Cybersecurity for Industrial Networks

Topic 7
Penetration Test - Reconnaissance

Dr Diarmuid Ó Briain

Version: 1.2

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Conrad Ekisa

Dr Diarmuid Ó Briain

Table of Contents

Illustration Index

1 Objectives

By the end of this topic, you will be able to:

- Carry out a reconnaissance on the Virtualised ICS Open-source Research Testbed (VICSORT) Operational Technology Simulation.

2 Introduction

Virtualised ICS Open-source Research Testbed (VICSORT) is a modified build of Graphical Realism Framework for Industrial Control Simulation Version 2 (GRFICSv2). VICSORT is a light-weight open-source Industrial Control Systems (ICS) testbed designed to be repeatable, scalable and easy to deploy. VICSORT, built upon Ubuntu 20.04 LTS, leverages LXD, a system container and virtual machine manager, Linux Containers (LXC) and the Kernel Virtual Machine (KVM) to provide a leaner over build requiring significantly less system resources to operate, compared to its predecessor GRFICSv2.

VICSORT maintains all the testbed components in GRFICSv2 i.e the Human Machine Interface (HMI), Programmable Logic Controller (PLC), engineering workstation, firewall, a physical process simulation and also interoperates an attacker workstation based on Kali Linux 2021.

2.1 The Purdue Model

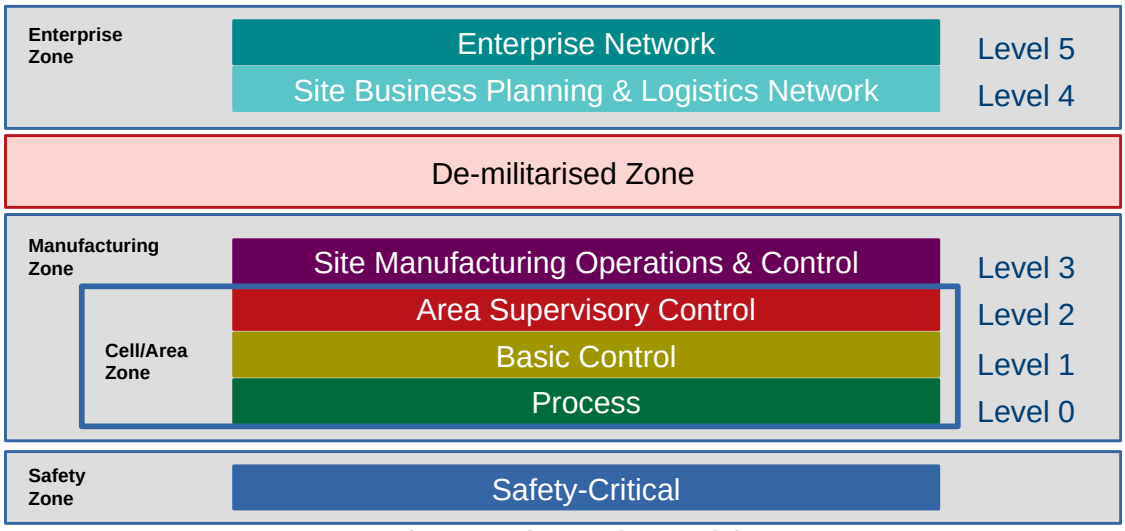


Figure 1: The Purdue Model

2.2 Network Topology

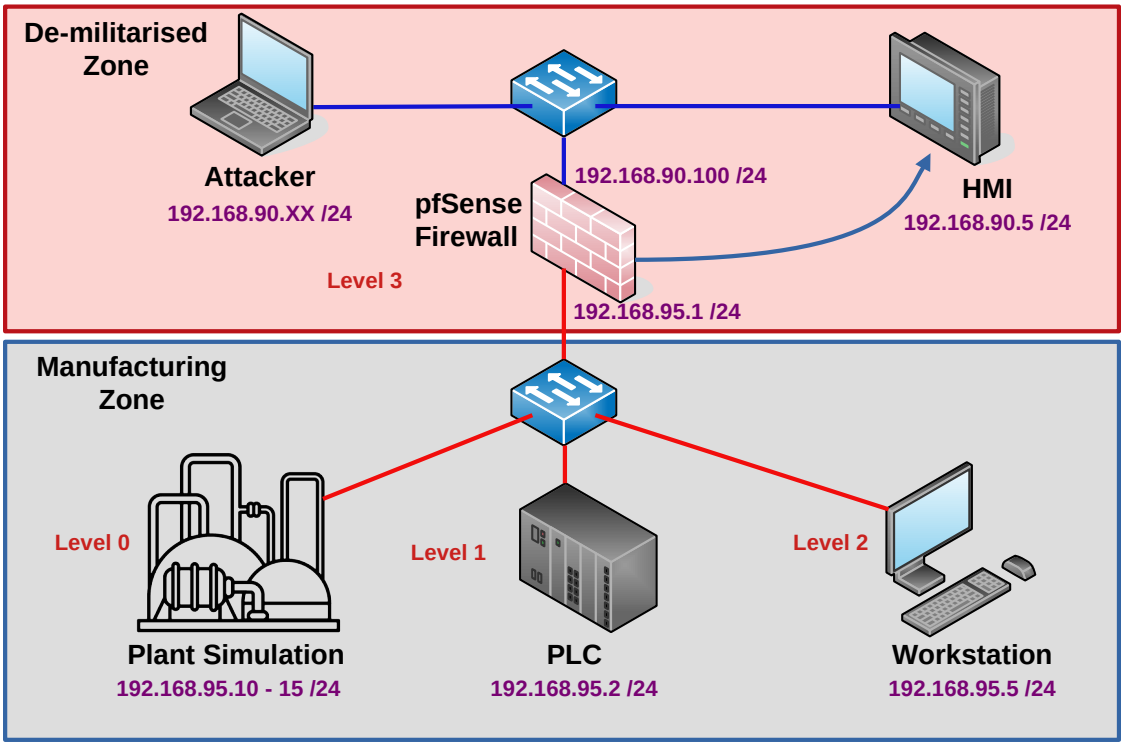


Figure 2: ICS Testbed

The VICSORT testbed assumes the network topology illustrated in Figure 2, with the IP address mapping and passwords listed in Figure 3.

Node	IP Address Mapping
HMI	192.168.90.5 /24
Firewall	- WAN: 192.168.90.100 /24 - LAN: 192.168.95.100 /24
PLC	192.168.95.2 /24
Engineering Workstation	192.168.95.5 /24
Plant Simulation	192.168.95.10 - 15 /24
Attacker	192.168.90.XX /24

Firewall Username: admin	Password: pfsense
HMI Username: admin	Password: admin
Kali Username: kali	Password: kali

Figure 3: Network Topology and Passwords

3 Pre-work with VICSORT

Using VirtualBox, import the Open Virtual Appliance (.ova).

As it boots it is very likely that the network interface error, illustrated in Figure 4, will occur. This is because the interface name on the computer the Virtual Machine (VM) was created on is different to the name on the computer opening the VM. **`Select Change Network Settings`**.

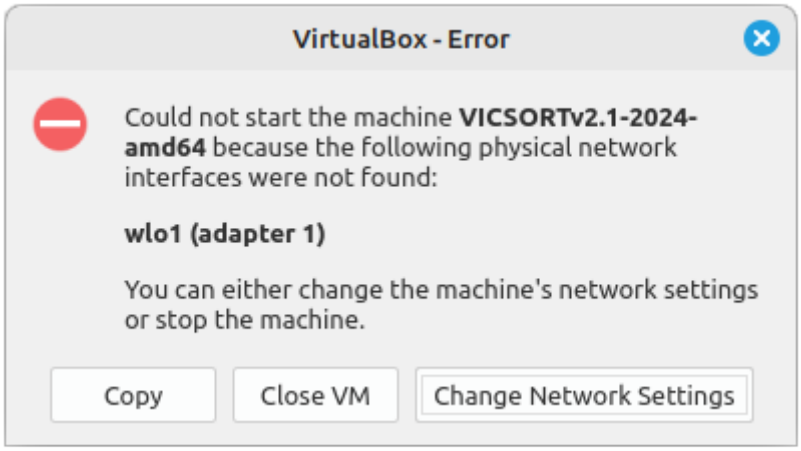


Figure 4: Network interface error

With the **`Network`** option open, as illustrated in Figure 5, Select the correct interface for the computer. The **`Attached to:`** option is set to **`Bridged Adapter`** if the physical connect network will assign Internet Protocol (IP) addresses from a Dynamic Host Configuration Protocol server to VMs, if not select the Network Address Translation (**`NAT`**) option.

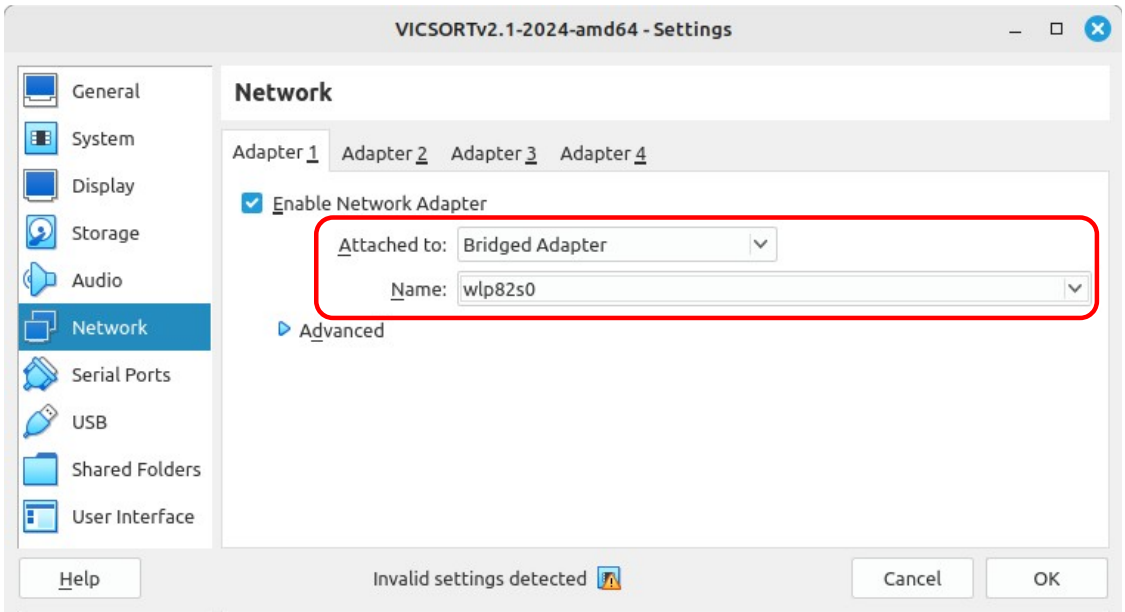


Figure 5: Network Adapter

Start the VM.

