5**”.


Capturing from enp0s8
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time         Source         Destination    srcPort dstPort Protocol Length Info
    1 0.000000000  192.168.70.6   192.168.70.5   35192      22 TCP        74 35192 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=697026419 TSecr=0 WS=128
    2 0.001503858  192.168.70.5   192.168.70.6      22   35192 TCP        74 22 → 35192 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2228877377 TSecr=697026419 WS=128
    3 0.003537927  192.168.70.6   192.168.70.5   35192      22 TCP        66 35192 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=697026423 TSecr=2228877377
    4 0.006422453  192.168.70.6   192.168.70.5   35192      22 SSHv2      98 Client: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3)
    5 0.006799800  192.168.70.5   192.168.70.6      22   35192 TCP        66 22 → 35192 [ACK] Seq=1 Ack=33 Win=65152 Len=0 TSval=2228877382 TSecr=697026426
    6 0.193988853  192.168.70.5   192.168.70.6      22   35192 SSHv2      98 Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3)
    7 0.195092232  192.168.70.6   192.168.70.5   35192      22 TCP        66 35192 → 22 [ACK] Seq=33 Ack=33 Win=64256 Len=0 TSval=697026615 TSecr=2228877569
    8 0.202942017  192.168.70.6   192.168.70.5   35192      22 SSHv2    1570 Client: Key Exchange Init
    9 0.202964515  192.168.70.5   192.168.70.6      22   35192 TCP        66 22 → 35192 [ACK] Seq=33 Ack=1537 Win=64128 Len=0 TSval=2228877578 TSecr=697026619
   10 0.211206140  192.168.70.5   192.168.70.6      22   35192 SSHv2    1146 Server: Key Exchange Init
   11 0.214206456  192.168.70.6   192.168.70.5   35192      22 SSHv2     114 Client: Elliptic Curve Diffie-Hellman Key Exchange Init

