

Module 12: Evading IDS, Firewalls, and Honeypots

Lab 1: Perform Intrusion Detection using Various Tools

Lab Scenario

The goal of the Intrusion Detection Analyst is to find possible attacks against a network. Recent years have witnessed a significant increase in Distributed Denial-of-Service (DDoS) attacks on the Internet, making network security a great concern. Analysts search for possible attacks by examining IDS logs and packet captures and corroborating them with firewall logs, known vulnerabilities, and general trending data from the Internet. IDS attacks are becoming more sophisticated; automatically reasoning the attack scenarios in real-time, and categorizing them has become a critical challenge. These processes result in huge amounts of data, which analysts must examine to detect a pattern. However, the overwhelming flow of events generated by IDS sensors make it difficult for security administrators to uncover hidden attack plans.

To become an expert penetration tester and security administrator, you must possess sound knowledge of network IPSs, IDSs, malicious network activity, and log information.

Lab Objectives

- Detect intrusions using Snort
- Deploy Cowrie honeypot to detect malicious network traffic

Overview of Intrusion Detection Systems

Intrusion detection systems are highly useful as they monitor both the inbound and outbound traffic of the network and continuously inspects the data for suspicious activities that may indicate a network or system security breach. The IDS checks traffic for signatures that match known intrusion patterns and signals an alarm when a match is detected. It can be categorized into active and passive, depending on its functionality: an IDS is generally passive and is used to detect intrusions, while an intrusion prevention system (IPS) is considered as an active IDS, as it is not only used to detect the intrusion on the network, but also prevent them.

Main Functions of IDS:

- Gathers and analyzes information from within a computer or a network, to identify the possible violations of security policy
- Also referred to as a “packet-sniffer,” which intercepts packets traveling along various communication mediums and protocols
- Evaluates traffic for suspected intrusions and signals an alarm after detection

Task 1: Detect Intrusions using Snort

Snort is an open-source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content

searching/matching and is used to detect a variety of attacks and probes such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts. It uses a flexible rules language to describe traffic to collect or pass, as well as a detection engine that utilizes a modular plug-in architecture.

Uses of Snort:

- Straight packet sniffer such as tcpdump
- Packet logger (useful for network traffic debugging, etc.)
- Network intrusion prevention system

Here, we will use Snort to detect network intrusions.

1. Click on [Windows 11](#) to switch to **Windows 11** machine. Click [**Ctrl+Alt+Delete**](#) to activate the machine and login using **Admin/Pa\$\$w0rd**.

Alternatively, you can also click **Pa\$\$w0rd** under **Windows 11** machine thumbnail in the **Resources** pane.

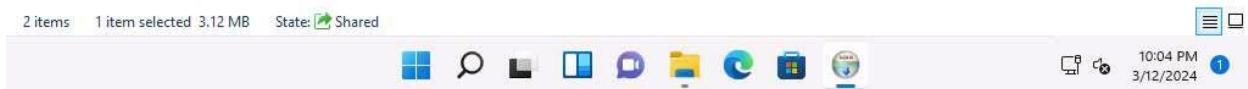
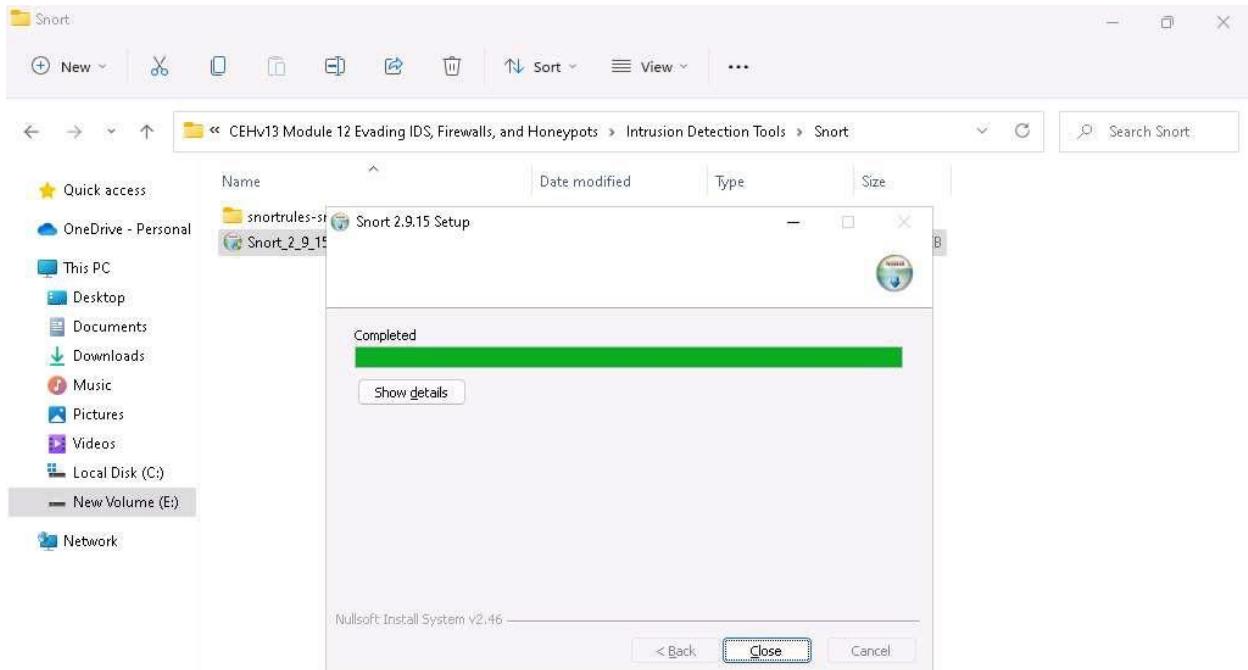
Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.

2. Navigate to **E:\CEH-Tools\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort** and double-click the **Snort_2_9_15_Installe.x64.exe** file to start the Snort installation.

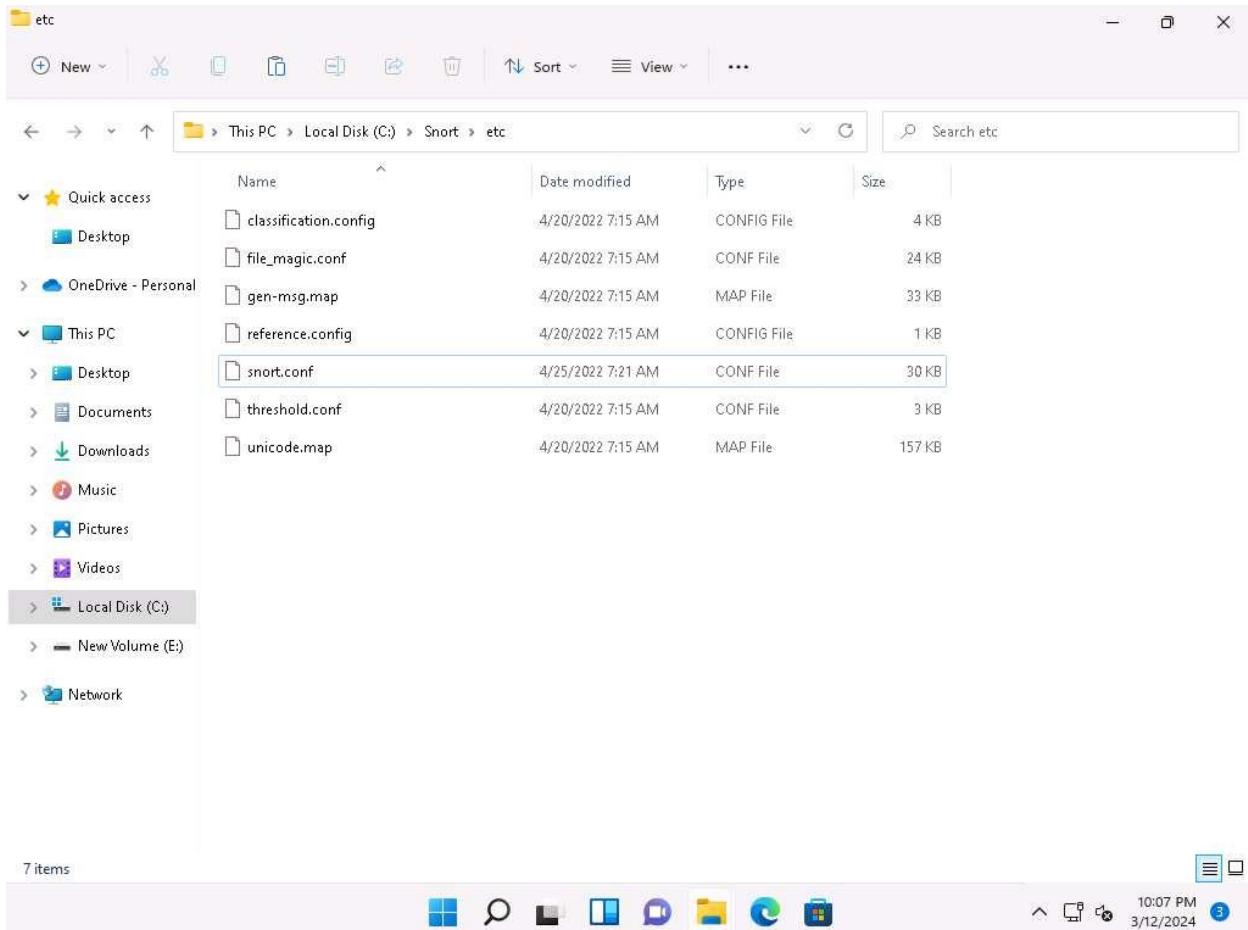
If an **Open File - Security warning** pop-up window appears, click **Yes**.

3. Accept the **License Agreement** and install Snort by selecting the default options that appear step by step in the wizard.
4. A window appears after the successful installation of Snort; click **Close**.
5. Click **OK** to exit the **Snort Installation** window.

Snort requires **WinPcap** to be installed on your machine. In this task environment, we have already installed WinPcap drivers for packet capturing.



6. By default, Snort installs itself in **C:\Snort** (C:\ or D:\, depending on the disk drive in which the OS is installed).
7. Navigate to the **etc** folder in the specified location, **E:\CEH-Tools\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort\snortrules-snapshot-29150\etc** of the Snort rules; copy **snort.conf** and paste it in **C:\Snort\etc**.
8. **snort.conf** is already present in **C:\Snort\etc**; replace the file with the newly copied file.



9. Copy the **so_rules** folder from **E:\CEH-Tools\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort\snortrules-snapshot-29150** and paste into **C:\Snort**.
 10. Copy the **preproc_rules** folder from **E:\CEH-Tools\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort\snortrules-snapshot-29150**, and paste it into **C:\Snort**. The **preproc_rules** folder is already present in **C:\Snort**; replace this folder with the **preproc_rules** folder taken from the specified location.
 11. Using the same method, copy the **rules** folder from **E:\CEH-Tools\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort\snortrules-snapshot-29150** and paste into **C:\Snort**.
 12. Now, right-click on the **Windows Start** icon and click **Run** from the menu.
 13. The **Run** window appears; type **cmd** in the **Open** field and click **OK** to launch command prompt window.
 14. The **Command Prompt** window appears; type **cd C:\Snort\bin** and press **Enter** to access the bin folder in the command prompt. Run **snort** command to initiate snort.

15. Snort initializes; wait for it to complete. Press **Ctrl+C** after some time, Snort exits and comes back to **C:\Snort\bin**.
 16. Now type **snort -W**. This command lists your machine's physical address, IP address, and Ethernet Drivers, but all are disabled by default.

```
cmd Select Command Prompt
GRE IP4:          0 ( 0.000%)
GRE IP6:          0 ( 0.000%)
GRE IP6 Ext:       0 ( 0.000%)
GRE PPTP:          0 ( 0.000%)
GRE ARP:          0 ( 0.000%)
GRE IPX:          0 ( 0.000%)
GRE Loop:          0 ( 0.000%)
MPLS:              0 ( 0.000%)
ARP:               0 ( 0.000%)
IPX:               0 ( 0.000%)
Eth Loop:          0 ( 0.000%)
Eth Disc:          0 ( 0.000%)
IP4 Disc:          0 ( 0.000%)
IP6 Disc:          0 ( 0.000%)
TCP Disc:          0 ( 0.000%)
UDP Disc:          0 ( 0.000%)
ICMP Disc:         0 ( 0.000%)
All Discard:        0 ( 0.000%)
    Other:          0 ( 0.000%)
Bad Chk Sum:        60 ( 50.420%)
    Bad TTL:         0 ( 0.000%)
    S5 G 1:          0 ( 0.000%)
    S5 G 2:          0 ( 0.000%)
Total:             119
=====
Snort exiting
C:\Snort\bin>snort -W
--> Snort! <--
Version 2.9.15-WIN32 GRE (Build 7)
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 PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3
Index Physical Address      IP Address      Device Name      Description
---- 00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:709f:40d1 \Device\NPF_{5A9B3588-F693-4023-B9B6-DCC29ADB1114}
Microsoft Corporation
C:\Snort\bin>
```

