Source: *Stable Diffusion AI*

Laboratory 8-9. Network Systems Enumeration

DEPARTMENT OF COMPUTER SCIENCE. UNIVERSITY OF OVIEDO

Computer Security | 2022 – 2023 (V2.7 *"S-81 Isaac Peral"*)

CONTENT

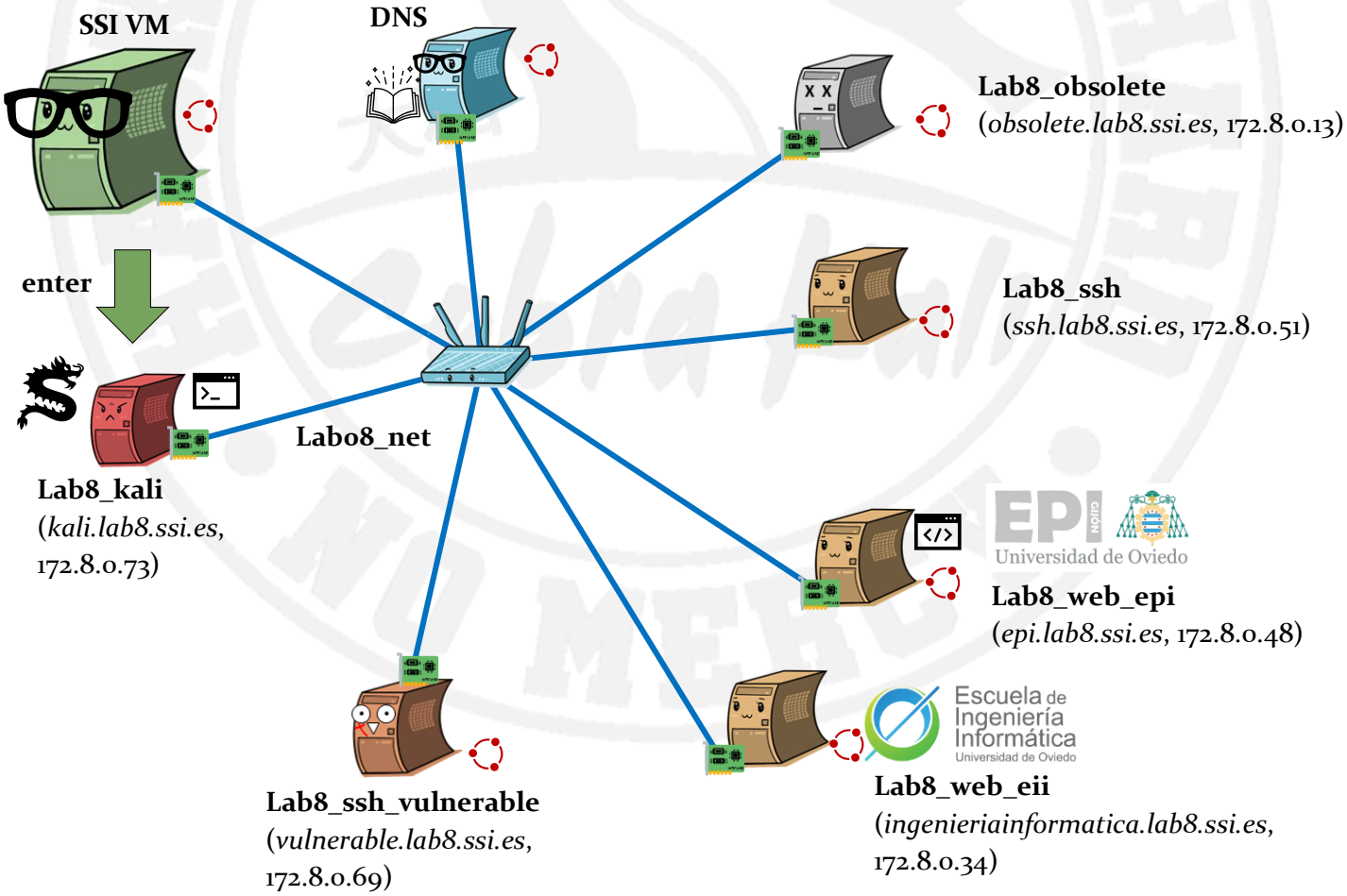
THE INFRASTRUCTURE OF THIS LABORATORY

This lab contains the following containers:

- An "attacking" *Kali*, who is the one that you access to once you run the `enter_kali_lab8.sh` script.
- A DNS server for *lab8.ssi.es*.
- Two web servers (EII, EPI).
- A container with SSH capabilities.
- A container with SSH capabilities and more potentially vulnerable services.
- An obsolete (*Ubuntu* 14.04) container running multiple services

NOTES:

- Please place the line `RUN apt-get update` after the line `USER root` in `<path where you unzipped the labs>\images\ubuntus\ubuntu_apache2\Dockerfile` and `<path where you unzipped the labs>\images\ubuntus\ubuntu_dns\Dockerfile` to ensure the labs could be built without problems.
- Also, some of you have experienced problems while building the lab due to a temporal problem in the distribution of the `theHarvester` program. While the vendor of the program fixes this, please go to `<path where you unzipped the labs>\images\kalis\kali_exploiting\Dockerfile` fixes and comment (`#`) the line that install this program in the `RUN apt-get` command inside this file. This is really a `theHarvester` tool issue, so maybe the problem with correct itself immediately, or even it has been already corrected. Do not bother if you do not find it. Of course, `theHarvester` exercise cannot be done while this issue exists.



BLOCK 1: ENUMERATION WITH NMAP LV1

Just getting it up and running

Practical application: *You need to learn common nmap usages*

Nmap is the reference network scanner, so knowing how to use it is necessary for any pentester / security engineer. We already reviewed how to use it to perform basic "alive" machine detection, but that is only a bit of its "power". This laboratory gives you enough resources to be able to unlock a sizable part of *Nmap* full potential. Practicing these techniques will make you a worthy "opponent" in any network you want to scan, giving you access to powerful techniques to use this tool. Mastering *Nmap* cannot be done straight away, so we have divided this laboratory in sections or levels. Each one will give you a deeper understanding of this powerful tool, giving you more possibilities. **This level is the most basic one**, although with these contents only you will be able to use *Nmap* effectively in most situations.

Nmap is a remote machine reconnaissance and enumeration tool able to locate machines, their exposed ports, the running services attached to these ports, their concrete service type and version, the OS type, and much, much more. It can also be used to perform exploiting tasks and very advanced operations thanks to their scripting engine. But, for now, we will keep it simple to get the most out of it without requiring much effort. To do that, we will present you several of the most typical (or useful) scans)

All-defaults typical scans

These scans are the most typical usages of `nmap` in common scenarios. Ranges of ports, IPs and networks can be combined with the next scan types:

- **Scanning multiple objectives**: `nmap <target 1> <target 2> … <target N>`. E. g.: `nmap 192.168.2.1 192.168.2.100 scanme.nmap.org`
- **Scanning an IP range**: E. g.: `nmap 192.168.2.1-192.168.2.100`
  Scanning a full network: E. g.: `nmap 192.168.0.0/24`
- **Scanning a full network excluding certain hosts**: E. g. `nmap 192.168.0.0/24 --exclude 192.168.2.10`
- **Identify OS of the alive machines**: `nmap -O 192.168.0.0/24`
- **Scan just a port range**: E. g. `nmap -p 22-80 192.168.0.0/24`

Expected results: This task will be finished when you are able to locate and perform an all-default scan to all "alive" machines in the network IP range of the Lab 8 infrastructure we gave to you. Remember to locate them first, as we taught you in a previous lab.

Quick target scan

Once target IPs are known, most of the times the next operation is to **perform a quick analysis** of its running services. We can do this with the following command: `nmap --top-ports 20 --open <target IP>`.