Login to the VM with the username: **`vicsort`** and password **`vicsort`**.

After logging in the display may need to be adjusted. As illustrated in Figure 6, from the menu View and Virtual Screen 1, select a comfortable display size for operation.

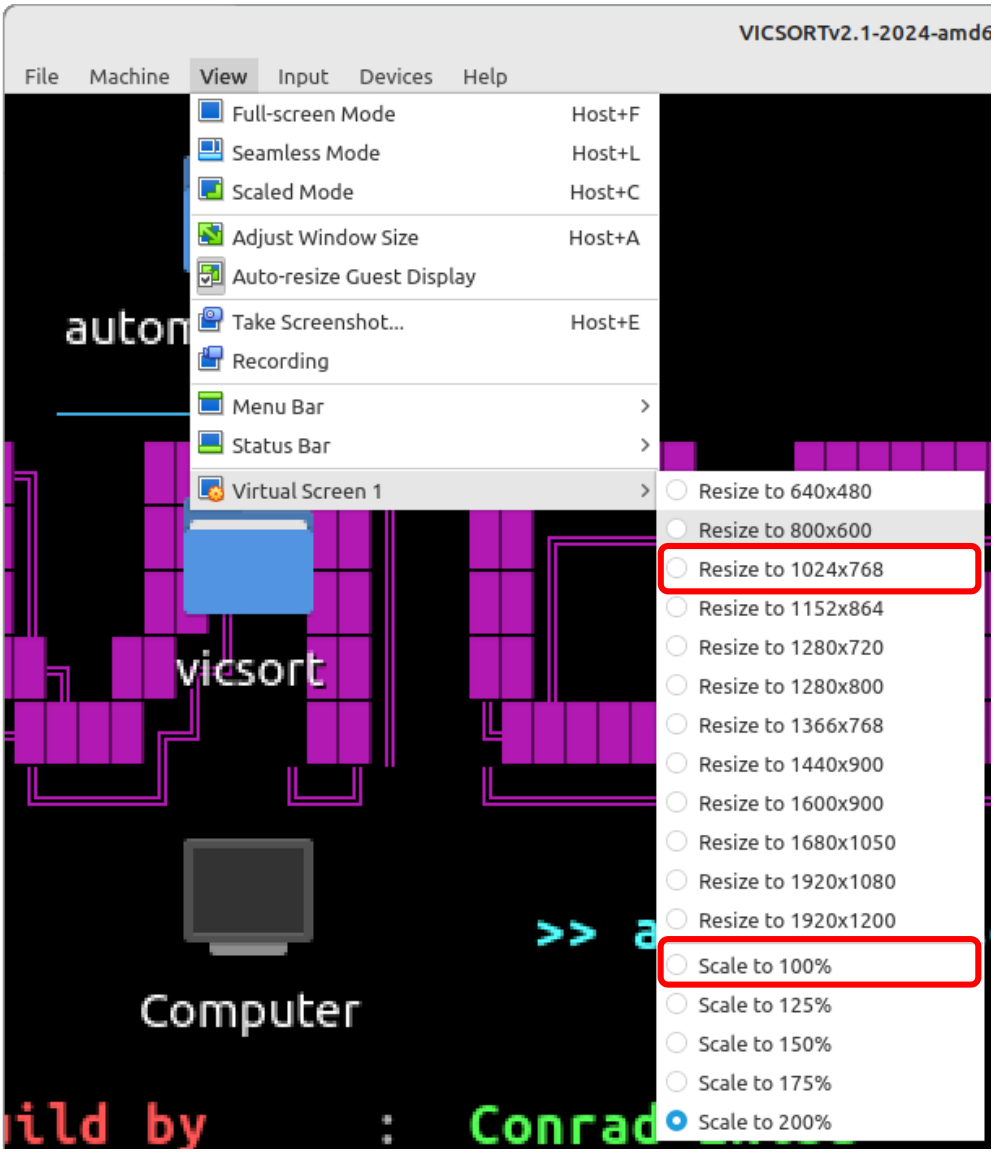


Figure 6: VirtualBox display

3.1 Operating System changes

3.1.1 Set preferred locale settings

Set the locale settings to best match the keyboard in use.

```
vicsort@vicsort:~$ isudo dpkg-reconfigure locales
[sudo] password for vicsort: vicsort
```

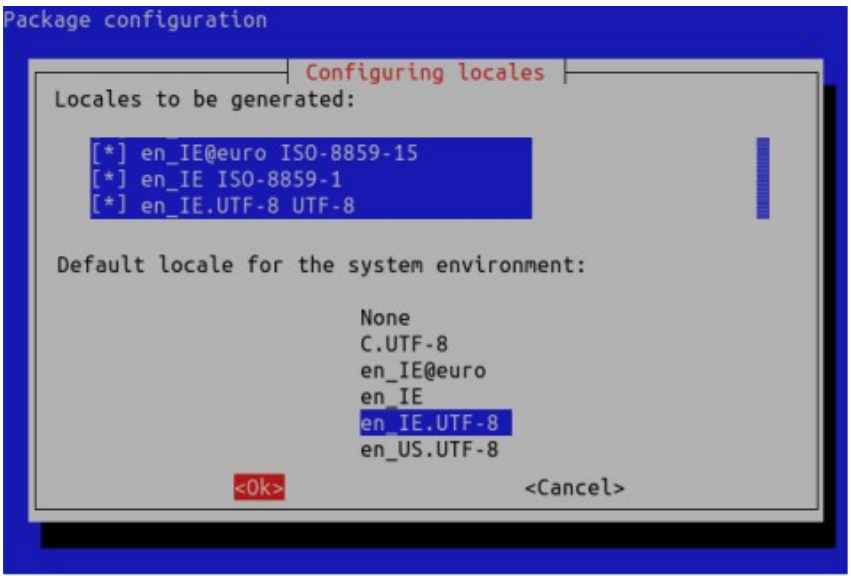


Figure 7: Configure locales

```
Generating locales (this might take a while)...
  en_IE.ISO-8859-1... done
  en_IE.UTF-8... done
  en_IE.ISO-8859-15@euro... done
  en_US.UTF-8... done
Generation complete.
```

3.1.2 Update the operating system

Update the operating system from the Ubuntu repositories.

```
vicsort@vicsort:~$ sudo apt update && sudo apt upgrade
Do you want to continue? [Y/n] Yes
```

3.1.3 Reboot the VM

Reboot to enable the locale and keyboard changes.

```
vicsort@vicsort:~$ sudo reboot now
```

After the reboot has completed.

3.1.4 Set the keyboard layout to GB

```
vicsort@vicsort:~$ sudo dpkg-reconfigure keyboard-configuration
```

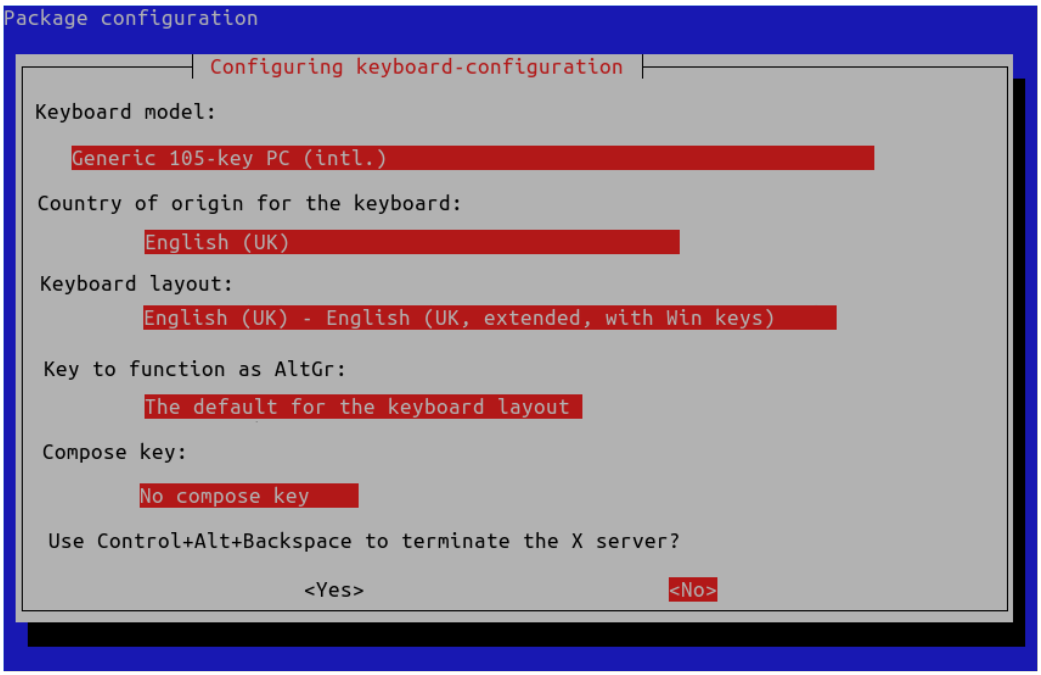


Figure 8: Keyboard layout

To update the LXQt Desktop, as illustrated in Figure 9, from the menu select **`Preferences`** >> **`LXQt Settings`** >> **`Keyboard and Mouse`**. Then select **`Keyboard Layout`** and the preferred layout before selecting **`Apply`**.

