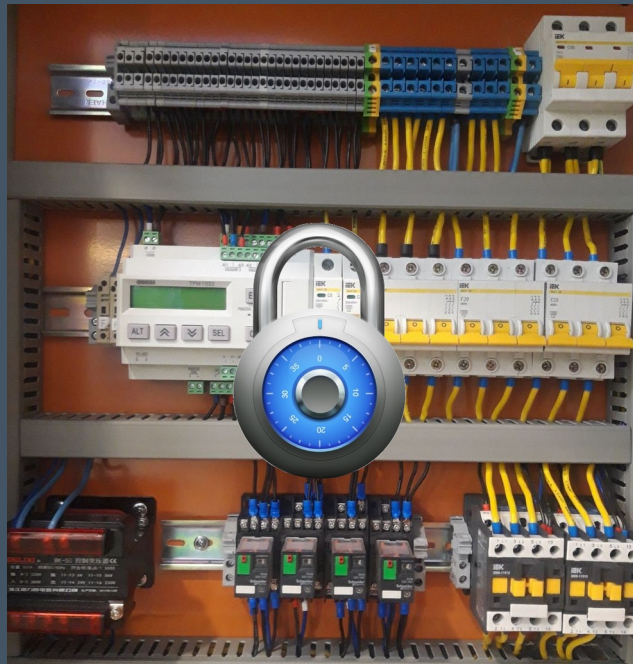


Topic 7

Proactive Defence & Offensive Reconnaissance



Dr Diarmuid Ó Briain
Version: 1.0

Copyright © 2026 C²S Consulting

Licensed under the EUPL, Version 1.2 or – as soon they will be approved by the European Commission - subsequent versions of the EUPL (the "Licence");

You may not use this work except in compliance with the Licence.

You may obtain a copy of the Licence at:

https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf

Unless required by applicable law or agreed to in writing, software distributed under the Licence is distributed on an "AS IS" basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the Licence for the specific language governing permissions and limitations under the Licence.

Dr Diarmuid Ó Briain



Table of Contents

1 Objectives.....	5
2 Introduction.....	5
3 Introduction to Penetration testing.....	6
3.1 Penetration testing steps.....	6
4 Open-Source Intelligence & Passive Reconnaissance.....	8
4.1 Utilising Shodan for internet-facing asset discovery.....	8
4.2 Exercise: Shodan Enumeration.....	9
5 Network Mapper Deep Dive.....	10
5.1 Network Mapper.....	10
5.2 Target specification and host discovery techniques.....	11
5.3 Scan techniques, port specification, and version/OS detection.....	14
5.4 Timing, performance optimisation, and output management.....	17
5.5 XML output.....	20
5.6 Firewall/IDS evasion, spoofing, and anonymous scanning.....	21
5.7 NMAP Scripting Engine & debugging options.....	25
6 Operational Technology & Legacy Scanning.....	29
6.1 The inherent dangers of scanning Automated Control Systems.....	29
6.2 IACS-specific scripts for NSE.....	31
6.3 Exercise: Safe OT Scanning.....	31
7 Active Defence & Deception.....	32
7.1 Deploying Honeypots and canaries to detect early-stage reconnaissance.....	32
8 Bibliography.....	34

Table of Figures

Figure 1: Penetration test steps.....	6
Figure 2: Shodan scan of Dublin City for Modbus devices.....	8
Figure 3: nmapsi4 graphical nmap front-end.....	10
Figure 4: Using the TOP network.....	22
Figure 5: IP addressing info.....	24
Figure 6: Source for connections through TOR.....	25

Index of Tables

Table 1: Nmap host discovery control options.....	14
Table 2: Nmap port scan techniques.....	15
Table 3: Nmap port specification.....	15
Table 4: Nmap Service and Version detection.....	16
Table 5: Nmap OS detection.....	16
Table 6: Nmap timing options.....	17
Table 7: Nmap fine-tuning scan performance options.....	17
Table 8: Nmap output options.....	18
Table 9: Nmap output examples.....	19
Table 10: Nmap commands and techniques for evasion and spoofing.....	21
Table 11: Nmap NSE Scripts.....	26
Table 12: Nmap miscellaneous options.....	26
Table 13: Other useful Nmap commands.....	27
Table 14: Nmap NSE script examples.....	28

1 Objectives

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

- Map an organisation's attack surface by leveraging Shodan for passive asset discovery and configuring Nmap for precise target specification and host discovery.
- Formulate advanced Network Mapper (Nmap) scanning strategies that optimise performance, utilise XML output, and implement Firewall/IDS evasion techniques while diagnosing issues via NMAP Scripting Engine debugging (NSE).
- Evaluate the risks of scanning Industrial Automated Control Systems (IACS) and safely execute non-disruptive, IACS-specific NSE scripts to prevent industrial downtime.
- Design and deploy honeypots and network canaries within an architecture to detect early-stage reconnaissance and alert security operations before a breach escalates.
-

2 Introduction

This topic provides a comprehensive introduction to security assessment methodologies, guiding you from early-stage reconnaissance to advanced scanning and active defence. You will develop the foundational skills necessary to map external attack surfaces using Open-Source INTelligence (OSINT), perform deep-dive network enumeration, and safely navigate the delicate environments of Operational Technology (OT). Finally, the module explores defensive deception techniques, teaching you how to proactively detect and neutralise adversarial reconnaissance before an intrusion can escalate.

3 Introduction to Penetration testing

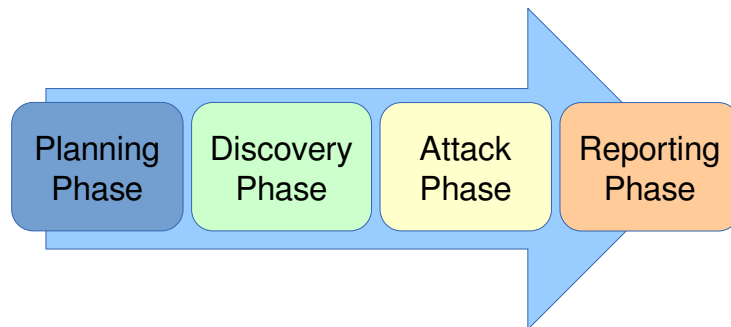


Figure 1: Penetration test steps

Penetration testing (pen-testing) is the practice of testing a computer system, network or web application to find vulnerabilities that an attacker can exploit. It is a proactive and authorised attempt to evaluate the security of an Information Technology (IT) or OT infrastructure by safely attempting to exploit system vulnerabilities, including OS, service and application flaws, improper configurations, and even risky end-user behaviour.

3.1 Penetration testing steps

In many ways the pen-test mimics the anatomy of a cyber attack but with different objectives. The pen-test is an attempt to identify existing vulnerabilities that could present an attack vector to a hacker. Once these vulnerabilities are identified the organisation needs to establish and implement a plan to remove or mitigate these.

3.1.1 Planning and Preparation

A kick-off meeting with the management to discuss in detail the scope and the overall objective of the pen-test. A clear objective is essential for the pen-test such as to demonstrate that exploitable vulnerabilities do in fact exist within the business infrastructure, whether that be IT or OT. As part of the scoping identify:

- Timing and duration allowed for the pen-tests
- Personnel involved
- Are staff being informed of the tests?
- Network, Computers, Devices and processes involved
- Operational requirements during the pen-test
- How the results are to be presented at the conclusion of the test.

