Joseph Tsai
SNORT Intrusion Detection System Lab

Introduction

SNORT is a network packet sniffing tool that can also be utilized as an Intrusion Detection System, or IDS. At a fundamental level, SNORT allows its users to create rules that take specific actions when packets meet a given criteria. This lab report explores SNORT and its practical uses as it relates to detecting scans, and alerting users when certain packet criteria are met.

Configuration of network environment

To complete this lab, the following network environment was utilized:

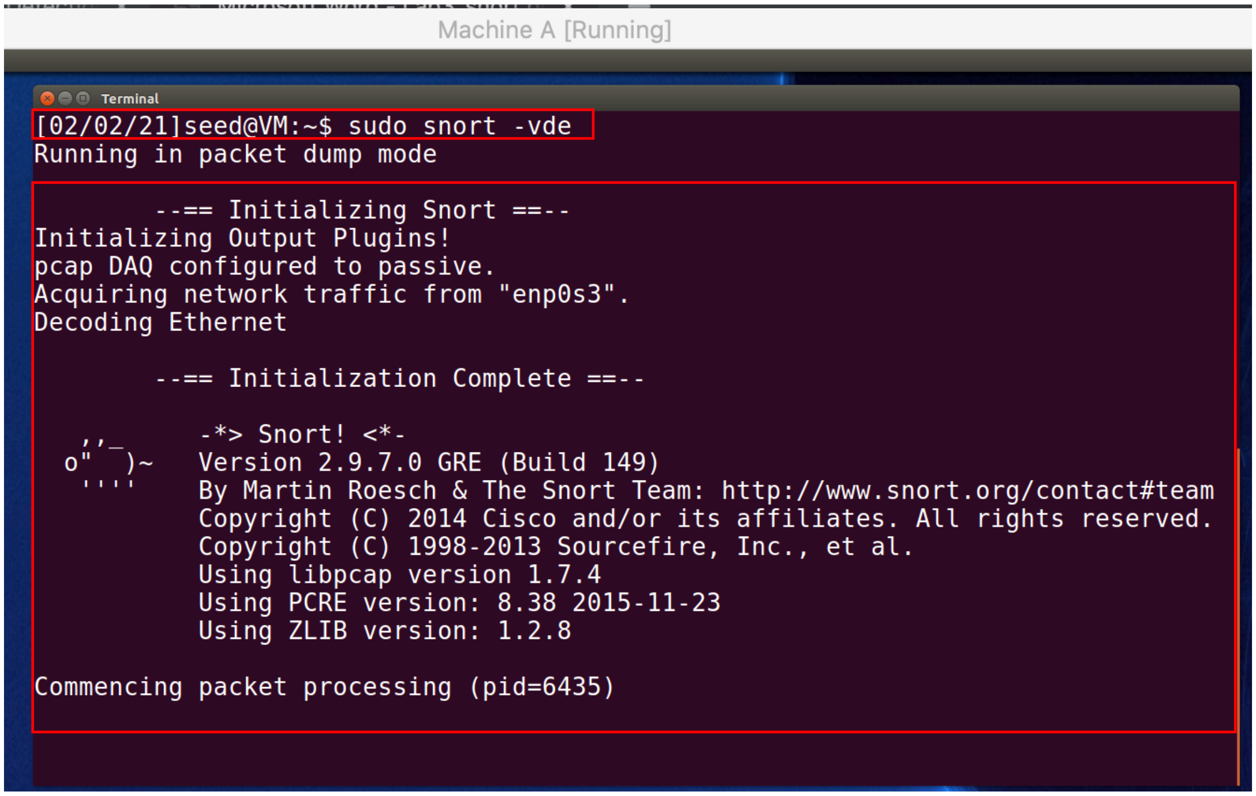
- Two hosts, both sitting on the same network.
- One host was a SEED lab machine which contains the SNORT program. This is referred to as "Machine A" in this report.
- The second host is a Metasploit machine which is running nmap. This is referred to as "Machine B" in this report.

Lab Tasks

Task 1: Use SNORT as a Packet Sniffer

The goal of this task is to test the proper setup and functionality of SNORT on Machine A, though pinging A from Machine B.

Exhibit 1: Enabling the SNORT program on Machine A. This enables Machine A to begin sniffing packets on the network it shares with Machine B. The command to enable SNORT has been indicated, below, along with the system message displaying the successful run of the command.