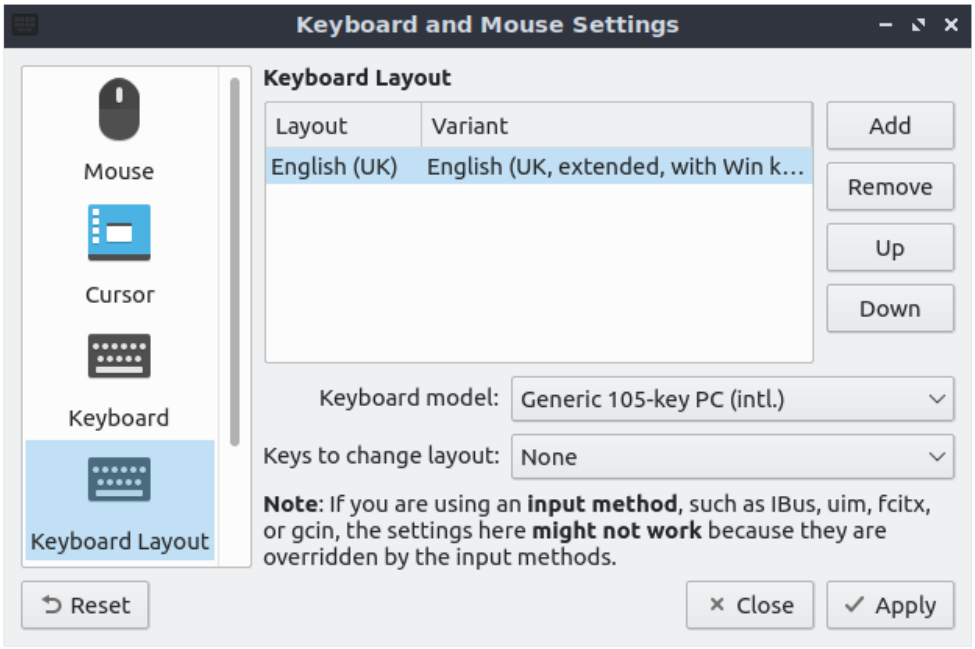


Figure 9: LXQt Keyboard Settings

Open a terminal and select the keys **`Shift-2`**, **`shift-3`** and **`alt gr-4`**, for a English (UK) keyboard layout the characters **`"£€`** should be displayed on the screen.

3.2 Understanding DNS on the VM

The VM uses **systemd-resolve** to resolve domain names and IP addresses. In the example below the VM is connected to an Internet Service Provider (ISP) with the domain **ripple.net** that is automatically picked up from the main network.

Note: In this case the VM Network was set to bridged adapter.

```
vicsort@vicsort:~$ systemd-resolve --status |grep "Current DNS Server"
  Current DNS Server: 89.34.154.5

vicsort@vicsort:~$ dig +short -x 89.34.154.5
limk1-dns02.ripple.net.
```

Test that resolution is working.

```
vicsort@vicsort:~$ sudo apt install fping

vicsort@vicsort:~$ fping www.google.com
www.google.com is alive
```

4 Running the VICSORT testbed

4.1 Start the testbed

With sudo privileges start the testbed and review the LinuX Containers (LXC).

```
vicsort@vicsort:~$ testbed_startup

**** Testbed Ready to go ****

vicsort@vicsort:~$ lxc list
+-----------------------+---------+-----------------------+-----------+-----------+
|         NAME          |  STATE  |         IPV4          |   TYPE    | SNAPSHOTS |
+-----------------------+---------+-----------------------+-----------+-----------+
| attacker-container    | RUNNING | 192.168.90.197 (eth1) | CONTAINER | 0         |
+-----------------------+---------+-----------------------+-----------+-----------+
| hmi-container         | RUNNING | 192.168.90.5 (eth1)   | CONTAINER | 0         |
+-----------------------+---------+-----------------------+-----------+-----------+
| plc-container         | RUNNING | 192.168.95.2 (eth1)   | CONTAINER | 0         |
+-----------------------+---------+-----------------------+-----------+-----------+
| simulation-container  | RUNNING | 192.168.95.15 (eth7)  | CONTAINER | 0         |
|                       |         | 192.168.95.14 (eth6)  |           |           |
|                       |         | 192.168.95.13 (eth5)  |           |           |
|                       |         | 192.168.95.12 (eth4)  |           |           |
|                       |         | 192.168.95.11 (eth3)  |           |           |
|                       |         | 192.168.95.10 (eth2)  |           |           |
+-----------------------+---------+-----------------------+-----------+-----------+
| workstation-container | RUNNING | 192.168.95.5 (eth1)   | CONTAINER | 0         |
+-----------------------+---------+-----------------------+-----------+-----------+
```

4.2 Firewall Rule

The management interfaces for each device can be accessed via the Google Chrome browser and the required tabs should pop up by default. Tabs that require credentials should have them saved and ready for autofill. If not, refer to Figure 3. Once the testbed is started, Internet access is disabled from all containers via the firewall; however, the **`attacker-container`** requires Internet access.

Note: The firewall does not appear in the container table because it is actually a separate VM that can be accessed on the IP address **`192.168.95.100`**.

4.3 Access the pfSense firewall

Access the firewall with the username: **`admin`** and password: **`pfsense`** as illustrated in Figure 10.

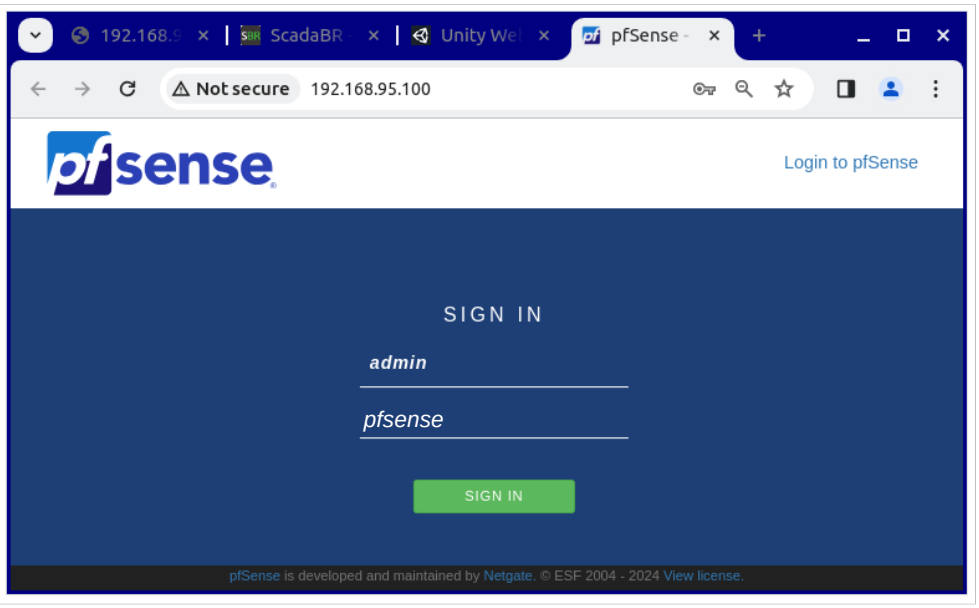


Figure 10: pfSense Firewall

To add Internet access add the following rule to the top of the WAN table. Confirm the following rule exists, and if not added it. This rule will allow the **`attacker-container`** access to the Internet.

Select from the top menu bar **`Firewall`** >> **`Rules`** and add the following entry:

- **Action**: Pass
- **Interface**: WAN
- **Protocol**: Any
- **Source**: single host or alias: **`192.168.90.197`**
- **Description**: Allow attacker-container to access the Internet
- **Advanced Options**:
  - **Gateway**: WANGW - **`192.168.90.1`** - WAN Gateway

The table entry can be visualised in Figure 11, Internet access should now be available on the **`attacker-container`**.

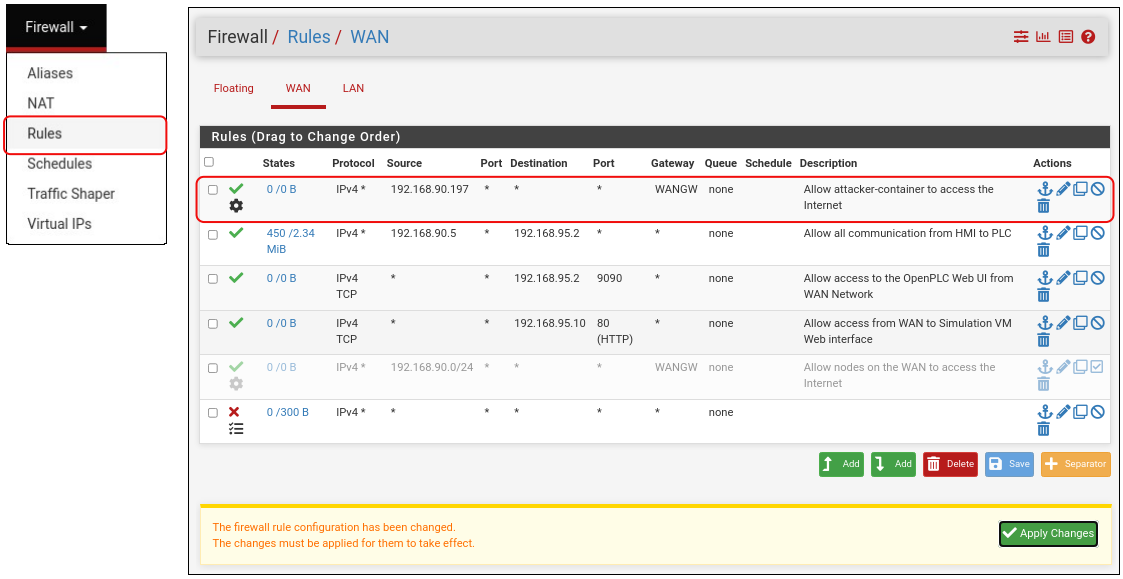


Figure 11: pfSense Firewall Rule

4.4 The attacker-container

DNS on the containers is not managed by **`systemd-resolve`** like on the VM. LXD configures new containers from its DHCP server and as well as performing the gateway function it also acts as the DNS server for each container.