17. Observe your Ethernet Driver **index number** and write it down (in this task, it is **1**).
18. To enable the Ethernet Driver, in the command prompt, run command **snort -dev -i 1**.
19. You see a rapid scroll text in the command prompt, which means that the Ethernet Driver is enabled and working properly.

```
cmd Select Command Prompt
1 00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:709f:40d1 \Device\NPF_{5A9B3588-F693-4023-B9B6-DCC29ADB1114}
Microsoft Corporation

C:\Snort\bin>snort -dev -i 1
Running in packet dump mode

--- Initializing Snort ---
Initializing Output Plugins!
pcap DAO configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{5A9B3588-F693-4023-B9B6-DCC29ADB1114}".
Decoding Ethernet

--- Initialization Complete ---

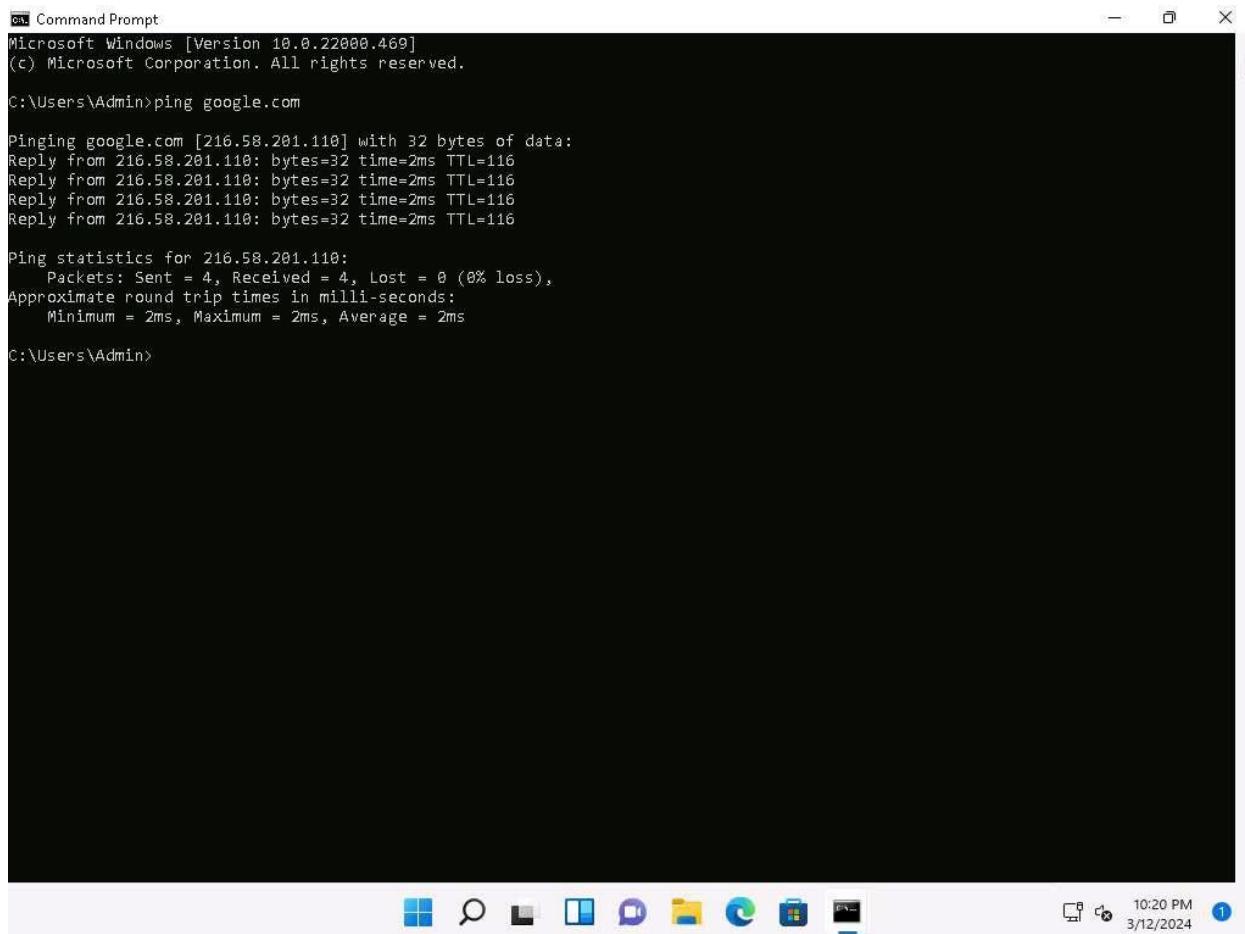
o"~  -*> Snort! <*-
.... Version 2.9.15-WIN32 GRE (Build 7)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3

Commencing packet processing (pid=7164)
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
05/30-23:02:04.569799 00:15:5D:01:80:00 -> 02:15:5D:02:7F:32 type:0x800 len:0x99
10.18.1.11:49870 -> 20.90.153.243:443 TCP TTL:128 TOS:0x0 ID:49458 IpLen:20 DgmLen:139 DF
***AP*** Seq: 0x478EF33E Ack: 0xFF48A026 Win: 0x400 TcpLen: 20
17 03 03 00 5E 00 00 00 00 00 00 08 BE CD 16 ....^..... .
B6 99 54 CD C5 E0 CE EA 1C 52 BF AD 5A D2 E1 C4 ..T.....R..Z...
FD 2C B8 5A 16 30 C9 A7 5C 0F 44 27 93 5F 1F 5C .,.Z.0..\\D'...\
1B AB C9 73 5F 53 BE DA E5 77 EC F8 22 FC 0B 0E ...S_S...W.."."
00 A6 EC AE 97 F1 2D E8 8F 0B 0B 02 CE 80 0D 40 .....-.....@...
14 5A 16 AF EC 7C AD 2F B3 5B 94 17 A6 25 17 D6 .Z...|/.|[...%.. .
FF 66 DF .f.

=====+
WARNING: No preprocessors configured for policy 0.
05/30-23:02:04.573541 02:15:5D:02:7F:32 -> 00:15:5D:01:80:00 type:0x800 len:0xDF
20.90.153.243:443 -> 10.10.1.11:49870 TCP TTL:117 TOS:0x0 ID:61924 IpLen:20 DgmLen:209 DF
***AP*** Seq: 0xFF48A026 Ack: 0x478EF3A1 Win: 0x1CA2 TcpLen: 20
```

20. Leave the Snort command prompt window open, and launch another command prompt window.

21. In a new command prompt, run **ping google.com** command.



```
Command Prompt
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>ping google.com

Pinging google.com [216.58.201.110] with 32 bytes of data:
Reply from 216.58.201.110: bytes=32 time=2ms TTL=116

Ping statistics for 216.58.201.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

C:\Users\Admin>
```

22. This ping command triggers a Snort alert in the Snort command prompt with rapid scrolling text.

The Google IP address will differ when you perform this task.

```
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
07/23/08:24:03.122378 02:15:5D:05:58:7D -> 33:33:00:00:00:FB type:0x86DD len:0x173
fe80:0000:0000:0000:0015:5dff:fe05:587d:5353 -> ff02:0000:0000:0000:0000:0000:00fb:5353 UDP TTL:255 TOS:0x0 ID:0 IpLen:4
0 DgmLen:357
Len: 309
00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 10 61 64 62 .....adb
20 75 0E 09 64 65 6E 74 69 66 69 05 64 04 5F 61 -unidentified._a
64 62 04 5F 74 63 70 05 6C 6F 63 61 6C 00 00 10 db._tcp.local...
00 01 00 00 11 94 00 01 00 09 5F 73 65 72 76 69 ....._servi
63 65 73 07 5F 64 6E 73 2D 73 64 04 5F 75 64 78 ces._dns-sd._udp
C0 27 00 0C 00 01 00 00 11 94 00 02 C0 1D C0 1D .'.
00 0C 00 01 00 00 11 94 00 02 C0 0C C0 0C 00 21 .....!
00 01 00 00 00 78 00 10 00 00 00 00 15 B3 87 41 .....x.....A
6E 04 72 6F 69 64 C0 27 01 44 01 37 01 38 01 35 ndroid.'D.7.8.5
01 35 01 01 30 01 45 01 46 01 46 01 46 01 44 01 35 .5.0.E.F.F.D.5
01 35 01 31 01 30 01 38 01 30 01 30 01 30 01 30 01 30 .5.1.0.0.0.0.0.0
01 30 01 30 01 30 01 38 01 30 01 30 01 30 01 30 01 30 .0.0.0.0.0.0.0.0
01 38 01 38 01 45 01 46 03 69 70 36 04 61 72 79 .0.8.E.F.ip6.arp
61 00 00 0C 80 01 00 00 00 78 00 02 C0 7E C0 7E a.....x..~.~
00 1C 80 01 00 00 00 78 00 10 FE 80 00 00 00 00 .....x.....
00 00 00 15 5D FF FE 05 58 7D C0 0C 0C 00 2F 80 01 ...]...X).../..
00 00 11 94 00 09 C0 0C 00 05 80 00 00 80 00 40 C0 .....@.
88 00 2F 80 81 00 00 00 78 00 06 00 88 00 02 00 ./...x.....
88 C0 7E 00 2F 80 01 00 00 00 78 00 08 C0 7E 00 ..~/...x..~.
84 00 00 00 88 .....+
```

=====
07/23/08:24:08.820629 00:15:5D:01:80:00 -> 02:15:5D:05:58:78 type:0x800 len:0x4A
10.10.1.11 -> 216.58.212.206 ICMP TTL:128 TOS:0x0 ID:6975 IpLen:20 DgmLen:60
Type:8 Code:0 ID:1 Seq:5 ECHO
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 abcdefghijklmnop
71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69 qrstuvwxyzabcdghi

=====
WARNING: No preprocessors configured for policy 0.
07/23/08:24:08.822969 02:15:5D:05:58:78 -> 00:15:5D:01:80:00 type:0x800 len:0x4A
216.58.212.206 -> 10.10.1.11 ICMP TTL:117 TOS:0x0 ID:0 IpLen:20 DgmLen:60
Type:0 Code:0 ID:1 Seq:5 ECHO REPLY
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 abcdefghijklmnop

23. Close both command prompt windows. The verification of Snort installation and the triggering alert is complete, and Snort is working correctly in verbose mode.
 24. Configure the **snort.conf** file, located at **C:\Snort\etc**.
 25. Open the **snort.conf** file with **Notepad++**.
 26. Scroll down to the **Step #1: Set the network variables** section (Line 41) of the **snort.conf** file. In the **HOME_NET** line (Line 45), replace **any** with the IP addresses of the machine (target machine) on which Snort is running. Here, the target machine is **Windows 11** and the IP address is **10.10.1.11**.
 27. Leave the **EXTERNAL_NET any** line as it is.
 28. If you have a **DNS Server**, then make changes in the **DNS_SERVERS** line by replacing **\$HOME_NET** with your DNS Server IP address; otherwise, leave this line as it is.

Here, the DNS server is **8.8.8.8**.

```
*C:\Snort\etc\snort.conf - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hosts fuzz.py findoff.py ovenwrite.py badchars.py jump.py shellcode.py snort.conf

34 # 6) Configure output plugins
35 # 7) Customize your rule set
36 # 8) Customize preprocessor and decoder rule set
37 # 9) Customize shared object rule set
38 #####
39 #####
40 ##### Step #1: Set the network variables. For more information, see README.variables
41 #####
42 #####
43 #
44 # Setup the network addresses you are protecting
45 ipvar HOME_NET 10.10.1.11
46 #
47 # Set up the external network addresses. Leave as "any" in most situations
48 ipvar EXTERNAL_NET any
49 #
50 # List of DNS servers on your network
51 ipvar DNS_SERVERS 8.8.8.8
52 #
53 # List of SMTP servers on your network
54 ipvar SMTP_SERVERS $HOME_NET
55 #
56 # List of web servers on your network
57 ipvar HTTP_SERVERS $HOME_NET
58 #
59 # List of sql servers on your network
60 ipvar SQL_SERVERS $HOME_NET
61 #
62 # List of telnet servers on your network
63 ipvar TELNET_SERVERS $HOME_NET
64 #
65 # List of ssh servers on your network
66 ipvar SSH_SERVERS $HOME_NET
67 #
68 # List of ftp servers on your network
69 ipvar FTP_SERVERS $HOME_NET
```

Normal text file length : 30,254 lines : 713 Ln:51 Col:26 Pos:2,028 Unix (LF) UTF-8 INS

10:22 PM 3/12/2024 ④

29. The same applies to SMTP_SERVERS, HTTP_SERVERS, SQL_SERVERS, TELNET_SERVERS, and SSH_SERVERS.
30. Remember that if you do not have any servers running on your machine, leave the line as it is. **DO NOT** make any changes in that line.
31. Scroll down to RULE_PATH (Line 104). In Line 104, replace ..//rules with C:\Snort\rules in Line 105, replace ..//so_rules with C:\Snort\so_rules and in Line 106, replace ..//preproc_rules with C:\Snort\preproc_rules.

C:\Snort\etc\snort.conf - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

hosts fuzz.py finhoff.py overwritte.py badchars.py jump.py shellcode.py snort.conf

```
81 portvar ORACLE_PORTS 1024;
82
83 # List of ports you want to look for SSH connections on:
84 portvar SSH_PORTS 22
85
86 # List of ports you run ftp servers on
87 portvar FTP_PORTS [21,2100,3535]
88
89 # List of ports you run SIP servers on
90 portvar SIP_PORTS [5060,5061,5600]
91
92 # List of file data ports for file inspection
93 portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]
94
95 # List of GTP ports for GTP preprocessor
96 portvar GTP_PORTS [2123,2152,3386]
97
98 # other variables, these should not be modified
99 ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24]
100
101 # Path to your rules files (this can be a relative path)
102 # Note for Windows users: You are advised to make this an absolute path,
103 # such as: c:\Snort\rules
104 var RULE_PATH C:\Snort\rules
105 var SO_RULE_PATH C:\Snort\so_rules
106 var PREPROC_RULE_PATH C:\Snort\preproc_rules
107
108 # If you are using reputation preprocessor set these
109 var WHITE_LIST_PATH ../rules
110 var BLACK_LIST_PATH ../rules
111
112 #####
113 # Step #2: Configure the decoder. For more information, see README.decode
114 #####
115
116 # Stop generic decode events:
```

32. In Lines 109 and 110, replace `./rules` with `C:\Snort\rules`. Minimize the **Notepad++** window.

*C:\Snort\etc\snort.conf - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

hosts fuzz.py findoff.py oopenwrite.py badchars.py jump.py shellcode.py snort.conf

```
81 portvar ORACLE_PORTS 1024;
82
83 # List of ports you want to look for SSH connections on:
84 portvar SSH_PORTS 22
85
86 # List of ports you run ftp servers on
87 portvar FTP_PORTS [21,2100,3535]
88
89 # List of ports you run SIP servers on
90 portvar SIP_PORTS [5060,5061,5600]
91
92 # List of file data ports for file inspection
93 portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]
94
95 # List of GTP ports for GTP preprocessor
96 portvar GTP_PORTS [2123,2152,3386]
97
98 # other variables, these should not be modified
99 ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24]
100
101 # Path to your rules files (this can be a relative path)
102 # Note for Windows users: You are advised to make this an absolute path,
103 # such as: c:\snort\rules
104 var RULE_PATH C:\Snort\rules
105 var SO_RULE_PATH C:\Snort\so_rules
106 var PREPROC_RULE_PATH C:\Snort\preproc_rules
107
108 # If you are using reputation preprocessor set these
109 var WHITE_LIST_PATH C:\Snort\rules
110 var BLACK_LIST_PATH C:\Snort\rules
111
112 ######
113 # Step #2: Configure the decoder. For more information, see README.decode
114 #####
115
116 # Stop generic decode events:
```

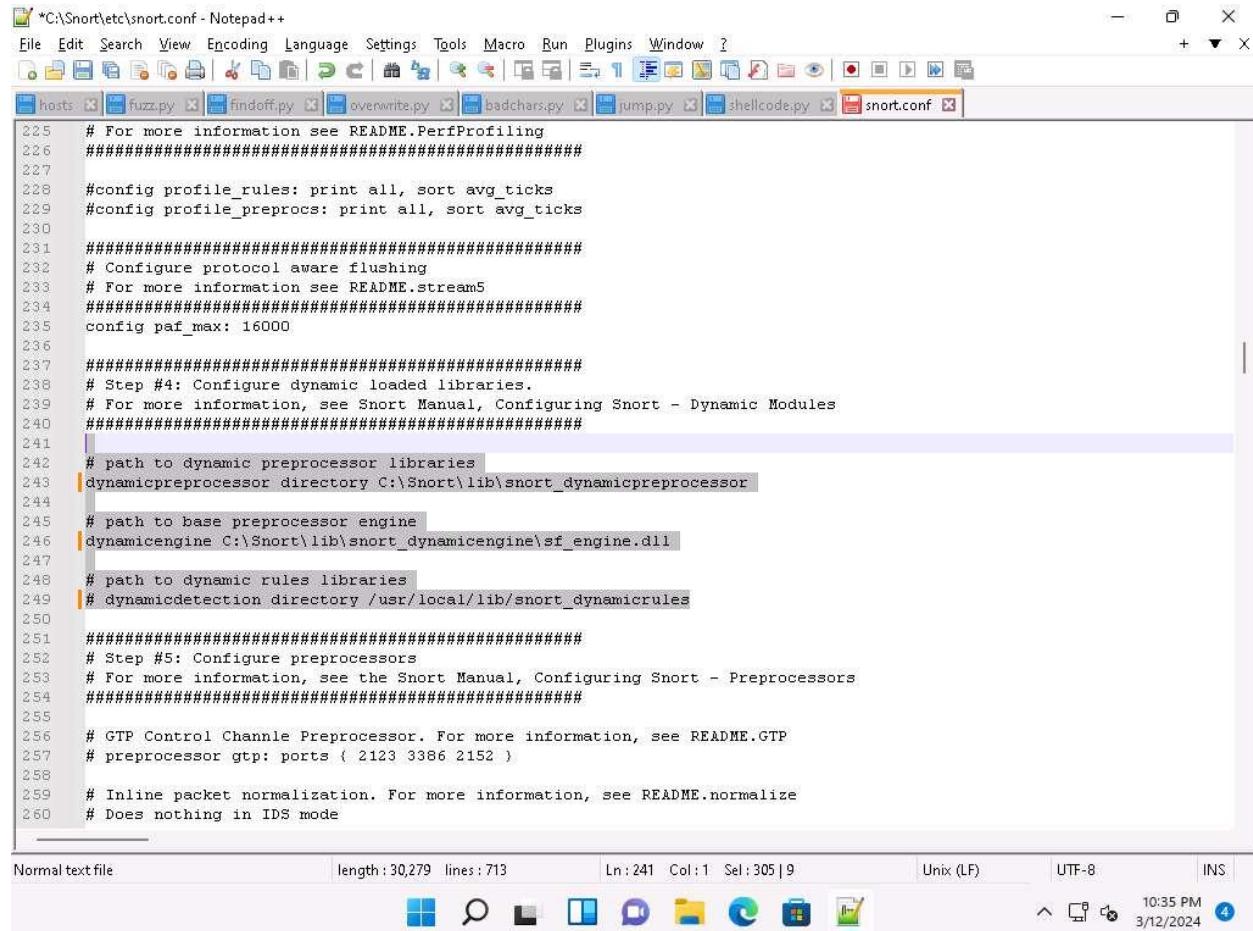
33. Navigate to **C:\Snort\rules**, and create two text files; name them **white_list** and **black_list** and change their file extensions from **.txt** to **.rules**.