After this scoping meeting the pen-testers need to develop a **Penetration Test Plan** which should be shared with the client company. It must include:

- The detailed test plan itself. What tests are to be performed and on what.
- A **Confidentiality statement** that is signed by both the pen-testers and the client.
- A clear **Acceptance sign-off sheet** that the **Penetration Test Plan** is acceptable to the client and affords legal protection to the pen-testers.

Remember the pen-testers are actually conducting tests that are deemed illegal and therefore require the indemnity of the Acceptance sign-off from the client company.

3.1.2 Reconnaissance, Information Gathering and Analysis

Gathering of as much information as possible as a reconnaissance is essential.

- What does the network look like?
- What devices are on the network?
- Who works at the company?
- What does the organogram of the company look like?

3.1.3 Vulnerability detection

Once a picture of the target organisation has been compiled a scan of vulnerabilities is the next step.

3.1.4 Penetration attempt

Once a list of vulnerabilities have been identified and logged it is time to attempt a penetration. Identifying the best targets from the machines showing vulnerability is important particularly if the time given is short. Identifying the juicy targets may be as simply as looking at the machine names as it is a habit of IT personnel to use functional names like MAILSVR or FTPSERVER etc...

Define the list of machines that are to be given special additional treatment. Try password cracking tools, dictionary, brute force and hybrid attacks.

3.1.5 Analysis and Reporting

A detailed report must be furnished to the client at the conclusion of the tests. It should include:

- A summary of successful penetration tests.
- A list of all information gathered during the pen-test.
- A complete list and description of vulnerabilities found (including on machines not singled out for a penetration attempt).
- A suggested list of next steps to close the vulnerabilities and increase security at the client company.

3.1.6 Tidy up

During the pen-testing a detailed list of steps taken should be maintained. On the conclusion of the testing the pen-testers work with the client staff ensure that the steps have not left and residual issues, like entries in configuration files, new users or groups etc..

The remainder of this topic will focus on the initial step of penetration testing, the reconnaissance.

4 Open-Source Intelligence & Passive Reconnaissance

4.1 Utilising Shodan for internet-facing asset discovery

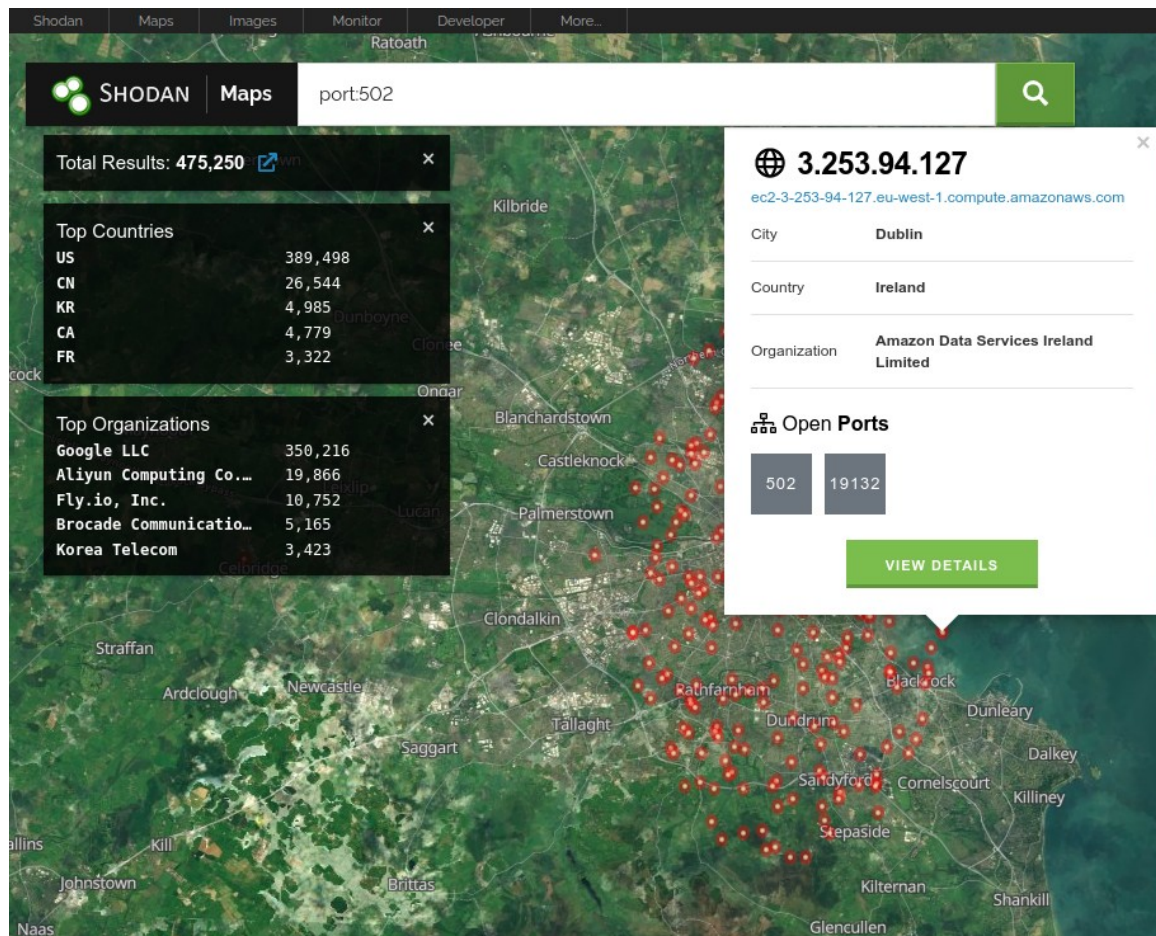


Figure 2: Shodan scan of Dublin City for Modbus devices

Shodan[1] is a search engine that finds devices connected to the Internet. It is a powerful tool for security researchers, hackers, and anyone who wants to learn more about the Internet of Things (IoT).

Shodan works by scanning the Internet for devices that respond to Internet Control Message Protocol (ICMP) ping requests. It then collects information about the devices, such as their IP addresses, operating systems, and open ports. Shodan also indexes the banners that devices send out, which can reveal additional information about the devices.

Shodan can be used to find a wide variety of devices, including:

- Switches
- Routers
- Webcams
- Security cameras
- IACS
- Heating, Ventilation, and Air Conditioning (HVAC) systems

- Smart TVs
- Refrigerators

Shodan can be a valuable tool for security researchers, as it can help them to discover new vulnerabilities and to track the deployment of malware. Shodan can also be used by hackers to find targets for attacks. However, it's important to note that Shodan is not a hacking tool. It's simply a search engine that helps people to find devices connected to the Internet. It's up to the user to decide how to use the information that Shodan provides.

Here are some examples of how Shodan can be used:

- A security researcher can use Shodan to find all of the Modbus devices that are publicly accessible on the Internet. This could help discover new vulnerabilities in Programmable Logic Controllers (PLC) or to track the deployment of malware that targets PLCs.
- A hacker could use Shodan to find all of the routers that are using a specific firmware with a known vulnerability. This could allow them to exploit the vulnerability and gain access to the routers.
- A business could use Shodan to find all of the devices on their network that are exposed to the Internet. This could help identify and mitigate security risks.

Shodan is a powerful tool that can be used for a variety of purposes. It's important to be aware of the potential risks of using Shodan, but it can also be a valuable tool for learning more about the internet and for improving security.

4.2 Exercise: Shodan Enumeration

- Create a personal Shodan account.
- Login to Shodan.
- Discover the Transmission Control Protocol (TCP) port number for the Distributed Network Protocol version 3 (DNP3) protocol.
- Search for DNP3 in Ireland and copy down the information relating to the nearest one to your current location.
 - Who owns it?
 - What is the device?
 - What is the device IP Address?