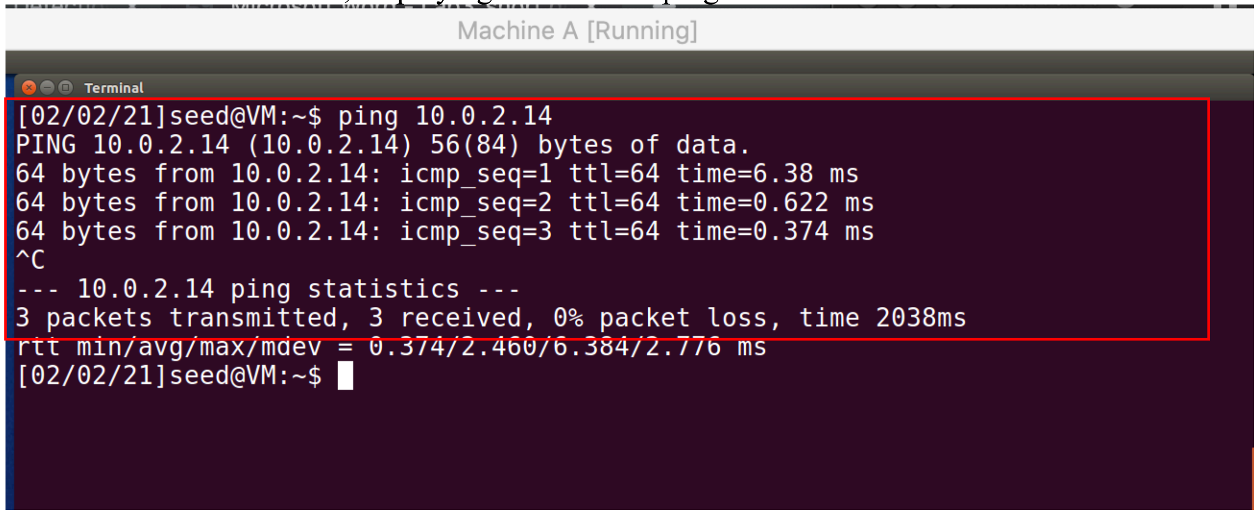


```
[02/02/21]seed@VM:~$ sudo snort -vde
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.7.4
           Using PCRE version: 8.38 2015-11-23
           Using ZLIB version: 1.2.8

Commencing packet processing (pid=6435)
```

Joseph Tsai
SNORT Intrusion Detection System Lab

To test if the SNORT program is working properly, a separate terminal window on Machine A is used to ping Machine B, which has the IP address of 10.0.2.14.

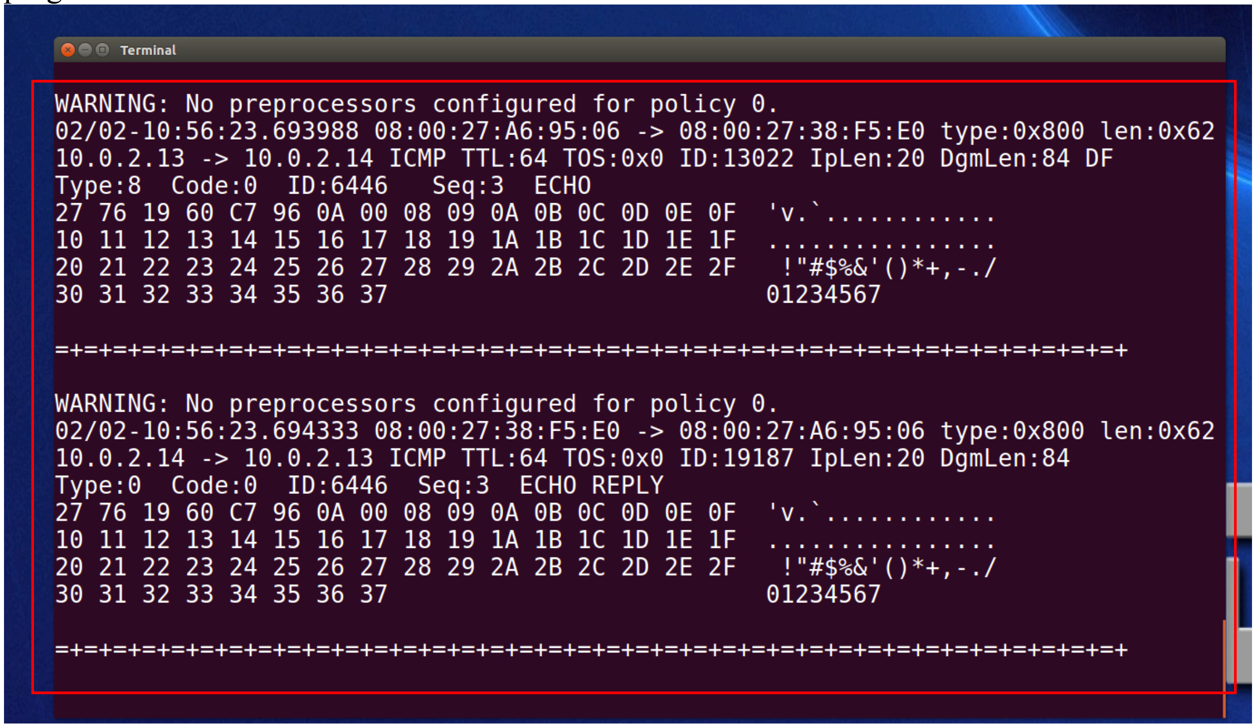
Exhibit 2: Pinging Machine B from Machine A by utilizing the *ping* command. Note how bytes are returned to Machine A, displaying a successful ping of Machine B.



```
[02/02/21]seed@VM:~$ ping 10.0.2.14
PING 10.0.2.14 (10.0.2.14) 56(84) bytes of data.
64 bytes from 10.0.2.14: icmp_seq=1 ttl=64 time=6.38 ms
64 bytes from 10.0.2.14: icmp_seq=2 ttl=64 time=0.622 ms
64 bytes from 10.0.2.14: icmp_seq=3 ttl=64 time=0.374 ms
^C
--- 10.0.2.14 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.374/2.460/6.384/2.776 ms
[02/02/21]seed@VM:~$
```