Instead of a single IP, you can provide an IP list in a file using the `-iL` option. This is a **fast and effective scan** that explore the N (=20 in the example) **statistically most used network ports**, only showing those that are open or potentially open. This scan is recommendable to have an initial idea of the services of the device very fast, so we can analyze the system more thoroughly later. How common is a port used is obtained from the file `/usr/share/nmap/nmap-services`. This file orders the port usage using the **prevalence** (ratio of servers observed having the port open, on a probability scale from 0 to 1; in other words: the expected likelihood of finding the port open in the field).

Expected results: This task will be finished when you are able to perform a quick scan over any of the IPs of the Lab 8 infrastructure you previously located

Fast OS and service detection

Using the `-A` parameter enables you to **perform OS and service detection with more detail**. At the same time, we are combining this with `-T4` for faster execution. For example: `nmap -A -T4 <target IP>`. The output size is large, so is preferable to dump the output to a file.

Expected results: This task will be finished when you are able to perform a fast OS and service detection scan over any of the IPs of the Lab 8 infrastructure you previously located

Standard detection of service/daemon versions

This can be done by using `-sV` parameters: `nmap -sV <target IP>`

Expected results: This task will be finished when you are able to perform a standard service detection scan over any of the IPs of the Lab 8 infrastructure you previously located

Slow detailed scan

The command `sudo nmap -sS -A -sV -O -p - <target IP>` runs a detailed and slow scan, recommendable when we **need to explore a chosen target with much more detail**, because it is interesting to our goals.

- `-sS`: Uses a TCP SYN scan type, potentially more able to evade firewalls.
- `-A`: Detects service and OS versions, perform a `traceroute`, and run some scanning scripts to complete the information it obtains.
- `-sV`: Investigate open ports to determine what service and version they are running.
- `-O`: Enable OS version and type detection.
- `-p -`: Scans a port range (`-` means every port).

Expected results: This task will be finished when you are able to perform a slow detailed scan over any of the IPs of the Lab 8 infrastructure you previously located.

Dealing with ports

Practical application: *You need to understand how to scan predeterminate port ranges only*

Ports are present in the results of most scans so we must understand the information *Nmap* outputs.

Port limitations

Nmap scans by default up to 1000 of the statistically most used ports worldwide (according to the file previously mentioned) in random order. This is equivalent to do a `-top-ports 1000`. This gives you substantial coverage

without the effort of scanning the full 65k port range, which is strongly discouraged (unless you have a very good reason to do so 😊), due to the time it takes: https://www.scottbrownconsulting.com/2018/11/nmap-top-ports-frequencies-study/. If more precision is required within an acceptable time, we can try up to 5000. There are other related modifiers too, and these are examples on how to use them:

- `nmap -p 80,443 scanme.nmap.org`: Scans only the ports 80 and 443 (useful if you only want to scan ports of a web server)
- `nmap -p 100-2000 scanme.nmap.org`: Random scan of ports from 100 to 2000.
- `nmap -p -2000 scanme.nmap.org`: Idem, but from 1 to 2000.
- `nmap -p 100- scanme.nmap.org`: Idem, but from 100 to 65536 (highly unrecommended).
- You can use protocols attached to any port number using letters (`T`: TCP, `U`: UDP, `S`: SCTP, `P`: IP). E. g.: `nmap -p U:53,11,13,T:22-25,80,443,8080 scanme.nmap.org`
- There are also two additional modifiers
  - `-F` **(Fast scan)**: Instead of 1000, it scans the most used 100 ports, randomly
  - `-r`: Instead of using random port order, it uses an ascending sequential one

Expected results: This task will be finished when you are able to modify any of the previous scans to restrict it to specific port ranges, and check if this affects the scan speed and/or modifies the `nmap` output somehow.

Port states

- **Open**: There is a service waiting for a connection on this port.
- **Closed**: No service seems to be waiting for connections on this port.
- **Filtered**: *Nmap* packages are not received in this port, so its current state is unknown.
- **Unfiltered**: *Nmap* packages are received in this port, but some reason prevents *Nmap* to know which state this port is.
- **Open/Filtered**: *Nmap* is unable to decide in which of these two states the port is.
- **Closed/Filtered**: Idem as the previous one.