▸ Ethernet II, Src: PcsCompu_15:bb:5b (08:00:27:15:bb:5b), Dst: PcsCompu_21:3c:9c (08:00:27:21:3c:9c)
▸ Internet Protocol Version 4, Src: 192.168.70.5, Dst: 192.168.70.6
▾ Transmission Control Protocol, Src Port: 22, Dst Port: 35192, Seq: 0, Ack: 1, Len: 0
    Source Port: 22
    Destination Port: 35192
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2790837125
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 4010281489
    1010 .... = Header Length: 40 bytes (10)
  ▾ Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ▾ .... .... ..1. = Syn: Set
      ▸ [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 22]
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A··S·]
    Window: 65160
    [Calculated window size: 65160]
```

Figure 9: ssh connected.

Snort rule to detect the port scanning.

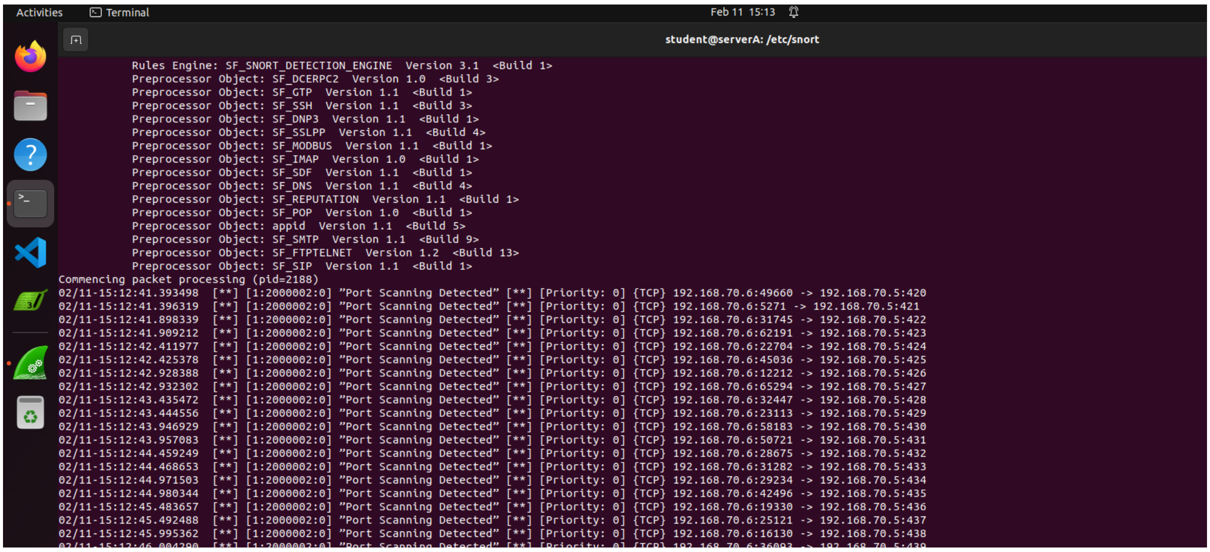
In server A a new snort rule is added in local.rules files which is used to detect the port scanning. IN this snort rules the source and destination Ip addresses are defined along with tcp protocol, message that need to be displayed and also flag and ttl are included. The snort rule is ***“alert tcp 192.168.70.6 any-> 192.168.70.5 any (msg: “Port Scanning Detected”; flags:S, ttl:32; fragbits: MDR!; sid:2000002) ”.***

The explanation of the above snort rule is in the below table.

alert	It generates the alert
tcp	It is the protocol name
192.168.70.6, 192.168.70.5	Source and destination IP address
->	Direction operation which takes care about the direction of the traffic
msg	The message that needs to be displayed.
flags	Keyword which is assigned to check the bit of the tcp protocol. S: indicates the syn sequence number
ttl	This value will impact the time-to-live of IP, it can take values between 0-255.
fragsbits	This key word is used to check the reserved and fragmentation bits in IP. M: indicates more fragemnets D: indicates don’t fragements. R : indicates reserved Bit

	!: it is a kind of regular expression matches if specific bits are not set.
sid	Unique number to identify the snort rule

The below figure is from Server A after executing the snort rule we can see that the snort is only detecting the port scanning but not the benign traffic.



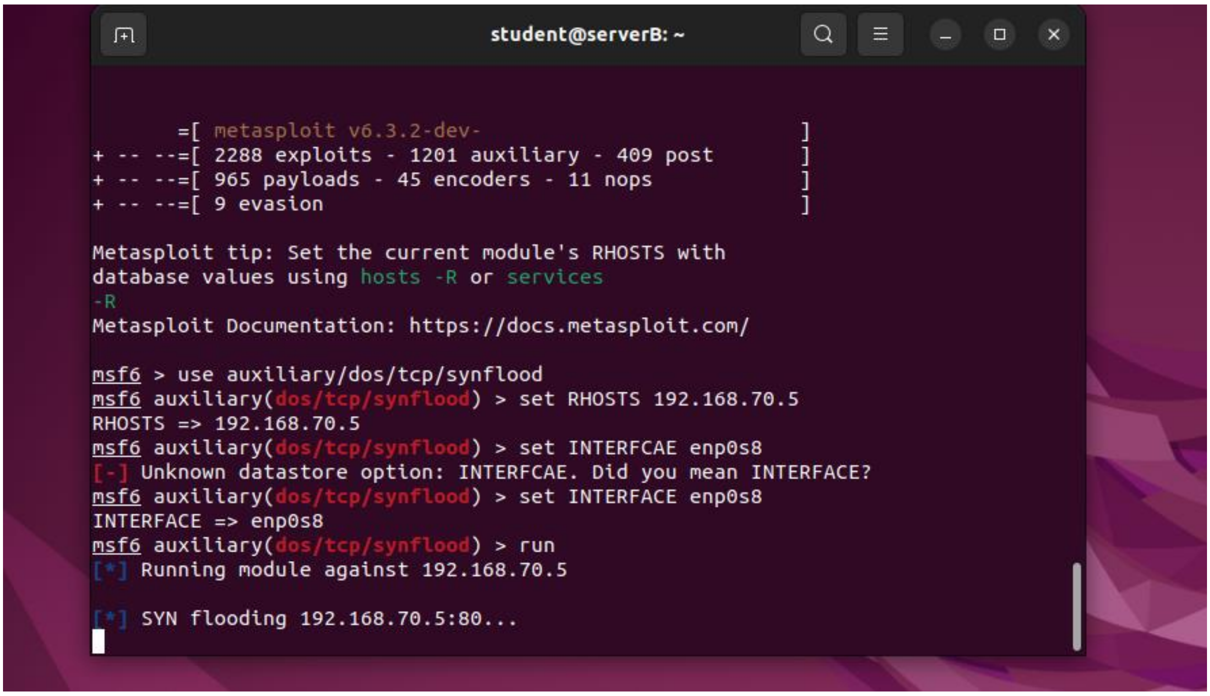
```
           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Commencing packet processing (pid=2188)
02/11-15:12:41.393498  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:49660 -> 192.168.70.5:420