5 Network Mapper Deep Dive

The following sections of this topic will focus on one aspect of the pen-test, reconnaissance, and in particular using Network Mapper (Nmap)[2] and the Python programming language to perform reconnaissance.

5.1 Network Mapper

nmap is an open source network exploration and security auditing tool. It was designed to rapidly scan large networks, although it works fine against single hosts too. **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.

5.1.1 nmapsi4

A Qt4 front-end interface for **nmap** exists as illustrated in Figure 3.

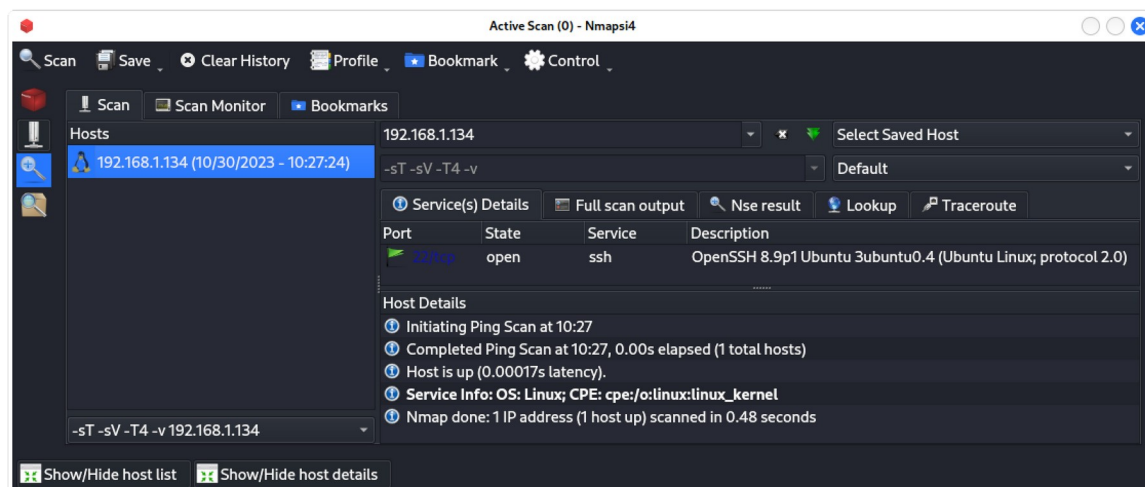


Figure 3: nmapsi4 graphical nmap front-end

5.1.2 Install NMAP

Install **nmap** and **nmapsi4** packages on the operating system.

```
(kali㉿kali)-[~]  
└─$ sudo apt install nmap nmapsi4
```

Test **nmap** version.

```
(kali㉿kali)-[~]  
└─$ nmap --version  
nmap --version  
Nmap version 7.94 ( https://nmap.org )  
Platform: x86_64-pc-linux-gnu  
Compiled with: liblua-5.4.4 openssl-3.0.11 libssh2-1.11.0 libz-1.2.13  
libpcre-8.39 libpcap-1.10.4 nmap-libdnet-1.12 ipv6  
Compiled without:  
Available nsock engines: epoll poll select
```

5.2 Target specification and host discovery techniques

5.2.1 Target Specification

At its simplest **nmap** checks the target for open ports. In the example below the Secure Shell (SSH) and Hyper Text Transfer Protocol (HTTP) daemons are active as TCP ports 22 and 80 are open. This scan of 1,000 TCP ports returns a granular list of states:

5.2.2 open

An application is actively accepting TCP connections, User Datagram Protocol (UDP) datagrams or Stream Control Transmission Protocol (SCTP) associations on this port. Finding these is often the primary goal of port scanning. Security-minded people know that each open port is an avenue for attack. Attackers and pen-testers want to exploit the open ports, while administrators try to close or protect them with firewalls without thwarting legitimate users. Open ports are also interesting for non-security scans because they show services available for use on the network.

5.2.3 closed

A closed port is accessible (it receives and responds to **nmap** probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (host discovery, or ping scanning), and as part of OS detection. Because closed ports are reachable, it may be worth scanning later in case some open up. Administrators may want to consider blocking such ports with a firewall. Then they would appear in the filtered state, discussed next.

5.2.4 filtered

nmap cannot determine whether the port is open because packet filtering prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. These ports frustrate attackers because they provide so little information. Sometimes they respond with ICMP error messages such as type 3 code 13 (destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common. This forces **nmap** to retry several times just in case the probe was dropped due to network congestion rather than filtering. This slows down the scan dramatically.

5.2.5 unfiltered

The unfiltered state means that a port is accessible, but **nmap** is unable to determine whether it is open or closed. Only the ACK scan, which is used to map firewall rulesets, classifies ports into this state. Scanning unfiltered ports with other scan types such as Window scan, SYN scan, or FIN scan, may help resolve whether the port is open.

5.2.6 open|filtered

nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in which open ports give no response. The lack of response could also mean that a packet filter dropped the probe or any response it elicited. So **nmap** does not know for sure whether the port is open or being filtered. The UDP, IP protocol, FIN, NULL, and Xmas scans classify ports this way.

5.2.7 closed|filtered

This state is used when **nmap** is unable to determine whether a port is closed or filtered. It is only used for the IP ID idle scan.

```
(kali㉿kali)-[~]  
$ nmap 10.0.2.7  
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 14:42 GMT  
Nmap scan report for 10.0.2.7  
Host is up (0.00087s latency).  
Not shown: 998 closed ports  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http
```

A scan can also be carried out on a range of addresses.

```
(kali㉿kali)-[~]  
$ nmap 10.0.2.0/24  
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 14:46 GMT  
Nmap scan report for _gateway (10.0.2.1)  
Host is up (0.00058s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
53/tcp    open  domain  
  
Nmap scan report for 10.0.2.2  
Host is up (0.00061s latency).  
Not shown: 998 closed ports
```

```
PORT      STATE SERVICE
22/tcp    open  ssh
631/tcp    open  ipp

Nmap scan report for scapy (10.0.2.6)
Host is up (0.00025s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap scan report for 10.0.2.7
Host is up (0.00020s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 256 IP addresses (4 hosts up) scanned in 3.15 seconds
```

5.2.8 Host Discovery

One of the very first steps in any network reconnaissance is to reduce a set of IP ranges into a list of active or interesting hosts. Scanning every port of every single IP address is slow and usually unnecessary. Of course what makes a host interesting depends greatly on the scan purposes. Network administrators may only be interested in hosts running a certain service, while security auditors may care about every single device with an IP address. An administrator may be comfortable using just an ICMP ping to locate hosts on his internal network, while an external penetration tester may use a diverse set of dozens of probes in an attempt to evade firewall restrictions.

Because host discovery needs are so diverse, **nmap** offers a wide variety of options for customising the techniques used. Host discovery is sometimes called ping scan, but it goes well beyond the simple ICMP echo request packets associated with the standard ping tool. Users can skip the discovery step entirely with a list scan (**-sL**) or by disabling host discovery (**-Pn**), or engage the network with arbitrary combinations of multi-port TCP SYN/ACK, UDP, SCTP INIT and ICMP probes. The goal of these probes is to solicit responses which demonstrate that an IP address is actually active (is being used by a host or network device). On many networks, only a small percentage of IP addresses are active at any given time. This is particularly common with RFC1918 private address blocks such as 10.0.0.0/8. That network has 16 million IPs but a company may only use a small fraction of them. Host discovery can find those machines in a sparsely allocated sea of IP addresses.