To create a text file, right-click anywhere inside the rules window and navigate to **New --> Text Document**.

34. While changing the extension, if any pop-up appears, click **Yes**.
 35. Switch back to **Notepad++**, scroll down to the **Step #4: Configure dynamic loaded libraries** section (Line 238). **Configure dynamic loaded libraries** in this section.
 36. Add the path to dynamic preprocessor libraries (Line 243);
replace **/usr/local/lib/snort_dynamicpreprocessor/** with your dynamic preprocessor libraries folder location.
 37. In this task, the dynamic preprocessor libraries are located
at **C:\Snort\lib\snort_dynamicpreprocessor**.
 38. At the path to base preprocessor (or dynamic) engine (Line 246),
replace **/usr/local/lib/snort_dynamicengine/libsf_engine.so** with your base preprocessor engine **C:\Snort\lib\snort_dynamicengine\sf_engine.dll**.

39. Ensure that the dynamic rules libraries (Line 249) is commented out, as you have already configured the libraries in dynamic preprocessor libraries.

Add (**space**) in between # and dynamicdetection (Line 249).



```
*C:\Snort\etc\snort.conf - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hosts fuzz.py findoff.py overwritex.py badchars.py jump.py shellcode.py snort.conf

225 # For more information see README.PerfProfiling
226 ######
227
228 #config profile_rules: print all, sort avg_ticks
229 #config profile_procs: print all, sort avg_ticks
230
231 ######
232 # Configure protocol aware flushing
233 # For more information see README.stream5
234 ######
235 config paf_max: 16000
236
237 ######
238 # Step #4: Configure dynamic loaded libraries.
239 # For more information, see Snort Manual, Configuring Snort - Dynamic Modules
240 ######
241
242 # path to dynamic preprocessor libraries
243 dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor
244
245 # path to base preprocessor engine
246 dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll
247
248 # path to dynamic rules libraries
249 # dynamicdetection directory /usr/local/lib/snort_dynamicrules
250
251 ######
252 # Step #5: Configure preprocessors
253 # For more information, see the Snort Manual, Configuring Snort - Preprocessors
254 ######
255
256 # GTP Control Channel Preprocessor. For more information, see README.GTP
257 # preprocessor gtp: ports { 2123 3386 2152 }
258
259 # Inline packet normalization. For more information, see README.normalize
260 # Does nothing in IDS mode
```

40. Scroll down to the **Step #5: Configure preprocessors** section (Line 253), the listed preprocessor. This does nothing in IDS mode, however, it generates errors at runtime.

41. Comment out all the preprocessors listed in this section by adding '#' and (**space**) before each preprocessor rule (261-265).

To 'comment out' is to render a block of code inert by turning it into a comment.

*C:\Snort\etc\snort.conf - Notepad++

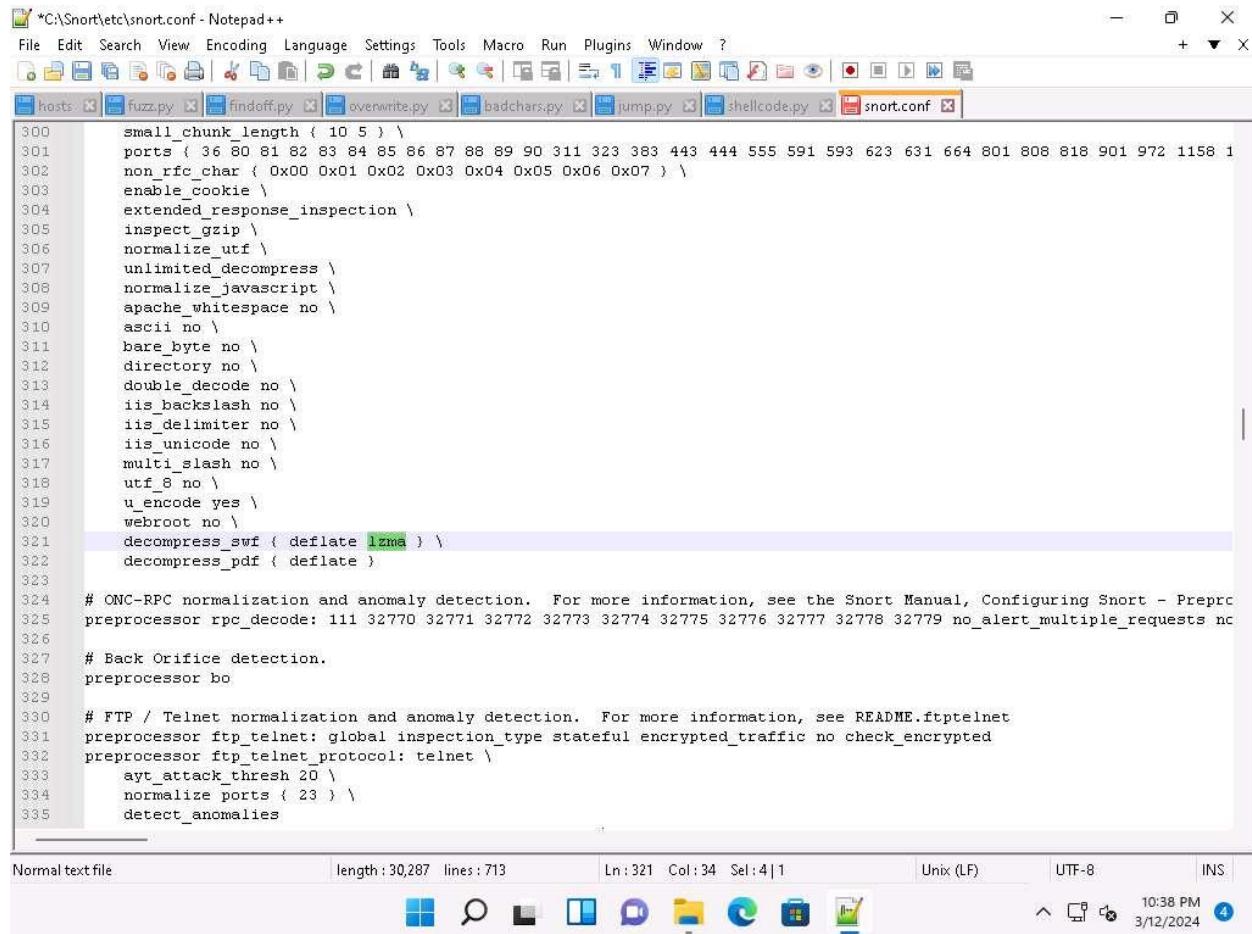
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

hosts fuzz.py findoff.py ovenwrite.py badchars.py jump.py shellcode.py snort.conf

```
243 dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor
244
245 # path to base preprocessor engine
246 dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll
247
248 # path to dynamic rules libraries
249 # dynamicdetection directory /usr/local/lib/snort_dynamicrules
250
251 ######
252 # Step #5: Configure preprocessors
253 # For more information, see the Snort Manual, Configuring Snort - Preprocessors
254 #####
255
256 # GTP Control Channel Preprocessor. For more information, see README.GTP
257 # preprocessor gtp: ports { 2123 3386 2152 }
258
259 # Inline packet normalization. For more information, see README.normalize
260 # Does nothing in IDS mode
261 # preprocessor normalize_ip4
262 # preprocessor normalize_tcp: block, rsv, pad, urp, req_urg, req_pay, req_urp, ips, ecn stream
263 # preprocessor normalize_icmp4
264 # preprocessor normalize_ip6
265 # preprocessor normalize_icmp6
266
267 # Target-based IP defragmentation. For more information, see README.frag3
268 preprocessor frag3_global: max_frgs 65536
269 preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180
270
271 # Target-Based stateful inspection/stream reassembly. For more information, see README.stream5
272 preprocessor stream5_global: track_tcp yes,
273     track_udp yes,
274     track_icmp no,
275     max_tcp 262144,
276     max_udp 131072,
277     max_active_responses 2,
278     min_response_seconds 5
```

42. Scroll down to line 321 and delete **Izma** keyword and a (space).

Make sure you only delete "Izma" keyword.



```
300     small_chunk_length ( 10 5 ) \
301     ports { 36 80 81 82 83 84 85 86 87 88 89 90 311 323 383 443 444 555 591 593 623 631 664 801 808 818 901 972 1158 1
302     non_rfc_char ( Ox00 Ox01 Ox02 Ox03 Ox04 Ox05 Ox06 Ox07 ) \
303     enable_cookie \
304     extended_response_inspection \
305     inspect_gzip \
306     normalize_utf \
307     unlimited_decompress \
308     normalize_javascript \
309     apache_whitespace no \
310     ascii no \
311     bare_byte no \
312     directory no \
313     double_decode no \
314     iis_backslash no \
315     iis_delimiter no \
316     iis_unicode no \
317     multi_slash no \
318     utf_8 no \
319     u_encode yes \
320     webroot no \
321     decompress_swf { deflate lzma } \
322     decompress_pdf { deflate } \
323
# ONC-RPC normalization and anomaly detection. For more information, see the Snort Manual, Configuring Snort - Preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no_alert_multiple_requests nc
325
326
327 # Back Orifice detection.
328 preprocessor bo
329
# FTP / Telnet normalization and anomaly detection. For more information, see README.ftptelnet
330 preprocessor ftp_telnet: global inspection_type stateful encrypted_traffic no check_encrypted
331 preprocessor ftp_telnet_protocol: telnet \
332     ayt_attack_thresh 20 \
333     normalize_ports ( 23 ) \
334     detect_anomalies
```

43. Scroll down to **Step #6: Configure output plugins** (Line 513). In this step, provide the location of the **classification.config** and **reference.config** files.
44. These two files are in **C:\Snort\etc**. Provide this location of files in the configure output plugins (in Lines 527 and 528)
(i.e., **C:\Snort\etc\classification.config** and **C:\Snort\etc\reference.config**).

The screenshot shows a Notepad++ window with the file 'snort.conf' open. The code is a configuration file for Snort, a network intrusion detection system. It includes sections for unified2 logging, syslog output, and pcap output. It also includes metadata reference data and a note about rule sets. The file ends with a series of include statements for various rule files. The Notepad++ interface includes a toolbar at the top, a tab bar with multiple files, status bars at the bottom, and a system tray icon.

```
510 #####  
511 # unified2  
512 # Recommended for most installs  
513 # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types  
514  
515 # Additional configuration for specific types of installs  
516 # output alert_unified2: filename snort.alert, limit 128, nostamp  
517 # output log_unified2: filename snort.log, limit 128, nostamp  
518  
519 # syslog  
520 # output alert_syslog: LOG_AUTH LOG_ALERT  
521  
522 # pcap  
523 # output log_tcpdump: tcpdump.log  
524  
525 # metadata reference data. do not modify these lines  
526 include C:\Snort\etc\classification.config  
527 include C:\Snort\etc\reference.config  
528  
529  
530 #####  
531 # Step #7: Customize your rule set  
532 # For more information, see Snort Manual, Writing Snort Rules  
533 #  
534 # NOTE: All categories are enabled in this conf file  
535 #####  
536  
537 # site specific rules  
538 include ${RULE_PATH}/local.rules  
539  
540 include ${RULE_PATH}/app-detect.rules  
541 include ${RULE_PATH}/attack-responses.rules  
542 include ${RULE_PATH}/backdoor.rules  
543 include ${RULE_PATH}/bad-traffic.rules  
544 include ${RULE_PATH}/blacklist.rules
```

45. In **Step #6**, add to line (529) **output alert_fast: alerts.ids:** this command orders Snort to dump all logs into the **alerts.ids** file.

The screenshot shows a Notepad++ window with the file 'snort.conf' open. The code in the file is as follows:

```
513 # Recommended for most installs
514 # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types
515
516 # Additional configuration for specific types of installs
517 # output alert_unified2: filename snort.alert, limit 128, nostamp
518 # output log_unified2: filename snort.log, limit 128, nostamp
519
520 # syslog
521 # output alert_syslog: LOG_AUTH LOG_ALERT
522
523 # pcap
524 # output log_tcpdump: tcpdump.log
525
526 # metadata reference data. do not modify these lines
527 include C:\Snort\etc\classification.config
528 include C:\Snort\etc\reference.config
529 output alert_fast: alerts.ids:
530
531 #####
532 # Step #7: Customize your rule set
533 # For more information, see Snort Manual, Writing Snort Rules
534 #
535 # NOTE: All categories are enabled in this conf file
536 #####
537
538
539 # site specific rules
540 include $RULE_PATH/local.rules
541
542 include $RULE_PATH/app-detect.rules
543 include $RULE_PATH/attack-responses.rules
544 include $RULE_PATH/backdoor.rules
545 include $RULE_PATH/bad-traffic.rules
546 include $RULE_PATH/blacklist.rules
547 include $RULE_PATH/botnet-cnc.rules
548 include $RULE_PATH/browser-chrome.rules
```

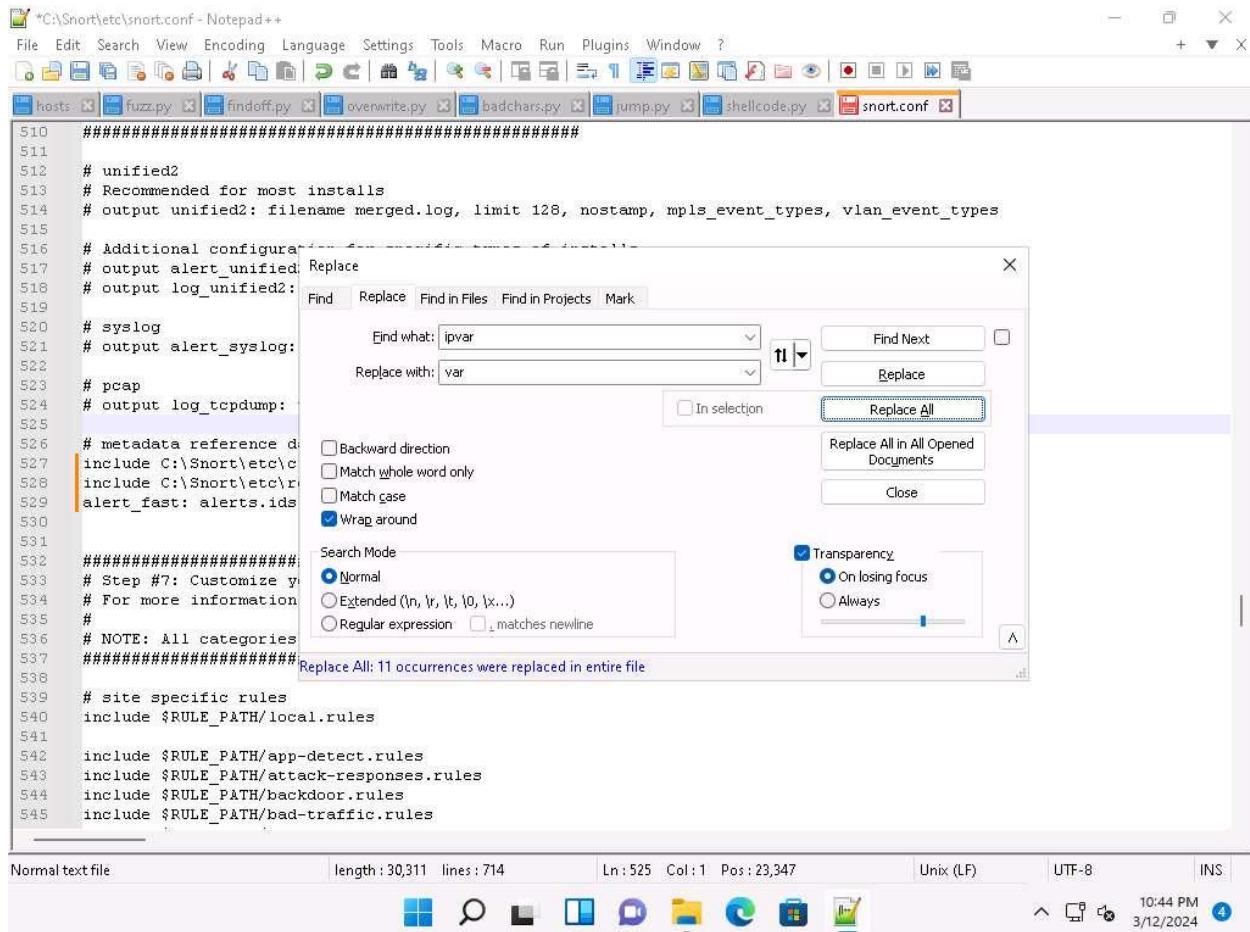
The status bar at the bottom shows: Normal text file, length: 30,318 lines: 714, Ln: 529 Col: 1 Sel: 31|1, Unix (LF), UTF-8, INS. The bottom right corner shows 10:52 PM 3/12/2024 4.