02/11-15:12:41.396319  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:5271 -> 192.168.70.5:421
02/11-15:12:41.898339  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:31745 -> 192.168.70.5:422
02/11-15:12:41.909212  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:62191 -> 192.168.70.5:423
02/11-15:12:42.411977  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:22704 -> 192.168.70.5:424
02/11-15:12:42.425378  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:45036 -> 192.168.70.5:425
02/11-15:12:42.928388  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:12212 -> 192.168.70.5:426
02/11-15:12:42.932302  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:65294 -> 192.168.70.5:427
02/11-15:12:43.435472  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:32447 -> 192.168.70.5:428
02/11-15:12:43.444556  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:23113 -> 192.168.70.5:429
02/11-15:12:43.946929  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:58183 -> 192.168.70.5:430
02/11-15:12:43.957083  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:50721 -> 192.168.70.5:431
02/11-15:12:44.459249  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:28675 -> 192.168.70.5:432
02/11-15:12:44.468653  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:31282 -> 192.168.70.5:433
02/11-15:12:44.971503  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:29234 -> 192.168.70.5:434
02/11-15:12:44.980344  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:42496 -> 192.168.70.5:435
02/11-15:12:45.483657  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:19330 -> 192.168.70.5:436
02/11-15:12:45.492488  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:25121 -> 192.168.70.5:437
02/11-15:12:45.995362  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:16130 -> 192.168.70.5:438
02/11-15:12:46.004290  [**] [1:2000002:0] ”Port Scanning Detected” [**] [Priority: 0] {TCP} 192.168.70.6:36093 -> 192.168.70.5:439
```

Figure 10: new rule in snort.

Task-4: Detect DoS Attack

This task is associated with DoS attack detection, to do that the **“auxiliary/dos/tcp/synflood”** is used the synflooder is used to create the traffic at the receiver side that legitimate connection cannot be accepted. Now configure the module by setting the **RHOSTS** as **192.168.70.5** and **INTERFACE** as **enp0s8**, when you start run command it will start the attack on server A.

Below figure is configuring the module on server B, to attack the server A.



