Cybersecurity for Industrial Networks

Topic 8
Penetration Test - Reconnaissance

Dr Diarmuid Ó Briain

Version: 2.0

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Conrad Ekisa

Dr Diarmuid Ó Briain

Table of Contents

Illustration Index

Index of Tables

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 v2.0 (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

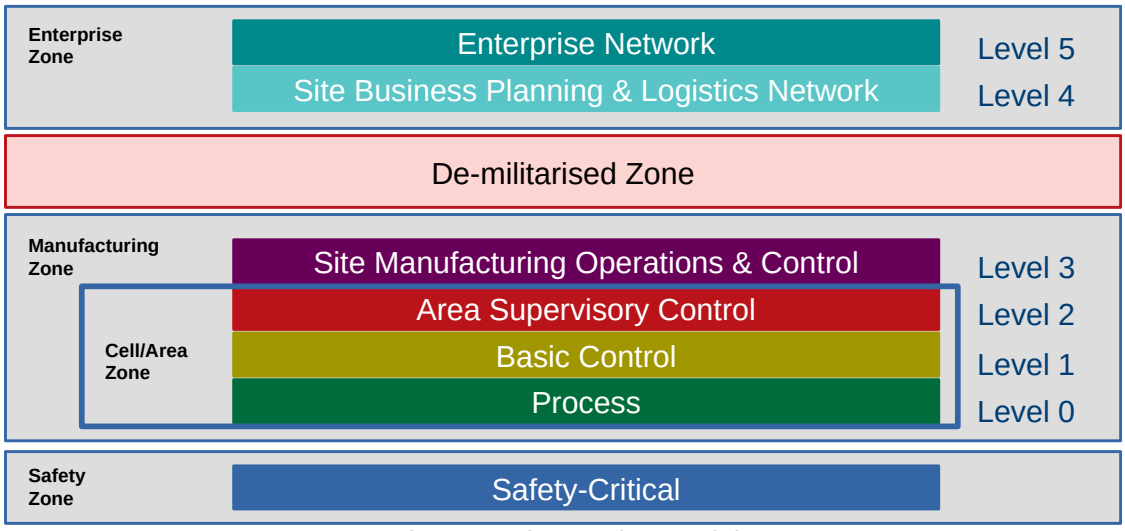


Figure 1: The Purdue Model

3 Virtualised ICS OpenSource Research (VICSORT 2.0)

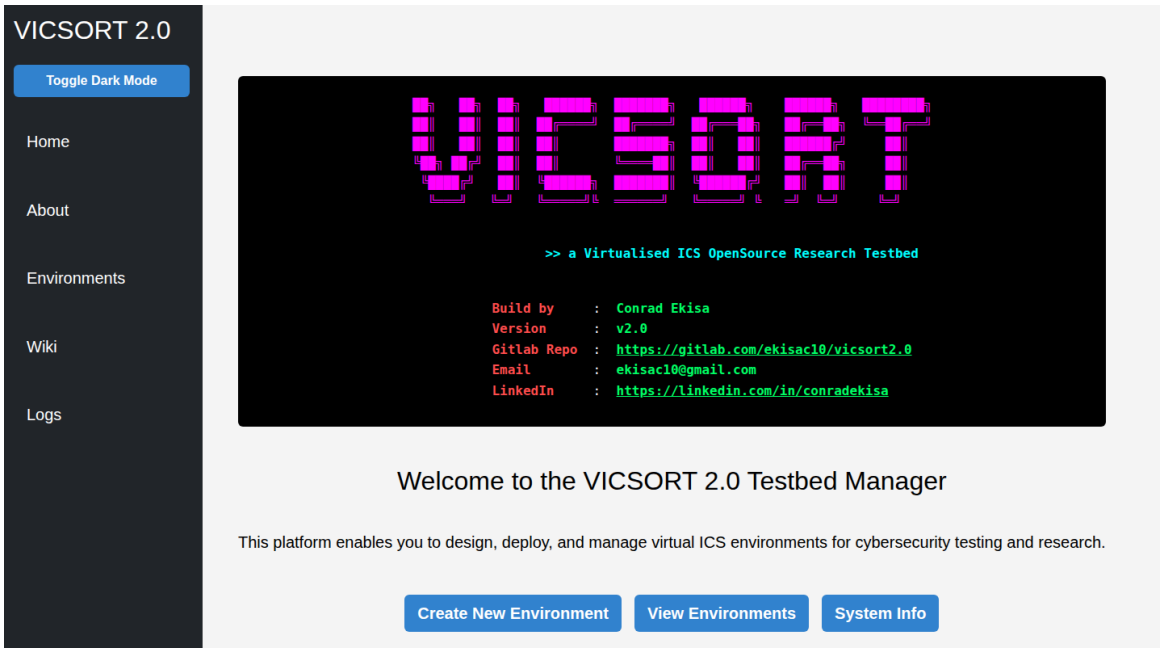


Figure 2: VICSORT Testbed Manager

A Virtualised ICS OpenSource Research Testbed *for cybersecurity research, training, and experimentation in ICS environments.*

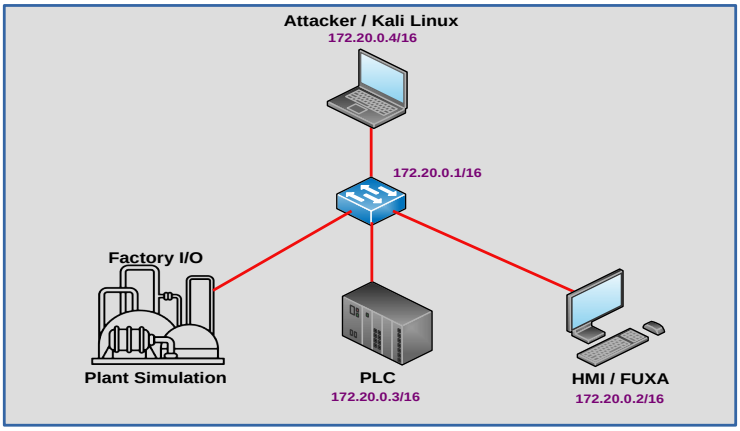


Figure 3: Basic scenario in VICSORT2

Table 1: IP Address and Password for VICSORT2 Basic Setup

Node	IP Address Mapping
Bridge	172.20.0.1 /16
HMI / FUXA	172.20.0.2 /16
PLC	172.20.0.3 /16
Kali Linux	172.20.0.4 /16

PLC Username: openplc	Password: openplc
Kali Username: kasm_user	Password: password

