

Arzadon, Christian Andrei T.
BSCS 3-B

UBUNTU (DEFENDER): Start snort

```
ubuntu@Ubuntu: ~/Desktop
ubuntu@Ubuntu:~/Desktop$ sudo nano /etc/snort/rules/local.rules
[sudo] password for ubuntu:
ubuntu@Ubuntu:~/Desktop$ sudo snort -A console -i eth0 -c /etc/snort/snort.conf -K ascii
Running in IDS mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2843 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined : [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:355 ]
PortVar 'CTP_PORTS' defined : [ 2123 2152 3386 ]
```

KALI LINUX (ATTACKER): Kali Linux (Attacker) using nmap to scan Ubuntu(Defender)

```
(kali@kali)-[~]
$ nmap -sS 192.168.1.30
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 22:27 PST
Nmap scan report for 192.168.1.30
Host is up (0.0053s latency).
All 1000 scanned ports on 192.168.1.30 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: E4:C7:67:66:46:7B (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 1.40 seconds
```

UBUNTU (DEFENDER): Snort Detecting attack from Kali Linux (Attacker)

```
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_S7COMMPPLUS Version 1.0 <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_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=2975)
05/20-14:27:54.079024  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:39801 -> 192.168.1.30:705
05/20-14:27:54.299120  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:39801 -> 192.168.1.30:161
05/20-14:27:56.030416  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.1.8:64961 -> 239.255.255.250:1900
```

KALI LINUX (ATTACKER): Using nikto

```
(kali@kali)-[~] memory: 27.3281
$ nikto -h http://192.168.1.30
- Nikto v2.5.0
+ 0 host(s) tested | to passive.
```

KALI LINUX (ATTACKER): Using msfconsole

```
kali@kali: ~
File Actions Edit View Help

.00000000. 1; 0 ; 0 ,00000000.
c0000000. .00c. 'o00. ,0000000c
servo000000: co.0000. :0000:o-s,000000o li
l00000. .0000le_1:0000. 2,00000l 208
;0000' .0000:ota:0000. 2;0000; 208
.d00o .0000occcx0000. x00d.
fast patt,k0lgr.00000000000000. .d0k,
:kk;.00000000000000.c0k:
;k0000000000000000k:
,x000000000000x,
searchengine (ac.l0000000l.
inst,d0d,: 2
patte:ns: 416
pattern chars: 2508
=[ metasploit v6.4.56-dev ]
+ -- --[ 2505 exploits - 1291 auxiliary - 431 post ]
+ -- --[ 1610 payloads - 49 encoders - 13 nops ]
+ -- --[ 9 evasion memory: 68.5879 ]
pattern memory: 18.6973
Metasploit Documentation: https://docs.metasploit.com/
transition memory: 24.3125
msf6 > use auxiliary/scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.1.30
RHOSTS => 192.168.1.30
msf6 auxiliary(scanner/portscan/tcp) > run
[*] 192.168.1.30: process - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) >
```

UBUNTU (DEFENDER): Detecting metasploit attack using msfconsole from the Kali VM

```
ubuntu@Ubuntu: ~/Desktop
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_S7COMMPPLUS Version 1.0 <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_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=3029)
05/20-14:38:09.962115 [**] [1:1418:11] SNMP request tcp [**] [Classification: A
tttempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:44557 -> 192.168.1.3
0:161
05/20-14:38:09.963170 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {TCP} 192.168.1.27:42661 -> 192.168.1.30:1
62
05/20-14:38:12.250483 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classifica
tion: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.27:35631 -> 192.
168.1.30:705
```

- **The source and destination** : 192.168.1.27:44557 > 192.168.1.3:161
- **The attack type** : SNMP (Simple Network Management Protocol) request tcp
- **Classification and priority** : Attempted Information Leak

UBUNTU (DEFENDER): add rule to /etc/snort/rules/local.rules:

```

ubuntu@Ubuntu: ~/Desktop
GNU nano 7.2 /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 tcp any any -> any 23 (msg:"Telnet attempt detected"; sid:1000002; rev:1;)
```

KALI LINUX (ATTACKER): telnet

```

(kali㉿kali)-[~]
$ telnet 192.168.1.30 300
Trying 192.168.1.30 ...
```

UBUNTU (DEFENDER):

```

05/20-14:52:44.861482  [**] [1:1000002:1] Telnet attempt detected [**] [Priority: 0] {TCP} 192.168.1.27:34758 -> 192.168.1.30:23
05/20-14:52:57.433505  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.1.8:52606 -> 239.255.255.250:1900
05/20-14:52:58.456328  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.1.8:52606 -> 239.255.255.250:1900
05/20-14:52:59.480974  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.1.8:52606 -> 239.255.255.250:1900
05/20-14:53:00.402598  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.1.8:52606 -> 239.255.255.250:1900
```

- What types of attacks did Snort detect during your tests?**
During the tests, Snort detected attacks such as Nmap TCP scans and Metasploit TCP port scans, as simulated using Kali Linux tools.
- How does Snort classify and prioritize threats?**
Snort classifies threats based on the nature of the activity "Attempted Information Leak" for scans like Nmap.
- What happens when you trigger your custom rule?**
When you trigger the custom Telnet rule by attempting a telnet 192.168.1.30 connection, Snort generates an alert with the message "Telnet attempt detected" (as defined in the rule), provided the rule is correctly added to /etc/snort/rules/local.rules
- How can Snort be used in real-world networks as both IDS and IPS?**
IDS, Snort monitors network traffic in real-time, logging and alerting on suspicious activities without blocking them. For IPS, it can be configured in inline mode using NFQUEUE and iptables to actively block malicious traffic based on defined rules, enhancing network security by preventing attacks in addition to detecting them.