```
       =[ metasploit v6.3.2-dev-                          ]
+ -- --=[ 2288 exploits - 1201 auxiliary - 409 post       ]
+ -- --=[ 965 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Set the current module's RHOSTS with
database values using hosts -R or services
-R
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.70.5
RHOSTS => 192.168.70.5
msf6 auxiliary(dos/tcp/synflood) > set INTERFCAE enp0s8
[-] Unknown datastore option: INTERFCAE. Did you mean INTERFACE?
msf6 auxiliary(dos/tcp/synflood) > set INTERFACE enp0s8
INTERFACE => enp0s8
msf6 auxiliary(dos/tcp/synflood) > run
[*] Running module against 192.168.70.5

[*] SYN flooding 192.168.70.5:80...
```

Figure 11: Metasploit connection

The snort is rule is added to the local.rules file on the server A which is written to characterize the high frequency SYN packets.

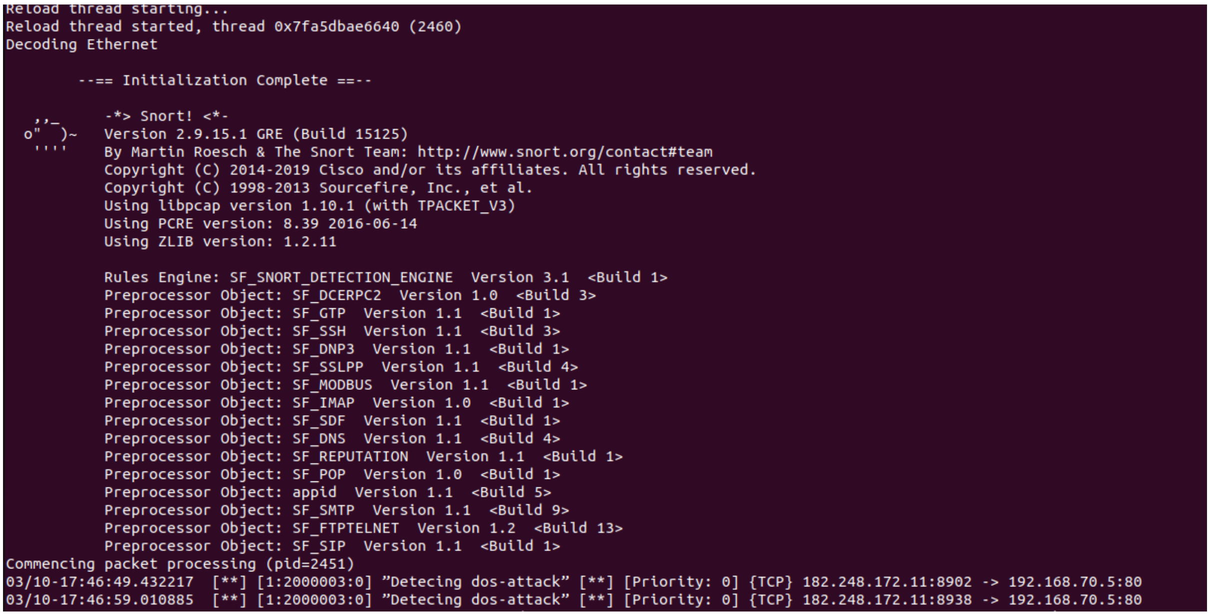
The snort rule is ***"alert tcp any any-> 192.168.70.5 80 (msg: "Detecting dos-attack"; flags:S, fragbits: MDR!; threshold: type both, track by_dst,count 10,seconds 10; sid:2000003) ".***

The explanation of the above snort rule is in the below table.

alert	It generates the alert
tcp	It is the protocol name
any, 192.168.70.5	Source and destination IP address, here any in source indicates from anywhere the source is accepted.
->	Direction operation which takes care about the direction of the traffic
msg	The message that needs to be displayed when alert is trigged.
flags	Keyword which is assigned to check the bit of the tcp protocol. S: indicates the syn sequence number
fragsbits	This key word is used to check the reserved and fragmentation bits in IP. M: indicates more fragments D: indicates don't fragments. R : indicates reserved Bit !: it is a kind of regular expression matches if specific bits are not set.
Threshold: type both	threshold: number of times an event is triggered with in each time.

Count, second	They both are related no of events occurred over a period.
sid	Unique number to identify the snort rule

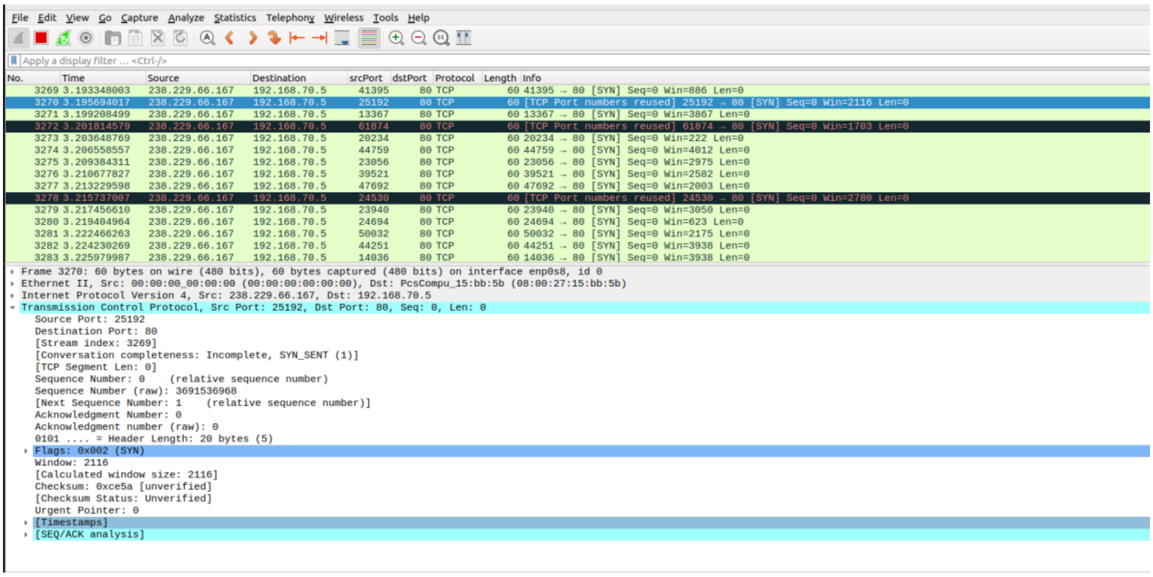
After adding the snort rule to the local.rules file, to verify whether the rule is working properly or not the rule need to be executed, the below figure represents the execution of the snort rule on server A. In the below figure the snort rule is not generating the alerts for the benign traffic.