If no host discovery options are given, **nmap** sends an ICMP echo request, a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request. (For IPv6, the ICMP timestamp request is omitted because it is not part of ICMPv6.) The exceptions to this are the Address Resolution Protocol (ARP) (for IPv4) and Neighbour Discovery (for IPv6) scans which are used for any targets on a local Ethernet network. For unprivileged users, the default probes are a SYN packet to ports 80 and 443 using the connect system call. This host discovery is often sufficient when scanning local networks, but a more comprehensive set of discovery probes is recommended for security auditing.

The **-P*** options (which select ping types) can be combined. The odds of penetrating strict firewalls can be increased by sending many probe types using different TCP ports/flags and ICMP codes. Also note that ARP/Neighbour Discovery is carried out

by default against targets on a local Ethernet network even if other **-P*** options are specified, because it is almost always faster and more effective.

By default, **nmap** performs host discovery and then performs a port scan against each host it determines is online. This is true even if non-default host discovery types are specified such as UDP probes (**-PU**). Table 1 lists the host discovery control options:

Table 1: Nmap host discovery control options

Switch	Example	Description
-sL	<code>nmap 192.168.1.1-3 -sL</code>	No Scan. List targets only
-sn	<code>nmap 192.168.1.1/24 -sn</code>	Disable port scanning. Host discovery only.
-Pn	<code>nmap 192.168.1.1-5 -Pn</code>	Disable host discovery. Port scan only.
-PS	<code>nmap 192.168.1.1-5 -PS22-25,80</code>	TCP SYN discovery on port x. Port 80 by default
-PA	<code>nmap 192.168.1.1-5 -PA22-25,80</code>	TCP ACK discovery on port x. Port 80 by default
-PU	<code>nmap 192.168.1.1-5 -PU53</code>	UDP discovery on port x. Port 40125 by default
-PR	<code>nmap 192.168.1.1-1/24 -PR</code>	ARP discovery on local network
-n	<code>nmap 192.168.1.1 -n</code>	Never do DNS resolution

5.3 Scan techniques, port specification, and version/OS detection

Most of the scan types are only available to privileged users. This is because they send and receive raw packets, which requires **root** access.

While **nmap** attempts to produce accurate results, bear in mind that all of its insights are based on packets returned by the target machines (or firewalls in front of them). Such hosts may be untrustworthy and send responses intended to confuse or mislead **nmap**. Much more common are non-RFC-compliant hosts that do not respond as they should to **nmap** probes. FIN, NULL, and Xmas scans are particularly susceptible to this problem. Such issues are specific to certain scan types and so are discussed in the individual scan type entries.

Table 2 lists the port scan techniques supported by **nmap**. Only one method may be used at a time, except that UDP scan (**-sU**) and any one of the SCTP scan types (**-sY**, **-sZ**) may be combined with any one of the TCP scan types. Port scan type options are of the form **-s<C>**, where **<C>** is a prominent character in the scan name, usually the first. The one exception to this is the deprecated FTP bounce scan (**-b**). By default, **nmap** performs a SYN Scan, though it substitutes a connect scan if the user does not have proper privileges to send raw packets. Of the scans listed in this section, unprivileged users can only execute connect and FTP bounce scans.

Table 2: Nmap port scan techniques

Switch	Example	Description
-sS	nmap 192.168.1.1 -sS	TCP SYN port scan (Default)
-sT	nmap 192.168.1.1 -sT	TCP connect port scan (Default without root privilege)
-sU	nmap 192.168.1.1 -sU	UDP port scan
-sA	nmap 192.168.1.1 -sA	TCP ACK port scan
-sW	nmap 192.168.1.1 -sW	TCP Window port scan
-sM	nmap 192.168.1.1 -sM	TCP Maimon port scan

```
(kali㉿kali)-[~]
└─$ sudo nmap 10.0.2.7 -sS
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 17:23 GMT
Nmap scan report for 10.0.2.7
Host is up (0.00016s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:A3:92:BA (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.73 seconds
```

```
(kali㉿kali)-[~]
└─$ sudo nmap 10.0.2.7 -sU
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 17:24 GMT
```

5.3.1 Port Specification

Table 3 outlines various Nmap commands and their uses for port scanning.

Table 3: Nmap port specification

Switch	Example	Description
-p	nmap 192.168.1.1 -p 21	Port scan for port x
-p	nmap 192.168.1.1 -p 21-100	Port range
-p	nmap 192.168.1.1 -p U:53,T:21-25,80	Port scan multiple TCP and UDP ports
-p-	nmap 192.168.1.1 -p-	Port scan all ports
-p	nmap 192.168.1.1 -p http,https	Port scan from service name
-F	nmap 192.168.1.1 -F	Fast port scan (100 ports)
--top-ports	nmap 192.168.1.1 --top-ports 2000	Port scan the top x ports
-p-65535	nmap 192.168.1.1 -p-65535	Leaving off initial port in range makes the scan start at port 1
-p0-	nmap 192.168.1.1 -p0-	Leaving off end port in range makes the scan go through to port 65535

5.3.2 Service and Version detection

Table 4 provides an overview of Nmap commands used for service and version detection.

Table 4: Nmap Service and Version detection

Switch	Example	Description
-sV	<code>nmap 192.168.1.1 -sV</code>	Attempts to determine the version of the service running on port
-sV --version-intensity	<code>nmap 192.168.1.1 -sV --version-intensity 8</code>	Intensity level 0 to 9. Higher number increases possibility of correctness
-sV --version-light	<code>nmap 192.168.1.1 -sV --version-light</code>	Enable light mode. Lower possibility of correctness. Faster
-sV --version-all	<code>nmap 192.168.1.1 -sV --version-all</code>	Enable intensity level 9. Higher possibility of correctness. Slower
-A	<code>nmap 192.168.1.1 -A</code>	Enables OS detection, version detection, script scanning, and traceroute

5.3.3 Operating System Detection

Table 5 details Nmap commands used for operating system (OS) detection.

Table 5: Nmap OS detection

Switch	Example	Description
-O	<code>nmap 192.168.1.1 -O</code>	Remote OS detection using TCP/IP stack fingerprinting
-O --osscan-limit	<code>nmap 192.168.1.1 -O --osscan-limit</code>	If at least one open and one closed TCP port are not found it will not try OS detection against host
-O --osscan-guess	<code>nmap 192.168.1.1 -O --osscan-guess</code>	Makes nmap guess more aggressively
-O --max-os-tries	<code>nmap 192.168.1.1 -O --max-os-tries 1</code>	Set the maximum number x of OS detection tries against a target
-A	<code>nmap 192.168.1.1 -A</code>	Enables OS detection, version detection, script scanning, and traceroute

5.4 Timing, performance optimisation, and output management

5.4.1 Timing

Table 6 lists the different Nmap timing templates and their descriptions.

Table 6: Nmap timing options

Switch	Example	Description
-T0	<code>nmap 192.168.1.1 -T0</code>	Paranoid (0) Intrusion Detection System evasion
-T1	<code>nmap 192.168.1.1 -T1</code>	Sneaky (1) Intrusion Detection System evasion
-T2	<code>nmap 192.168.1.1 -T2</code>	Polite (2) slows down the scan to use less bandwidth and use less target machine resources
-T3	<code>nmap 192.168.1.1 -T3</code>	Normal (3) which is default speed
-T4	<code>nmap 192.168.1.1 -T4</code>	Aggressive (4) speeds scans; assumes you are on a reasonably fast and reliable network
-T5	<code>nmap 192.168.1.1 -T5</code>	Insane (5) speeds scan; assumes you are on an extraordinarily fast network

5.4.2 Performance

Table 7 outlines various Nmap commands for fine-tuning scan performance.