4 Architecture Overview

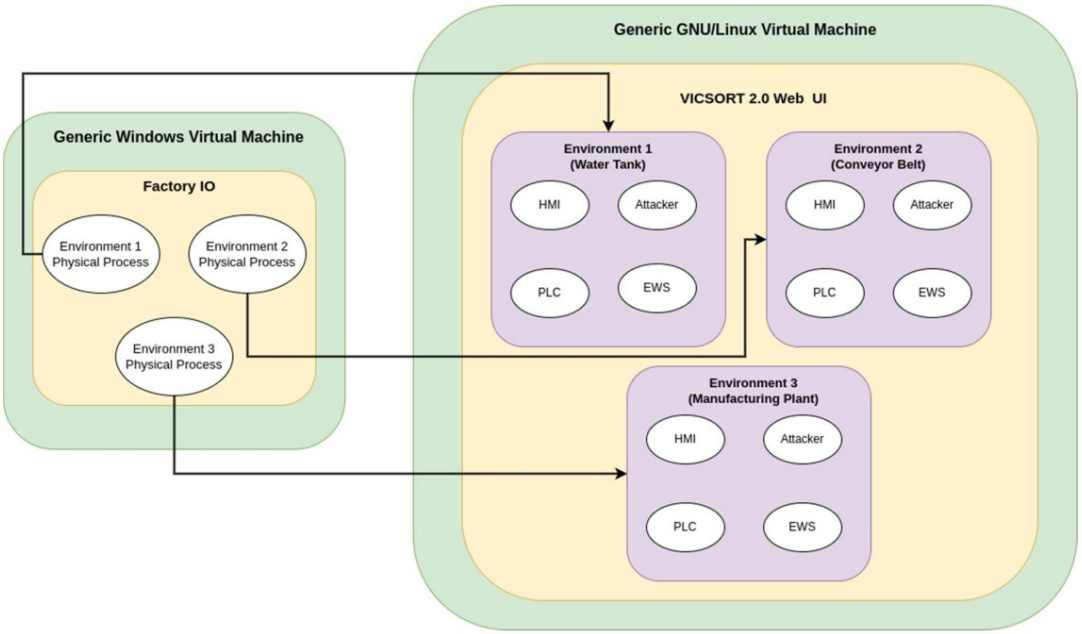


Figure 4: Architecture

VICSORT 2.0 enables the creation of multiple isolated ICS environments within a virtualised GNU/Linux host, each containing:

- **HMI** (FUXA)
- **PLC** (OpenPLC)
- **Engineering Workstation** (OpenPLC Editor)
- **Attacker Node** (Kali Linux)

These can be linked to **Factory I/O** running on a separate Windows VM to simulate realistic 3D industrial processes via **Modbus TCP**.

4.1 Core Features

4.1.1 Environment Management

- Create **blank** or **imported** environments.
- Import pre-packaged environments containing PLC, HMI, and PLC Editor configurations.
- Maintain multiple environments simultaneously.
- Full environment deletion with cleanup of containers, networks, and persistent files.

4.1.2 Automated Deployment

- Deploys containers for:
  - ScadaLTS (HMI)
  - OpenPLC (PLC runtime)
  - OpenPLC Editor
  - Kali Linux (attacker tools)
- Each environment has its own dedicated Docker network.
- Real-time deployment logs streamed to the browser.

4.1.3 Import Validation & Guided Setup

- Validates uploaded ZIP files for required ICS components.
- Detects missing or duplicate files before deployment.
- Provides step-by-step guidance to prevent configuration issues.

4.1.4 Container & Service Management

- Start/stop individual containers or grouped HMI containers.
- Monitor container status and uptime.
- View/download environment and system logs.
- Real-time log tailing.

4.1.5 Factory I/O Integration

- Connects OpenPLC to **Factory I/O** for realistic 3D process visualisation.
- Factory I/O can be:
  - On a separate VM
  - On the same physical host as VICSORT 2.0 (network accessible)
- **Requirements:**
  - IP address of the Factory I/O instance
  - Modbus TCP port configured in Factory I/O

4.1.6 Built-in Documentation & Logs

- Integrated and Updatable **Wiki** for guides and tutorials.
- Centralised **Logs Dashboard** for environment/system logs.

5 Installing the VICSORT v2.0 Platform

This platform was initially built on Ubuntu 22.04 LTS. Recommended OS is Ubuntu 22.04 LTS or Ubuntu 24.04 LTS with VirtualBox.

5.1.1 Prerequisites

- A downloaded Ubuntu 22.04 LTS or 24.04 LTS Desktop version VM and set it up within VirtualBox.
- Assumes a clean VM.
- Google Chrome downloaded onto the VM.

5.1.2 Clone the repository

```
~$ sudo apt update && sudo apt install git -y
~$ git clone https://gitlab.com/ekisac10/vicsort2.0.git
~$ cd vicsort2.0
```

5.1.3 Run Initial Setup

```
~$ sudo bash scripts/setup.sh
```

5.1.4 Run the application

Once the setup script has completed execution, access the web UI via **`http://localhost:8000/`** or as guided by the setup script.

5.1.5 Gunicorn Manager

A helper script is added to assist with managing **gunicorn** (the web server running VISCORT 2.0) e.g. for viewing logs as the process runs detached in the background. Below is the usage:

```
# Start in background
~$ ./scripts/gunicorn_manager.sh start

#  Status / PID
~$ ./scripts/gunicorn_manager.sh status

# Error log. Also great for monitoring Docker install progress
~$ ./scripts/gunicorn_manager.sh logs

  # access log
~$ ./scripts/gunicorn_manager.sh logs access

#  Foreground (for live debugging)
~$ ./scripts/gunicorn_manager.sh foreground

#  Stop / Restart
~$ ./scripts/gunicorn_manager.sh stop
~$ ./scripts/gunicorn_manager.sh restart
```

5.1.6 FactoryIO Initial Setup

Once the application is running, the inbuilt Wiki contains information on how to setup and connect FactoryIO. However, the FactoryIO Windows VM can be downloaded from here and run with VirtualBox.

5.2 Connecting to Factory I/O

- Ensure **Factory I/O** is running and configured for **Modbus TCP Server** mode.
- Note its **IP address** and **Modbus TCP port**.
- In **VICSORT 2.0**, configure **OpenPLC** to connect to the Factory I/O instance.

A full **Factory I/O Integration Guide** will be provided in the Wiki.

5.3 Creating First Environment

Once VISCORT 2.0 has been installed and is accessible via **`http://localhost:8000`**, you can create your first environment. This will trigger the download of the required containers.

Navigate to **`Environments >> Create New Environment >> Create Blank Environment`**. Simply provide the name **`WaterTank`** under the Environment. Everything else can be left blank and Click **`Create New Environment`**.

Monitor the progress from the terminal too by running

```
~$ bash ./scripts/gunicorn_manager.sh logs
```

First Environment creation may take a while due to container download. Subsequent Environment creations will be much quicker.

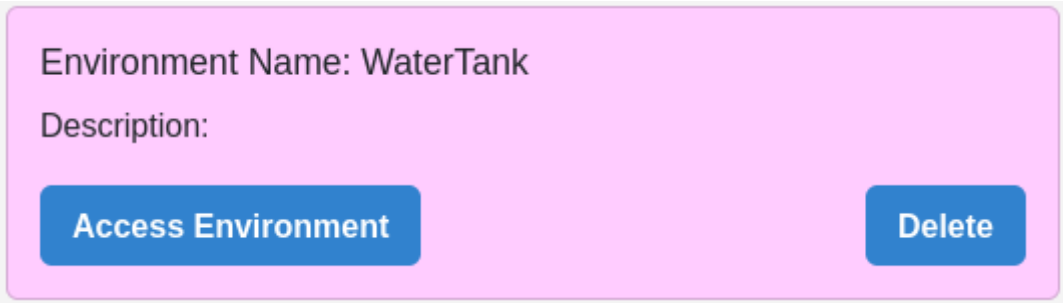


Figure 5: First Environment

To get started, some sample environments are available for import under the **Sample Environments** directory. You can import these environments under **`Environments >> Import Environment`**.

6 Uninstall the VICSORT v2-0 Platform

VICSORT includes an uninstaller script at **scripts/uninstall_viscort.sh**. It is safe-by-default and only removes the VICSORT service + app-local runtime files. More destructive clean-up requires explicit flags.