```
Reload thread starting...
Reload thread started, thread 0x7fa5dbae6640 (2460)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Commencing packet processing (pid=2451)
03/10-17:46:49.432217  [**] [1:2000003:0] "Detecing dos-attack" [**] [Priority: 0] {TCP} 182.248.172.11:8902 -> 192.168.70.5:80
03/10-17:46:59.010885  [**] [1:2000003:0] "Detecing dos-attack" [**] [Priority: 0] {TCP} 182.248.172.11:8938 -> 192.168.70.5:80
```

Figure 12: server A execution of snort rule

The below figure is the malicious traffic detected by the snort rule that is captured using wire shark.

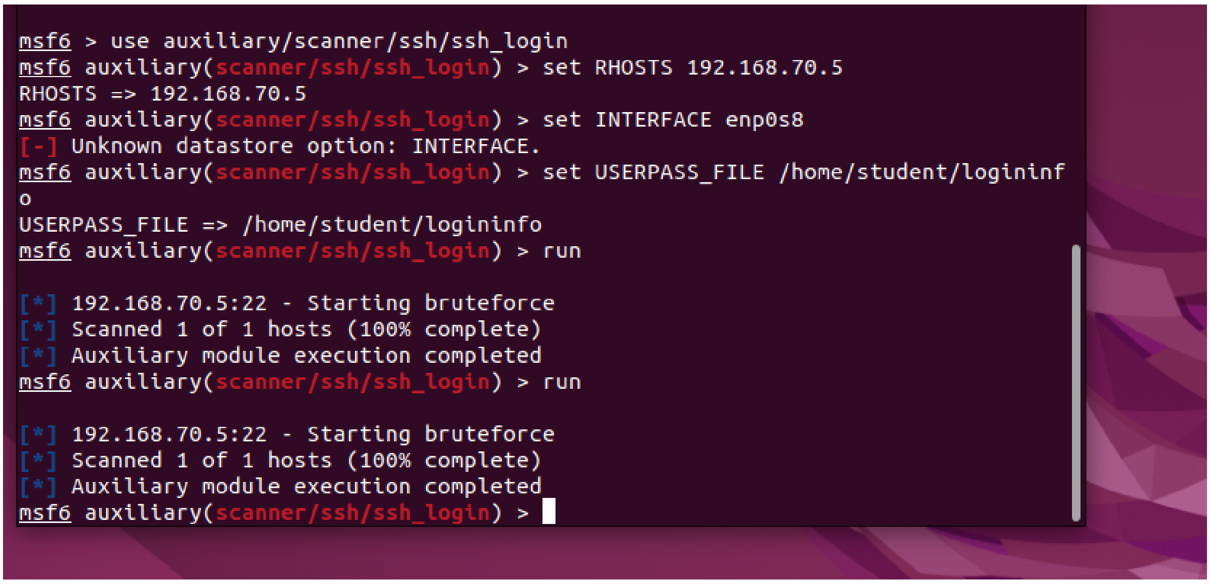

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time         Source          Destination    srcPort  dstPort Protocol  Length Info
 3269 3.193348003  238.229.66.167  192.168.70.5   41395    80 TCP            60 41395 → 80 [SYN] Seq=0 Win=886 Len=0
 3270 3.195694017  238.229.66.167  192.168.70.5   25192    80 TCP            60 [TCP Port numbers reused] 25192 → 80 [SYN] Seq=0 Win=2116 Len=0
 3271 3.199208499  238.229.66.167  192.168.70.5   13367    80 TCP            60 13367 → 80 [SYN] Seq=0 Win=3867 Len=0
 3272 3.201814579  238.229.66.167  192.168.70.5   61874    80 TCP            60 [TCP Port numbers reused] 61874 → 80 [SYN] Seq=0 Win=1703 Len=0
 3273 3.203648769  238.229.66.167  192.168.70.5   20234    80 TCP            60 20234 → 80 [SYN] Seq=0 Win=222 Len=0
 3274 3.206558557  238.229.66.167  192.168.70.5   44759    80 TCP            60 44759 → 80 [SYN] Seq=0 Win=4012 Len=0
 3275 3.209384311  238.229.66.167  192.168.70.5   23056    80 TCP            60 23056 → 80 [SYN] Seq=0 Win=2975 Len=0
 3276 3.210677827  238.229.66.167  192.168.70.5   39521    80 TCP            60 39521 → 80 [SYN] Seq=0 Win=2582 Len=0
 3277 3.213229598  238.229.66.167  192.168.70.5   47692    80 TCP            60 47692 → 80 [SYN] Seq=0 Win=2003 Len=0
 3278 3.215737007  238.229.66.167  192.168.70.5   24530    80 TCP            60 [TCP Port numbers reused] 24530 → 80 [SYN] Seq=0 Win=2780 Len=0
 3279 3.217456610  238.229.66.167  192.168.70.5   23940    80 TCP            60 23940 → 80 [SYN] Seq=0 Win=3050 Len=0
 3280 3.219404964  238.229.66.167  192.168.70.5   24694    80 TCP            60 24694 → 80 [SYN] Seq=0 Win=623 Len=0
 3281 3.222466263  238.229.66.167  192.168.70.5   50032    80 TCP            60 50032 → 80 [SYN] Seq=0 Win=2175 Len=0
 3282 3.224230269  238.229.66.167  192.168.70.5   44251    80 TCP            60 44251 → 80 [SYN] Seq=0 Win=3938 Len=0
 3283 3.225979987  238.229.66.167  192.168.70.5   14036    80 TCP            60 14036 → 80 [SYN] Seq=0 Win=3938 Len=0
› Frame 3270: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s8, id 0
› Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: PcsCompu_15:bb:5b (08:00:27:15:bb:5b)
› Internet Protocol Version 4, Src: 238.229.66.167, Dst: 192.168.70.5
˅ Transmission Control Protocol, Src Port: 25192, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 25192
    Destination Port: 80
    [Stream index: 3269]
    [Conversation completeness: Incomplete, SYN_SENT (1)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3691536968
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x002 (SYN)
    Window: 2116
    [Calculated window size: 2116]
    Checksum: 0xce5a [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  › [Timestamps]
  › [SEQ/ACK analysis]
```