Table 7: Nmap fine-tuning scan performance options

Switch	Example input	Description
<code>--host-timeout <time></code>	<code>1s; 4m; 2h</code>	Give up on target after this long
<code>--min-rtt-timeout/ max-rtt-timeout/ initial-rtt-timeout <time></code>	<code>1s; 4m; 2h</code>	Specifies probe round trip time
<code>--min-hostgroup/ max-hostgroup <size></code>	<code>50; 1024</code>	Parallel host scan group sizes
<code>--min-parallelism/max-parallelism <numprobes></code>	<code>10; 1</code>	Probe parallelisation
<code>--scan-delay/ --max-scan-delay <time></code>	<code>20ms; 2s; 4m; 5h</code>	Adjust delay between probes
<code>--max-retries <tries></code>	<code>3</code>	Specify the maximum number of port scan probe retransmissions
<code>--min-rate <number></code>	<code>100</code>	Send packets no slower than <number> per second
<code>--max-rate <number></code>	<code>100</code>	Send packets no faster than <number> per second

5.4.3 Output

Table 8 provides an overview of Nmap commands for controlling output and debugging and Table 9 presents a collection of Nmap command-line examples and their descriptions, demonstrating various uses of the tool.

Table 8: Nmap output options

Switch	Example	Description
-oN	<code>nmap 192.168.1.1 -oN normal.file</code>	Normal output to the file <code>normal.file</code>
-oX	<code>nmap 192.168.1.1 -oX xml.file</code>	XML output to the file <code>xml.file</code>
-oG	<code>nmap 192.168.1.1 -oG grep.file</code>	Grepable output to the file <code>grep.file</code>
-oA	<code>nmap 192.168.1.1 -oA results</code>	Output in the three major formats at once
-oG -	<code>nmap 192.168.1.1 -oG -</code>	Grepable output to screen. -oN -, -oX - also usable
--append-output	<code>nmap 192.168.1.1 -oN file.file --append-output</code>	Append a scan to a previous scan file
-v	<code>nmap 192.168.1.1 -v</code>	Increase the verbosity level (use -vv or more for greater effect)
-d	<code>nmap 192.168.1.1 -d</code>	Increase debugging level (use -dd or more for greater effect)
--reason	<code>nmap 192.168.1.1 --reason</code>	Display the reason a port is in a particular state, same output as -vv
--open	<code>nmap 192.168.1.1 --open</code>	Only show open (or possibly open) ports
--packet-trace	<code>nmap 192.168.1.1 -T4 --packet-trace</code>	Show all packets sent and received
--iflist	<code>nmap --iflist</code>	Shows the host interfaces and routes
--resume	<code>nmap --resume results.file</code>	Resume a scan

5.4.3.1 NMAP Output examples

Table 9: Nmap output examples

Command	Description
<code>nmap -p80 -sV -oG - --open 192.168.1.1/24 grep open</code>	Scan for web servers and grep to show which IPs are running web servers
<code>nmap -iR 10 -n -oX out.xml grep "Nmap" cut -d " " -f5 > live-hosts.txt</code>	Generate a list of the IPs of live hosts
<code>nmap -iR 10 -n -oX out2.xml grep "Nmap" cut -d " " -f5 >> live-hosts.txt</code>	Append IP to the list of live hosts
<code>ndiff scan1.xml scan2.xml</code>	Compare output from nmap using the ndif
<code>xsltproc nmap.xml -o nmap.html</code>	Convert nmap xml files to html files
<code>grep " open " results.nmap sed -r 's/ +/ /g' sort uniq -c sort -rn less</code>	Reverse sorted list of how often ports turn up

5.5 XML output

- **-oX <file>**: Output scan in format, using [dash] i.e. '-' in lieu of <file> redirects to standard out (**stdout**).
- **-p 22-1024**: The well-known port range.
- **-sV**: Enables version detection.

```
(kali㉿kali)-[~]
$ nmap -oX - -p 22-1024 -sV 192.168.0.1
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xml-stylesheet href="file:///usr/bin/./share/nmap/nmap.xsl"
type="text/xsl"?>
<!-- Nmap 7.80 scan initiated Tue Jul  6 09:52:09 2021 as: nmap -oX
- -p 22-1024 -sV 192.168.0.1 -->
<nmaprun scanner="nmap" args="nmap -oX - -p 22-1024 -sV
192.168.0.1" start="1625561529" startstr="Tue Jul  6 09:52:09 2021"
version="7.80" xmloutputversion="1.04">
<scaninfo type="connect" protocol="tcp" numservices="1003"
services="22-1024"/>
<verbose level="0"/>
<debugging level="0"/>
<host starttime="1625561529" endtime="1625561536"><status
state="up" reason="syn-ack" reason_ttl="0"/>
<address addr="192.168.0.1" addrtype="ipv4"/>
<hostnames>
<hostname name="_gateway" type="PTR"/>
</hostnames>
<ports><extraports state="closed" count="998">
<extrareasons reason="conn-refused" count="998"/>
</extraports>
<port protocol="tcp" portid="22"><state state="filtered"
reason="no-response" reason_ttl="0"/><service name="ssh"
method="table" conf="3"/></port>
<port protocol="tcp" portid="23"><state state="filtered"
reason="no-response" reason_ttl="0"/><service name="telnet"
method="table" conf="3"/></port>
<port protocol="tcp" portid="53"><state state="open" reason="syn-
ack" reason_ttl="0"/><service name="domain" product="dnsmasq"
version="2.78" method="probed"
conf="10"><cpe>cpe:/a:thekelleys:dnsmasq:2.78</cpe></service></port>
<port protocol="tcp" portid="80"><state state="open" reason="syn-
ack" reason_ttl="0"/><service name="http" product="lighttpd"
method="probed"
conf="10"><cpe>cpe:/a:lighttpd:lighttpd</cpe></service></port>
<port protocol="tcp" portid="111"><state state="filtered"
reason="no-response" reason_ttl="0"/><service name="rpcbind"
method="table" conf="3"/></port>
</ports>
<times srtt="5523" rttvar="178" to="100000"/>
</host>
<runstats><finished time="1625561536" timestr="Tue Jul  6 09:52:16
2021" elapsed="7.53" summary="Nmap done at Tue Jul  6 09:52:16
2021; 1 IP address (1 host up) scanned in 7.53 seconds"
exit="success"/><hosts up="1" down="0" total="1"/>
</runstats>
</nmaprun>
```

5.6 Firewall/IDS evasion, spoofing, and anonymous scanning

Table 10 details various Nmap commands and techniques for evasion and spoofing.

Table 10: Nmap commands and techniques for evasion and spoofing

Switch	Example	Description
-f	<code>nmap 192.168.1.1 -f</code>	Requested scan (including ping scans) use tiny fragmented IP packets. Harder for packet filters
--mtu	<code>nmap 192.168.1.1 --mtu 32</code>	Set your own offset size
-D	<code>nmap -D 192.168.1.101,192.168.1.102,192.168.1.103,192.168.1.23 192.168.1.1</code>	Send scans from spoofed IPs
-D	<code>nmap -D decoy-ip1,decoy-ip2,your-own-ip,decoy-ip3,decoy-ip4 remote-host-ip</code>	Above example explained
-S	<code>nmap -S www.microsoft.com www.facebook.com</code>	Scan Facebook from Microsoft (-e eth0 -Pn may be required)
-g	<code>nmap -g 53 192.168.1.1</code>	Use given source port number
--proxies	<code>nmap --proxies http://192.168.1.1:8080,http://192.168.1.2:8080 192.168.1.1</code>	Relay connections through HTTP/SOCKS4 proxies
--data-length	<code>nmap --data-length 200 192.168.1.1</code>	Appends random data to sent packets

5.6.1 Example IDS Evasion command

```
(kali㉿kali)-[~]
└─$ nmap -f -t 0 -n -Pn --data-length 200 -D 192.168.1.101,192.168.1.102, 192.168.1.103,192.168.1.23 192.168.1.1
```

5.6.2 Use NMAP anonymously

For anonymous use of **nmap** it is possible to do so using The Onion Router (**TOR**) and **ProxyChains**. ProxyChains redirects TCP connections through proxy servers.