The *ping* traffic generated by Machine A and B is then displayed within the SNORT program, as displayed below.

Exhibit 3: Corresponding capture from the SNORT program, displaying the traffic being sent between the two machines. Note how the packet type is ICMP, as expected when using the *ping* program.



```
WARNING: No preprocessors configured for policy 0.
02/02-10:56:23.693988 08:00:27:A6:95:06 -> 08:00:27:38:F5:E0 type:0x800 len:0x62
10.0.2.13 -> 10.0.2.14 ICMP TTL:64 TOS:0x0 ID:13022 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6446   Seq:3  ECHO
27 76 19 60 C7 96 0A 00 08 09 0A 0B 0C 0D 0E 0F  'v.`............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
02/02-10:56:23.694333 08:00:27:38:F5:E0 -> 08:00:27:A6:95:06 type:0x800 len:0x62
10.0.2.14 -> 10.0.2.13 ICMP TTL:64 TOS:0x0 ID:19187 IpLen:20 DgmLen:84
Type:0  Code:0  ID:6446  Seq:3  ECHO REPLY
27 76 19 60 C7 96 0A 00 08 09 0A 0B 0C 0D 0E 0F  'v.`............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

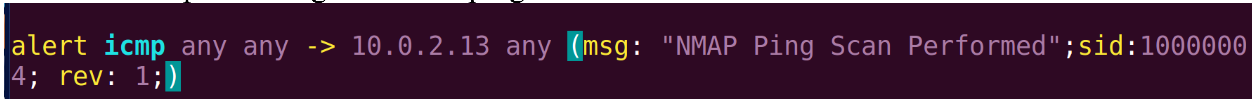
Joseph Tsai
SNORT Intrusion Detection System Lab

Task 2: Run SNORT as IDS to detect ping scans

SNORT can also be used to detect certain kinds of scans. This section explores how SNORT rules can be configured to detect such scans.

To begin, the following rule is implemented into the local.rules file for SNORT. This rule creates an alert whenever there is a captured ICMP packet originating from any destination IP or destination port with a destination of Machine A's (10.0.2.13) IP address, on any Machine A port. A corresponding message is displayed when a relevant packet is captured.

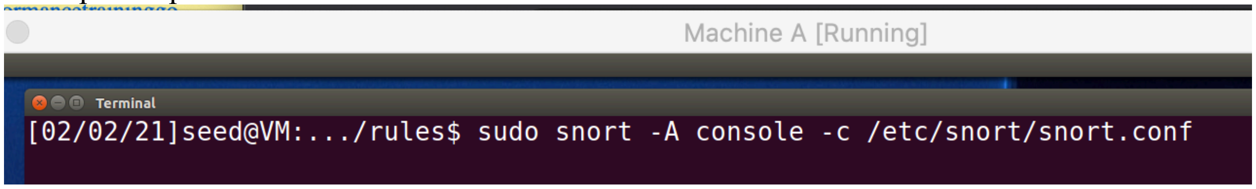
Exhibit 4: Implementing the ICMP ping scan detection rules within the SNORT rules.



```
alert icmp any any -> 10.0.2.13 any (msg: "NMAP Ping Scan Performed";sid:1000000
4; rev: 1;)
```

To test the rule, Machine A then begins to run SNORT.

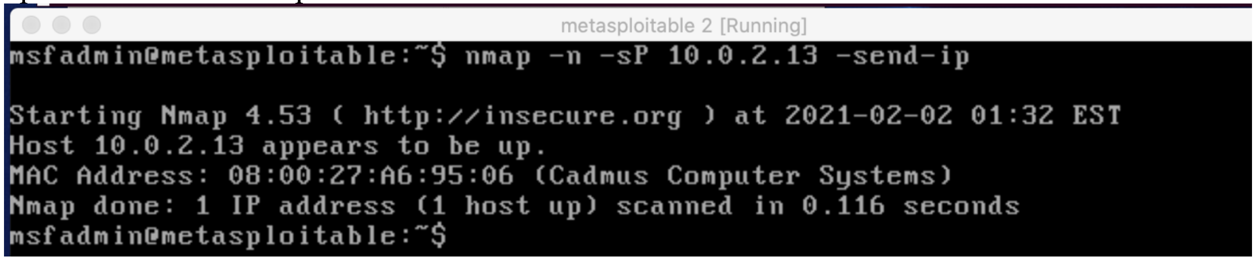
Exhibit 5: Enabling SNORT on Machine A to sniff packets and configuring SNORT to output the captured packets to the console.