46. In the **snort.conf** file, find and replace the **ipvar** string with **var**. To do this, press **Ctrl+H** on the keyboard. The **Replace** window appears; enter **ipvar** in the **Find what** : text field, enter **var** in the **Replace with** : text field, and click **Replace All**.

You will get a notification saying 11 occurrences were replaced.

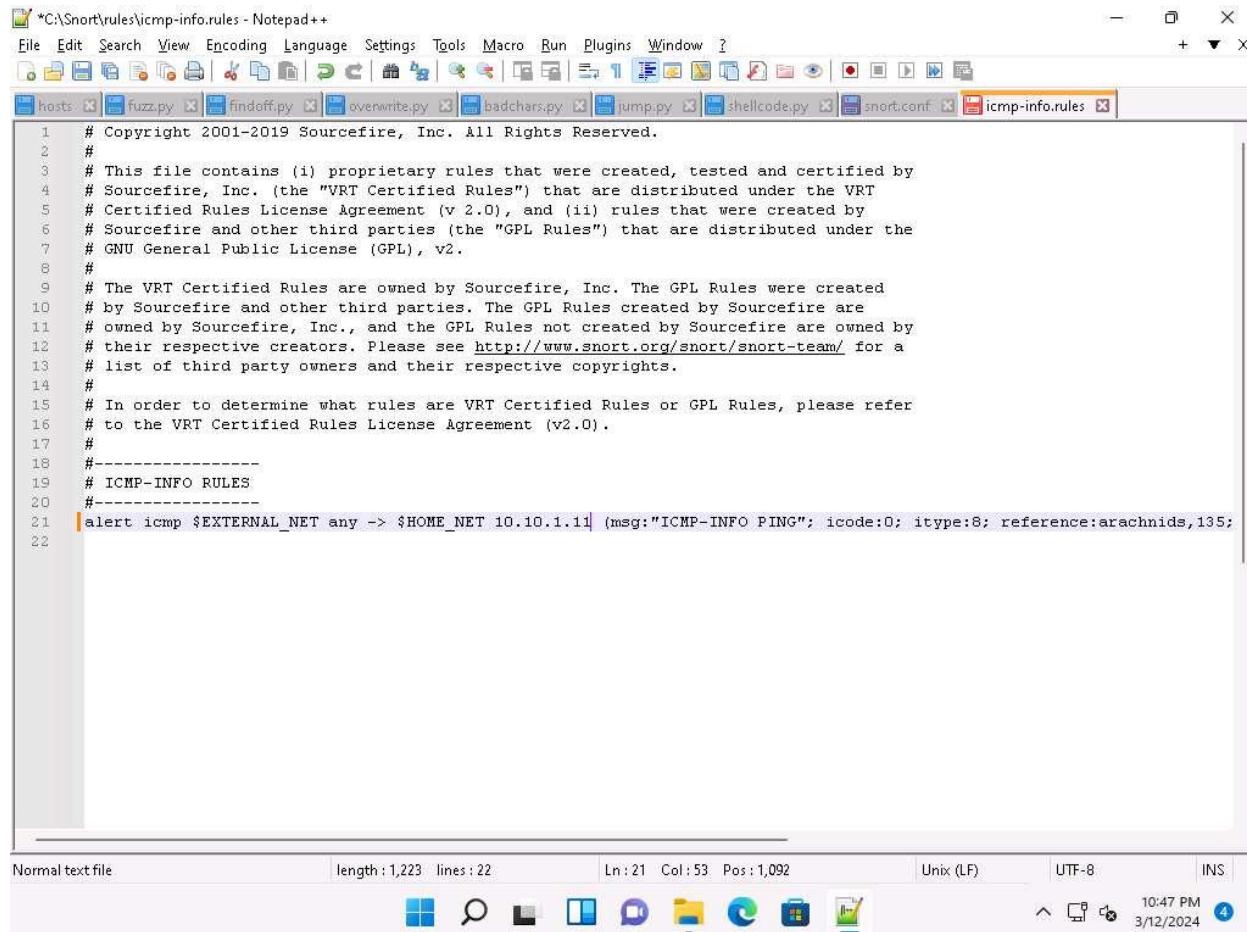
47. By default, the string is **ipvar**, which is not recognized by Snort: replace with the **var** string, and then **close** the window.

Snort now supports multiple configurations based on VLAN Id or IP subnet within a single instance of Snort. This allows administrators to specify multiple snort configuration files and bind each configuration to one or more VLANs or subnets rather than running one Snort for each configuration required.



48. Click **Close** to close the **Replace** window.
49. Save the **snort.conf** file by pressing **Ctrl+S** and close Notepad++ window.
50. Before running Snort, you need to enable detection rules in the Snort rules file. For this task, we have enabled the ICMP rule so that Snort can detect any host discovery ping probes directed at the system running Snort.
51. Navigate to **C:\Snort\rules** and open the **icmp-info.rules** file with **Notepad++**.
52. In line 21, type **alert icmp \$EXTERNAL_NET any -> \$HOME_NET 10.10.1.11 (msg:"ICMP-INFO PING"; icode:0; itype:8; reference:arachnids,135; reference:cve,1999-0265; classtype:bad-unknown; sid:472; rev:7;)** and save. Close the **Notepad++** window.

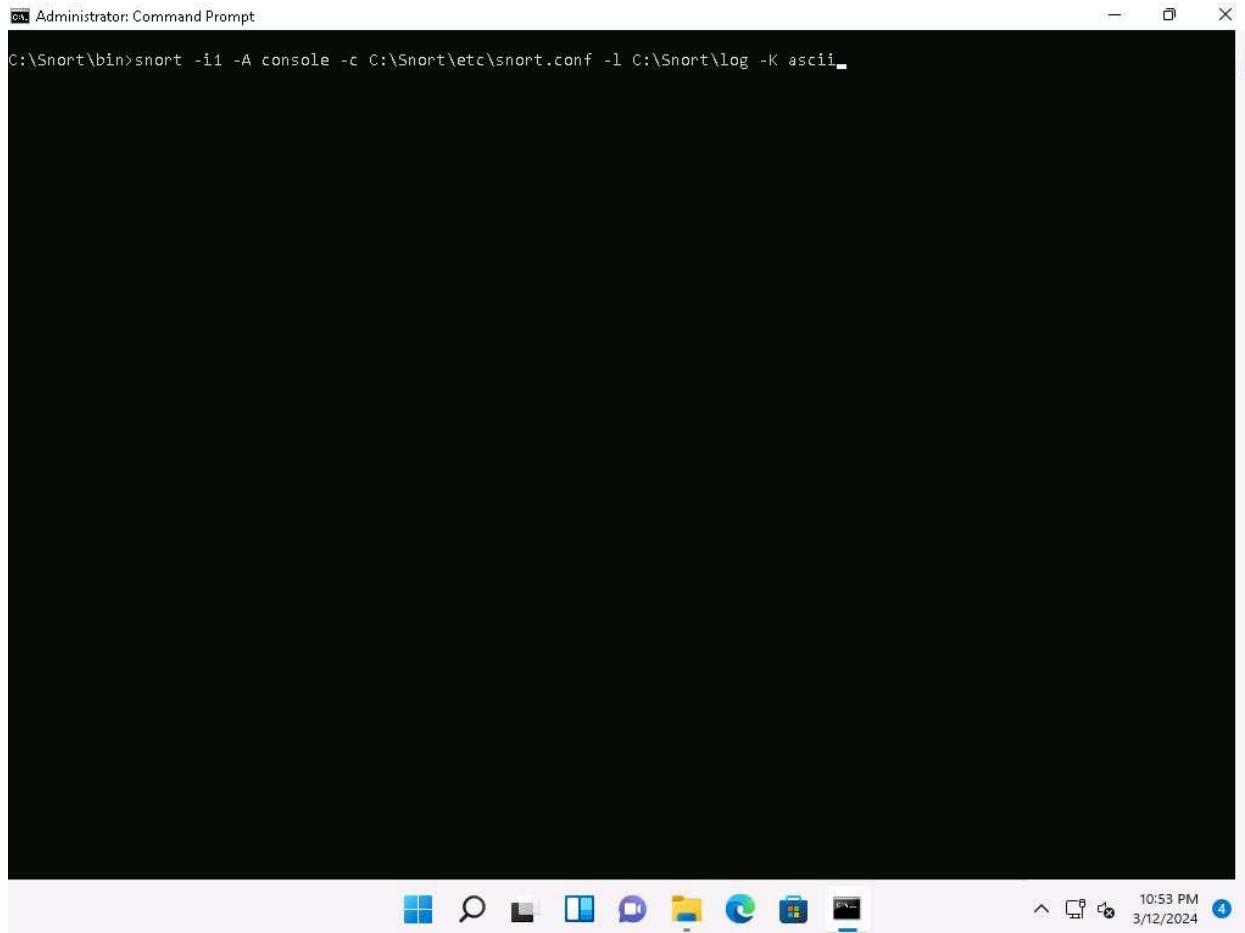
The IP address (10.10.1.11) mentioned in **\$HOME_NET** may vary when you perform this task.



The screenshot shows a Notepad++ window with the file 'C:\Snort\rules\icmp-info.rules' open. The code in the editor is as follows:

```
1 # Copyright 2001-2019 Sourcefire, Inc. All Rights Reserved.
2 #
3 # This file contains (i) proprietary rules that were created, tested and certified by
4 # Sourcefire, Inc. (the "VRT Certified Rules") that are distributed under the VRT
5 # Certified Rules License Agreement (v 2.0), and (ii) rules that were created by
6 # Sourcefire and other third parties (the "GPL Rules") that are distributed under the
7 # GNU General Public License (GPL), v2.
8 #
9 # The VRT Certified Rules are owned by Sourcefire, Inc. The GPL Rules were created
10 # by Sourcefire and other third parties. The GPL Rules created by Sourcefire are
11 # owned by Sourcefire, Inc., and the GPL Rules not created by Sourcefire are owned by
12 # their respective creators. Please see http://www.snort.org/snort/snort-team/ for a
13 # list of third party owners and their respective copyrights.
14 #
15 # In order to determine what rules are VRT Certified Rules or GPL Rules, please refer
16 # to the VRT Certified Rules License Agreement (v2.0).
17 #
18 #-----
19 # ICMP-INFO RULES
20 #-----
21 alert icmp $EXTERNAL_NET any -> $HOME_NET 10.10.1.11 (msg:"ICMP-INFO PING"; icode:0; itype:8; reference:arachnids,135;
22
```

53. Now right-click on the **Windows Start** icon and click **Run** from the menu.
54. In the **Run** window, type **cmd** in the **Open** field and press **Enter**: This will launch a command prompt window.
55. In the command prompt window, type **cd C:\Snort\bin** and press **Enter**.
56. Run command **snort -iX -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii** to start Snort (replace **X** with your device index number; in this task: **X** is 1).



```
C:\Snort\bin>snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii
```

57. If you receive a **fatal error**, you should first **verify** that you have typed all modifications correctly into the **snort.conf** file, and then search through the file for **entries** matching your fatal error message.
58. If you receive an error stating “**Could not create the registry key**,” then run the command prompt as **Administrator**.
59. Snort starts running in IDS mode. It first initializes output plug-ins, preprocessors, plug-ins, loads dynamic preprocessors libraries, rule chains of Snort, and then logs all signatures.
60. If you have entered all command information correctly, you receive a comment stating **Commencing packet processing (pid=xxxx)** (the value of xxxx may be any number; in this task, it is 2132), as shown in the screenshot.

```

Select Administrator: Command Prompt - snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii
Patterns      : 19862
Match States   : 21068
Memory (MB)    : 277.12
  Patterns     : 2.32
  Match Lists   : 5.46
  DFA          :
    1 byte states : 1.66
    2 byte states : 47.54
    4 byte states : 219.64
[ Number of patterns truncated to 20 bytes: 1083 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{5A9B3588-F693-4023-B9B6-DCC29ADB1114}".
Decoding Ethernet

--== Initialization Complete ==--

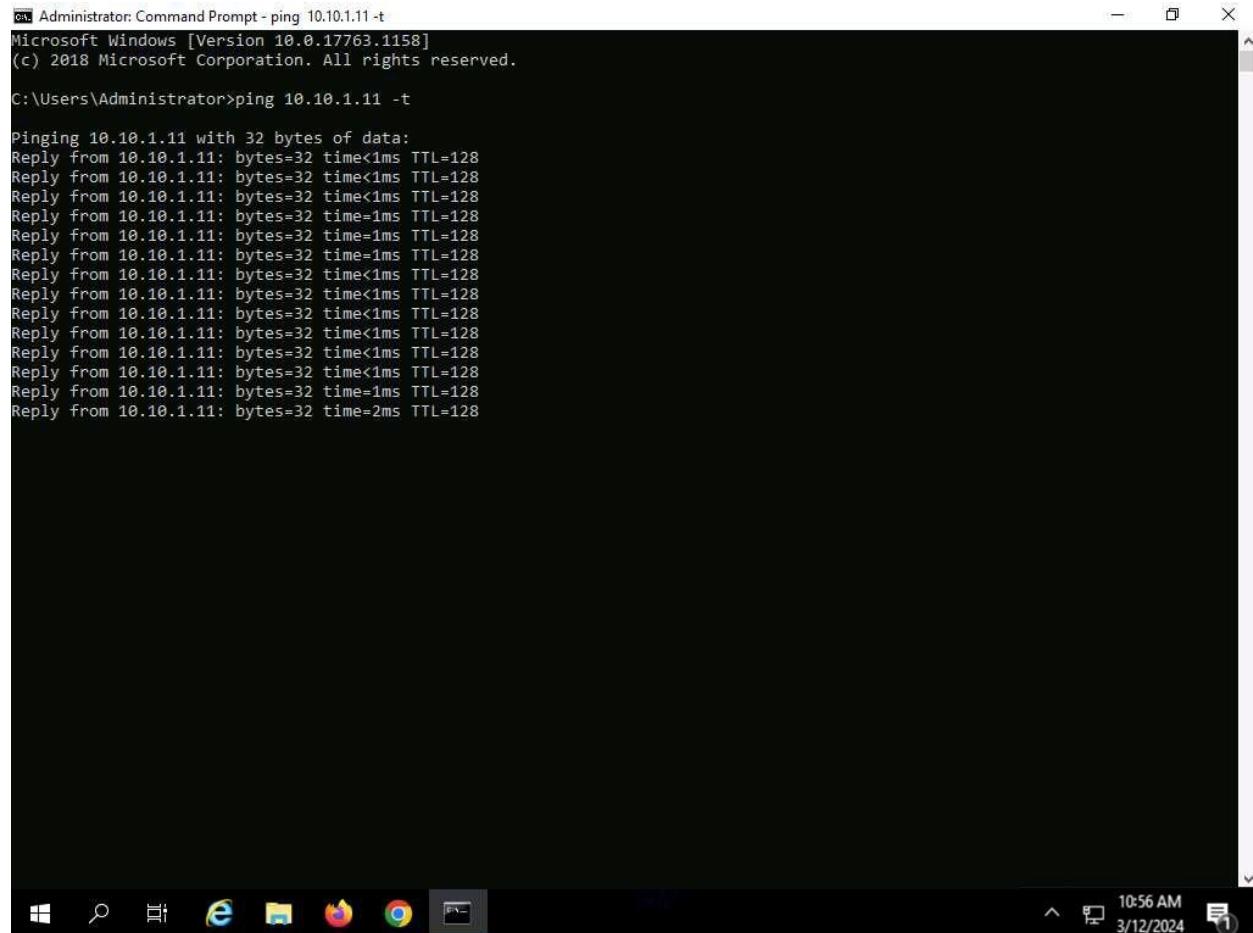
-> Snort! <-
Version 2.9.15-WIN64 GRE (Build 82)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE Version: 8.10 2010-06-25
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.2 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=2132)