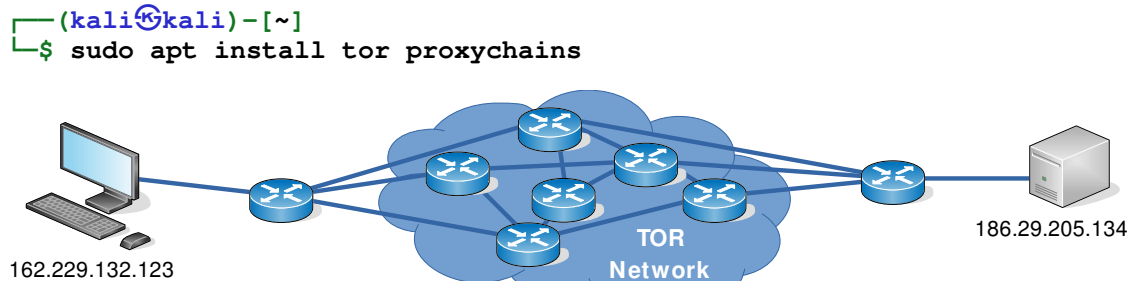


Figure 4: Using the TOR network

Consider the following **nmap** scan through a proxy chain via the TOR network. Some additional options here:

- **-sT**: TCP connect scan, instead of writing raw packets as most other scan types do, **nmap** asks the underlying OS to establish a connection with the target machine and port by issuing the connect system call. This more exactly simulates what network enabled applications would do. Basically, **nmap** is making use of the OS own Berkeley Socket API.

```
(kali@kali)-[~]
$ proxychains nmap -Pn -sT -p 22,80 186.29.205.134
```

ProxyChains-3.1 (<http://proxychains.sf.net>)

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-11-04 22:07 EAT
|S-chain|-<>-127.0.0.1:9050-<>-186.29.205.134:80-<>-OK
|S-chain|-<>-127.0.0.1:9050-<>-186.29.205.134:22-<>-OK
Nmap scan report for 186.29.205.13
Host is up (0.61s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
```

Adding an additional option to detect the OS.:

- **-sV**: Enable version detection. It can be used to help differentiate the truly open ports from the filtered ones.

```
(kali㉿kali)-[~]  
$ proxychains nmap -Pn -sV -sT -p 22,80 186.29.205.134  
  
ProxyChains-3.1 (http://proxychains.sf.net)  
  
Starting Nmap 6.40 ( http://nmap.org ) at 2015-11-10 12:13 EAT  
|S-chain|-<>-127.0.0.1:9050-<><>-186.29.205.134:22-<><>-OK  
|S-chain|-<>-127.0.0.1:9050-<><>-186.29.205.134:80-<><>-OK  
Nmap scan report for li489-237.members.linode.com (186.29.205.134)  
Host is up (0.71s latency).  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)  
80/tcp    open  http     Apache httpd 2.2.22 ((Debian))  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Attempt at SSH connection using user root failed but as it passed through the TOR network the attempt was anonymous.

```
(kali㉿kali)-[~]  
$ proxychains ssh root@186.29.205.134  
  
ProxyChains-3.1 (http://proxychains.sf.net)  
|S-chain|-<>-127.0.0.1:9050-<><>-186.29.205.134:22-<><>-OK  
root@186.29.205.134's password:  
Permission denied, please try again.  
root@186.29.205.134's password:  
Permission denied, please try again.  
root@186.29.205.134's password:  
Permission denied (publickey,password).
```

On the server that the compromise attempt occurred check the authentication logs.

```
root@ece:~# tail /var/log/auth.log  
  
Nov  4 19:09:26 ece sshd[1146]: Failed password for root from  
207.244.70.35 port 45909 ssh2  
Nov  4 19:09:33 ece sshd[1146]: Failed password for root from  
207.244.70.35 port 45909 ssh2  
Nov  4 19:09:40 ece sshd[1146]: Failed password for root from  
207.244.70.35 port 45909 ssh2  
Nov  4 19:09:40 ece sshd[1146]: Connection closed by 207.244.70.35  
[preauth]  
Nov  4 19:09:40 ece sshd[1146]: PAM 2 more authentication failures;  
logname= uid=0 euid=0 tty=ssh ruser= rhost=207.244.70.35 user=root
```

Note the IP address from where the attempt connection originated, it is not from the source 62.229.132.123 but from 207.244.70.35 which is the edge of the TOR at that time for that connection.


```
root@ece:~# tail /var/log/auth.log
Nov 10 09:46:10 ece sshd[21706]: Failed password for root from
43.229.53.25 port 11978 ssh2
Nov 10 09:46:12 ece sshd[21706]: Failed password for root from
43.229.53.25 port 11978 ssh2
Nov 10 09:46:12 ece sshd[21706]: Received disconnect from
43.229.53.25: 11: [preauth]
Nov 10 09:46:12 ece sshd[21706]: PAM 2 more authentication failures;
logname= uid=0 euid=0 tty=ssh ruser= rhost=43.229.53.25 user=root
Nov 10 09:46:13 ece sshd[21708]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=43.229.53.25
user=root
Nov 10 09:46:15 ece sshd[21708]: Failed password for root from
43.229.53.25 port 28216 ssh2
Nov 10 09:46:17 ece sshd[21708]: Failed password for root from
43.229.53.25 port 28216 ssh2
Nov 10 09:46:19 ece sshd[21708]: Failed password for root from
43.229.53.25 port 28216 ssh2
Nov 10 09:46:19 ece sshd[21708]: Received disconnect from
43.229.53.25: 11: [preauth]
Nov 10 09:46:19 ece sshd[21708]: PAM 2 more authentication failures;
logname= uid=0 euid=0 tty=ssh ruser= rhost=43.229.53.25 user=root
```

Each time the source is a different address as the exit point from TOR changes.

IP ADDRESS INFORMATION	
IP Address	43.229.53.25
Hostname	43.229.53.25
Network	Asia Pacific Network Information Centre
Country	JP - JAPAN
Latitude	36
Longitude	138
IP Range	43.0.0.0 - 43.233.35.255
IP Network	American Registry for Internet Numbers (ARIN)

IP ADDRESS INFORMATION	
IP Address	81.7.15.115
Hostname	81-7-15-115.blue.kundencontroller.de
Network	RIPE Network Coordination Centre
Country	DE - GERMANY
Latitude	51
Longitude	9
IP Range	81.7.0.0 - 81.7.63.255
IP Network	American Registry for Internet Numbers (ARIN)

Figure 6: Source for connections through TOR

5.7 NMAP Scripting Engine & debugging options

The NSE[3] is a powerful and advanced functionality within **nmap**, enabling users to develop custom scripts for automated scanning and extending the capabilities of **nmap**. These scripts are written in Lua, a fast and potent programming language, allowing for tasks like host discovery, service scanning, and version detection. NSE aids in the identification of running services, detecting vulnerabilities, and gathering additional target information. Security professionals can leverage NSE to automate diverse scanning tasks, such as identifying specific services on devices or detecting vulnerabilities across IP address ranges. NSE proves to be an indispensable tool for efficiently scanning networks and pinpointing potential vulnerabilities.

```
└─(kali🌀kali)-[~]
```

```

└─$ ls /usr/share/nmap/scripts | grep .nse | wc -l
598

```

Table 11 details Nmap commands used for running NSE scripts and Table 14 provides examples of Nmap commands for various scanning tasks, including web server analysis, DNS enumeration, and vulnerability detection.