```
[02/02/21]seed@VM:.../rules$ sudo snort -A console -c /etc/snort/snort.conf
```

Machine B is then used to perform a ping scan on Machine A

Exhibit 6: Performing a ping scan on Machine A from Machine B. This is indicated in the *-sP* option used in the nmap command.

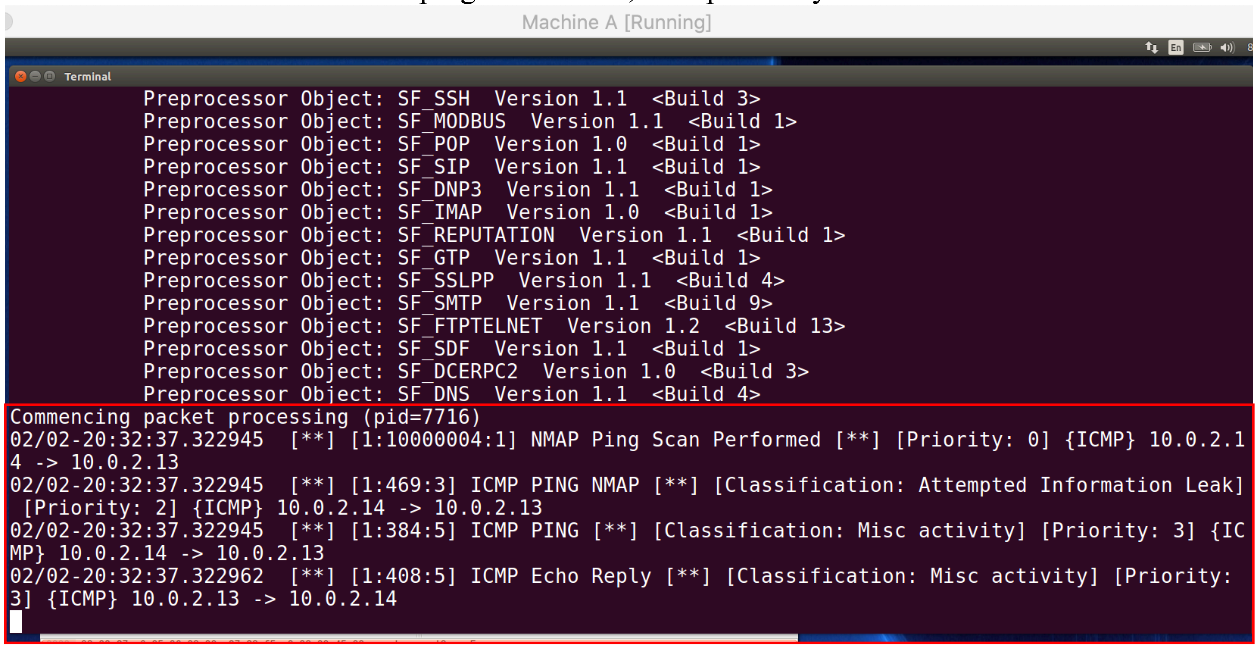