6.1 Option A: uninstall the service + app-local files

This stops and removes the vicsort systemd service, and deletes:

/.venv (VICSORT virtualenv)

logs/, **temp_uploads/**, PID/lock files used by the app

```
~$ sudo bash scripts/uninstall_viscort.sh
```

6.2 Option B: also remove Docker containers/images created for VICSORT

Use this if you want to reclaim disk space and you do not need the local images/containers.

```
~$ sudo bash scripts/uninstall_viscort.sh –purge–docker
```

6.3 Option C: also remove persistent host data

Use this option if you want to remove environment data saved on the VM host, including (best-effort):

- **~/OpenPLCEditorFiles**
- **~/Kali_Container_Files**
- ScadaLTS bind-mount DB directory under **backend/src/Docker_Setup_Files/ScadaLTS/docker/volumes/**

```
~$ sudo bash scripts/uninstall_viscort.sh ––purge–data
```

6.4 Option D (full wipe): remove service + Docker + persistent data + source folder

This is the most destructive option.

```
~$ sudo bash scripts/uninstall_viscort.sh --purge-docker --purge-data --remove-source --yes
```

This does not uninstall Docker / Python / system packages installed by **`scripts/setup.sh`**.

The Docker purge is best-effort and may remove containers if you used the same image tags outside VICSORT. Review the script before running **`--purge-docker`**.

7 Access Environment

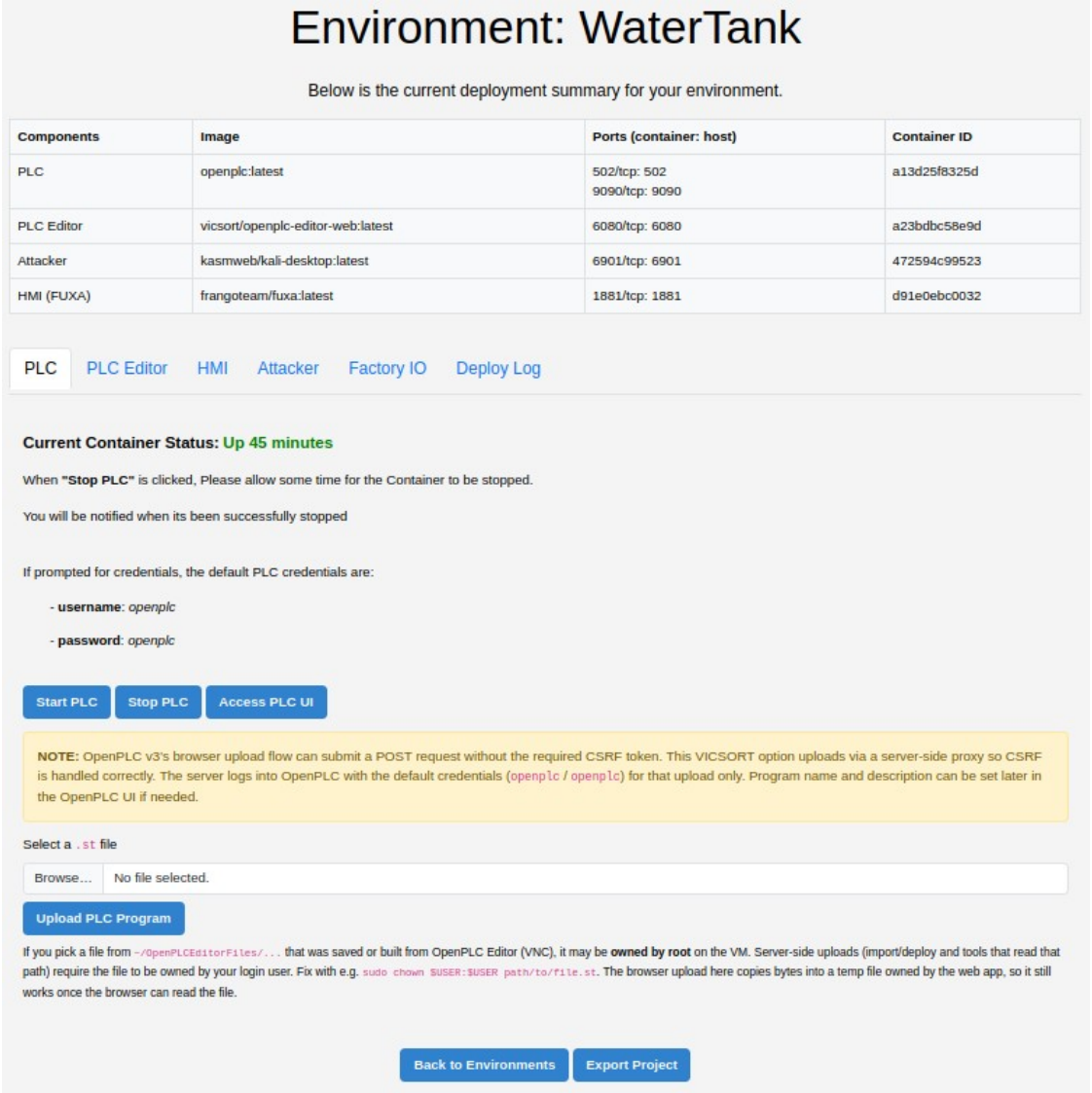


Figure 6: Environment Dashboard

7.1 Programmable Logic Controller (PLC)

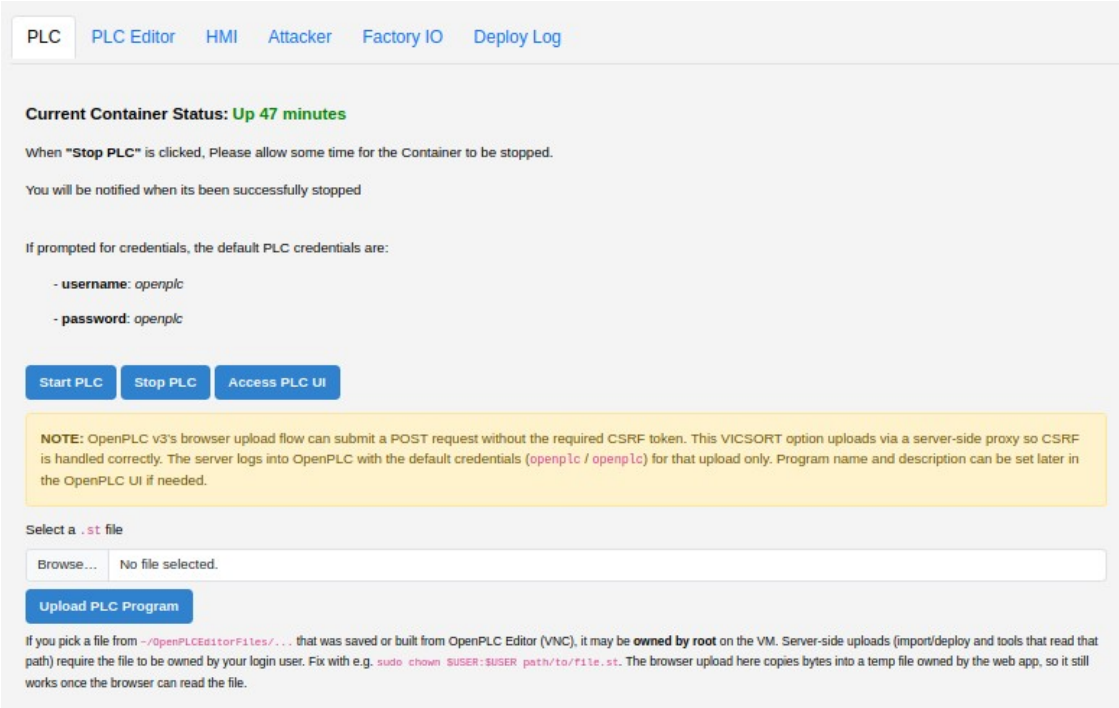


Figure 7: Programmable Logic Controller Dashboard

7.2 Start the PLC

Access the PLC UI with the **[Access PLC UI]** button.
On the left side pane, click in the **[Start PLC]** button.
The dashboard will show **Running**.

8 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.

8.1 Kali Linux on the Attacker Container

As illustrated in 16, click on the **`[Start Attacker]`** and then the **`[Access Attacker UI]`** to launch the Kali Linux instance on the **`attacker-container`**.