Table 11: Nmap NSE Scripts

Switch	Example	Description
-sC	nmap 192.168.1.1 -sC	Scan with default NSE scripts. Considered useful for discovery and safe
--script default	nmap 192.168.1.1 --script default	Scan with default NSE scripts. Considered useful for discovery and safe
--script	nmap 192.168.1.1 --script=banner	Scan with a single script. Example banner
--script	nmap 192.168.1.1 --script=http*	Scan with a wildcard. Example http
--script	nmap 192.168.1.1 --script=http,banner	Scan with two scripts. Example http and banner
--script	nmap 192.168.1.1 --script "not intrusive"	Scan default, but remove intrusive scripts
--script-args	nmap --script snmp-sysdescr --script-args snmpcommunity=admin 192.168.1.1	NSE script with arguments

5.7.1 Miscellaneous Options

Table 12 shows some common Nmap switches, including a brief description and an example of how to use them.

Table 12: Nmap miscellaneous options

Switch	Example	Description
-6	nmap -6 2607:f0d0:1002:51::4	Enable IPv6 scanning
-h	nmap -h	nmap help screen

5.7.2 Other useful nmap Commands

Table 13 shows some other useful Nmap commands that are useful for network discovery and host enumeration without performing a full port scan.

Table 13: Other useful Nmap commands

Command	Description
<code>nmap -iR 10 -PS22-25,80,113,1050,35000 -v -sn</code>	Discovery only on ports x, no port scan
<code>nmap 192.168.1.1-1/24 -PR -sn -vv</code>	ARP discovery only on local network, no port scan
<code>nmap -iR 10 -sn -traceroute</code>	Traceroute to random targets, no port scan
<code>nmap 192.168.1.1-50 -sL --dns-server 192.168.1.1</code>	Query the Internal DNS for hosts, list targets only

5.7.3 Debugging, Verbosity and Reason

-v: Increase verbosity level, use **-vv** or more for greater effect.

-d: Increase debugging level, use **-dd** or more for greater effect.

--reason: Display the reason a port is in a particular state.

```
(kali㉿kali)-[~]
$
nmap 192.168.0.1 --reason -vv -d
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-26 14:11 IST
PORTS: Using top 1000 ports found open (TCP:1000, UDP:0, SCTP:0)
----- Timing report -----
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0
-----
Initiating Ping Scan at 14:11
Scanning 192.168.0.1 [2 ports]
Completed Ping Scan at 14:11, 0.00s elapsed (1 total hosts)
Overall sending rates: 962.93 packets / s.
mass_rdns: Using DNS server 127.0.0.53
Initiating Parallel DNS resolution of 1 host. at 14:11
mass_rdns: 0.04s 0/1 [#: 1, OK: 0, NX: 0, DR: 0, SF: 0, TR: 1]
Completed Parallel DNS resolution of 1 host. at 14:11, 0.04s elapsed
DNS resolution of 1 IPs took 0.04s. Mode: Async [#: 1, OK: 1, NX: 0, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating Connect Scan at 14:11
Scanning _gateway (192.168.0.1) [1000 ports]
Discovered open port 53/tcp on 192.168.0.1
Discovered open port 80/tcp on 192.168.0.1
Discovered open port 49152/tcp on 192.168.0.1
Completed Connect Scan at 14:11, 1.20s elapsed (1000 total ports)
Overall sending rates: 834.20 packets / s.
Nmap scan report for _gateway (192.168.0.1)
Host is up, received syn-ack (0.0057s latency).
Scanned at 2021-05-26 14:11:42 IST for 2s
Not shown: 994 closed ports
Reason: 994 conn-refused
PORT      STATE      SERVICE REASON
22/tcp    filtered  ssh      no-response
```

```

23/tcp    filtered telnet  no-response
53/tcp    open      domain  syn-ack
80/tcp    open      http    syn-ack
111/tcp   filtered rpcbind no-response
49152/tcp open      unknown syn-ack
Final times for host: srth: 5686 rttvar: 131  to: 100000

```

Read from /usr/bin/./share/nmap: nmap-payloads nmap-services.
 Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds

5.7.3.1 Useful NSE Scripts

Table 14: Nmap NSE script examples

Command	Description
<code>nmap -Pn --script=http-sitemap-generator scanme.nmap.org</code>	http site map generator
<code>nmap -n -Pn -p 80 --open -sV -vvv --script banner,http-title -iR 1000</code>	Fast search for random web servers
<code>nmap -Pn --script=dns-brute domain.com</code>	Brute forces DNS hostnames guessing subdomains
<code>nmap -n -Pn -vv -O -sV --script smb-enum*,smb-ls,smb-mbenum,smb-os-discovery,smb-s*,smb-vuln*,smbv2* -vv 192.168.1.1</code>	Safe SMB scripts to run
<code>nmap --script whois* domain.com</code>	Whois query
<code>nmap -p80 --script http-unsafe-output-escaping scanme.nmap.org</code>	Detect cross site scripting vulnerabilities
<code>nmap -p80 --script http-sql-injection scanme.nmap.org</code>	Check for SQL injections

6 Operational Technology & Legacy Scanning

6.1 The inherent dangers of scanning Automated Control Systems

Scanning legacy IACS can result in device freezing or even permanent malfunction (commonly referred to as "*bricking*"). The absence of sufficient security mechanisms and the utilisation of outdated software significantly contribute to these issues, especially when the system receives an NMAP-TCP packet.

Legacy IACS are primarily engineered for real-time functionality, lacking inherent security features. This design approach renders them more susceptible to cyber threats. Given their integration into Critical National Infrastructure (CNI), any compromise, even as minimal as an NMAP scan, carries potentially severe consequences. Safeguarding these systems against such risks is paramount.

Adversaries are indifferent to a device's resistance to scanning; their objective is to infiltrate the network and enumerate connected devices. Regardless of a device's robustness to scanning, determined adversaries seek unauthorised access to the network, emphasising the critical importance of comprehensive security measures to thwart potential breaches.

6.1.1 Scanning IACS in a Penetration Test

To conduct or abstain from scanning poses a critical decision. An NMAP scan functions as a practical stress test for your OT, revealing its response to such probing. It serves as a way to gauge the system's resilience under simulated scanning conditions. However, caution must be exercised due to potential negative impacts on system stability and functionality during the scanning process. A thorough evaluation of the benefits and risks of scanning is vital before proceeding, ensuring that the benefits outweigh the potential disruptions or damage that may occur. Ultimately, informed judgment and readiness are essential in making an appropriate decision regarding scanning.

6.1.1.1 *Never scan a live system*

Prioritise obtaining appropriate permissions before initiating any network scan. It is crucial to restrict scanning activities to instances when the target is in a maintenance or downtime phase. This precaution ensures that the scanning process doesn't interfere with active operations or disrupt critical services. Timing the scan with precision is essential to minimise any potential impact on the target system's performance. Adhering to this practice safeguards against unintended consequences and aligns with responsible scanning procedures. Always exercise caution and the target system before proceeding with a network scan.