```
msfadmin@metasploitable:~$ nmap -n -sP 10.0.2.13 -send-ip

Starting Nmap 4.53 ( http://insecure.org ) at 2021-02-02 01:32 EST
Host 10.0.2.13 appears to be up.
MAC Address: 08:00:27:A6:95:06 (Cadmus Computer Systems)
Nmap done: 1 IP address (1 host up) scanned in 0.116 seconds
msfadmin@metasploitable:~$
```

Joseph Tsai
SNORT Intrusion Detection System Lab

The results of the traffic are captured by SNORT on Machine A.

Exhibit 7: Results of the ICMP ping scan traffic, as captured by SNORT on Machine A.



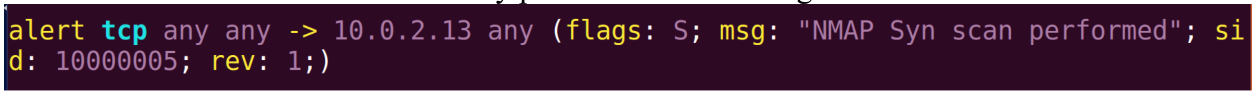
```
          Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
          Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
          Preprocessor Object: SF_POP  Version 1.0  <Build 1>
          Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
          Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
          Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
          Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
          Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
          Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
          Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
          Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
          Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
          Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Commencing packet processing (pid=7716)
02/02-20:32:37.322945  [**] [1:10000004:1] NMAP Ping Scan Performed [**] [Priority: 0] {ICMP} 10.0.2.1
4 -> 10.0.2.13
02/02-20:32:37.322945  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Attempted Information Leak]
 [Priority: 2] {ICMP} 10.0.2.14 -> 10.0.2.13
02/02-20:32:37.322945  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {IC
MP} 10.0.2.14 -> 10.0.2.13
02/02-20:32:37.322962  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority:
3] {ICMP} 10.0.2.13 -> 10.0.2.14
```

Task 3: Run SNORT as IDS to detect port scans

SNORT can also be utilized to detect different types of port scans. The following steps explore different types of port scans, and how SNORT can be configured to detect such scans. This section will follow the format of: Displaying the ruleset on Machine A in SNORT, running the nmap command on Machine B, and displaying the output of the captured traffic on Machine A.

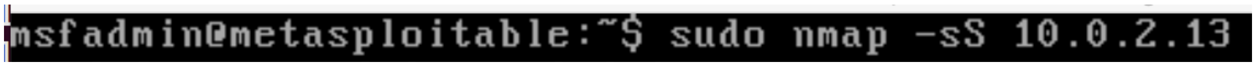
Step 1: Use Nmap to launch a SYN scan attack, create a rule to detect the attack, test the rule and check the logs

Exhibit 8: Rule set on Machine A to detect TCP scans originating from any host from any port with a destination of Machine A on any port with the SYN flag indicated.



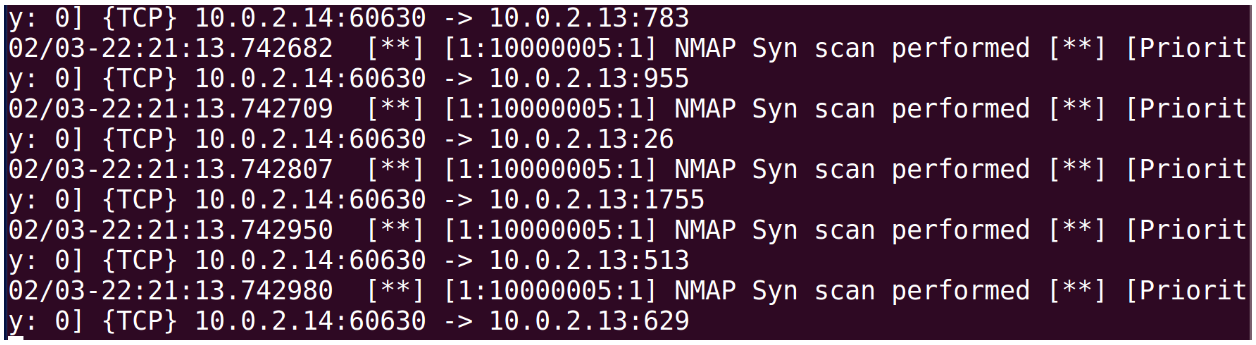
```
alert tcp any any -> 10.0.2.13 any (flags: S; msg: "NMAP Syn scan performed"; si
d: 10000005; rev: 1;)
```

Exhibit 9: Performing the SYN scan on Machine B with the following command, as indicated by the *-sS* option.



```
msfadmin@metasploitable:~$ sudo nmap -sS 10.0.2.13
```

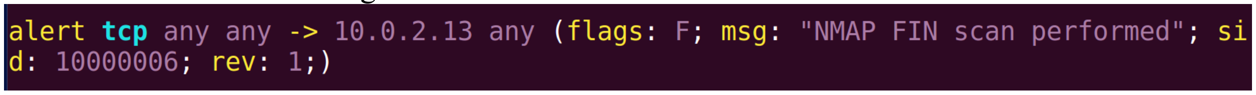
Exhibit 10: Resulting TCP traffic captured on Machine A, with corresponding alert message which was set in Exhibit 8.