```

61. After initializing interface and logged signatures, Snort starts and waits for an attack and triggers alerts when attacks occur on the machine.
62. Leave the Snort command prompt running.
63. Attack your own machine, and check whether Snort detects it or not.
64. Now, click on [Windows Server 2019](#) to switch to the **Windows Server 2019** machine (**Attacker Machine**). Click [Ctrl+Alt+Delete](#) to activate the machine and login with **Administrator/Pa\$\$w0rd**.
65. Open the command prompt and issue the command **ping 10.10.1.11 -t** from the **Attacker Machine**

10.10.1.11 is the IP address of the Windows11. This IP address may differ when you perform the task.



```
Administrator: Command Prompt - ping 10.10.1.11 -t
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ping 10.10.1.11 -t

Pinging 10.10.1.11 with 32 bytes of data:
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time<1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=1ms TTL=128
Reply from 10.10.1.11: bytes=32 time=2ms TTL=128
```

66. Click [Windows 11](#) to return to the **Windows 11** machine. Observe that Snort triggers an alarm, as shown in the screenshot:

```
Administrator: Command Prompt - snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=2132)
03/12-22:58:40.886674 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:41.898290 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:42.912081 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:43.915598 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:44.918642 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:45.935950 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:46.943234 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:47.960549 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:48.973067 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:49.985462 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:50.997071 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:52.006096 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:53.020941 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:54.032603 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:55.049286 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:56.055036 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:57.070372 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
03/12-22:58:58.073774 [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.1
0.1.19 -> 10.10.1.11
```

67. Press **Ctrl+C** to stop Snort; snort exits.

68. Go to the **C:\Snort\log\10.10.1.19** folder and open the **ICMP_ECHO.ids** file with **Notepad++**. You see that all the log entries are saved in the **ICMP_ECHO.ids** file.

The folder name **10.10.1.19** might vary when you perform the task, depending on the IP address of the **Windows 11** machine.

This means that whenever an attacker attempts to connect or communicate with the machine, Snort immediately triggers an alarm

This will make you aware of the intrusion and can thus take certain security measures to disconnect the lines of communication with the attacker's machine.

69. Close all open windows in the **Windows 11** and **Windows Server 2019** machines.

Question 12.1.1.1

Install Snort in the Windows Server 2019 machine. The necessary files are available at Z:\CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots\Intrusion Detection Tools\Snort. Configure and initialize the Snort tool. Initialize the Snort interfaces and attack a target machine from the attacker machine (10.10.1.11) to check whether Snort detects it or not. Enter the Snort command to view the index number of the Ethernet driver.

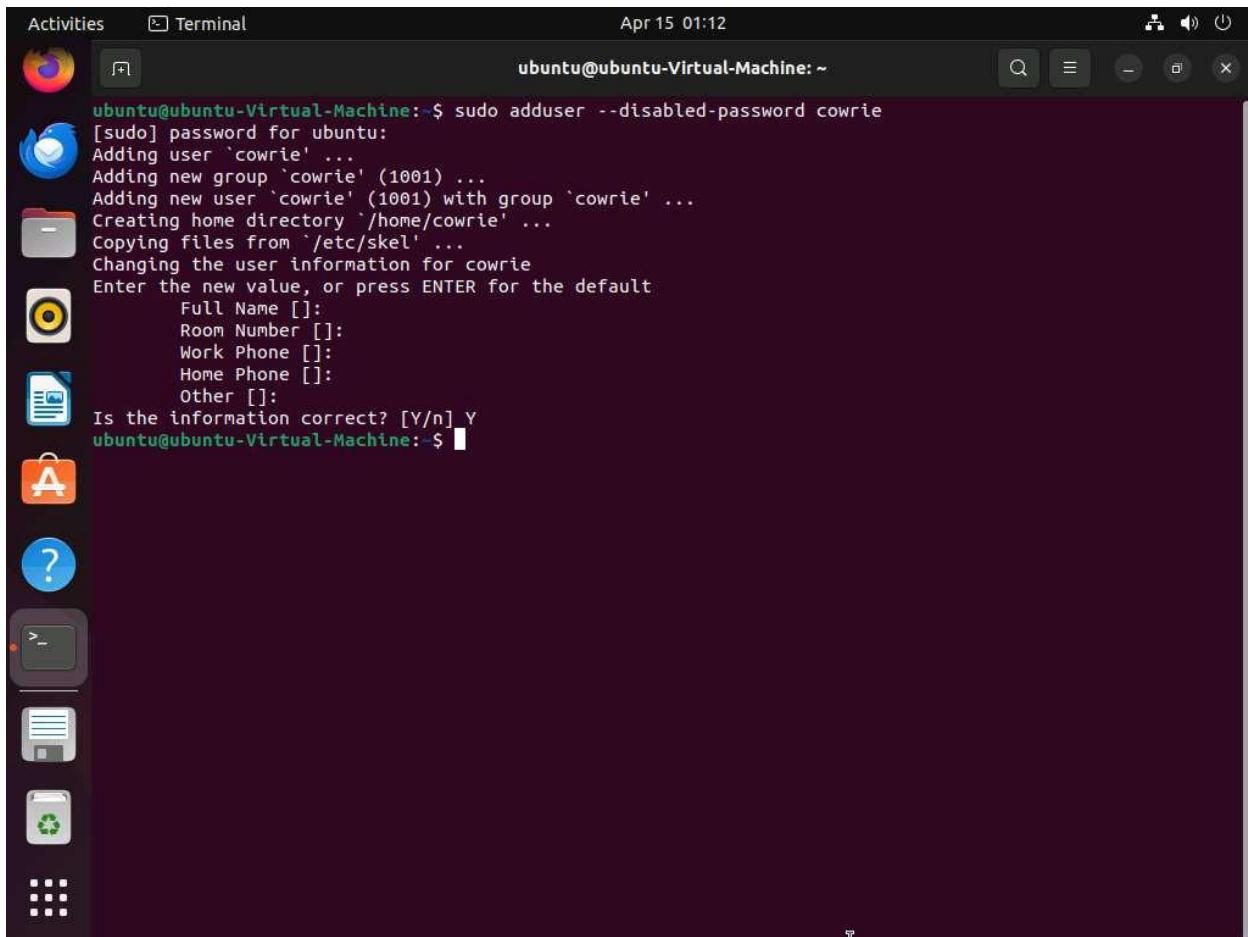
Task 2: Deploy Cowrie Honeypot to Detect Malicious Network Traffic

Cowrie serves as an SSH and Telnet honeypot, capable of capturing brute-force attacks and the actions taken by attackers within the shell. When operating in medium interaction mode, it replicates UNIX behavior using Python. In high interaction mode, acting as a proxy, it allows observation of attacker actions on another system via SSH and Telnet.

Here, we will use Cowrie honeypot to capture incoming malicious traffic from the attacker's machine (here, Parrot Security).

1. Click [Ubuntu](#) to switch to the **Ubuntu** machine and login with **Ubuntu/toor**.
2. In the **Ubuntu** machine, we will deploy **Cowrie** honeypot.
3. Open a **Terminal** window and execute **sudo adduser --disabled-password cowrie** to create a new user named **cowrie** without password (When prompted, enter the password **toor**, The password that you type will not be visible). Close the terminal.

Leave Full Name, Room Number, Work Phone, Home Phone, Other blank and type **Y** when prompted **Is the information correct? [Y/n]**.



The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open in the center, displaying the following command and its execution:

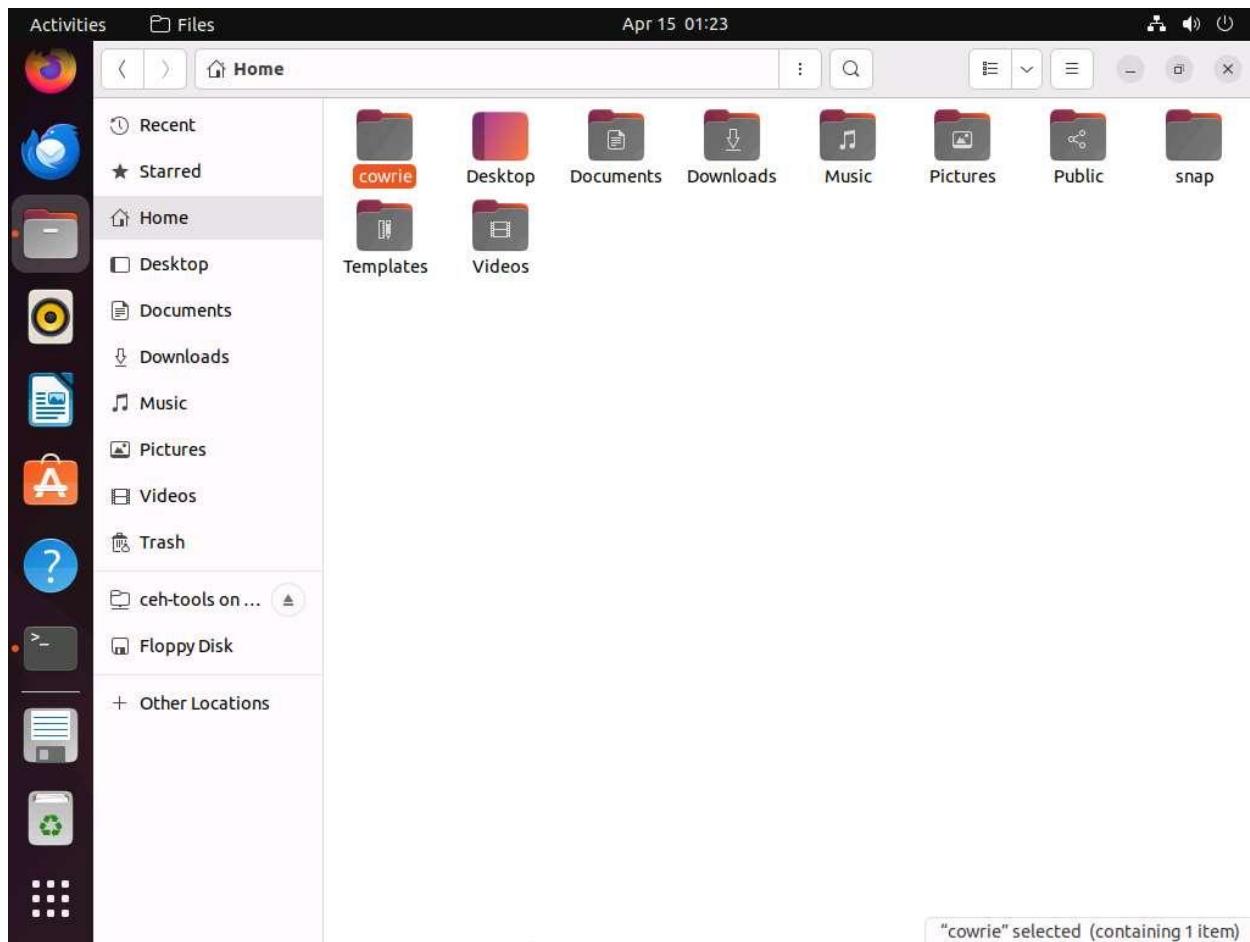
```
ubuntu@ubuntu-Virtual-Machine:~$ sudo adduser --disabled-password cowrie
[sudo] password for ubuntu:
Adding user `cowrie' ...
Adding new group `cowrie' (1001) ...
Adding new user `cowrie' (1001) with group `cowrie' ...
Creating home directory `/home/cowrie' ...
Copying files from `/etc/skel' ...
Changing the user information for cowrie
Enter the new value, or press ENTER for the default
      Full Name []:
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] Y
ubuntu@ubuntu-Virtual-Machine:~$
```

The terminal window has a title bar "Activities Terminal" and a status bar "Apr 15 01:12". To the left of the terminal is a dock with various icons, including a browser, file manager, terminal, and system tray. The desktop background is dark with a subtle grid pattern.

4. Click on **Files** icon and navigate to **ceh tools on 10.10.1.11/CEHv13 Module 12 Evading IDS, Firewalls, and Honeypots/Honeypot Tools** and copy **cowrie** folder and paste it into **/home/ubuntu**.

If **ceh-tools** on **10.10.1.11** option is not present then follow the below steps to access **CEH-Tools** folder:

- Open **Files** and navigate to the **+ Other Locations** from the left pane
- In the **Connect to Server** field, type **smb://10.10.1.11** and press **Enter** to access **Windows 11** shared folders.
- The security pop-up appears; enter the **Windows 11** machine credentials (**Admin/Pa\$\$w0rd**) and click **Connect**.
- The **Windows shares on 10.10.1.11** window appears; double-click the **CEH-Tools** folder.



5. Open a new terminal and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**). Now, jump into Cowrie directory using **cd cowrie** command.

The password that you type will not be visible.