6.1.1.2 Scan Option: Timing -Tx

Enhancing the scan's safety involves leveraging the timing options provided by NMAP. Utilise the **-T** option followed by a number ranging from 0 to 5 to tailor the scan's aggressiveness and mitigate risks. For OT environments, a recommended approach is to opt for either **-T1** or **-T2** to strike a balance between thoroughness and caution. However, if the intent is to stress-test the system, consider the more aggressive option, **-T5**, for a rigorous and exhaustive assessment. Choose the appropriate timing option in line with the specific security and operational requirements of the OT network.

6.1.1.3 Scan Option: Full TCP handshake -sT

Another measure to enhance scanning safety involves utilising the **-sT** option in NMAP. This option specifies a TCP connect scan, which is generally safer than other scan types. **-sT** establishes a full TCP connection with the target, minimising the chance of disrupting services or causing system damage during the scan. When conducting scans in sensitive OT environments, prioritising safety through a TCP connect scan is recommended to reduce potential negative impacts while still gaining valuable insights into the target network.

6.1.1.4 Scan Option: Limit parallel probing

Utilising the **--max-parallelism 1** option in the scan configuration limits the parallel operations to one at a time. This setting minimises the load on the target system, significantly reducing the risk of disruptions or potential damage during the scan. In OT environments, where cautious scanning is vital, employing this option ensures a meticulous and safe scanning process, aligning with the imperative to prioritise system stability and security.

6.1.2 Quiz

1. Which scan timing options are recommended for OT environments to balance thoroughness and caution?
☐ **-T1** ☐ **-T2** ☐ **-T3** ☐ **-T4** ☐ **-T5**
2. What does the **-sT** option in NMAP signify for enhancing scanning safety in OT networks?
☐ UDP Connect Scan ☐ TCP Connect Scan ☐ SYN Stealth Scan
3. How does the **--max-parallelism 1** option in a scan configuration contribute to safe scanning in OT environments?
☐ It increases the number of parallel operations, enhancing scan speed.
☐ It limits parallel probing to one at a time, reducing the risk of disruptions
☐ It scans multiple hosts simultaneously to save time.

6.2 IACS-specific scripts for NSE

The NSE is a powerful and advanced functionality within NMAP, and it can be enhanced by adding more OT specific scripts from github. To integrate a collection of ICS enumeration tools from the Redpoint repository, clone this repository into the NMAP script folder:

```
(kali㉿kali)-[~]
└─$ cd /usr/share/nmap/scripts

(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo git clone https://github.com/digitalbond/Redpoint.git
Cloning into 'Redpoint'...
remote: Enumerating objects: 343, done.
remote: Total 343 (delta 0), reused 0 (delta 0), pack-reused 343
Receiving objects: 100% (343/343), 191.10 KiB | 1.59 MiB/s, done.
Resolving deltas: 100% (194/194), done.

(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ ls Redpoint/
atg-info.nse      dnp3-info.nse      modicon-info.nse      proconos-info.nse
BACnet-discover-enumerate.nse  enip-enumerate.nse  omrontcp-info.nse
README.md        codesys-v2-discover.nse  fox-info.nse          omronudp-info.nse
s7-enumerate.nse  cspv4-info.nse      LICENSE               pcworx-info.nse

(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -p 502 --script Redpoint/modicon-info.nse -sV
192.168.1.134
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-30 11:21 EDT
NSE: DEPRECATION WARNING: bin.lua is deprecated. Please use Lua 5.3
string.pack
Nmap scan report for riomhaire-OB (192.168.1.134)
Host is up (0.00050s latency).

PORT      STATE SERVICE VERSION
502/tcp   open  mbap?

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 135.69 seconds
```

6.3 Exercise: Safe OT Scanning

Carry out a pen-test reconnaissance on the IP address given to you by the lecturer.

7 Active Defence & Deception

7.1 Deploying Honeypots and canaries to detect early-stage reconnaissance.

7.1.1 Honey Pot

A honey pot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorised use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated, (un)protected, and monitored, and which seems to contain information or a resource of value to attackers.

A honey pot that masquerades as an open proxy to monitor and record those using the system is a '*sugarcane*'. Honey pots should have no production value, and hence should not see any legitimate traffic or activity. Whatever they capture can then be surmised as malicious or unauthorised. One practical implication of this is honey pots that thwart spam by masquerading as the type of systems abused by spammers. They categorise trapped material 100% accurately.

7.1.1.1 Enticement

A Honey Pot placed with open security vulnerabilities and services with known exploits is enticement. The opportunity for illegal or unauthorised activity is provided but the perpetrator makes his/her own decision to perform the exploit.

7.1.1.2 Entrapment

This is where the honey pot actively solicits subjects to access it and then the owner charges them with unauthorised intrusion. This type of activity is illegal.

7.1.1.3 Vulnerability scanner

A vulnerability scanner is a software program designed to search for and map systems for weaknesses in an application, computer or network. This is a four step process:

1. The scanner will first look for active Internet Protocol (IP) addresses, open ports, Operating Systems (OS) and any applications running.
2. The scanner creates a report or moves to the next step.
3. The scanner attempts to determine the patch level of the OS or applications. In this process the scanner can cause an exploit of the vulnerability such as crash the OS or application.
4. In the final phase the scanner may attempt to exploit the vulnerability.

Scanners may either be malicious or friendly. Friendly scanners usually stop at step 2 and occasionally step 3 but never go to step 4.

7.1.1.4 Types of vulnerability scanners

- **Port Scanner**
 - Application designed to probe a network host for open ports, example: NMAP.
- **Network Scanner**
 - Application used to retrieve user names, and info on groups, shares and services of networked computers. This type of program scans networks for vulnerabilities in the security of that network, examples OpenVAS and Metasploit.
- **Web Application Security Scanner**
 - Application that communicates with a web application through the web front-end in order to identify potential security vulnerabilities in the web application and architectural weaknesses, example: OWASP Zed Attack Proxy (ZAP).
- **Computer worm**
 - A computer worm is a self-replicating computer program. It uses a network to send copies of itself to other nodes (computers on the network) and it may do so without any user intervention. Unlike a virus, it does not need to attach itself to an existing program. Worms almost always cause at least some harm to the network, if only by consuming bandwidth, whereas viruses almost always corrupt or devour files on a targeted computer.

7.1.2 Penetration Testing

Penetration testing is a method of evaluating the security of a computer system or network by simulating an attack from a malicious source, known as a Black Hat Hacker, or Cracker.

The process involves an active analysis of the system for any potential vulnerabilities that may result from poor or improper system configuration, known and/or unknown hardware or software flaws, or operational weaknesses in process or technical countermeasures. This analysis is carried out from the position of a potential attacker, and can involve active exploitation of security vulnerabilities. Any security issues that are found will be presented to the system owner together with an assessment of their impact and often with a proposal for mitigation or a technical solution. The intent of a penetration test is to determine feasibility of an attack and the amount of business impact of a successful exploit, if discovered. It is a component of a full security audit.

7.1.3 Padded cell

A padded cell is like a honey pot but is used for intruder isolation. When the IDS detects an intruder s/he is transferred to the padded cell. The padded cell has the look of an actual system but with fake programs and data, a simulated environment of sorts.

8 Bibliography

- [1] 'Shodan'. Accessed: Oct. 06, 2026. [Online]. Available: <https://www.shodan.io>
- [2] 'Network Mapper (nmap)'. Accessed: Oct. 06, 2026. [Online]. Available: <https://nmap.org>
- [3] 'Nmap Scripting Engine (NSE)'. Accessed: Oct. 06, 2026. [Online]. Available: <https://nmap.org/book/nse.html>