```
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:783
02/03-22:21:13.742682  [**] [1:10000005:1] NMAP Syn scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:955
02/03-22:21:13.742709  [**] [1:10000005:1] NMAP Syn scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:26
02/03-22:21:13.742807  [**] [1:10000005:1] NMAP Syn scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:1755
02/03-22:21:13.742950  [**] [1:10000005:1] NMAP Syn scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:513
02/03-22:21:13.742980  [**] [1:10000005:1] NMAP Syn scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:60630 -> 10.0.2.13:629
```

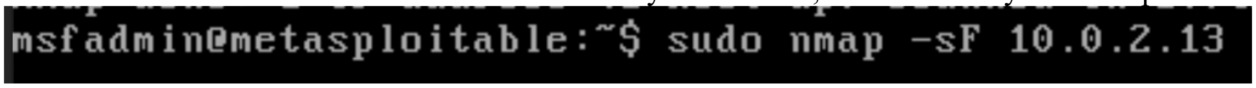
Step 2: Use Nmap to launch a FIN scan attack, create a rule to detect the attack, test the rule and check the logs

Exhibit 11: Rule set on Machine A to detect TCP traffic from any host and port to any port on Machine A with the FIN flag set.



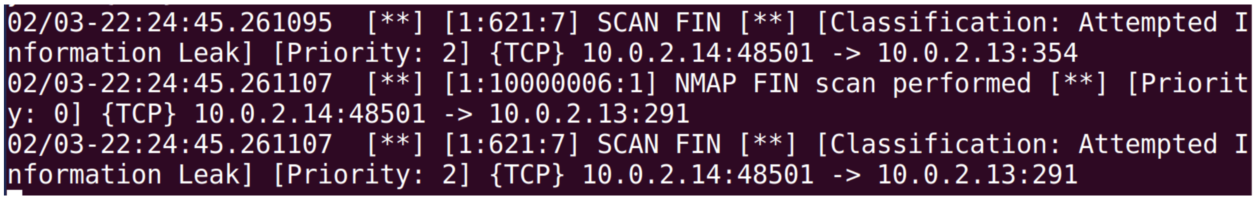
```
alert tcp any any -> 10.0.2.13 any (flags: F; msg: "NMAP FIN scan performed"; si
d: 10000006; rev: 1;)
```

Exhibit 12: NMAP FIN scan run on Machine A by Machine B, as indicated by the *-sF* option.



```
msfadmin@metasploitable:~$ sudo nmap -sF 10.0.2.13
```

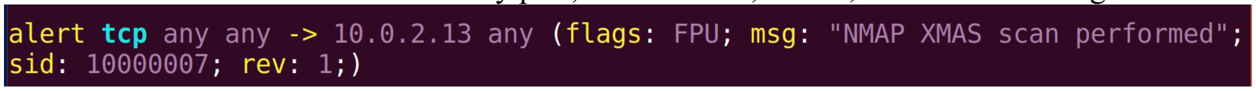
Exhibit 13: Resulting TCP traffic captured by Machine A, with corresponding alert message which was set in Exhibit 11.



```
02/03-22:24:45.261095  [**] [1:621:7] SCAN FIN [**] [Classification: Attempted I
nformation Leak] [Priority: 2] {TCP} 10.0.2.14:48501 -> 10.0.2.13:354
02/03-22:24:45.261107  [**] [1:10000006:1] NMAP FIN scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:48501 -> 10.0.2.13:291
02/03-22:24:45.261107  [**] [1:621:7] SCAN FIN [**] [Classification: Attempted I
nformation Leak] [Priority: 2] {TCP} 10.0.2.14:48501 -> 10.0.2.13:291
```

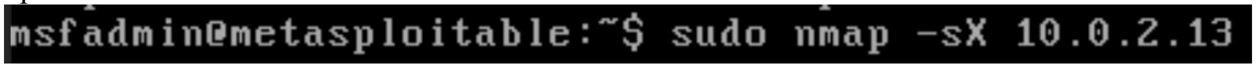
Step 3: Use Nmap to launch a XMAS scan attack, create a rule to detect the attack, test the rule and check the logs

Exhibit 14: Rules on Machine A to alert on any TCP traffic originating from any host and port with a destination of Machine A on any port, with the FIN, PUSH, and URGENT flags set.



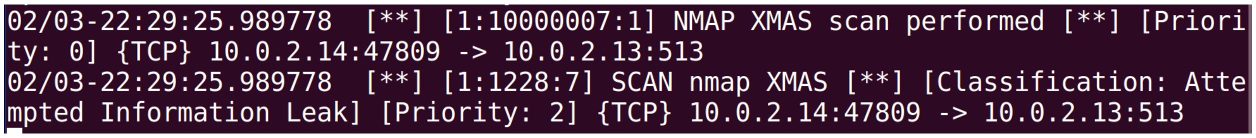
```
alert tcp any any -> 10.0.2.13 any (flags: FPU; msg: "NMAP XMAS scan performed";
sid: 10000007; rev: 1;)
```

Exhibit 15: NMAP XMAS scan performed on Machine A by Machine B, as indicated by the *-sX* option.