6. Run **pip install --upgrade -r requirements.txt** to install all the required dependencies.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "root@ubuntu-Virtual-Machine: /home/ubuntu/cowrie". The command being run is "pip install --upgrade -r requirements.txt". The output shows the upgrade process for various Python packages, including appdirs, attrs, bcrypt, configparser, cryptography, packaging, pyasn1_modules, pyparsing, python-dateutil, service_identity, tftp, treq, and twisted. The progress bar indicates the download and upgrade status for each package.

```

Activities Terminal Apr 15 01:38
root@ubuntu-Virtual-Machine:~$ sudo su
[sudo] password for ubuntu:
root@ubuntu-Virtual-Machine:/home/ubuntu# cd cowrie/
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie# pip install --upgrade -r requirements.txt
Collecting appdirs==1.4.4
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting attrs==23.2.0
  Downloading attrs-23.2.0-py3-none-any.whl (60 kB) 60.8/60.8 KB 1.8 MB/s eta 0:00:00
Collecting bcrypt==4.1.2
  Downloading bcrypt-4.1.2-cp39-abi3-manylinux_2_28_x86_64.whl (698 kB) 698.9/698.9 KB 22.2 MB/s eta 0:00:00
Collecting configparser==6.0.1
  Downloading configparser-6.0.1-py3-none-any.whl (19 kB)
Collecting cryptography==42.0.5
  Downloading cryptography-42.0.5-cp39-abi3-manylinux_2_28_x86_64.whl (4.6 MB) 4.6/4.6 MB 67.7 MB/s eta 0:00:00
Collecting packaging==24.0
  Downloading packaging-24.0-py3-none-any.whl (53 kB) 53.5/53.5 KB 8.3 MB/s eta 0:00:00
Collecting pyasn1_modules==0.3.0
  Downloading pyasn1_modules-0.3.0-py2.py3-none-any.whl (181 kB) 181.3/181.3 KB 26.6 MB/s eta 0:00:00
Collecting pyparsing==3.1.1
  Downloading pyparsing-3.1.1-py3-none-any.whl (103 kB) 103.1/103.1 KB 23.1 MB/s eta 0:00:00
Collecting python-dateutil==2.8.2
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB) 247.7/247.7 KB 22.0 MB/s eta 0:00:00
Collecting service_identity==24.1.0
  Downloading service_identity-24.1.0-py3-none-any.whl (12 kB)
Collecting tftp==0.8.2
  Downloading tftp-0.8.2.tar.gz (34 kB)
  Preparing metadata (setup.py) ... done
Collecting treq==23.11.0
  Downloading treq-23.11.0-py3-none-any.whl (65 kB) 65.2/65.2 KB 6.2 MB/s eta 0:00:00
Collecting twisted==24.3.0
  Downloading twisted-24.3.0-py3-none-any.whl (2.3 kB)

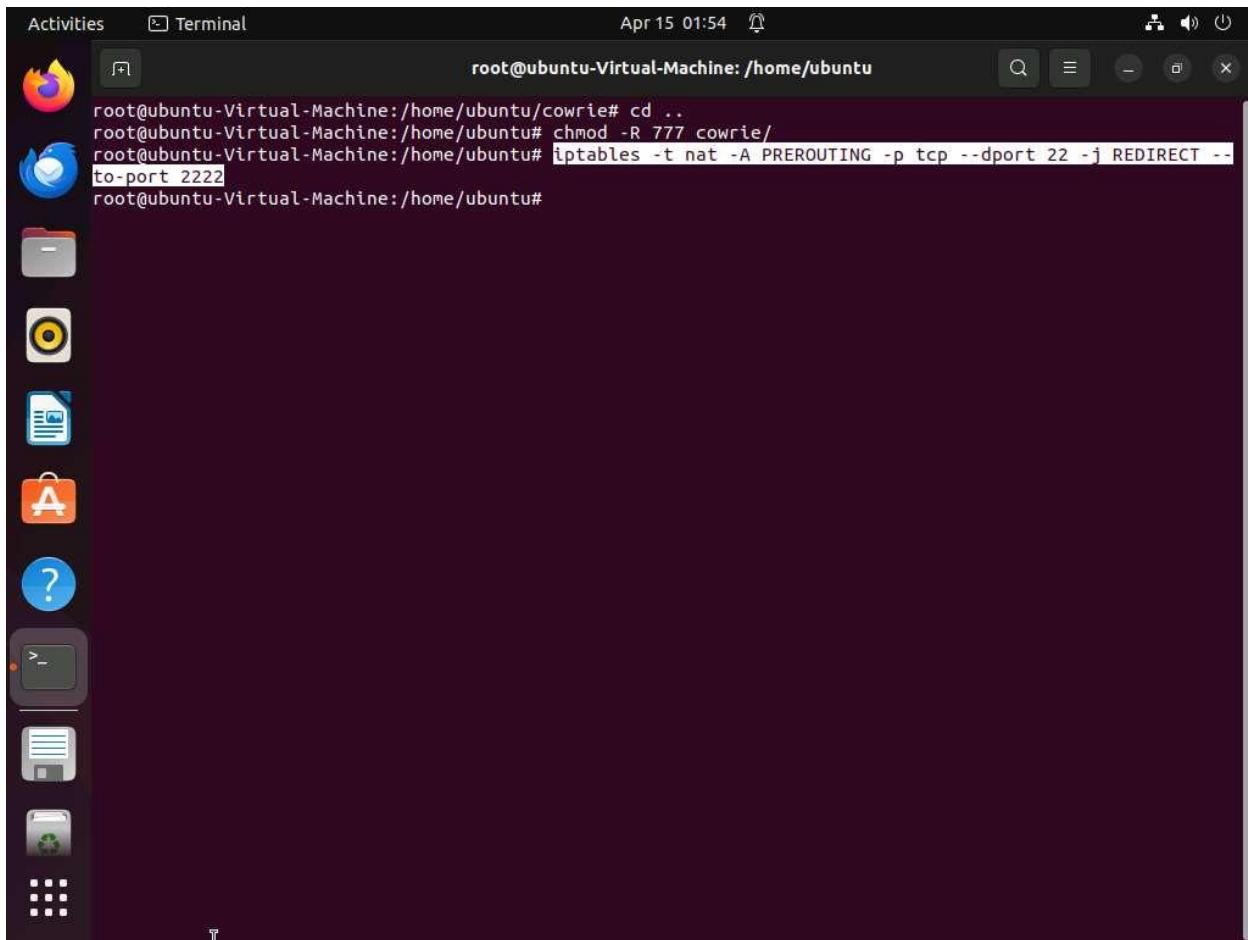
```

7. Run `cd ..` command to jump back to `/home/ubuntu` and run `chmod -R 777 cowrie` to modify the file permissions of cowrie folder.
8. Run `iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222` to redirect the traffic coming on port 22 to port 2222. This redirection channels incoming traffic to a Cowrie honeypot, ensuring the protection of the primary system by segregating potential risks.

Here,

- **-t nat** specifies the table in which the rule should be added. Here, it is the network address translation (NAT) table.
- **-A PREROUTING** specifies that the rule should be appended to the PREROUTING chain. The PREROUTING chain is traversed by packets as soon as they come in, before any routing decisions are made.
- **-p tcp** specifies the protocol to which the rule should apply. Here, it is TCP.
- **--dport 22** specifies the destination port. Here, it is port 22, which is commonly used for SSH (Secure Shell) connections.

- **-j REDIRECT** specifies the target of the rule. It instructs iptables to redirect the packet to another destination instead of its original destination. Here, the destination will be redirected.
- **--to-port 2222** specifies the port to which the packet should be redirected. Here, it is port 2222. So, any incoming TCP packets to port 22 will be redirected to port 2222.

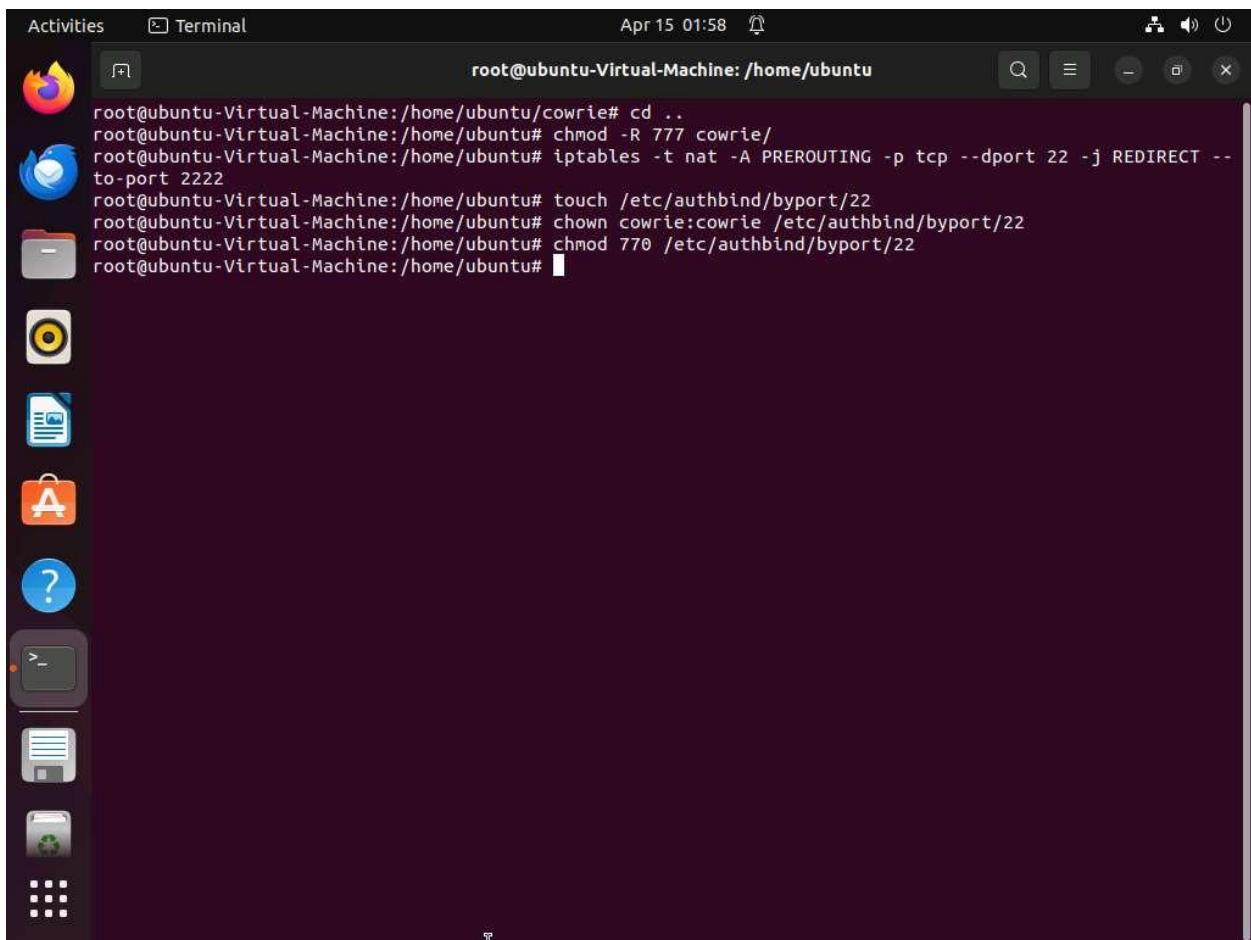


The screenshot shows a terminal window titled "Terminal" with the command prompt "root@ubuntu-Virtual-Machine: /home/ubuntu". The terminal window displays the following command:

```
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie# cd ..
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod -R 777 cowrie/
root@ubuntu-Virtual-Machine:/home/ubuntu# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
root@ubuntu-Virtual-Machine:/home/ubuntu#
```

The terminal window is part of a desktop environment, with a dock on the left containing icons for various applications like a browser, file manager, terminal, and system settings.

9. To facilitate enhanced security measures, run the following commands to configure authbind, enabling the Cowrie honeypot to operate on port 22 without requiring root privileges:
 - Run **touch /etc/authbind/byport/22** to create an authbind configuration file for port 22, essential for granting permission to the Cowrie honeypot to listen on that specific port.
 - Run **chown cowrie:cowrie /etc/authbind/byport/22** to change the ownership of the authbind configuration file for port 22 to the user and group "cowrie," ensuring proper access permissions for the Cowrie honeypot to operate on that port.
 - Run **chmod 770 /etc/authbind/byport/22** to set appropriate permissions on the authbind configuration file for port 22, ensuring that the Cowrie honeypot has the necessary access to operate on the port without requiring root privileges.



A screenshot of an Ubuntu desktop environment. On the left, there's a vertical dock containing icons for various applications: a browser, a mail client, a file manager, a terminal, a system settings gear, a file folder, a document, a terminal, a help symbol, a terminal, a file folder, a terminal, and a terminal. The main window is a terminal window titled "root@ubuntu-Virtual-Machine: /home/ubuntu". The terminal shows the following command sequence:

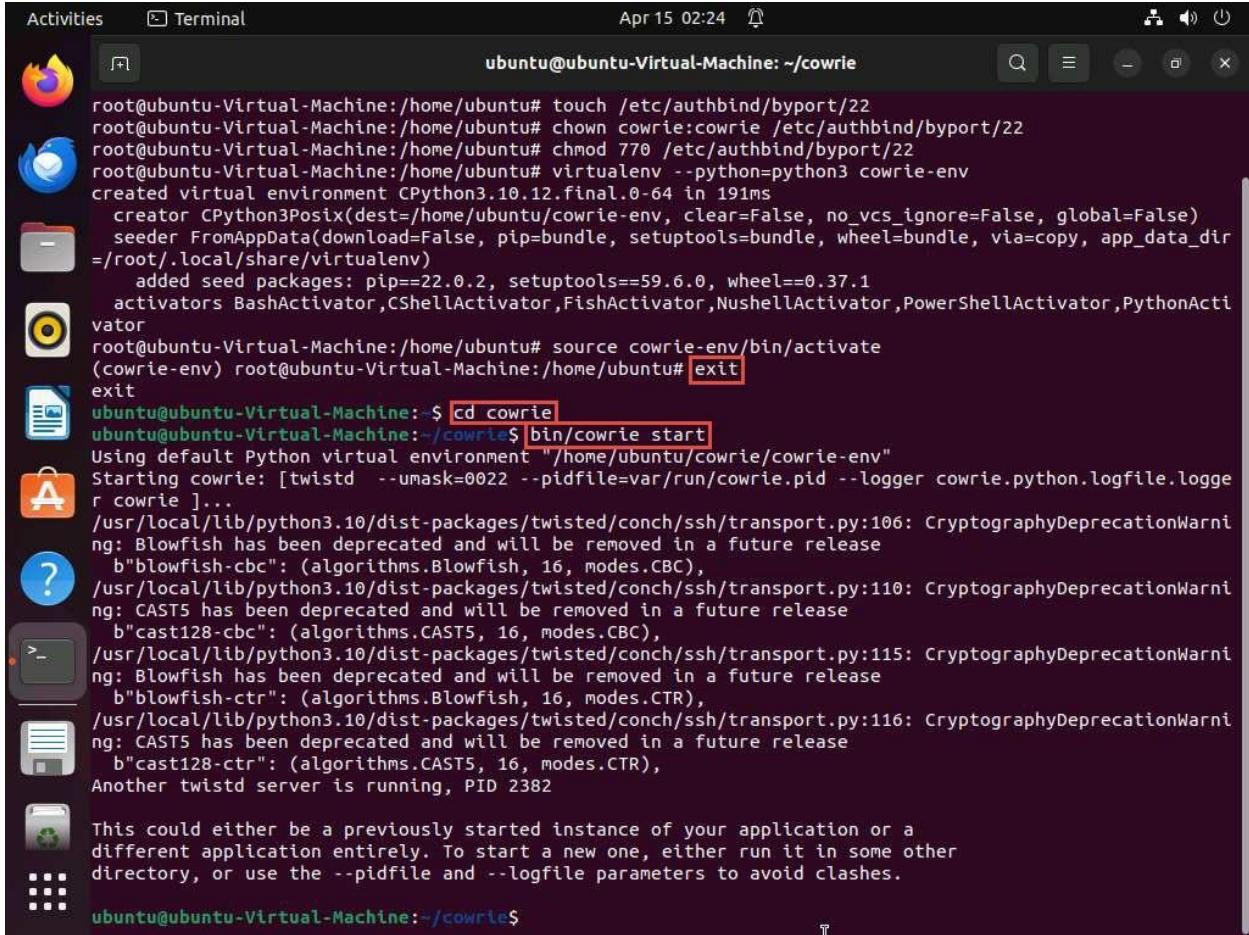
```
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie# cd ..  
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod -R 777 cowrie/  
root@ubuntu-Virtual-Machine:/home/ubuntu# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222  
root@ubuntu-Virtual-Machine:/home/ubuntu# touch /etc/authbind/byport/22  
root@ubuntu-Virtual-Machine:/home/ubuntu# chown cowrie:cowrie /etc/authbind/byport/22  
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod 770 /etc/authbind/byport/22  
root@ubuntu-Virtual-Machine:/home/ubuntu#
```

10. Run **virtualenv --python=python3 cowrie-env** to create a virtual environment. Run **source cowrie-env/local/bin/activate** to activate the environment.