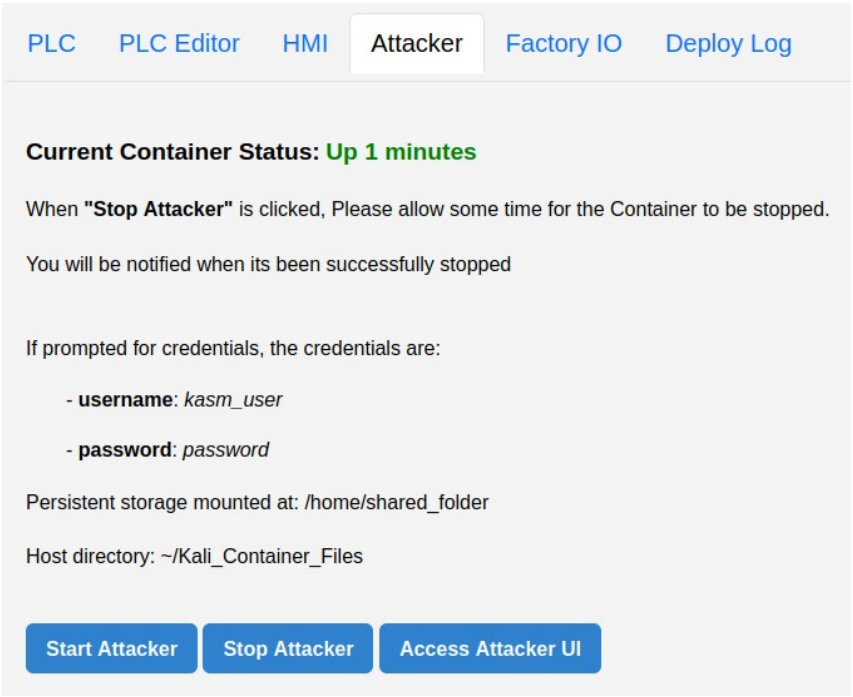


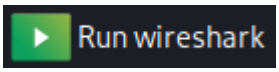
Figure 8: Launch the Attacker Kali Linux Instance

Figure 9: Attacker-container Desktop

8.2 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.

Select the Kali applications button



Type **`wireshark`** in search bar click **`Run wireshark`**.

Click on the **`Start capturing packets`** icon.

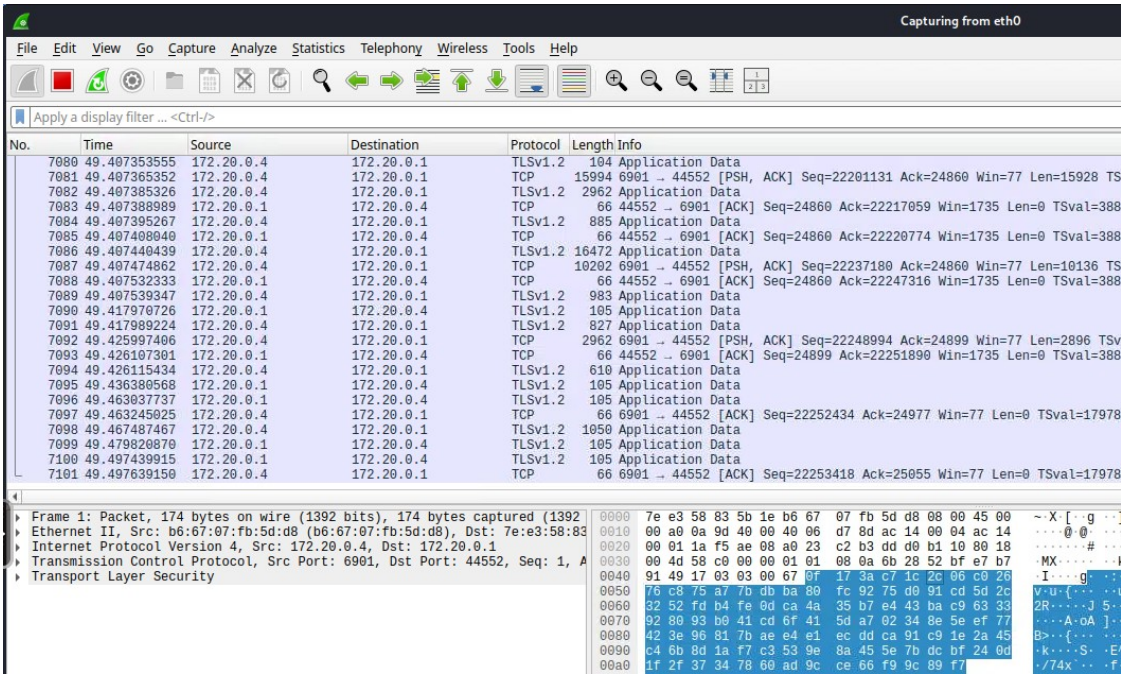


Figure 10: Wireshark in the attacker-container

8.3 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**.

Click on the Terminal Emulator icon in the Kali Linux Desktop.

```
Session    Actions    Edit    View    Help

default:~$ tshark -D
1. eth0
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. sshdig (SSH remote syscall capture)
14. sshdump (SSH remote capture)
15. udpdump (UDP Listener remote capture)
16. wifidump (Wi-Fi remote capture)
default:~$


default:~$  tshark -F pcap -V > ~/tshark_out.pcap
Capturing on 'eth0'
903 ^C                          [Control+C]
default:~$
```

When the process was stopped after a few seconds there was over 903 frames captured. Have a look into the first 20 bytes of the first frame captured within the file using the head command.

```
Session    Actions    Edit    View    Help

default:~$ head -20 ~/tshark_out.pcap
Frame 1: Packet, 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
on interface eth0, id 0
Section number: 1
Interface id: 0 (eth0)
Interface name: eth0
Encapsulation type: Ethernet (1)
Arrival Time: Apr 3, 2026 15:10:34.161889543 UTC
UTC Arrival Time: Apr 3, 2026 15:10:34.161889543 UTC
Epoch Arrival Time: 1775229034.161889543
[Time shift for this packet: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
```

```
Frame Length: 110 bytes (880 bits)
Capture Length: 110 bytes (880 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth: ethertype:ip:tcp:tls]
Character encoding: ASCII (0)
Ethernet II, Src: b6:67:07:fb:5d:d8 (b6:67:07:fb:5d:d8), Dst: 7e:e3:58:83:5b:
1e (7e:e3:58:83:5b:1e)
Destination: 7e:e3:58:83:5b:1e (7e:e3:58:83:5b:1e)
.... LG bit: Locally administered address
( this is NOT the factory default)
default:~$
```

8.4 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.

```
Session    Actions    Edit    View    Help

default:~$ sudo netdiscover -i eth0 -r 172.20.0.0/16
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 2 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 84
 ___________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 ___________________________________________________________________________

 172.20.0.2      16:ee:df:9d:2f:7d      1      42  Unknown vendor
 172.20.0.3      72:23:f9:f6:77:80      1      42  Unknown vendor
```

8.5 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

```
Session    Actions    Edit    View    Help

default:~$ sudo p0f -i eth0 -d -o ~/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 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Log file '/home/kasm-user/p0f-output.txt' opened for
writing.
[+] Daemon process created, PID 24646 (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 24646.