```
msfadmin@metasploitable:~$ sudo nmap -sX 10.0.2.13
```

Exhibit 16: Resulting TCP traffic captured by Machine A, with corresponding alert message which was set in Exhibit 14.

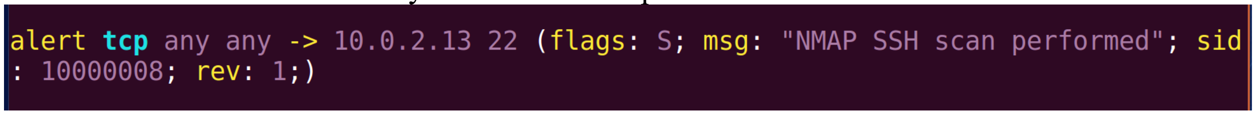


```
02/03-22:29:25.989778  [**] [1:10000007:1] NMAP XMAS scan performed [**] [Priori
ty: 0] {TCP} 10.0.2.14:47809 -> 10.0.2.13:513
02/03-22:29:25.989778  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {TCP} 10.0.2.14:47809 -> 10.0.2.13:513
```

Task 4: Write your own custom rule to detect ANY other attack on the victim machine. Demonstrate the attack and show how SNORT is capable of detecting this attack.

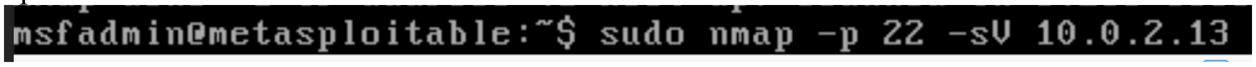
Other types of scans can also be detected by SNORT. This section explores scans that may be performed on SSH ports, specifically. This section will follow the same format as that of Task 3, where the SNORT alert configuration is provided, followed by the scan performed by Machine B, followed by the traffic captured by Machine A's SNORT rules.

Exhibit 17: SNORT Rule on Machine A to detect any scans for open SSH ports. This will detect ssh version scans that nmap may attempt to perform from any host, from any port, with the destination of Machine A on any of Machine A's ports.



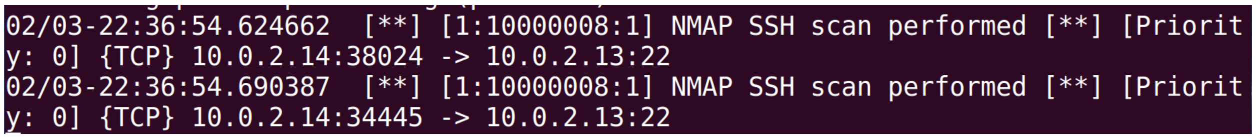
```
alert tcp any any -> 10.0.2.13 22 (flags: S; msg: "NMAP SSH scan performed"; sid
: 10000008; rev: 1;)
```

Exhibit 18: SSH version scan run by Machine B on Machine A. This is indicated by the *-sV* option in the command.



```
msfadmin@metasploitable:~$ sudo nmap -p 22 -sV 10.0.2.13
```

Exhibit 19: Resulting network traffic captured by Machine A, with corresponding alert message which was set in Exhibit 17.

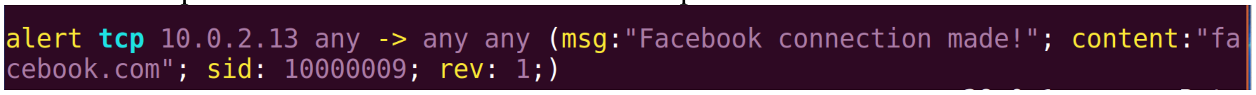


```
02/03-22:36:54.624662  [**] [1:10000008:1] NMAP SSH scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:38024 -> 10.0.2.13:22
02/03-22:36:54.690387  [**] [1:10000008:1] NMAP SSH scan performed [**] [Priorit
y: 0] {TCP} 10.0.2.14:34445 -> 10.0.2.13:22
```

Task 5: Write a rule that will fire when you browse to facebook.com from the machine SNORT is running on; it should look for any outbound TCP request to facebook.com and alert on it.

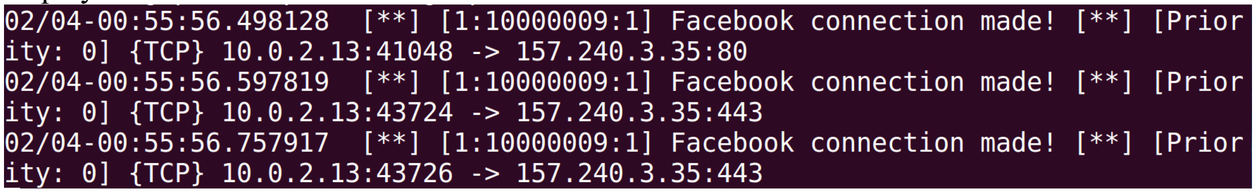
SNORT can also be configured to view the contents of packets as they are captured on the network. This is explored in the following section.

Exhibit 20: Creating an alert for TCP traffic originating from Machine A on any port to any destination IP and destination source port that contains the content "facebook.com". Essentially, this alert will produce a result whenever a user attempts to access facebook.com.



```
alert tcp 10.0.2.13 any -> any any (msg:"Facebook connection made!"; content:"fa
cebook.com"; sid: 10000009; rev: 1;)
```

Exhibit 21: Upon attempting to connect to Facebook via a browser on Machine A, an alert is displayed which matches the alert shown in Exhibit 20.