Expected results: This task will be finished when you are able to locate port states in the output of any of the previous scans

"Fire and forget" scripting engine usage (default script scan)

Practical application: *You need to use the NSE script library without going into any detail about these scripts*

The *Nmap* scripting engine enhances the capabilities of the tool, but its usage exceeds this level. However, we can launch a series of default scripts against the scanned ports that may enhance our scan results, even without knowing exactly what the tool is doing. This can be achieved adding the `-sC` option to any of the previous scans and let `nmap` do the rest. We will detail the scripting engine later.

Expected results: This task will be finished when you are able to launch the default script scan over any of the IPs of the Lab 8 infrastructure you previously located.

BLOCK 2: ENUMERATION WITH NMAP LV2

Understanding (a bit) of the TCP Protocol

Although this is not a networking course, it is necessary to know a bit about **how the underlying TCP protocol works** to truly master `nmap`. Fortunately, it is not much 😊. We will just define a bit about how a TCP connection is established and how a TCP package is structured. These things will be used in later explanations and exercises.

According to the Wikipedia: *"TCP protocol operations may be divided into three phases. Connection establishment is a multi-step handshake process that establishes a connection before entering the data transfer phase. After data transmission is completed, the connection termination closes the connection and releases all allocated resources."*

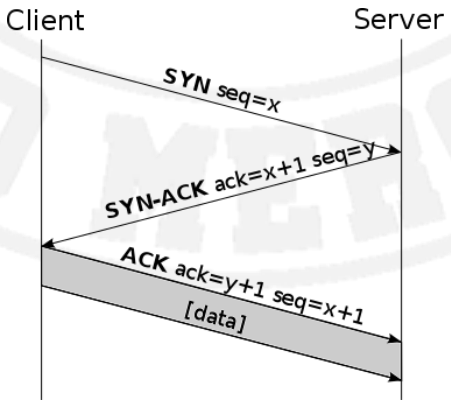
Connection establishment

According to the Wikipedia: *"To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:*

- ***SYN**: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.*
- ***SYN-ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.*
- ***ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e., B+1.*

At this point, both the client and server have received an acknowledgment of the connection. The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established."

Understanding this is important, because manipulation of the connection procedure is extensively used by `nmap` to determine alive ports and its running services. The connection sequence is represented in this image

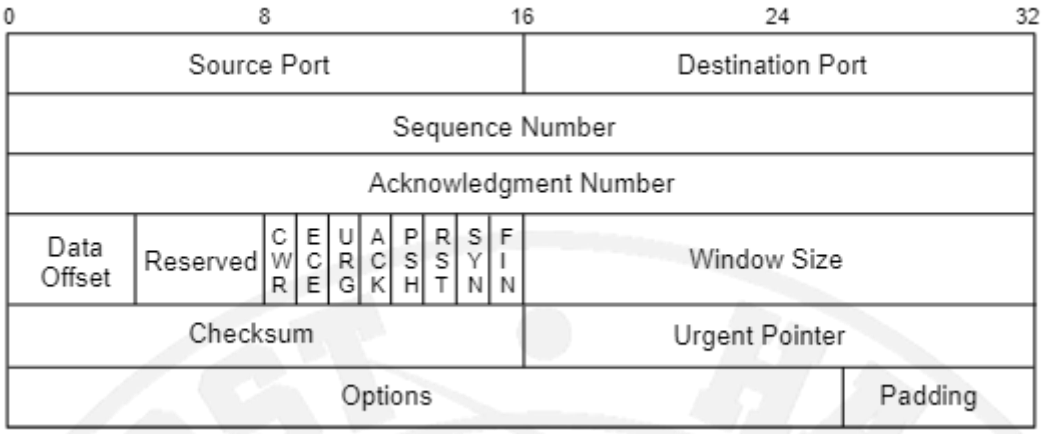


Source: https://es.wikipedia.org/wiki/Protocolo_de_control_de_transmisi%C3%B3n

TCP Packet structure

This is the TCP packet structure. Apart from data, other important fields to remember is the **source and destination ports** (indicating which ports are the origin and the destination of the data transmission), the six

flags (URG, ACK…) that are used in some of the scan types, and the **Window size**, also used in another scan type.