```
default:~$ ps -ef | grep p0f
root        24646       1  0 16:35 ?        00:00:00 p0f -i eth0 -d -o
/home/kasm-user/p0f-output.txt
kasm-us+    24769   24032  0 16:38 pts/3    00:00:00 grep p0f
```

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

```
default:~$ sudo tail -f ~/p0f-output.txt
2026/04/03 15:34:09] mod=syn|cli=172.20.0.1/34348|srv=172.20.0
.2/1881 subj=c lilos Linux 2.2.x-3.x dist=0|params=generic|raw
_sig=4:64+0:0:1460:mss*44,10:m ss,sok,ts,nop,ws:df,id+:0
[2026/04/03 15:34:09] mod=mtu|cli=172.20.0.1/34348|srv=172.20.0
.2/1881 subj=cli|link=Ethernet or modem|raw_mtu=1500
[2026/04/03 15:34:09] mod=syn+ack|cli=172.20.0.1/34348 srv=172
.20.0.2/1881 su bj=srvlos=???|dist=0|params=none|raw_sig=4:64
+0:0:1460:mss*45,10:mss,sok,ts,n op,ws:df:0
[2026/04/03 15:34:09] mod=mtu|cli=172.20.0.1/34348 srv=172.20
.0.2/1881 subj=srv|link=Ethernet or modem|raw_mtu=1500
[2026/04/03 15:34:11] mod=uptime|cli=172.20.0.1/34348|srv=172
.20.0.2/1881 sub j=srv|uptime=43 days 6 hrs 46 min (modulo 49
days)|raw_freq=1000.00 Hz
```

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

```
default:~$ sudo kill -SIGKILL 24646

default:~$ ps -ef | grep p0f

default:~$  ps -ef | grep p0f
kasm-us+    25922   24032  0 16:38 pts/3    00:00:00 grep p0f
```

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

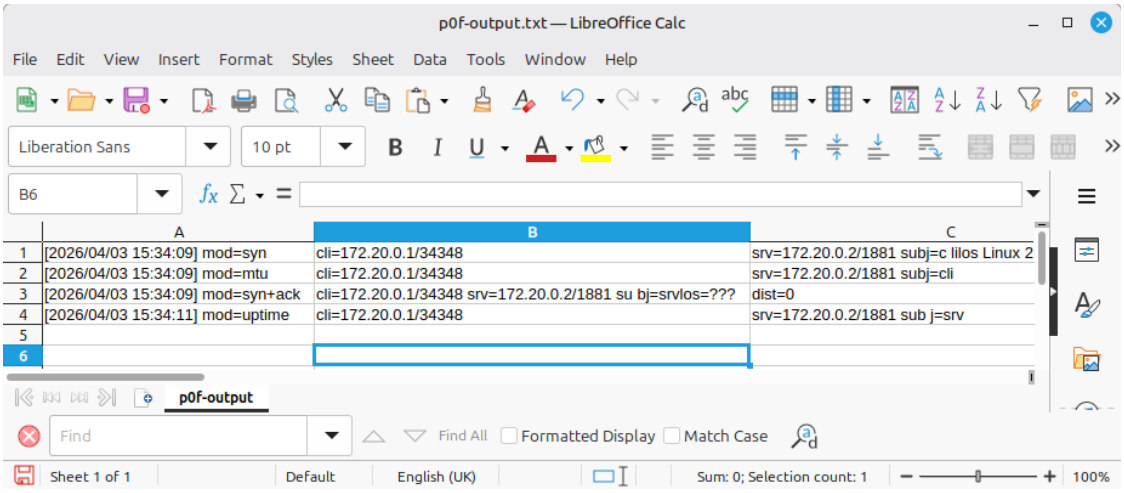


Figure 11: p0f imported into a spreadsheet with pipe (|) as a delimiter

8.6 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 172.20.0.0/16 network, removing the **attacker-container** computer itself.

```
Session    Actions    Edit    View    Help

default:~$  nmap -sn 172.20.0.0/16 --exclude 172.20.0.4
Starting Nmap 7.98 (https://nmap.org) at 2026-04-03 16:01 +0000
Nmap scan report for vicsort2-VirtualBox.local (172.20.0.1)
Host is up (0.000056s latency).
MAC Address: 7E:E3:58:83:5B:1E (Unknown)
Nmap scan report for Water_Tank_fuxa. Water_Tank_network
(172.20.0.2)
Host is up (0.000024s latency).
MAC Address: 16:EE:DF:9D:2F:7D (Unknown)
Nmap scan report for Water_Tank_plc.Water_Tank_network
(172.20.0.3)
Host is up (0.00012s latency).
MAC Address: 72:23:F9:F6:77:80 (Unknown)
```

Host **172.20.0.2** is a HMI. And **172.20.0.3** is the PLC. 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.

```
default:~$ nmap -v -sn 172.20.0.2
Starting Nmap 7.98 (https://nmap.org) at 2026-04-03 16:04 +0000
Initiating ARP Ping Scan at 16:04
Scanning 172.20.0.2 [1 port]
Completed ARP Ping Scan at 16:04, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 16:04
Completed Parallel DNS resolution of 1 host. at 16:04, 0.00s
elapsed
Nmap scan report for Water_Tank_fuxa.Water_Tank_network
(172.20.0.2)
Host is up (0.000050s latency).
MAC Address: 16:EE:DF:9D:2F:7D (Unknown)
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds Raw
packets sent: 1 (28B) | Rcvd: 1 (288)

default:~$ nmap -v sn 172.20.0.3
Starting Nmap 7.98 (https://nmap.org) at 2026-04-03 16:05 +0000
Initiating ARP Ping Scan at 16:05
Scanning 172.20.0.3 [1 port]
Completed ARP Ping Scan at 16:05, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 16:05
Completed Parallel DNS resolution of 1 host. at 16:05, 0.00s
elapsed
Nmap scan report for Water_Tank_plc.Water_Tank_network
(172.20.0.3)
Host is up (0.000061s latency).
MAC Address: 72:23:F9:F6:77:80 (Unknown)
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds Raw
packets sent: 1 (288) | Rcvd: 1 (288)
```

Now target the HMI and PLC 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.

```
Session    Actions    Edit    View    Help

default:~$ nmap -A -T4 -p- 172.20.0.2
Starting Nmap 7.98 (https://nmap.org) at 2026-04-03 16:07 +0000
Nmap scan report for Water_Tank_fuxa.Water_Tank_network (172.20.0.2)
Host is up (0.00014s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
1881/tcp open  http    Node.js Express framework
|_http-title: FUXA
MAC Address: 16:EE:DF:9D:2F:7D (Unknown)
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel: 4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.155.19
Network Distance: 1 hop

TRACEROUTE
HOP   RTT     ADDRESS
1     0.14 ms Water_Tank_fuxa.Water_Tank_network (172.20.0.2)