To demonstrate this, connect to the **`attacker-container`**, check the DNS Server and connectivity to the wider network.

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─#

┌──(root💀attacker-container)-[~]
└─# cat /etc/resolv.conf | grep nameserver
nameserver 192.168.90.1

┌──(root💀attacker-container)-[~]
└─# dig +short -x 192.168.90.1
_gateway.lxd.

┌──(root💀attacker-container)-[~]
└─# fping www.google.com
www.google.com is alive
```

4.5 Kali Linux Archive GPG Key

Get the Kali Linux archive key, dearmour it and add it to the keyring. This key is necessary to upgrade from the Kali Linux repository.

```
┌──(root💀attacker-container)-[~]
└─# cd /usr/share/keyrings
```

```
┌──(root💀attacker-container)-[/usr/share/keyrings]
└─# curl https://archive.kali.org/archive-key.asc | gpg --dearmor >
archive-key.gpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3155  100  3155    0     0   7346      0 --:--:-- --:--:-- --:--:--  7354
```

```
┌──(root💀attacker-container)-[/usr/share/keyrings]
└─# cp archive-key.gpg /etc/apt/trusted.gpg.d
```

Confirm the key is a binary file in the correct location.

```
┌──(root💀attacker-container)-[/usr/share/keyrings]
└─# file /etc/apt/trusted.gpg.d/archive-key.gpg
/etc/apt/trusted.gpg.d/archive-key.gpg: OpenPGP Public Key Version 4,
Created Mon Mar  5 14:56:40 2012, RSA (Encrypt or Sign, 4096 bits); User
ID; Signature; OpenPGP Certificate
```

As the shell is a root shell the upgrade of the Kali Linux operating system can be achieved without using **sudo**.

```
┌──(root💀attacker-container)-[~]
└─# apt update && apt upgrade -y
```

5 Reconnaissance

To gain a better understanding of the environment, the attacker begins gathering information about the nodes available on the network. Since the attacker had successfully breached the IT network and is now in the DMZ, the focus is on gathering information about the nodes visible to the attacker container.

Footprinting is the process of collecting as much information as possible about a target system or network. The objective of footprinting is to obtain specific details about the target, such as its operating systems, the service versions of running applications, and any other relevant network information. The information collected during footprinting can be used in various ways to gain further access to the target system, network, or organisation.

To passively monitor network activity, the attacker launched **`wireshark`** on the **`attacker-container`** using Remote Desktop Protocol (RDP). This allows for the monitoring of network traffic and the identification of any potential vulnerabilities or points of entry.

5.1 XRDP server on the attacher-container

As illustrated in Figure 12, access the X Remote Desktop Protocol (XRDP) desktop of the **`attacker-container`**.

```
vicsort@vicsort:~$ rdp_attacker
[1] 25887
```

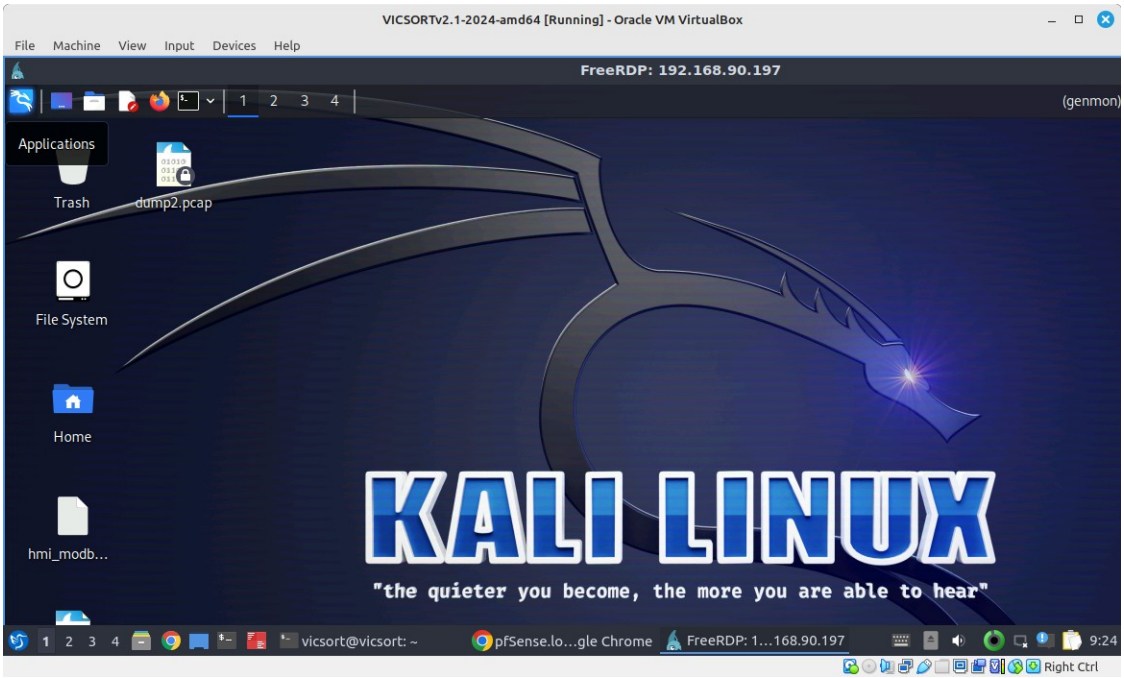


Figure 12: XRDP attacker-container Desktop

5.2 Wireshark

Wireshark is a Graphical User Interface (GUI) network protocol analyser. It facilitates interactive browsing of packet data from a live network or from a previously saved capture file. Wireshark's native capture file formats are **`pcapng`** format and **`pcap`** format; it can read and write both formats. **`pcap`** format is also the format used by **`tcpdump`** and various other tools; **`tcpdump`**, when using newer versions of the **`libpcap`** library, can also read some **`pcapng`** files.

Open a terminal in the XRDP **`attacker-container`** window and the following command will open **`wireshark`** in the **`attacker-container`**.

```
kali@attacker-container:~$ sudo wireshark
```

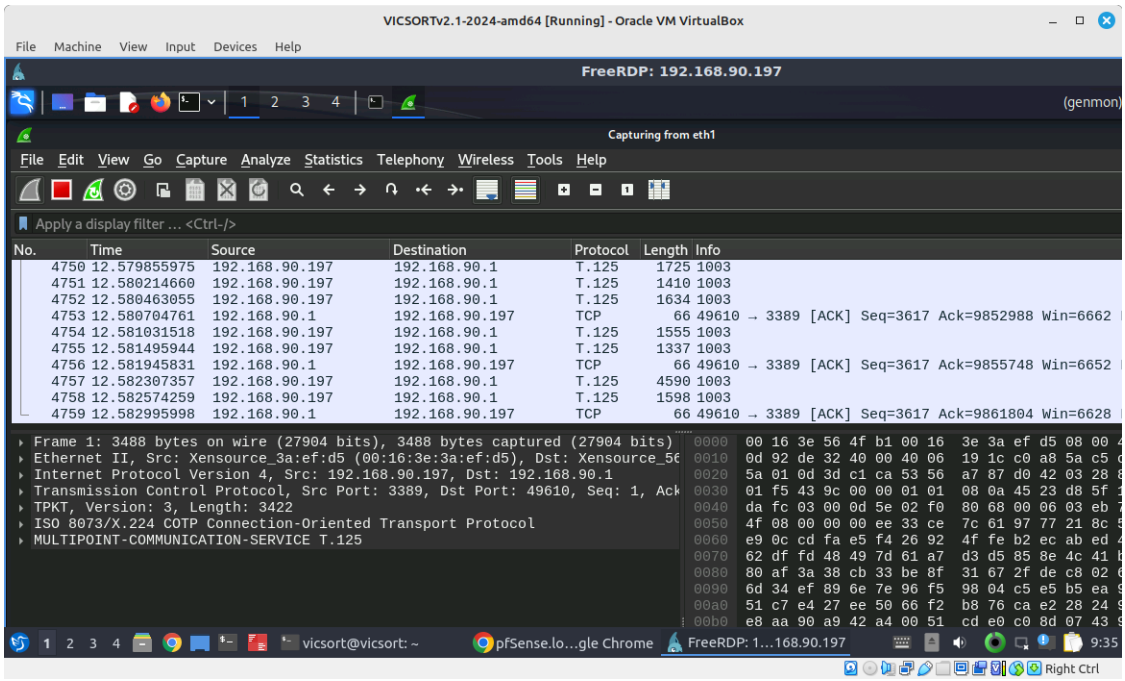


Figure 13: Wireshark in the attacker-container

5.3 Tshark

tshark is a network protocol analyser that facilitates the capture of packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. **tshark** native capture file format is **pcapng** format, which is also the format used by Wireshark and various other tools.

Determine the interfaces that are available to **tshark**.