```
02/04-00:55:56.498128  [**] [1:10000009:1] Facebook connection made! [**] [Prior
ity: 0] {TCP} 10.0.2.13:41048 -> 157.240.3.35:80
02/04-00:55:56.597819  [**] [1:10000009:1] Facebook connection made! [**] [Prior
ity: 0] {TCP} 10.0.2.13:43724 -> 157.240.3.35:443
02/04-00:55:56.757917  [**] [1:10000009:1] Facebook connection made! [**] [Prior
ity: 0] {TCP} 10.0.2.13:43726 -> 157.240.3.35:443
```

Task 6: Answer the following questions

Q1: State how each of following real rules from the SNORT home page work:

i. alert icmp any any -> any any (msg:"ICMP Source Quench"; itype: 4; icode: 0;)

This SNORT rule generates an alert for any packet on the network which is an ICMP source quench message that has been redirected. This is indicated by the itype of 4, which specifies searching for ICMP source quench messages. Icode 0 indicates that it is a redirected ICMP packet.

ii. alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI view-source access";flags: A+; content:"/view source?../../../../../../../etc/passwd"; nocase;reference:cve,CVE-1999-0174;)

This SNORT rule generates an alert for any TCP packet originating from the external network on any port with the destination IP of the defined HTTP servers on the host machine with a destination port of port 80. The rule looks for packets where ACK flags *and* content attempting to access the /view source?../../../../../../../etc/passwd filepath, and ignores cases.

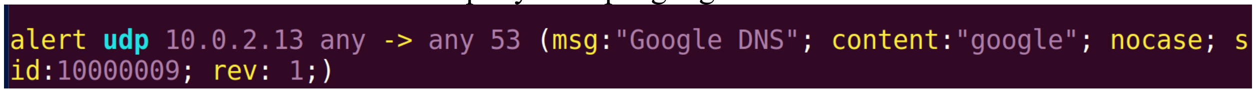
The reference keyword allows the user to use the commonly used CFE reference system to reference a common attack. This allows the user to provide additional information in the alert related to the specific attack, when such packets are detected. In this case, the packet is attempting to access the passwords stored in the /etc/password directory.

Q2: Develop your own SNORT signature to capture DNS queries directed against the host the you choose to connect to via HTTPS. Make sure that your SNORT rule references the DNS data and not simply IP address of the server.

SNORT rules can also analyze DNS queries. This is explored in the following section.

Firstly, a rule can be created to capture DNS queries to a specified URL, such as that of “google.com”.

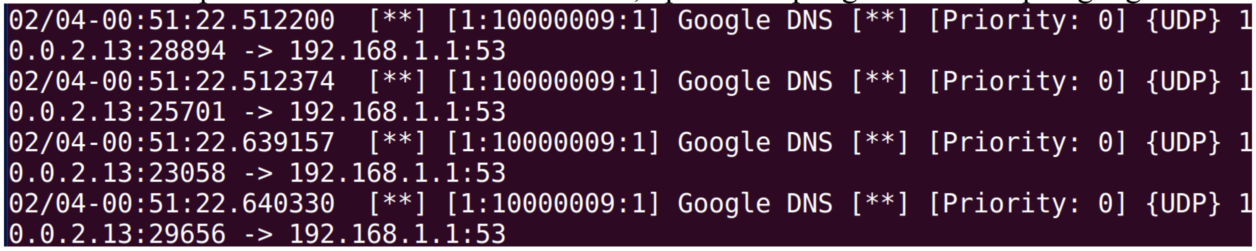
Exhibit 22: SNORT rule to capture UDP traffic originating from Machine A on any port to any destination IP address on port 53. Port 53 is the commonly used port for DNS queries. The rule also searched the given packets for any packets containing the keyword "google". Hence, the rule will alert when there is a DNS query to https//google.com.



```
alert udp 10.0.2.13 any -> any 53 (msg:"Google DNS"; content:"google"; nocase; s
id:10000009; rev: 1;)
```

Upon running SNORT, and attempting to access google.com, the following traffic is displayed:

Exhibit 23: Captured network traffic on SNORT, upon attempting to access https://google.com.



```
02/04-00:51:22.512200  [**] [1:10000009:1] Google DNS [**] [Priority: 0] {UDP} 1
0.0.2.13:28894 -> 192.168.1.1:53
02/04-00:51:22.512374  [**] [1:10000009:1] Google DNS [**] [Priority: 0] {UDP} 1
0.0.2.13:25701 -> 192.168.1.1:53
02/04-00:51:22.639157  [**] [1:10000009:1] Google DNS [**] [Priority: 0] {UDP} 1
0.0.2.13:23058 -> 192.168.1.1:53
02/04-00:51:22.640330  [**] [1:10000009:1] Google DNS [**] [Priority: 0] {UDP} 1
0.0.2.13:29656 -> 192.168.1.1:53
```