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 15.50 seconds

default:~$ nmap -A -T4 -p- 172.20.0.3
Starting Nmap 7.98 (https://nmap.org) at 2026-04-03 16:08 +0000
Nmap scan report for Water_Tank_plc.Water_Tank_network (172.20.0.3)
Host is up (0.00011s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
8443/tcp open  ssl/http Werkzeug httpd 2.3.7 (Python 3.10.12)
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost, IP Address: 127.0.0.1
| Not valid before: 2026-04-02T19:52:12
|-Not valid after: 2126-03-09T19:52:12
9090/tcp open http Werkzeug httpd 2.3.7 (Python 3.10.12)
|_http-server-header: Werkzeug/2.3.7 Python/3.10.12
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was /login
MAC Address: 72:23:F9:F6:77:80 (Unknown)
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel: 4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 5.19
Network Distance: 1 hop
TRACEROUTE
HOP RTT     ADDRESS
1   0.11 ms Water_Tank_plc.Water_Tank_network (172.20.0.3)
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 47.17 seconds
```

Figure 12: Port 181 on HMI host and 8443 on the PLC

The **`nmap`** command, illustrated in Figure 12, reveals that the host 172.20.0.2 appears to be the **Human-Machine Interface (HMI)** or visualisation layer of the system.

8.6.1 Host 1: 172.20.0.2 (FUXA HMI)

This host appears to be the Human-Machine Interface (HMI) or visualisation layer of the system.

- Port 1881 (TCP)
  - Service: **HTTP**
  - Technology: **Node.js** Express framework
  - Software: **FUXA**
  - FUXA is an open-source web-based SCADA/HMI (Supervisory Control and Data Acquisition) system. It is used here to provide a visual dashboard to monitor or control the water tank levels and pump statuses.

8.6.2 Host 2: 172.20.0.3 (Water Tank PLC)

This host is labelled as the **PLC** and is running web services likely used for management or an API.

- Port 8443 (TCP)
  - Service: **SSL/HTTP** (HTTPS)
  - Technology: **Werkzeug httpd** 2.3.7 (Python 3.10.12).
  - This is a secure web server. The SSL certificate is self-signed (issued to **`localhost`**), which is common in internal lab or ICS environments.
- Port 9090 (TCP)
  - Service: **HTTP**
  - Technology: **Werkzeug httpd** 2.3.7 (Python 3.10.12).
  - This is a standard web interface. The scan notes that the requested resource was redirected to **`/login`**, suggesting a web-based management portal or a login page for the PLC simulator/controller.

Table 2: Service Ports on network hosts

IP Address	Port	Service	Software/Platform	Role
172.20.0.2	1881	HTTP	FUXA (Node.js)	Visualisation / Dashboard
172.20.0.3	8443	HTTPS	Werkzeug (Python)	Secure Admin/API
172.20.0.3	9090	HTTP	Werkzeug (Python)	Web Login / Management

Browse to the open port, as illustrated in Figure 13, Figure 14, and Figure 15 confirms that an **`Apache Tomcat`** webserver is running.

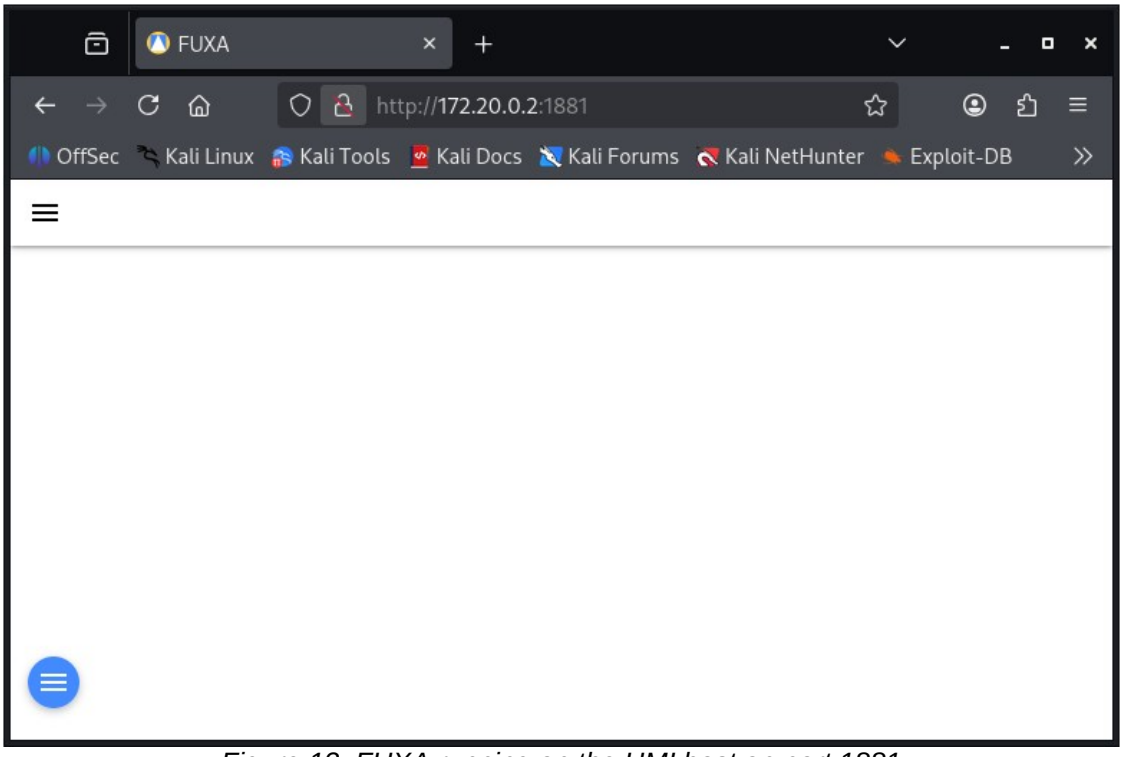


Figure 13: FUXA running on the HMI host on port 1881

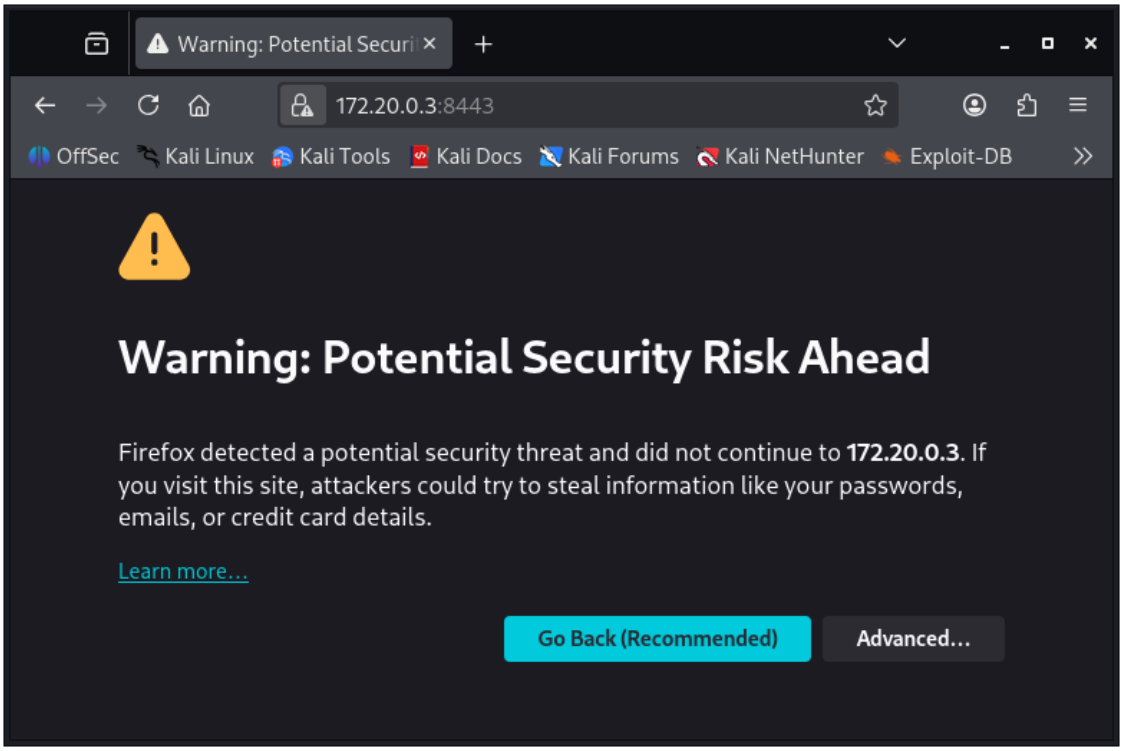


Figure 14: OpenPLC running on the PLC host on port 8443

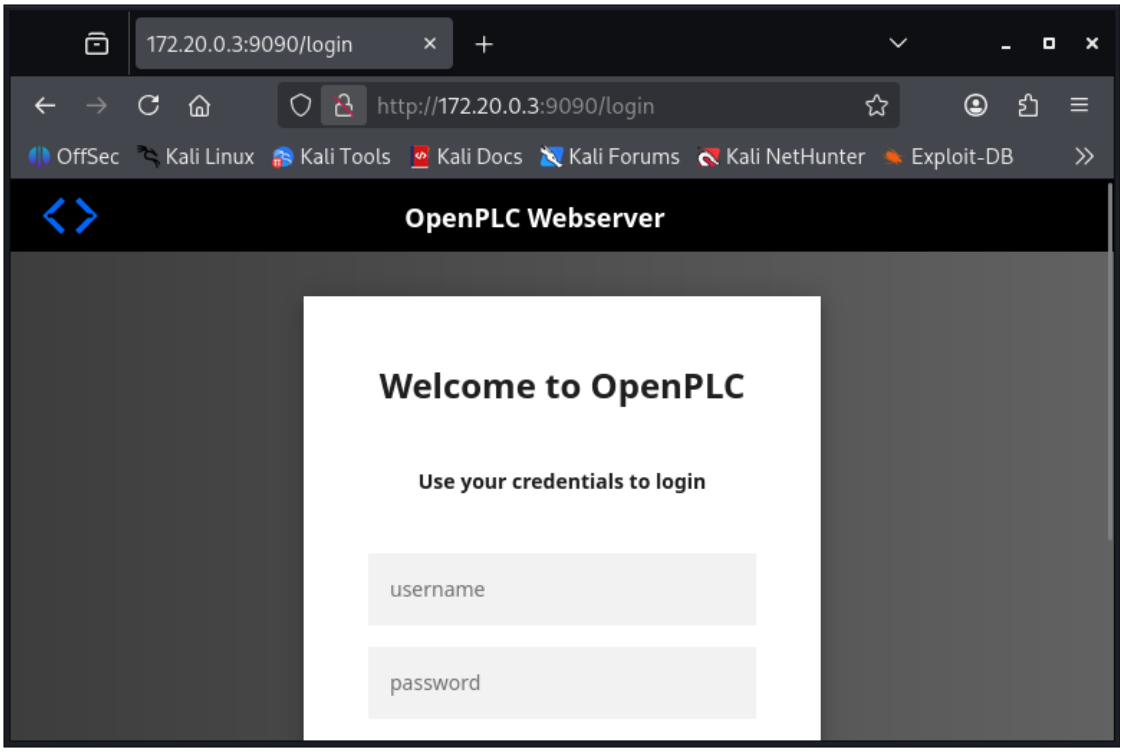


Figure 15: OpenPLC running on the PLC host on port 9090

8.7 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.

8.7.1 Install and update Nikto

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