```
┌──(root💀attacker-container)-[~]
└─# tshark -D
Running as user "root" and group "root". This could be dangerous.
1. eth1
2. any
3. lo (Loopback)
4. bluetooth-monitor
5. nflog
6. nfqueue
7. dbus-system
8. dbus-session
9. ciscodump (Cisco remote capture)
10. dpauxmon (DisplayPort AUX channel monitor capture)
11. randpkt (Random packet generator)
12. sdjournal (systemd Journal Export)
13. sshdump (SSH remote capture)
14. udpdump (UDP Listener remote capture)
```

```
┌──(root💀attacker-container)-[~]
└─# tshark -F pcap -V > /root/tshark_out.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth1'
 ** (tshark:2274) 12:56:46.982335 [Main MESSAGE] -- Capture started.
   **   (tshark:2274)   12:56:46.982394   [Main   MESSAGE]   --   File:
"/tmp/wireshark_eth1H1V7G2.pcapng"
```

When the process was stopped after a few seconds there was over 10,500 frames captured. Have a look into the first frame captured within the file using the head command.

```
┌──(root💀attacker-container)-[~]
└─# head -94 /root/tshark_out.pcap
Frame 1: 4875 bytes on wire (39000 bits), 4875 bytes captured (39000 bits) on
interface eth1, id 0
    Interface id: 0 (eth1)
        Interface name: eth1
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan  3, 2024 12:56:46.986995902 GMT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1704286606.986995902 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 4875 bytes (39000 bits)
    Capture Length: 4875 bytes (39000 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:tls]
Ethernet II, Src: 00:16:3e:3a:ef:d5 (00:16:3e:3a:ef:d5), Dst: 00:16:3e:56:4f:b1
(00:16:3e:56:4f:b1)
    Destination: 00:16:3e:56:4f:b1 (00:16:3e:56:4f:b1)
        Address: 00:16:3e:56:4f:b1 (00:16:3e:56:4f:b1)
```

```
          .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
          .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 00:16:3e:3a:ef:d5 (00:16:3e:3a:ef:d5)
        Address: 00:16:3e:3a:ef:d5 (00:16:3e:3a:ef:d5)
          .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
          .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.90.197, Dst: 192.168.90.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
          .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport
(0)
    Total Length: 4861
    Identification: 0x6e8f (28303)
    Flags: 0x40, Don't fragment
        0... .... = Reserved bit: Not set
        .1.. .... = Don't fragment: Set
        ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x8354 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.90.197
    Destination Address: 192.168.90.1
Transmission Control Protocol, Src Port: 3389, Dst Port: 41172, Seq: 1, Ack: 1,
Len: 4809
    Source Port: 3389
    Destination Port: 41172
    [Stream index: 0]
    [Conversation completeness: Incomplete (0)]
    [TCP Segment Len: 4809]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2241397658
    [Next Sequence Number: 4810    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2731214310
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x018 (PSH, ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 1... = Push: Set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·····AP···]
    Window: 685
    [Calculated window size: 685]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x4907 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
        TCP Option - No-Operation (NOP)
            Kind: No-Operation (1)
        TCP Option - No-Operation (NOP)
            Kind: No-Operation (1)
        TCP Option - Timestamps: TSval 303084322, TSecr 2891965769
            Kind: Time Stamp Option (8)
            Length: 10
            Timestamp value: 303084322
            Timestamp echo reply: 2891965769
    [Timestamps]
        [Time since first frame in this TCP stream: 0.000000000 seconds]
        [Time since previous frame in this TCP stream: 0.000000000 seconds]
    [SEQ/ACK analysis]
        [Bytes in flight: 4809]
        [Bytes sent since last PSH flag: 4809]
    TCP payload (4809 bytes)
Transport Layer Security
```

5.4 Netdiscover

netdiscover is an active/passive ARP reconnaissance tool, It was built upon **libnet** and **libpcap**, it can passively detect online hosts or search for them by sending ARP requests. Additionally, it can be used to inspect a network's ARP traffic, or find network addresses using auto scan mode, which will scan for common local networks.

```
┌──(root💀attacker-container)-[~]
└─# netdiscover -i eth1 -r 192.168.90.0/24
Currently scanning: Finished!   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 126
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.90.1    00:16:3e:56:4f:b1      1      42  Xensource, Inc.
192.168.90.5    00:16:3e:63:0d:8b      1      42  Xensource, Inc.
192.168.90.100  52:54:00:d7:8f:bd      1      42  Unknown vendor
```

5.5 p0f

p0f is a passive fingerprinting technique that identifies remote systems, based on analysis of the structure of a TCP/IP packets to determine the operating system and other configuration properties of a remote host. The process is completely passive and does not generate any suspicious network traffic. Identified hosts are either:

- **Connected to the network** - either spontaneously or in an induced manner, for example when trying to establish a ftp data stream, returning a bounced mail, performing auth lookup, using IRC DCC, external html mail image reference, etc..
- **Is contacted by some entity on the network** - using some standard means (such as a web browsing); it can either accept or refuse the connection.

The method can see through packet firewalls and does not have the restrictions of an active fingerprinting. The main uses of passive operating system fingerprinting are attacker profiling (IDS and honeypots), visitor profiling (content optimisation), customer/user profiling (policy enforcement), pen-testing, etc..

Run the **p0f** server to monitor the Ethernet interface and output results to a file. It runs in daemon mode in the background.

- **-i**: Interface
- **-d**: Daemon mode, Fork in the background
- **-o**: Output file

Install **p0f**.

```
┌──(root💀attacker-container)-[~]
└─# apt install p0f
```

```
┌──(root💀attacker-container)-[~]
└─# p0f -i eth1 -d -o /root/p0f-output.txt
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---

[!] Consider specifying -u in daemon mode (see README).
[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on interface 'eth1'.
[+] Default packet filtering configured [+VLAN].
[+] Log file '/root/p0f-output.txt' opened for writing.
[+] Daemon process created, PID 2882 (stderr not kept).

Good luck, you're on your own now!
```

Confirm the daemon is running and note the Process IDentifier (PID), in this case 2882.

```
┌──(root💀attacker-container)-[~]
└─# ps -ef | grep p0f
root        2882       1  0 16:35 ?        00:00:00 p0f -i eth1 -d -o
/root/p0f-output.txt
root        2891    1376  0 16:38 pts/1    00:00:00 grep --color=auto p0f
```

Monitor activity in the **pof-output.txt** file.

```
┌──(root💀attacker-container)-[~]
└─# tail -f /root/p0f-output.txt
[2024/01/05 16:42:18] mod=syn|cli=192.168.90.197/52988|
srv=209.85.202.95/443|subj=cli|os=Linux 2.2.x-3.x|dist=0|
params=generic|
raw_sig=4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
[2024/01/05 16:42:18] mod=mtu|cli=192.168.90.197/52988|
srv=209.85.202.95/443|subj=cli|link=Ethernet or modem|raw_mtu=1500
[2024/01/05 16:42:18] mod=syn+ack|cli=192.168.90.197/52988|
srv=209.85.202.95/443|subj=srv|os=???|dist=7|params=none|
raw_sig=4:121+7:0:1412:65535,8:mss,sok,ts,nop,ws:df:0
[2024/01/05 16:42:18] mod=mtu|cli=192.168.90.197/52988|
srv=209.85.202.95/443|subj=srv|link=DSL|raw_mtu=1452
[2024/01/05 16:42:18] mod=uptime|cli=192.168.90.197/52988|
srv=209.85.202.95/443|subj=cli|uptime=23 days 5 hrs 32 min (modulo 49
days)|raw_freq=964.29 Hz
[2024/01/05 16:42:35] mod=syn|cli=192.168.90.197/57430|
srv=34.107.243.93/443|subj=cli|os=Linux 2.2.x-3.x|dist=0|
params=generic|
raw_sig=4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
[2024/01/05 16:42:35] mod=mtu|cli=192.168.90.197/57430|
srv=34.107.243.93/443|subj=cli|link=Ethernet or modem|raw_mtu=1500
[2024/01/05 16:42:35] mod=syn+ack|cli=192.168.90.197/57430|
srv=34.107.243.93/443|subj=srv|os=???|dist=7|params=none|
raw_sig=4:121+7:0:1412:65535,8:mss,sok,ts,nop,ws:df:0
[2024/01/05 16:42:35] mod=mtu|cli=192.168.90.197/57430|
srv=34.107.243.93/443|subj=srv|link=DSL|raw_mtu=1452
[2024/01/05 16:42:36] mod=uptime|cli=192.168.90.197/57430|
srv=34.107.243.93/443|subj=cli|uptime=25 days 5 hrs 39 min (modulo 49
days)|raw_freq=960.00 Hz
[2024/01/05 16:42:36] mod=syn|cli=192.168.90.197/57436|
srv=34.107.243.93/443|subj=cli|os=Linux 2.2.x-3.x|dist=0|
params=generic|
raw_sig=4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
[2024/01/05 16:42:36] mod=mtu|cli=192.168.90.197/57436|
srv=34.107.243.93/443|subj=cli|link=Ethernet or modem|raw_mtu=1500
[2024/01/05 16:42:36] mod=uptime|cli=192.168.90.197/57436|
srv=34.107.243.93/443|subj=cli|uptime=25 days 5 hrs 39 min (modulo 49
days)|raw_freq=1000.00 Hz
```

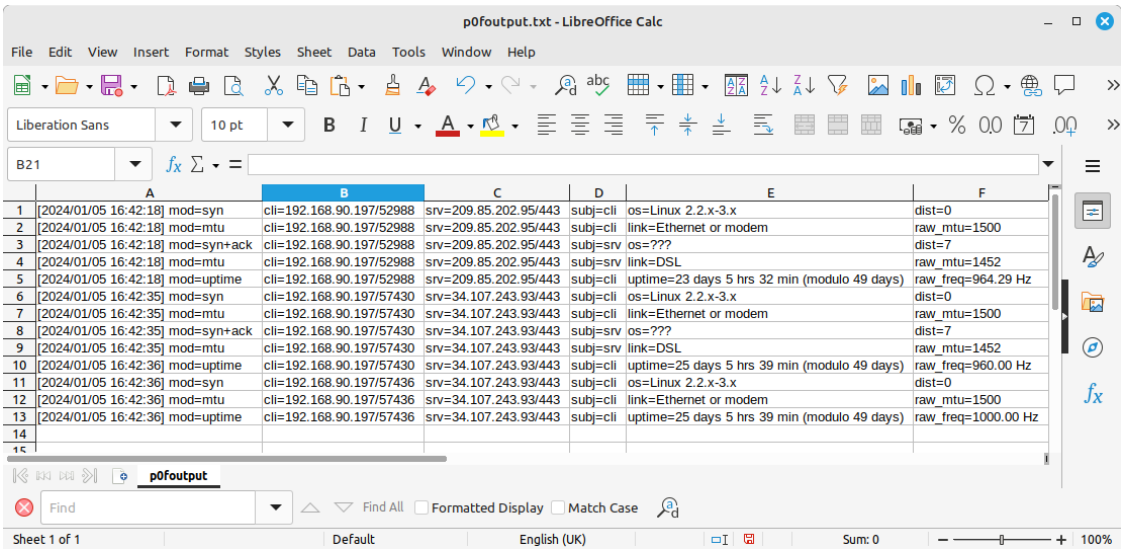
Terminate the **p0f** daemon when it is no longer required.