Figure – 13Wireshark of malicious traffic

Task-5 Detect incoming rogue SSH connections.

In this task we will detect the incoming rogue ssh connections to do that a text file which is the collection of login names and passwords is created in the home/student and that file is named as logininfo. In server B the **auxiliary/scanner/ssh/ssh_login** module is selected then the RHOSTS and interface is configured. Now in Metasploit **USERPASS_FILE** variable is set to the path where the logininfo file is present then we will run .



```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.70.5
RHOSTS => 192.168.70.5
msf6 auxiliary(scanner/ssh/ssh_login) > set INTERFACE enp0s8
[-] Unknown datastore option: INTERFACE.
msf6 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE /home/student/logininf
o
USERPASS_FILE => /home/student/logininfo
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.70.5:22 - Starting bruteforce
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.70.5:22 - Starting bruteforce
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > 
```

Figure:14 server B pic

The snort rule is added to the local.rules files in server A which is used to identify the brute force attack which is done by the server B and below is the snort rule.

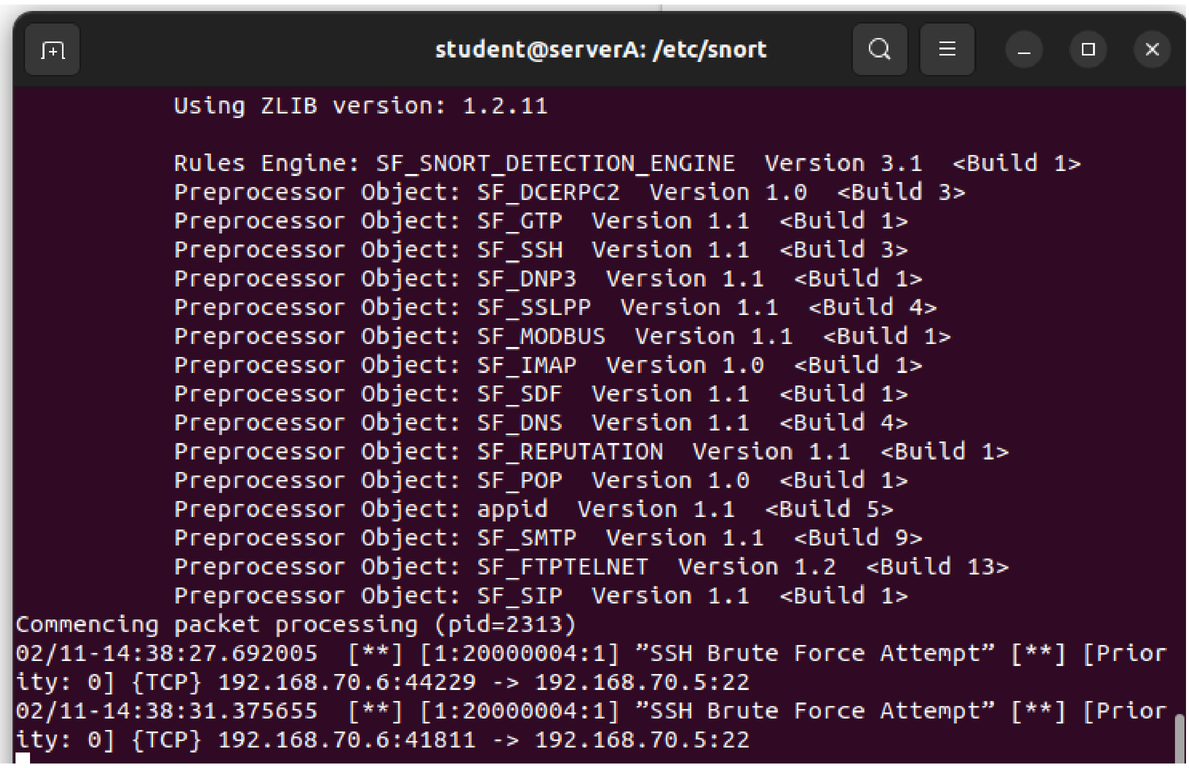
"alert tcp 192.168.70.6 any ->192.168.70.5 22 (msg: "SSH Brute Force Attempt"; flow: established, to_server; content: "SSH-2.0-OpenSSH"; nocase; offset:0; depth:16; detection_filter: track by_src, count 1, seconds 60; sid:2000004; rev:1;)".