```
Session    Actions    Edit    View    Help
default:~$ sudo apt purge nikto
default:~$ sudo apt install nikto
```

8.7.2 Running Nikto

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

```
default:~$  nikto -host 172.20.0.3 -port 9090
+ Target IP:          172.20.0.3
+ Target Hostname:    172.20.0.3
+ Target Port:        9090
+ Platform:           Unknown
+ Start Time:         2026-04-03 23:04:31 (GMT0)
---------------------------------------------------------------------------
+ Server: Werkzeug/2.3.7 Python/3.10.12
+ No CGI Directories found (use -C all' to force check all possible dirs). C
GI tests skipped.
+ [013587]/: Suggested security header missing: referrer-policy. See: https:
//developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy
+ [013587] /: Suggested security header missing: strict-transport-security. S
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transpor  ee:
t-Security
+ [013587] /: Suggested security header missing: permissions-policy. See: htt
ps://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Permissions-Policy
+ [013587] /: Suggested security header missing: x-content-type-options. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Opt
ions
+ [013587] /: Suggested security header missing: content-security-policy. See
: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
+ [600652] Python/3.10.12 appears to be outdated (current is at least 3.13.1)
+ [999990] OPTIONS: Allowed HTTP Methods: HEAD, OPTIONS, GET.
+ [007342] /: X-Frame-Options header is deprecated and was replaced with the
Content-Security-Policy HTTP header with the frame-ancestors directive. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-O
ptions
```

```
+ [007352] /: 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 th
e MIME type. See: https://ww.netsparker.com/web-vulnerability-scanner/vulner
abilities/missing-content-type-header/
+ 8268 requests: 0 errors and 9 items reported on the remote host
+ End Time:
2026-04-03 23:05:04 (GMTO) (33 seconds)
+ 1 host(s) tested
Portions of the server's headers (Werkzeug/2.3.7) are not in
the Nikto 2.6.0 database or are newer than the known string. Would you to
submit this information (*no server specific data*) to CIRT.net for a Nikto
update (or you may email to sullo@cirt.net) (y/n)? n
like
default:~$
```

9 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.

9.1 Run the Metasploit Framework and Console

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

```
Session    Actions    Edit    View    Help
```

```
default:~$ sudo apt update
```

Start the Postgresql Database.

```
default:~$ service postgresql status
18/main (port 5432): down

default:~$ sudo service postgresql start
Starting PostgreSQL 18 database server: main.
```

9.1.1 MSFDB Tool

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 **service postgresql status**.

```
default:~$ sudo msfdb reinit
[+] Starting database
[+] Deleting configuration file
[i] Database already stopped
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file
'/usr/share/metasploit-framework/config/datab ase.yml'
[+] Creating initial database schema
default:~$
```

Note that the container instance of Linux does not have **systemd**.

sudo service postgresql start: This ignored **systemctl** and used the system V (**SysV**) **init** script to kickstart the database.

sudo msfdb init: This script automatically tries to check service status via **systemctl** (which causes the error message), but it' then falls back to the **SysV init** script.

9.2 Start Postgresql database

```
default:~$ sudo msfdb init
[+] Starting database
[i] The database appears to be already configured, skipping
initialization
```

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

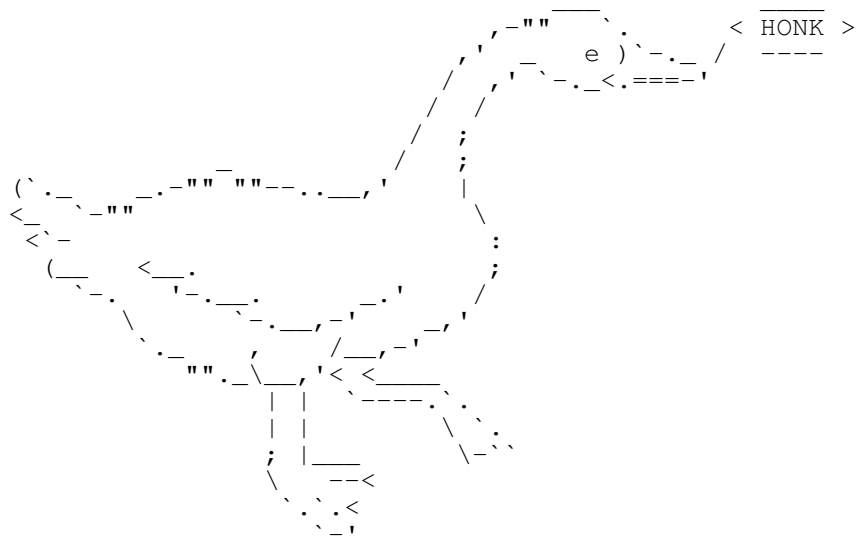
9.3 Check Postgresql database

```
default:~$ sudo msfdb status
[i] No network service running
[+] Detected configuration file
(/usr/share/metasploit-framework/config/database.yml)
```

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

Upon completion, your **metasploit** Framework is ready for use.

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




```
       =[ metasploit v6.4.124-dev                                 ]
+ -- --=[ 2,632 exploits - 1,328 auxiliary - 1,707 payloads     ]
+ -- --=[ 431 post - 49 encoders - 14 nops - 12 evasion         ]


Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework os a Rapid7 Open Source Project

msf >
```

9.4 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:

```
default:~$ sudo apt update; sudo apt install metasploit-framework
```

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

9.5 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**.