```
┌──(root💀attacker-container)-[~]
└─# kill -SIGKILL 2882

┌──(root💀attacker-container)-[~]
└─# ps -ef | grep p0f

┌──(root💀attacker-container)-[~]
└─# ps -ef | grep p0f
root         3013    1376  0 16:50 pts/1    00:00:00 grep --color=auto p0f
```

The output is easily imported into a spreadsheet using pipe (|) as a delimiter.



	A	B	C	D	E	F
1	[2024/01/05 16:42:18] mod=syn	cli=192.168.90.197/52988	srv=209.85.202.95/443	subj=cli	os=Linux 2.2.x-3.x	dist=0
2	[2024/01/05 16:42:18] mod=mtu	cli=192.168.90.197/52988	srv=209.85.202.95/443	subj=cli	link=Ethernet or modem	raw_mtu=1500
3	[2024/01/05 16:42:18] mod=syn+ack	cli=192.168.90.197/52988	srv=209.85.202.95/443	subj=srv	os=???	dist=7
4	[2024/01/05 16:42:18] mod=mtu	cli=192.168.90.197/52988	srv=209.85.202.95/443	subj=srv	link=DSL	raw_mtu=1452
5	[2024/01/05 16:42:18] mod=uptime	cli=192.168.90.197/52988	srv=209.85.202.95/443	subj=cli	uptime=23 days 5 hrs 32 min (modulo 49 days)	raw_freq=964.29 Hz
6	[2024/01/05 16:42:35] mod=syn	cli=192.168.90.197/57430	srv=34.107.243.93/443	subj=cli	os=Linux 2.2.x-3.x	dist=0
7	[2024/01/05 16:42:35] mod=mtu	cli=192.168.90.197/57430	srv=34.107.243.93/443	subj=cli	link=Ethernet or modem	raw_mtu=1500
8	[2024/01/05 16:42:35] mod=syn+ack	cli=192.168.90.197/57430	srv=34.107.243.93/443	subj=srv	os=???	dist=7
9	[2024/01/05 16:42:35] mod=mtu	cli=192.168.90.197/57430	srv=34.107.243.93/443	subj=srv	link=DSL	raw_mtu=1452
10	[2024/01/05 16:42:36] mod=uptime	cli=192.168.90.197/57430	srv=34.107.243.93/443	subj=cli	uptime=25 days 5 hrs 39 min (modulo 49 days)	raw_freq=960.00 Hz
11	[2024/01/05 16:42:36] mod=syn	cli=192.168.90.197/57436	srv=34.107.243.93/443	subj=cli	os=Linux 2.2.x-3.x	dist=0
12	[2024/01/05 16:42:36] mod=mtu	cli=192.168.90.197/57436	srv=34.107.243.93/443	subj=cli	link=Ethernet or modem	raw_mtu=1500
13	[2024/01/05 16:42:36] mod=uptime	cli=192.168.90.197/57436	srv=34.107.243.93/443	subj=cli	uptime=25 days 5 hrs 39 min (modulo 49 days)	raw_freq=1000.00 Hz
14						

Figure 14: p0f imported into a spreadsheet with pipe (|) as a delimiter

5.6 Nmap

Network Mapper (**nmap**) is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. **nmap** uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While **nmap** is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Using **nmap** scan the hostspace on the 192.168.90.0/24 network, removing the **attacker-container** computer itself.

```
┌──(root💀attacker-container)-[~]
└─# nmap -sn 192.168.90.0/24 --exclude 192.168.90.197
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-05 17:50 GMT
Nmap scan report for _gateway.lxd (192.168.90.1)
Host is up (0.000084s latency).
MAC Address: 00:16:3E:56:4F:B1 (Xensource)
Nmap scan report for hmi-container.lxd (192.168.90.5)
Host is up (0.000044s latency).
MAC Address: 00:16:3E:63:0D:8B (Xensource)
Nmap scan report for 192.168.90.100
Host is up (0.0017s latency).
MAC Address: 52:54:00:D7:8F:BD (QEMU virtual NIC)
Nmap done: 255 IP addresses (3 hosts up) scanned in 2.02 seconds
```

Host 192.168.90.5 is a HMI. After discovering the hosts on a network, the next phase is to identify any open service ports on the target system and determine which services are mapped to those open ports.

```
┌──(root💀attacker-container)-[~]
└─# nmap -v -sn 192.168.90.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-05 17:54 GMT
Initiating ARP Ping Scan at 17:54
Scanning 192.168.90.5 [1 port]
Completed ARP Ping Scan at 17:54, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:54
Completed Parallel DNS resolution of 1 host. at 17:54, 0.00s elapsed
Nmap scan report for hmi-container.lxd (192.168.90.5)
Host is up (0.000054s latency).
MAC Address: 00:16:3E:63:0D:8B (Xensource)
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
           Raw packets sent: 1 (28B) | Rcvd: 1 (28B)
```

Target the HMI with a specific scan.

- **–A**: This enables **nmap** to profile the target to identify its operating system, service versions, and script scanning, as well as perform a traceroute.
- **-T**: This syntax specifies the timing options for the scan, which ranges from 0 – 5, where 0 is very slow and 5 is the fastest. This command is good for preventing too many probes from being sent to the target too quickly.
- **-p**: Specify which port(s) to identify as opened or closed on a target. For example specifying **–p80** to scan for port 80 only on the target and **–p–** to scan for all 65,535 open ports on a target.

```
┌──(root💀attacker-container)-[~]
└─# nmap -A -T4 -p- 192.168.90.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-05 12:52 GMT
Nmap scan report for hmi-container.lxd (192.168.90.5)
Host is up (0.000093s latency).

Not shown: 65533 closed tcp ports (reset)

PORT     STATE SERVICE VERSION

8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
| ajp-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
|   Potentially risky methods: PUT DELETE
|_  See https://nmap.org/nsedoc/scripts/ajp-methods.html
9090/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
| http-methods:
|_  Potentially risky methods: PUT DELETE
MAC Address: 00:16:3E:63:0D:8B (Xensource)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.09 ms hmi-container.lxd (192.168.90.5)

OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.97 seconds
```

Figure 15: Port 9090 on HMI host

The **nmap** command, illustrated in Figure 15, illustrates that there is an **Apache Tomcat** server running on port **9090** on this HMI host. The HMI web GUI must be hosted here. The scan also reveals that the host is also running a GNU/Linux operating system.

Browse to the open port, as illustrated in Figure 16, confirms that an **Apache Tomcat** webserver is running.

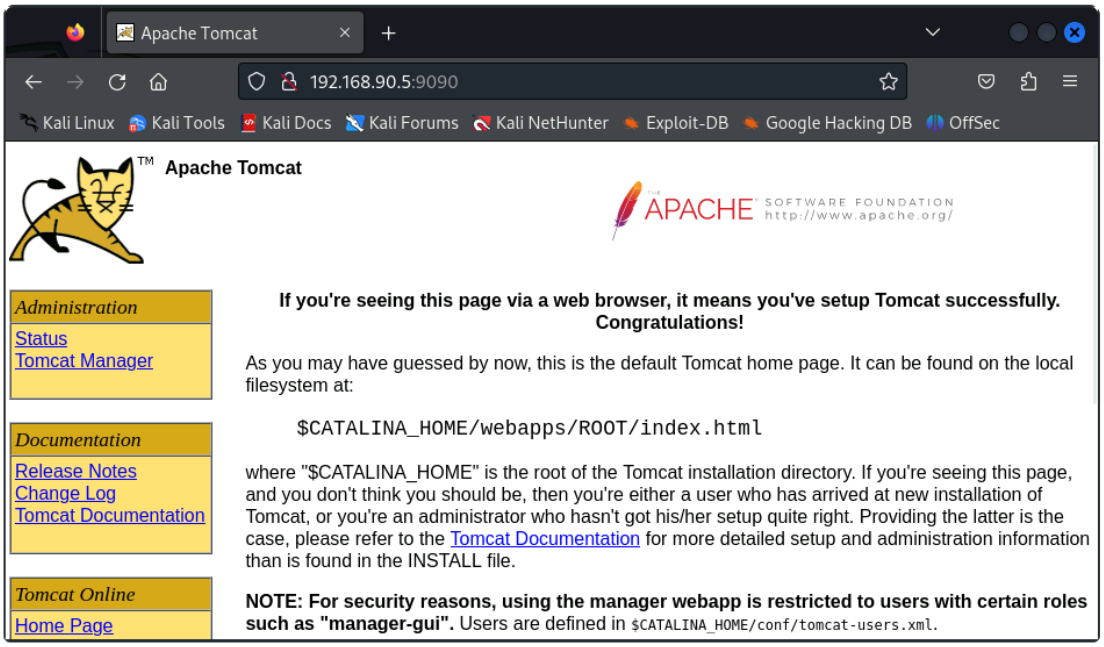


Figure 16: Apache Tomcat running on host

5.7 Nikto

This is a shell utility to scan web servers for known vulnerabilities. With **nikto** a web server can be examined for potential problems and security vulnerabilities, including:

- Server and software misconfigurations
- Default files and programs
- Insecure files and programs
- Outdated servers and programs.

nikto is built on **LibWhisker** (by RFP) and can run on any platform which has a Perl environment. It supports SSL, proxies, host authentication, IDS evasion and more.

5.7.1 Install and update Nikto

Install **nikto** and before use it is important to update the plugins and databases directly from **cirt.net**.