The explanation of the above snort rule is in the below table.

alert	It generates the alert
tcp	It is the protocol name
192.168.70.6, 192.168.70.5	Source and destination IP address, here any in source indicates from anywhere the source is accepted.
->	Direction operation which takes care about the direction of the traffic
msg	The message that needs to be displayed when alert is trigged.
to_server	it activities upon the request from A to B.
nocase	don't consider the case
depth	this tells about how far the rule should need to search for the pattern
offset	this helps where to start the search for the pattern.
Detection_filter	track_by_src tracking by either source or destination IP address.

Count, second	They both are related no of events occurred over a period.
sid	Unique number to identify the snort rule
rev	

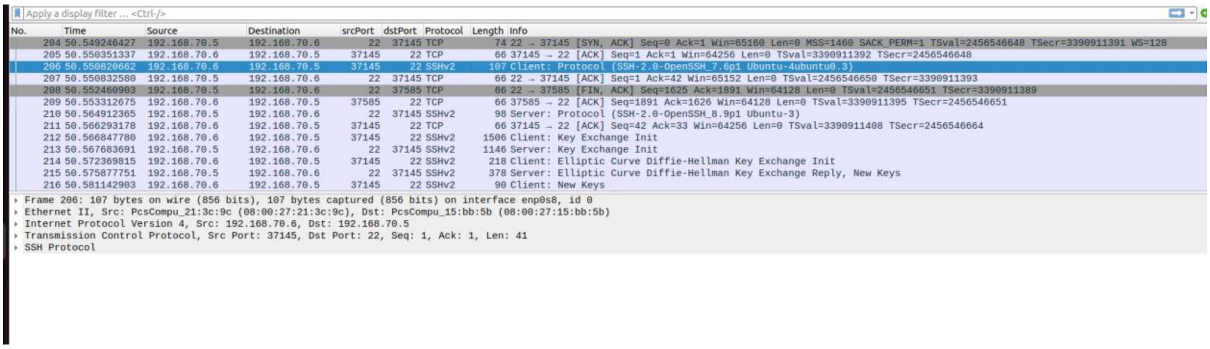
After saving the snort rule to the local.rule file then execute the file in server A. we can see that rule alerts only on malicious connection attempts, and that alerts are not generated for benign traffic.



```
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Commencing packet processing (pid=2313)
02/11-14:38:27.692005  [**] [1:20000004:1] "SSH Brute Force Attempt" [**] [Prior
ity: 0] {TCP} 192.168.70.6:44229 -> 192.168.70.5:22
02/11-14:38:31.375655  [**] [1:20000004:1] "SSH Brute Force Attempt" [**] [Prior
ity: 0] {TCP} 192.168.70.6:41811 -> 192.168.70.5:22
```