```
msf > nmap -Pn -sS -A -OX netscan.xml 172.20.0.0/29 -exclude 172.20.0.4
[*] exec: nmap -Pn -sS -A -OX netscan 172.20.0.0/29 -exclude 172.20.0.4

Starting Nmap 7.98 (https://nmap.org) at 2026-04-11 06:43 +0000
Nmap scan report for vicsort2-VirtualBox.local (172.20.0.1)
Host is up (0.000099s latency).
Not shown: 996 closed tcp ports (reset)
PORT       STATE   SERVICE            VERSION
6901/tcp   open    ssl/jetstream?
| ssl-cert: Subject:
commonName=kasm/organizationName=None/stateOrProvinceName=VA/countryName=US
| Not valid before: 2026-04-11T05:48:11
|_Not valid after: 2036-04-08T05:48:11
|_ssl-date: TLS randomness does not represent time
8000/tcp open tcpwrapped
|_http-server-header: gunicorn
|_http-title: Welcome
8080/tcp open http     Apache Tomcat (language: en)
|_http-title: HTTP Status 404 \xE2\x80\x93 Not Found
9090/tcp open http     Werkzeug httpd 2.3.7 (Python 3.10.12)
|_http-server-header: Werkzeug/2.3.7 Python/3.10.12
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was /login
MAC Address: 52:7F:8A:CD:E1:63 (Unknown)
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 5.19
Network Distance: 1 hop

TRACEROUTE
HOP   RTT       ADDRESS
1     0.10 ms   vicsort2-VirtualBox.local (172.20.0.1)

Nmap scan report for Water_Tank_fuxa.Water_Tank_network (172.20.0.2)
Host is up (0.00012s latency). All 1000 scanned ports on
Water_Tank_fuxa.Water_Tank_network (172.20.0.2) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 42:50:72:48:CD:80 (Unknown)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP   RTT      ADDRESS
1     0.12 ms  Water_Tank_fuxa.Water_Tank_network (172.20.0.2)

Nmap scan report for Water_Tank_plc.Water_Tank_network (172.20.0.3)
Host is up (0.00010s latency).
Not shown: 998 closed tcp ports (reset)
PORT STATE SERVICE VERSION
| 8443/tcp open ssl/http Werkzeug httpd 2.3.7 (Python 3.10.12)
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost, IP Address: 127.0.0.1
| Not valid before: 2026-04-02T19:52:12
|_Not valid after: 2126-03-09T19:52:12
|_ssl-date: TLS randomness does not represent time
9090/tcp open http Werkzeug httpd 2.3.7 (Python 3.10.12)
```

```
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was /login
|_http-server-header: Werkzeug/2.3.7 Python/3.10.12
MAC Address: BA:D0:F6:A0:21:20 (Unknown)
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 5.19
Network Distance: 1 hop

TRACEROUTE
HOP   RTT      ADDRESS
1     0.10 ms  Water_Tank_plc.Water_Tank_network (172.20.0.3)

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
Nmap done: 7 IP address (3 host up) scanned in 172.85 seconds
msf >
```

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

```
msf > db_import netscan.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.14.5'
[*] Importing host 172.20.0.1
[*] Importing host 172.20.0.2
[*] Importing host 172.20.0.3
[*] Successfully imported /root/netscan

msf > hosts

Hosts
=====

address     mac                name          os_name       os_flavor  os_sp  purpose
-------     ---                ----          -------       ---------  -----  -------
172.20.0.1  52:7f:8a:cd:e1:63  vicsort2-VirtualBox.local   Linux      4.X   server
172.20.0.2  42:5c:72:48:cd:80  Water_Tank_fuxa.Water_Tan   Unknown          device
                               k__network
172.20.0.3  ba:d0:f6:20:21:2d  Water_Tank_plc.Water_Tank   Linux     4.X   server
                               __network
```

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

```
msf > hosts -o /home/kasm-user/scanned_hosts.csv
[*] Wrote hosts to /home/kasm-user/scanned_hosts.csv



default:~$ cat /home/kasm-user/scanned_hosts.csv
address,mac, name, os_name,os_flavor, os_sp, purpose, info,comments
"172.20.0.1",  "52:7f:8a:cd:e1:63",  "vicsort2-VirtualBox.local",  "Linux","",
"4.X", "server"
"172.20.0.2","42:5c:72:48:cd:80",    "Water_Tank_fuxa.    Water_Tank_network",
"Unknown","","", "device"
"   172.20.0.3",   "ba:d0:f6:a0:21:2d",   "Water_Tank_plc.Water_Tank_network",
"Linux","", "4.X", "server"
default:~$
```

Get the services that are running on the network.

```
msf > services
Services
========

host            port  proto  name           state  info
----            ----  -----  ----           -----  ----
172.20.0.1      6901  tcp    ssl/jetstream  open
172.20.0.1      8000  tcp    tcpwrapped     open
172.20.0.1      8080  tcp    http           open   Apache Tomcat language: en
172.20.0.1      9090  tcp    http           open   Werkzeug httpd 2.3.7 Python 3.10.12
172.20.0.3      8443  tcp    ssl/http       open   Werkzeug httpd 2.3.7 Python 3.10.12
172.20.0.3      9090  tcp    http           open   Werkzeug httpd 2.3.7 Python 3.10.12
```

```
msf > services -o /home/kasm-user/scanned_services.csv
[*] Wrote hosts to /home/kasm-user/scanned_services.csv
```

```
default:~$ cat /home/kasm-user/scanned_hosts.csv
host, port, proto, name, state, info, resource, parents
"172.20.0.1", "6901", "tcp", "ssl/jetstream", "open",
"172.20.0.1", "8000", "tcp", "tcpwrapped", "open","","{}"
" 172.20.0.1", "8080' "tcp", "http", "open", "Apache Tomcat language:
en", "{}","" "172.20.0.1", "9090' tcp "http", "open", "Werkzeug httpd
2.3.7 Python 3.10.12","{}",""
"172.20.0.3", "8443", "tcp", "ssl/http", "open", "Werkzeug httpd
2.3.7 Python 3.10.12""""" "172.20.0.3", "9090", "tcp", "http",
"open", "Werkzeug httpd 2.3.7 Python 3.10.12","{}",
```

9.6 Searching for Modules

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

```
msf > search <search-term>
```

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

```
msf > 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
```

9.6.1 *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:

```
msf > use 6
msf auxiliary(scanner/portscan/tcp) >
```

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

9.7 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.

```
msf 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 +/-
PORTS        8443,9090        yes       Ports to scan (e.g. 22-25,80,110-900)
RHOSTS       172.20.0.3       yes       The target host(s), see
https://docs.metasploit.com/docs/using-metasploit/basics/usi
                                        ng-metasploit.html
THREADS      50               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.
```

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

```
msf auxiliary(scanner/portscan/tcp) > set threads 50
threads => 50
msf auxiliary(scanner/portscan/tcp) > set rhosts 172.20.0.3
rhosts => 192.168.90.5
msf auxiliary(scanner/portscan/tcp) > set ports 8443,9090
ports => 8443,9090
```

9.8 Executing the Module

With all parameters set, you can launch the module:

```
msf auxiliary(scanner/portscan/tcp) > run
[+] 172.20.0.3:          - 172.20.0.3:9090 - TCP OPEN
[+] 172.20.0.3:          - 172.20.0.3:8443 - TCP OPEN
[*] 172.20.0.3:          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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

This page is intentionally blank