```
┌──(root💀attacker-container)-[~]
└─# apt purge nikto

┌──(root💀attacker-container)-[~]
└─# apt install nikto
```

5.7.2 Running Nikto

Run **nikto** against the Apache Tomcat webserver host.

```
┌──(root💀attacker-container)-[~]
└─# nikto -host 192.168.90.5 -port 9090
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.90.5
+ Target Hostname:    192.168.90.5
+ Target Port:        9090
+ Start Time:         2024-03-19 19:06:46 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache-Coyote/1.1
+ /: The anti-clickjacking X-Frame-Options header is not present. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type.                                                                See:
https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-
content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /favicon.ico: identifies this app/server as: Apache Tomcat (possibly 5.5.26
through          8.0.15),          Alfresco          Community.          See:
https://en.wikipedia.org/wiki/Favicon
+ Multiple index files found: /index.jsp, /index.html.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save
files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on
the web server.
+ /: Appears to be a default Apache Tomcat install.
+ /examples/servlets/index.html: Apache Tomcat default JSP pages present.
+ /examples/jsp/snp/snoop.jsp: Cookie JSESSIONID created without the httponly
flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /examples/jsp/snp/snoop.jsp: Displays information about page retrievals,
including other users. See: http://cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-2004-2104
+ /manager/html: The detailed Tomcat version is disclosed in error pages.
+ /manager/html: Default Tomcat Manager / Host Manager interface found.
+ /host-manager/html: The detailed Tomcat version is disclosed in error
pages.
+ /host-manager/html: Default Tomcat Manager / Host Manager interface found.
+ /manager/status: The detailed Tomcat version is disclosed in error pages.
+ /manager/status: Default Tomcat Server Status interface found.
+ 8406 requests: 0 error(s) and 17 item(s) reported on remote host
+ End Time:           2024-03-19 19:07:06 (GMT0) (20 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

6 Metasploit Framework

Metasploit is a penetration testing framework from Rapid7 that enables a pen tester to find, exploit, and validate vulnerabilities. It is known for its robust penetration testing and vulnerability assessment capabilities. Key characteristics of the **metasploit** Framework are:

- **Comprehensive Testing**: **metasploit** provides extensive options for penetration testing, helping identify vulnerabilities in systems and networks.
- **Exploit Development**: It aids in developing and testing exploits for identified vulnerabilities, enhancing system security.
- **Payload Crafting**: Users can create payloads to gain control over compromised systems, providing a deeper understanding of potential threats.
- **Post-Exploitation Tools**: **metasploit** includes tools for extracting valuable data and maintaining access after a successful breach.
- **Network Analysis**: It offers capabilities to analyse network structures and identify potential entry points for securing the network.

6.1 Run the Metasploit Framework and Console

To get started with **metasploit** install the **metasploit-framework**.

```
┌──(root💀attacker-container)-[~]
└─# apt update; apt install metasploit-framework
```

The **msfdb** tool facilitates the management of the metasploit framework database. **init** initialises a new database. The status can be seen using the **msfdb status** command which essentially runs the command **systemctl status postgresql**.

```
┌──(root💀attacker-container)-[~]
└─# msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file
'/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

6.2 Check Postgresql database

```
┌──(root💀attacker-container)-[~]
└─# msfdb status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service;
enabled; preset: disabled)
    Drop-In: /run/systemd/system/service.d
             └─zzz-lxc-service.conf
     Active: active (exited) since Sun 2024-04-21 22:05:49 IST; 1min
3s ago
    Process: 2484 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 2484 (code=exited, status=0/SUCCESS)

Apr 21 22:05:49 attacker-container systemd[1]: Starting
postgresql.service - PostgreSQL RDBMS...
Apr 21 22:05:49 attacker-container systemd[1]: Finished
postgresql.service - PostgreSQL RDBMS.

COMMAND   PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
postgres 2447 postgres    5u  IPv6 121586      0t0  TCP
localhost:5432 (LISTEN)
postgres 2447 postgres    6u  IPv4 121587      0t0  TCP
localhost:5432 (LISTEN)

UID          PID    PPID  C STIME TTY      STAT   TIME CMD
postgres    2447       1  0 22:05 ?        Ss     0:00
/usr/lib/postgresql/14/bin/postgres -D /var/li

[+] Detected configuration file
(/usr/share/metasploit-framework/config/database.yml)


┌──(root💀attacker-container)-[~]
└─# msfconsole
```

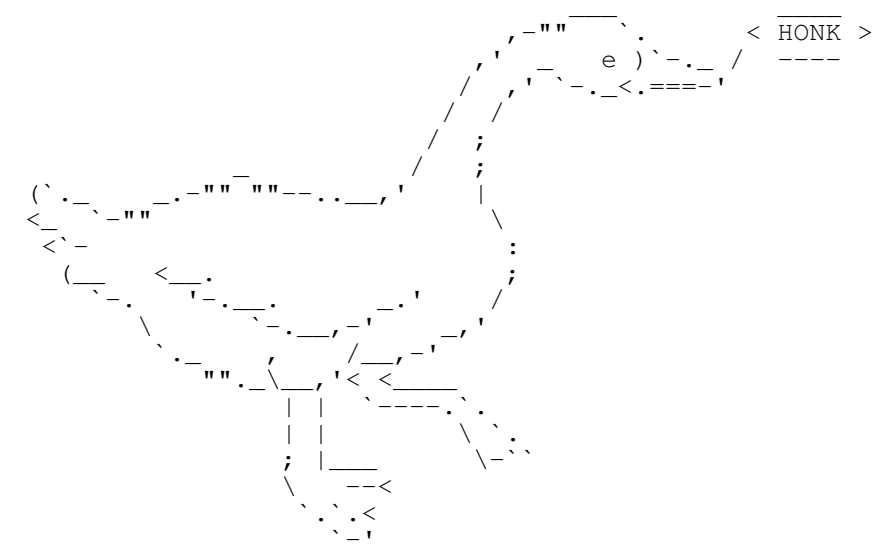
During this setup, several prompts maybe encountered, particularly the first time it is run:

```
Would you like to use and setup a new database (recommended)? Yes

Would you like to init the webservice? (Not Required) [no]: no
```

After addressing these prompts, the system will take a few minutes to finalise configurations. Upon completion, your **metasploit** Framework is ready for use.

```
Metasploit tip: When in a module, use back to go back to the top
level prompt
```




```
       =[ metasploit v6.3.55-dev                          ]
+ -- --=[ 2397 exploits - 1235 auxiliary - 422 post       ]
+ -- --=[ 1388 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

[*] Starting persistent handler(s)…

msf6 >
```

6.3 Keeping Metasploit Updated

The **metasploit** Framework is regularly enhanced with new modules, features, and fixes. To ensure the latest version is being use update it as follows:

```
┌──(root💀attacker-container)-[~]
└─# apt update; apt install metasploit-framework
```

This command fetches and installs the most recent iteration of the **metasploit** Framework.

6.4 Using nmap within Metasploit for reconnaissance

nmap exists as a module within **metasploit**, use it to get a list of IP addresses on the network. Note that this command will take some time, go for a tea break perhaps?

-Pn: Treat all hosts as online -- skip host discovery.

-sS: Use the TCP SYN scan technique.

-A: Enable OS detection, version detection, script scanning, and traceroute.

-oX netscan: Output as eXtensible Markup Language (XML) to the file **netscan**.