Figure:15 server A

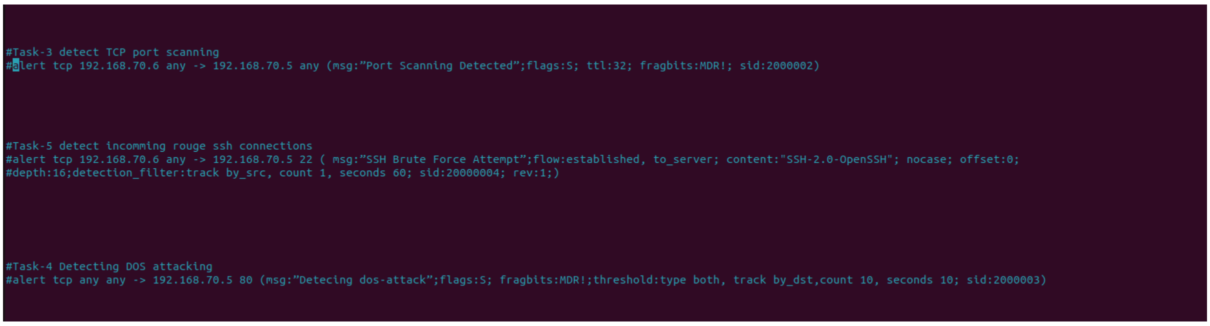
The below figure is the wire shark picture which is used to study and verifies the malicious connections attempts.



```
No.   Time          Source         Destination    srcPort dstPort Protocol Length Info
  204 50.549246427  192.168.70.5   192.168.70.6        22   37145 TCP         74 22 → 37145 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2456546648 TSecr=3390911391 WS=128
  205 50.550351337  192.168.70.6   192.168.70.5     37145      22 TCP         66 37145 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3390911392 TSecr=2456546648
  206 50.550820662  192.168.70.6   192.168.70.5     37145      22 SSHv2      107 Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
  207 50.550832580  192.168.70.5   192.168.70.6        22   37145 TCP         66 22 → 37145 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=2456546650 TSecr=3390911393
  208 50.552460903  192.168.70.5   192.168.70.6        22   37585 TCP         66 22 → 37585 [FIN, ACK] Seq=1625 Ack=1891 Win=64128 Len=0 TSval=2456546651 TSecr=3390911389
  209 50.553312675  192.168.70.6   192.168.70.5     37585      22 TCP         66 37585 → 22 [ACK] Seq=1891 Ack=1626 Win=64128 Len=0 TSval=3390911395 TSecr=2456546651
  210 50.564912365  192.168.70.5   192.168.70.6        22   37145 SSHv2       98 Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3)
  211 50.566293178  192.168.70.6   192.168.70.5     37145      22 TCP         66 37145 → 22 [ACK] Seq=42 Ack=33 Win=64256 Len=0 TSval=3390911408 TSecr=2456546664
  212 50.566847780  192.168.70.6   192.168.70.5     37145      22 SSHv2     1506 Client: Key Exchange Init
  213 50.567683691  192.168.70.5   192.168.70.6        22   37145 SSHv2     1146 Server: Key Exchange Init
  214 50.572369815  192.168.70.6   192.168.70.5     37145      22 SSHv2      218 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
  215 50.575877751  192.168.70.5   192.168.70.6        22   37145 SSHv2      378 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
  216 50.581142903  192.168.70.6   192.168.70.5     37145      22 SSHv2       90 Client: New Keys
▸ Frame 206: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface enp0s8, id 0
▸ Ethernet II, Src: PcsCompu_21:3c:9c (08:00:27:21:3c:9c), Dst: PcsCompu_15:bb:5b (08:00:27:15:bb:5b)
▸ Internet Protocol Version 4, Src: 192.168.70.6, Dst: 192.168.70.5
▸ Transmission Control Protocol, Src Port: 37145, Dst Port: 22, Seq: 1, Ack: 1, Len: 41
▸ SSH Protocol
```

Figure:16 wire shark

The below figure is the local.rules file where the snort for the tasks 3,4 and 5 are stored. The snort rules are gathered from the official documentation[1].



```
#Task-3 detect TCP port scanning
#alert tcp 192.168.70.6 any -> 192.168.70.5 any (msg:”Port Scanning Detected”;flags:S; ttl:32; fragbits:MDR!; sid:2000002)



#Task-5 detect incomming rouge ssh connections
#alert tcp 192.168.70.6 any -> 192.168.70.5 22 ( msg:”SSH Brute Force Attempt”;flow:established, to_server; content:"SSH-2.0-OpenSSH"; nocase; offset:0;
#depth:16;detection_filter:track by_src, count 1, seconds 60; sid:20000004; rev:1;)



#Task-4 Detecting DOS attacking
#alert tcp any any -> 192.168.70.5 80 (msg:”Detecing dos-attack”;flags:S; fragbits:MDR!;threshold:type both, track by_dst,count 10, seconds 10; sid:2000003)
```

Figure:17 all snort rules

References

[1] “Snort - Network Intrusion Detection & Prevention System.” https://www.snort.org/#documents (accessed Mar. 04, 2023).