Activities Terminal Apr 15 02:21 root@ubuntu-Virtual-Machine: /home/ubuntu

```
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie# cd ..
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod -R 777 cowrie/
root@ubuntu-Virtual-Machine:/home/ubuntu# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --
to-port 2222
root@ubuntu-Virtual-Machine:/home/ubuntu# touch /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# chown cowrie:cowrie /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod 770 /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# virtualenv --python=python3 cowrie-env
created virtual environment CPython3.10.12.final.0-64 in 191ms
  creator CPython3Posix(dest=/home/ubuntu/cowrie-env, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir
=/root/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
root@ubuntu-Virtual-Machine:/home/ubuntu# source cowrie-env/local/bin/activate
(cowrie-env) root@ubuntu-Virtual-Machine:/home/ubuntu#
```

11. Exit the root privileged terminal using **exit** command. Navigate to cowrie directory using **cd cowrie** command. Here, run **bin/cowrie start** to initiate the Cowrie honeypot and begin monitoring and logging incoming traffic for security analysis and threat detection purposes.

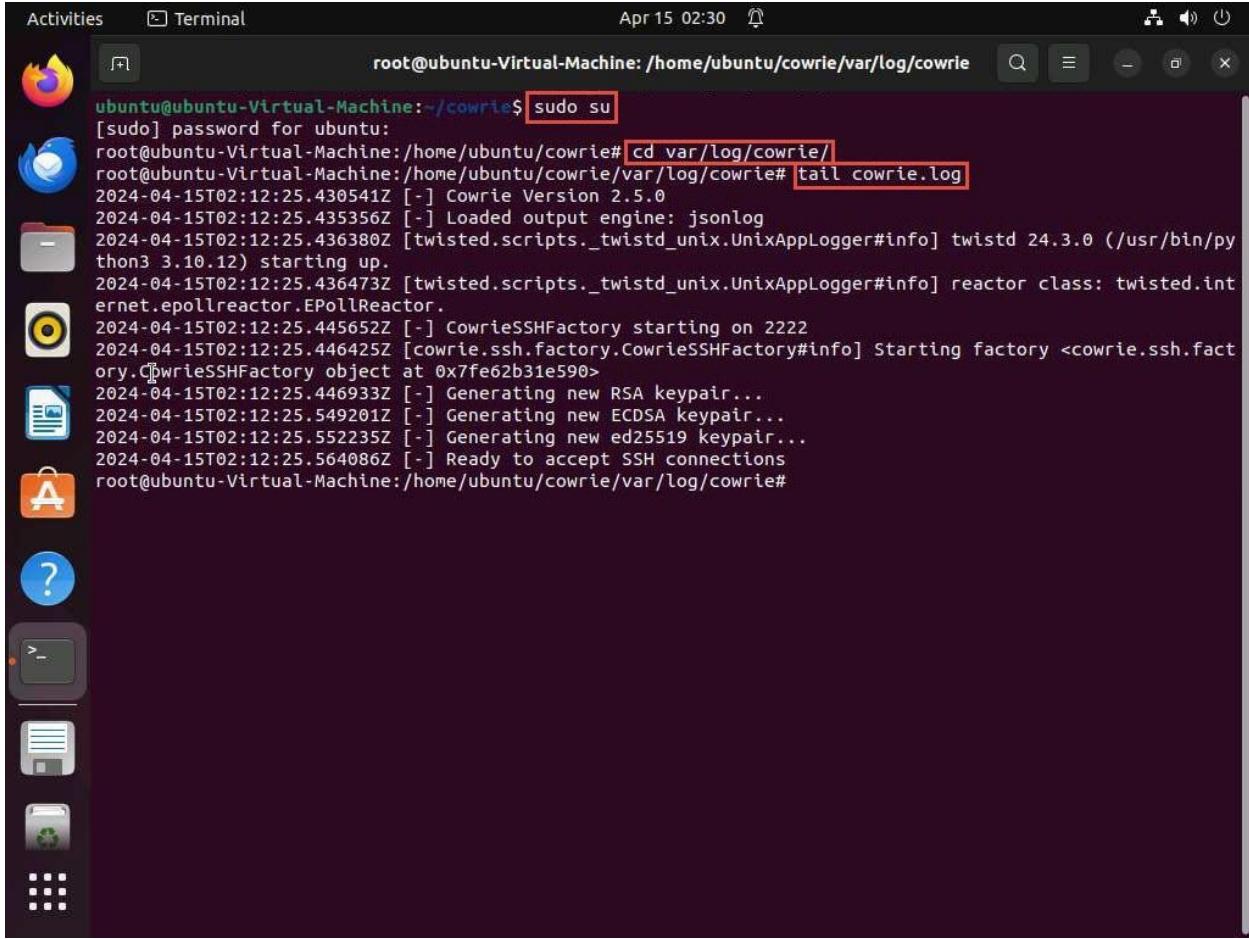


The screenshot shows a terminal window on an Ubuntu desktop. The terminal title is "ubuntu@ubuntu-Virtual-Machine: ~/cowrie". The terminal content shows the following steps to set up the Cowrie honeypot:

```
root@ubuntu-Virtual-Machine:/home/ubuntu# touch /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# chown cowrie:cowrie /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# chmod 770 /etc/authbind/byport/22
root@ubuntu-Virtual-Machine:/home/ubuntu# virtualenv --python=python3 cowrie-env
created virtual environment CPython3.10.12.final.0-64 in 191ms
  creator CPython3Posix(dest=/home/ubuntu/cowrie-env, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir
=/root/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
root@ubuntu-Virtual-Machine:/home/ubuntu# source cowrie-env/bin/activate
(cowrie-env) root@ubuntu-Virtual-Machine:/home/ubuntu# exit
exit
ubuntu@ubuntu-Virtual-Machine:~$ cd cowrie
ubuntu@ubuntu-Virtual-Machine:~/cowrie$ bin/cowrie start
Using default Python virtual environment "/home/ubuntu/cowrie/cowrie-env"
Starting cowrie: [twistd --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.looger cowrie]...
/usr/local/lib/python3.10/dist-packages/twisted/conch/ssh/transport.py:106: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in a future release
  b"blowfish-cbc": (algorithms.Blowfish, 16, modes.CBC),
/usr/local/lib/python3.10/dist-packages/twisted/conch/ssh/transport.py:110: CryptographyDeprecationWarning: CAST5 has been deprecated and will be removed in a future release
  b"cast128-cbc": (algorithms.CAST5, 16, modes.CBC),
/usr/local/lib/python3.10/dist-packages/twisted/conch/ssh/transport.py:114: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in a future release
  b"blowfish-ctr": (algorithms.Blowfish, 16, modes.CTR),
/usr/local/lib/python3.10/dist-packages/twisted/conch/ssh/transport.py:118: CryptographyDeprecationWarning: CAST5 has been deprecated and will be removed in a future release
  b"cast128-ctr": (algorithms.CAST5, 16, modes.CTR),
Another twistd server is running, PID 2382
This could either be a previously started instance of your application or a different application entirely. To start a new one, either run it in some other directory, or use the --pidfile and --logfile parameters to avoid clashes.
```

12. In the terminal, acquire root privileges using **sudo su** command and execute **cd var/log/cowrie/** to navigate to **var/log/cowrie**. Now, run **tail cowrie.log** to view the log file.

Here, **Ready to accept SSH connections** confirms that the honeypot is successfully setup.

A screenshot of an Ubuntu desktop environment. On the left is a dock with various icons: a browser, a terminal, a file manager, a mail client, a calendar, a file viewer, a terminal, a help icon, a terminal, and a terminal. The main window is a terminal window titled 'Terminal' with the command line 'root@ubuntu-Virtual-Machine: /home/ubuntu/cowrie#'. The terminal shows the following log output:

```
root@ubuntu-Virtual-Machine:~/cowrie$ sudo su
[sudo] password for ubuntu:
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie# cd var/log/cowrie/
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie# tail cowrie.log
2024-04-15T02:12:25.430541Z [-] Cowrie Version 2.5.0
2024-04-15T02:12:25.435356Z [-] Loaded output engine: jsonlog
2024-04-15T02:12:25.436380Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.3.0 (/usr/bin/python3 3.10.12) starting up.
2024-04-15T02:12:25.436473Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2024-04-15T02:12:25.445652Z [-] CowrieSSHFactory starting on 2222
2024-04-15T02:12:25.446425Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0x7fe62b31e590>
2024-04-15T02:12:25.446933Z [-] Generating new RSA keypair...
2024-04-15T02:12:25.549201Z [-] Generating new ECDSA keypair...
2024-04-15T02:12:25.552235Z [-] Generating new ed25519 keypair...
2024-04-15T02:12:25.564086Z [-] Ready to accept SSH connections
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie#
```

13. Now, we will use **Parrot Security** machine as an attacker to attack the honeypot running SSH service. To do so, firstly we will scan the target machine for open SSH port.
14. Click [Parrot Security](#) to switch to **Parrot Security** machine and use **toor** as password. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
15. In the terminal, run **nmap -p- -sV 10.10.1.9** command to discover open ports and services.
16. You can observe that the port 22 is open and is the default port used for SSH connection. Since the port is open we will try to connect to the machine using SSH.

Here, you can also observe the SSH service running on port 2222 that is a honeypot that we have deployed. Attackers will target the SSH service running on default SSH port (22).

The screenshot shows a terminal window titled "nmap -p- -sV 10.10.1.9 - Parrot Terminal". The terminal output is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nmap -p- -sV 10.10.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-15 02:34 EDT
Nmap scan report for 10.10.1.9
Host is up (0.00051s latency).

Not shown: 65532 closed tcp ports (reset)

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
2222/tcp  open  ssh      OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
MAC Address: 02:15:5D:54:3E:44 (Unknown)

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.23 seconds
#
```

17. To establish a connection to the target machine via SSH, we will use PuTTY. Initiate the PuTTY interface by executing **putty** command.

The screenshot shows a terminal window titled "nmap -p- -sV 10.10.1.9 - Parrot Terminal". The terminal session starts with the user "attacker" at the root prompt, entering "sudo su" and providing the password for attacker. The user then runs an Nmap scan on the target IP 10.10.1.9. The output shows the host is up with two open ports: 22/tcp (ssh) and 80/tcp (http). Both services are identified as OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0). The MAC address is listed as 02:15:5D:38:8B:87 (Unknown). Service information indicates the OS is Linux and the CPE is cpe:/o:linux:linux_kernel. A note about service detection and reporting is present. Finally, the user runs "putty" at the root prompt.

```
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
#nmap -p- -sV 10.10.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-16 01:39 EDT
Nmap scan report for 10.10.1.9
Host is up (0.00018s latency).

Not shown: 65532 closed tcp ports (reset)

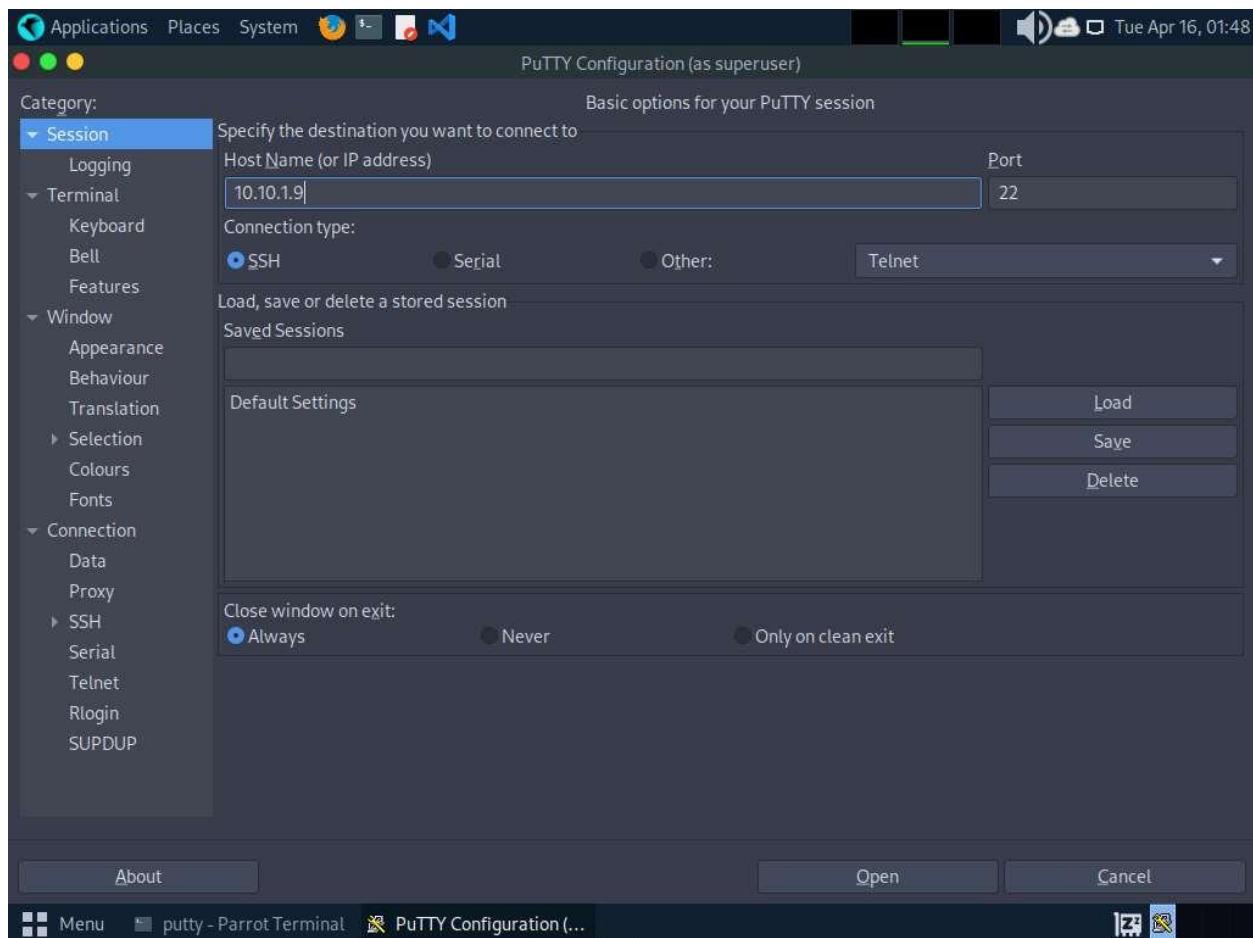
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.52 ((Ubuntu))
2222/tcp  open  ssh    OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
MAC Address: 02:15:5D:38:8B:87 (Unknown)