```
msf6 > nmap -Pn -sS -A -oX netscan 192.168.90.0/24 --exclude 192.168.90.197
[*] exec: nmap -Pn -sS -A -oX netscan 192.168.90.0/24 --exclude
192.168.90.197

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-28 11:01 GMT
Nmap scan report for 192.168.90.1
Host is up (0.00012s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 00:ea:24:9b:d9:e1:47:42:af:1b:60:2f:f2:26:f1:3e (RSA)
|   256 91:24:20:29:d9:04:0a:90:51:e2:fe:90:07:cb:e0:18 (ECDSA)
|_  256 63:6f:76:b5:66:3f:e3:b7:0d:15:87:ab:a8:03:a9:6f (ED25519)
53/tcp   open  domain        dnsmasq 2.80
| dns-nsid:
|_  bind.version: dnsmasq-2.80
3389/tcp open  ms-wbt-server xrdp
MAC Address: 00:16:3E:56:4F:B1 (Xensource)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.12 ms 192.168.90.1

Nmap scan report for hmi-container.lxd (192.168.90.5)
Host is up (0.000047s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
| ajp-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
|   Potentially risky methods: PUT DELETE
|_  See https://nmap.org/nsedoc/scripts/ajp-methods.html
9090/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-favicon: Apache Tomcat
|_http-title: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
MAC Address: 00:16:3E:63:0D:8B (Xensource)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.05 ms hmi-container.lxd (192.168.90.5)
```

```
Nmap scan report for 192.168.90.100
Host is up (0.0024s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9 (protocol 2.0)
53/tcp open  domain  (generic dns response: REFUSED)
80/tcp open  http    nginx
|_http-title: pfSense - Login
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.94SVN%I=7%D=2/28%Time=65DF1288%P=x86_64-pc-linux-gnu%r(D
SF:NSVersionBindReqTCP,E,"\0\x0c\0\x06\x81\x05\0\0\0\0\0\0\0");
MAC Address: 52:54:00:D7:8F:BD (QEMU virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least
1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): FreeBSD 11.X (97%)
OS CPE: cpe:/o:freebsd:freebsd:11.2
Aggressive OS guesses: FreeBSD 11.2-RELEASE (97%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   2.38 ms 192.168.90.100

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 255 IP addresses (3 hosts up) scanned in 151.30 seconds
msf6 >
```

Import the retrieved data, in the XML file, into **metasploit**.

```
msf6 > db_import netscan
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.13.10'
[*] Importing host 192.168.90.1
[*] Importing host 192.168.90.5
[*] Importing host 192.168.90.100
[*] Successfully imported /root/netscan

msf6 > hosts

Hosts
=====

address         mac                name          os_name  os_flavor  os_sp  purpose
-------         ---                ----          -------  ---------  -----  -------
66.96.161.141                                    Unknown             device
192.168.90.1    00:16:3e:56:4f:b1                Linux               4.X    server
192.168.90.5    00:16:3e:63:0d:8b  192.168.90.5  Linux               4.X    server
192.168.90.100  52:54:00:d7:8f:bd                FreeBSD             11.X   device
192.168.95.2                                     Unknown             device
```

This list can easily be output as a **csv** file to the computer.

```
msf6 > hosts -o /root/scanned_hosts.csv
[*] Wrote hosts to /root/scanned_hosts.csv

msf6 > exit

┌──(root💀attacker-container)-[~]
└─# cat scanned_hosts.csv
address,mac,name,os_name,os_flavor,os_sp,purpose,info,comments
"66.96.161.141","","","Unknown","","","device","",""
"192.168.90.1","00:16:3e:56:4f:b1","","Linux","","4.X","server","",""
"192.168.90.5","00:16:3e:63:0d:8b","192.168.90.5","Linux","","4.X","server","
",""
"192.168.90.100","52:54:00:d7:8f:bd","","FreeBSD","","11.X","device","",""
"192.168.95.2","","","Unknown","","","device","",""
```

Get the services that are running on the network.

```
msf6 > services
Services
========

host            port  proto  name           state  info
----            ----  -----  ----           -----  ----
66.96.161.141   80    tcp    http           open
192.168.90.1    22    tcp    ssh            open   OpenSSH 8.2p1 4ubuntu0.11 Linux; 2.0
192.168.90.1    53    tcp    domain         open   dnsmasq 2.80
192.168.90.1    3389  tcp    ms-wbt-server  open   xrdp
192.168.90.5    8009  tcp    ajp13          open   Apache Jserv Protocol v1.3
192.168.90.5    9090  tcp    http           open   Apache Tomcat/Coyote JSP engine 1.1
192.168.90.100  22    tcp    ssh            open   OpenSSH 7.9 protocol 2.0
192.168.90.100  53    tcp    domain         open   generic dns response: REFUSED
192.168.90.100  80    tcp    http           open   nginx
192.168.95.2    22    tcp                   open
192.168.95.2    502   tcp                   open
192.168.95.2    8080  tcp                   open


msf6 > hosts -o /root/scanned_services.csv
[*] Wrote hosts to /root/scanned_services.csv

msf6 > cat scanned_services.csv
host,port,proto,name,state,info
"66.96.161.141","80","tcp","http","open",""
"192.168.90.1","22","tcp","ssh","open","OpenSSH 8.2p1 Ubuntu
4ubuntu0.11 Ubuntu Linux; protocol 2.0"
"192.168.90.1","53","tcp","domain","open","dnsmasq 2.80"
"192.168.90.1","3389","tcp","ms-wbt-server","open","xrdp"
"192.168.90.5","80","tcp","http","open",""
"192.168.90.5","8009","tcp","ajp13","open","Apache Jserv Protocol
v1.3"
"192.168.90.5","9090","tcp","http","open","Apache Tomcat/Coyote JSP
engine 1.1"
"192.168.90.100","22","tcp","ssh","open","OpenSSH 7.9 protocol 2.0"
"192.168.90.100","53","tcp","domain","open","generic dns response:
REFUSED"
"192.168.90.100","80","tcp","http","open","nginx"
"192.168.95.2","22","tcp","","open",""
"192.168.95.2","502","tcp","","open",""
"192.168.95.2","8080","tcp","","open",""
```

6.5 Searching for Modules

A core functionality of the **metasploit** Framework is its extension via modules. To hunt for specific modules use the command format:

```
msf6 > search <search-term>
```

Replace **<search-term>** with relevant keywords or terms. For instance, to find exploits associated with port scanning:

```
msf6 > search portscan
```

This returns a list of modules linked to the Port Scanning activity. For example:

```
Matching Modules
================

#  Name                                  Disclosure Date  Rank    Check Description
-  ----                                  ---------------  ----    ----- -----------
0  auxiliary/scanner/portscan/ftpbounce                   normal  No    FTP Bounce Port Scanner
1  auxiliary/scanner/natpmp/natpmp_portscan               normal  No    NAT-PMP External Port Scanner
2  auxiliary/scanner/sap/sap_router_portscan ner          normal  No    SAPRouter Port Scanner
3  auxiliary/scanner/portscan/xmas                        normal  No    TCP "XMas" Port Scanner
4  auxiliary/scanner/portscan/ack                         normal  No    TCP ACK Firewall Scanner
5  auxiliary/scanner/portscan/tcp                         normal  No    TCP Port Scanner
6  auxiliary/scanner/portscan/syn                         normal  No    TCP SYN Port Scanner
7  auxiliary/scanner/http/wordpress_pingback_access       normal  No    Wordpress Pingback Locator


Interact with a module by name or index. For example info 7, use 7 or use
auxiliary/scanner/http/wordpress_pingback_access
```

6.5.1 Engaging with Modules

After identifying a desired module, activate it with the following command:

```
use <number | exploit-name>
```

Replace **<exploit-name>** with the number or the exact exploit module name. For example:

```
msf6 > use 6
msf6 auxiliary(scanner/portscan/syn) >
```

This action activates the exploit module, revealing details like its name, author, target platform, and associated payload.

6.6 Configuring Module Parameters

Before deploying a module, adjusting specific parameters, such as target IP, port, or chosen payload, is often necessary. To view an module's configurable options use the **show options** command which lists all tweakable parameters for the active exploit module.

```
msf6 auxiliary(scanner/portscan/syn) > show options

Module options (auxiliary/scanner/portscan/syn):

Name       Current Setting  Required  Description
----       ---------------  --------  -----------
BATCHSIZE  256              yes       The number of hosts to scan per set
DELAY      0                yes       The delay between connections, per thread, in ms
INTERFACE                   no        The name of the interface
JITTER     0                yes       The delay jitter factor (max value by which to +/- DELAY) in
ms.
PORTS      1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
RHOSTS                      yes       The target host(s), see
https://docs.metasploit.com/docs/using-metasploit/basics/using
                                        -metasploit.html
SNAPLEN    65535            yes       The number of bytes to capture
THREADS    1                yes       The number of concurrent threads (max one per host)
TIMEOUT    500              yes       The reply read timeout in ms


View the full module info with the info, or info -d command.
```

There are many more parameters for this module, set the options as required.

```
msf6 auxiliary(scanner/portscan/syn) > set threads 50
threads => 50
msf6 auxiliary(scanner/portscan/syn) > set rhosts 192.168.90.5
rhosts => 192.168.90.5
msf6 auxiliary(scanner/portscan/syn) > set ports 80,9090
ports => 80,9090
```

6.7 Executing the Module

With all parameters set, you can launch the module:

```
msf6 auxiliary(scanner/portscan/syn) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

If the module succeeds, a confirmation message will appear, indicating a successful operation.

6.8 TCP Scan

Trying the TCP scan.

```
msf6 auxiliary(scanner/portscan/syn) > use 5

msf6 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

Name         Current Setting  Required  Description
----         ---------------  --------  -----------
CONCURRENCY  10               yes       The number of concurrent ports to check per host
DELAY        0                yes       The delay between connections, per thread, in ms
JITTER       0                yes       The delay jitter factor (maximum value by which to +/-
DELAY) in ms.
PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
RHOSTS                        yes       The target host(s), see
https://docs.metasploit.com/docs/using-metasploit/basics/usi
                                           ng-metasploit.html
THREADS      1                yes       The number of concurrent threads (max one per host)
TIMEOUT      1000             yes       The socket connect timeout in ms


View the full module info with the info, or info -d command.


msf6 auxiliary(scanner/portscan/tcp) > set threads 10
threads => 10

msf6 auxiliary(scanner/portscan/tcp) > set rhosts 192.168.90.5
rhosts => 192.168.90.5

msf6 auxiliary(scanner/portscan/tcp) > set ports 9090
ports => 9090

msf6 auxiliary(scanner/portscan/tcp) > run
[+] 192.168.90.5:         - 192.168.90.5:9090 - TCP OPEN
[*] 192.168.90.5:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```