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.72 seconds
[root@parrot]~[/home/attacker]
#putty
```

18. The **PuTTY Configuration (as superuser)** window appears. Enter the IP address of the target machine (here, **10.10.1.9**) in the **Host Name (or IP address)** field, and proceed by clicking on **Open** button.

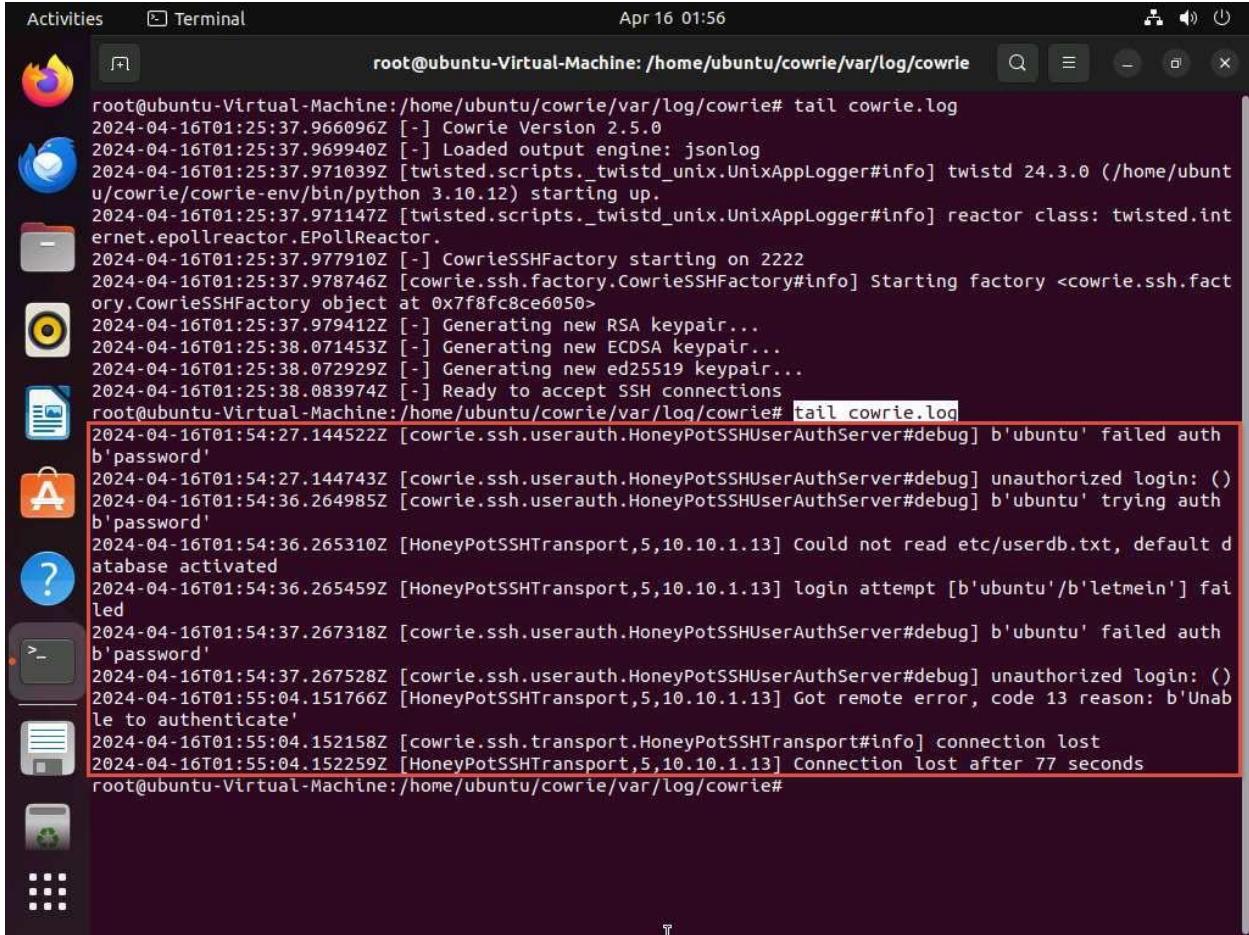
PuTTY Security Alert (as superuser) window appears, click **Accept**.



19. In **10.10.1.9 - PuTTY (as superuser)** window, type **ubuntu** in **login as** field and password as **toor**, it responds with **Access denied**. Try a random set of passwords to bruteforce into the target machine.

```
login as: ubuntu
ubuntu@10.10.1.9's password:
Access denied
ubuntu@10.10.1.9's password:
Access denied
ubuntu@10.10.1.9's password:
Access denied
ubuntu@10.10.1.9's password:
```

20. Click [Ubuntu](#) to switch back to **Ubuntu** machine, and run **tail cowrie.log**. Here, observe the logs generated when the attacker tried to connect to the SSH service on the **Ubuntu** machine using PuTTY. These log entries serve as evidence of the attacker's unauthorized access attempts.



A screenshot of a Linux desktop environment (Ubuntu) showing a terminal window. The terminal window title is "root@ubuntu-Virtual-Machine: /home/ubuntu/cowrie/var/log/cowrie". The terminal content displays log entries from the "cowrie" service, specifically from "cowrie.log". The log entries show the startup of the service, generation of RSA and ECDSA keypairs, and multiple failed login attempts from the IP address 10.10.1.13, where the user 'ubuntu' tried to log in with the password 'password'. The terminal window is part of a desktop interface with other application icons visible in the background.

```
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie# tail cowrie.log
2024-04-16T01:25:37.966096Z [-] Cowrie Version 2.5.0
2024-04-16T01:25:37.969940Z [-] Loaded output engine: jsonlog
2024-04-16T01:25:37.971039Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.3.0 (/home/ubuntu/cowrie-env/bin/python 3.10.12) starting up.
2024-04-16T01:25:37.971147Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2024-04-16T01:25:37.977910Z [-] CowrieSSHFactory starting on 2222
2024-04-16T01:25:37.978746Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0x7f8fc8ce6050>
2024-04-16T01:25:37.979412Z [-] Generating new RSA keypair...
2024-04-16T01:25:38.071453Z [-] Generating new ECDSA keypair...
2024-04-16T01:25:38.072929Z [-] Generating new ed25519 keypair...
2024-04-16T01:25:38.083974Z [-] Ready to accept SSH connections
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie# tail cowrie.log
2024-04-16T01:54:27.144522Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'ubuntu' failed auth b'password'
2024-04-16T01:54:27.144743Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2024-04-16T01:54:36.264985Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'ubuntu' trying auth b'password'
2024-04-16T01:54:36.265310Z [HoneyPotSSHTransport,5,10.10.1.13] Could not read etc/userdb.txt, default database activated
2024-04-16T01:54:36.265459Z [HoneyPotSSHTransport,5,10.10.1.13] login attempt [b'ubuntu'/b'letmein'] failed
2024-04-16T01:54:37.267318Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'ubuntu' failed auth b'password'
2024-04-16T01:54:37.267528Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2024-04-16T01:55:04.151766Z [HoneyPotSSHTransport,5,10.10.1.13] Got remote error, code 13 reason: b'Unable to authenticate'
2024-04-16T01:55:04.152158Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-04-16T01:55:04.152259Z [HoneyPotSSHTransport,5,10.10.1.13] Connection lost after 77 seconds
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie#
```

21. Click [Parrot Security](#) to switch back to **Parrot Security** machine. In the terminal window, open the PuTTY interface by executing **putty** command.
22. In the **PuTTY Configuration (as superuser)** window, type the IP address of the target machine (here, **10.10.1.9**) under **Host Name (or IP address)** section and click on **Open**.

If **PuTTY Security Alert (as superuser)** window appears, click **Accept**.

23. In **10.10.1.9 - PuTTY (as superuser)** window, type login as **root** and type a random password of your choice.

```
login as: root
root@10.10.1.9's password:
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@svr04:~#
```

24. You are now in the honeypot, execute various commands to gather intelligence and explore the system.

- Run **id** command to retrieve information about the current user and group.
- To identify the username associated with the active session, execute **whoami** command.
- Run **pwd** to determine the present working directory within the system.
- Run **cd ..** to navigate to the root directory.
- Run **ls** to list the files available in the directory.
- Continue the reconnaissance, run **ls -la** command to list detailed information about files and directories in the current location.

The Cowrie honeypot has default credentials set to **root / *** (with root as username and taking any character or word as a correct password). After obtaining access to the SSH service, attacker runs the above commands giving him a sense of compromising the target system successfully. However, all these interactions are captured by the virtual environment which was created while setting up Cowrie honeypot, without affecting actual SSH service running on the target system.

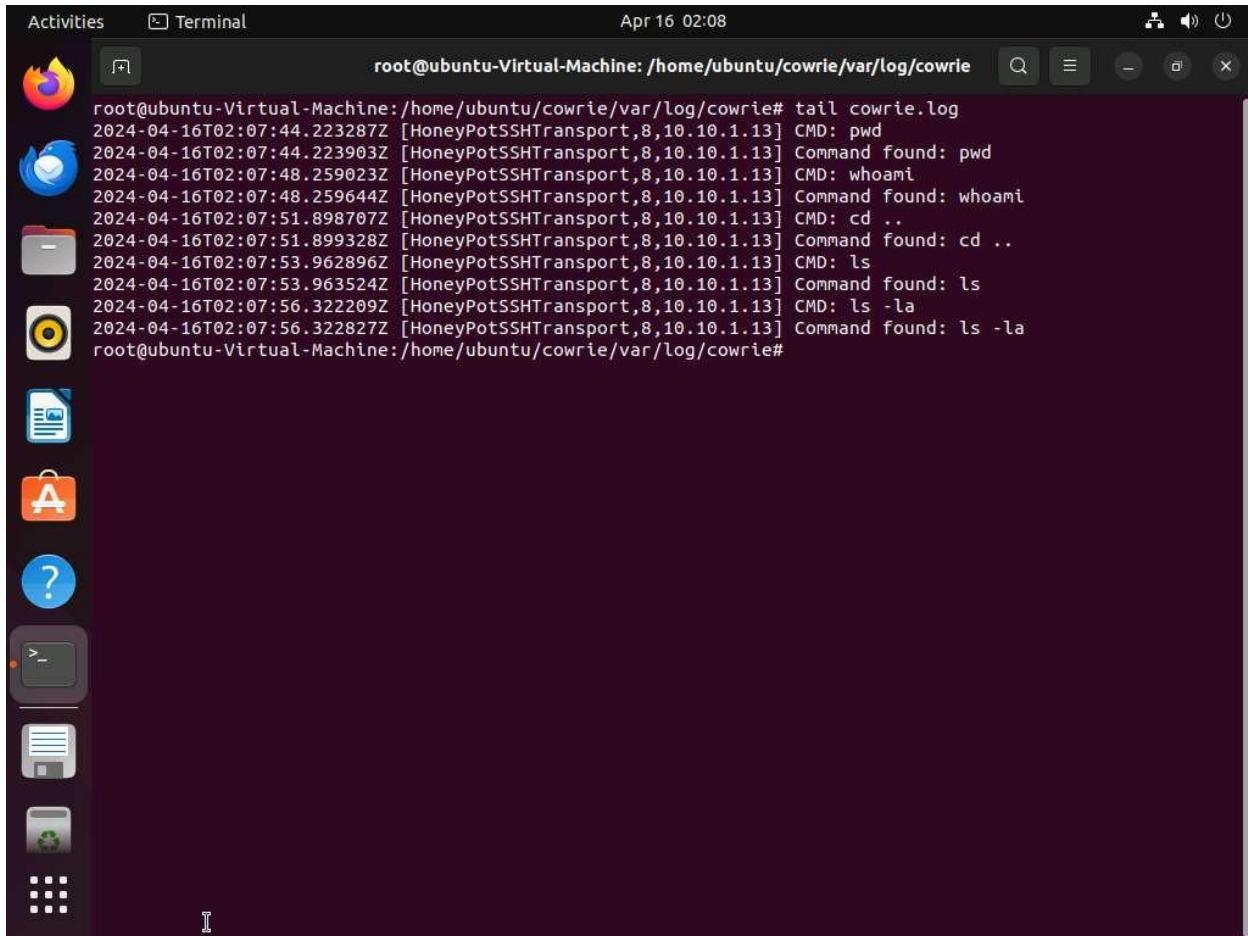
[more...](#)

```
login as: root
root@10.10.1.9's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@svr04:~# id
uid=0(root) gid=0(root) groups=0(root)
root@svr04:~# whoami
root
root@svr04:~# pwd
/root
root@svr04:~# cd ..
root@svr04:~/#
root@svr04:~/# ls
bin      boot      dev      etc      home      initrd.img lib      lost+found media      mnt      opt      proc      root      run      sbin
selinux   srv       sys      test2    tmp       user      var      vmlinuz
root@svr04:~/# ls -la
drwxr-xr-x 1 root root  4096 2013-04-05 08:03 .
drwxr-xr-x 1 root root  4096 2013-04-05 08:03 ..
drwxr-xr-x 1 root root  4096 2013-04-05 07:53 bin
drwxr-xr-x 1 root root  4096 2013-04-05 08:02 boot
drwxr-xr-x 1 root root  3060 2013-04-05 08:03 dev
drwxr-xr-x 1 root root  4096 2013-04-05 08:06 etc
drwxr-xr-x 1 root root  4096 2013-04-05 08:02 home
lrwxrwxrwx 1 root root   32 2013-04-05 07:53 initrd.img -> /boot/initrd.img-3.2.0-4-686-pae
drwxr-xr-x 1 root root  4096 2013-04-05 08:01 lib
drwxr-xr-x 1 root root 16384 2013-04-05 07:52 lost+found
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 media
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 mnt
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 opt
dr-xr-xr-x 1 root root     0 2013-04-05 08:03 proc
drwxr-xr-x 1 root root  4096 2013-04-05 08:25 root
drwxr-xr-x 1 root root   380 2013-04-05 08:06 run
drwxr-xr-x 1 root root  4096 2013-04-05 08:03sbin
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 selinux
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 srv
drwxr-xr-x 1 root root     0 2013-04-05 08:03 sys
-rwxr-xr-x 1 root root   500 2021-06-30 00:44 test2
drwxrwxrwt 1 root root  4096 2013-04-05 08:17 tmp
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 usr
drwxr-xr-x 1 root root  4096 2013-04-05 07:52 var
lrwxrwxrwx 1 root root   28 2013-04-05 07:53 vmlinuz -> /boot/vmlinuz-3.2.0-4-686-pae
root@svr04:~#
```

25. Click [Ubuntu](#) to switch back to the **Ubuntu** machine and execute **tail cowrie.log** command to view the logs. These logs will capture the activities performed by the attacker within the Cowrie honeypot environment, including the execution of commands.



A screenshot of an Ubuntu desktop environment. On the left is a vertical dock with icons for various applications: a browser, email, file manager, terminal, help, terminal, file manager, and a grid icon. The main window is a terminal titled "root@ubuntu-Virtual-Machine: /home/ubuntu/cowrie/var/log/cowrie". The terminal shows the following log entries from the "cowrie.log" file:

```
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie# tail cowrie.log
2024-04-16T02:07:44.223287Z [HoneyPotSSHTransport,8,10.10.1.13] CMD: pwd
2024-04-16T02:07:44.223903Z [HoneyPotSSHTransport,8,10.10.1.13] Command found: pwd
2024-04-16T02:07:48.259023Z [HoneyPotSSHTransport,8,10.10.1.13] CMD: whoami
2024-04-16T02:07:48.259644Z [HoneyPotSSHTransport,8,10.10.1.13] Command found: whoami
2024-04-16T02:07:51.898707Z [HoneyPotSSHTransport,8,10.10.1.13] CMD: cd ..
2024-04-16T02:07:51.899328Z [HoneyPotSSHTransport,8,10.10.1.13] Command found: cd ..
2024-04-16T02:07:53.962896Z [HoneyPotSSHTransport,8,10.10.1.13] CMD: ls
2024-04-16T02:07:53.963524Z [HoneyPotSSHTransport,8,10.10.1.13] Command found: ls
2024-04-16T02:07:56.322209Z [HoneyPotSSHTransport,8,10.10.1.13] CMD: ls -la
2024-04-16T02:07:56.322827Z [HoneyPotSSHTransport,8,10.10.1.13] Command found: ls -la
root@ubuntu-Virtual-Machine:/home/ubuntu/cowrie/var/log/cowrie#
```

This will notify you of the intrusion, allowing you to implement specific security measures to sever communication with the attacker's machine.

26. This concludes the demonstration of detecting malicious network traffic using Cowrie.
27. Close all open windows and document all acquired information.

Question 12.1.2.1

In ubuntu machine (10.10.1.9) setup Cowrie honeypot to capture incoming malicious traffic on port 22 from the attacker's machine (10.10.1.13). Enter the port number to which the port 22 traffic is redirected to.