Source: https://intronetworks.cs.luc.edu/current1/uhtml/tcp.html

Nmap advanced host discovery types

Practical application: *You need to discover hosts that are resistant to standard nmap scans*

Nmap 7.80 Cheatsheet series (ingenieriainformatica.uniovi.es)

Part 1: Reconnaissance (Advanced)

https://nmap.org/

GENERAL USAGE

```
nmap [Scan Type(s)] [Options] {target specification}
```

NOTES

Target specifications can be host names, IP addresses, ranges, networks, etc. (scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254)

Use these options to locate "alive machines" (sometimes only that, sometimes they also return some port / service information)

```
operario@kali:~$ sudo nmap --script targets-sniffer 192.168.20.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-21 18:47 CEST
Nmap scan report for 192.168.20.10
Host is up (0.000097s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:67:7A:EF (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.20.1
Host is up (0.0000030s latency).
All 1000 scanned ports on 192.168.20.1 are closed

Nmap done: 256 IP addresses (2 hosts up) scanned in 29.35 seconds
```

TARGET SPECIFICATION

- **--exclude <host1[,host2][,host3],...>**: Exclude hosts/networks
- **--excludefile <exclude_file>**: Exclude list from file
- **-iL <inputfilename>**: Input from file a list of hosts/networks
- **-iR <num hosts>**: Choose random targets

SCRIPT SCAN (https://nmap.org/book/man-nse.html)

- **-sC**: equivalent to --script=default
- **--script=<NSE scripts>**: <NSE scripts> is a comma separated list of directories, script-files or script-categories
- **--script-args=<n1=v1,[n2=v2,...]>**: provide arguments to scripts (see each script documentation to consult argument names, number, and valid value types)
- **--script-args-file=filename**: provide NSE script args in a file
- **--script-trace**: Show all data sent and received

HOST DISCOVERY OPTIONS (WAYS TO CHECK "ALIVE" MACHINES)

- **--dns-servers <serv1[,serv2],...>**: Specify custom DNS servers
- **-n/-R**: Never do DNS resolution/Always resolve [default: sometimes]
- **-PE/PP/PM**: ICMP echo, timestamp, and netmask request discovery probes
- **-Pn**: Treat all provided hosts as online -- skip host discovery
- **-PO[protocol list]**: IP Protocol Ping
- **-PS/PA/PU/PY[portlist]**: TCP SYN/ACK, UDP or SCTP discovery to given ports
- **-sL**: List Scan - simply list targets to scan
- **-sn**: Ping Scan - disable port scan
- **--system-dns**: Use OS's DNS resolver
- **--traceroute**: Trace hop path to each host

RECOMMENDED SCRIPT CATEGORIES FOR ENUMERATION (https://nmap.org/nsedoc/)

broadcast: Scripts in this category typically do discovery of hosts not listed on the command line by broadcasting on the local network. Use the newtargets script argument to allow these scripts to automatically add the hosts they discover to the Nmap scanning queue.

https://nmap.org/nsedoc/categories/broadcast.html

RECONOISSANCE EXAMPLES

sudo nmap --script targets-sniffer 192.168.20.0/24

sudo nmap --script broadcast-dropbox-listener 192.168.20.0/24

sudo nmap --script mringo scanme.nmap.org

sudo nmap -iL ips_to_scan.txt

In a previous lab we learn the basics of reconnaissance, but now we can go much further using two additional techniques to discover when a machine is "alive".

- **The scripting engine** (`--script=<script name and parameters>`): As the reconnaissance cheatsheet shows, there is a category of scripts (*broadcast*) entirely dedicated to discovering several types of

services on a network (ATA over Ethernet, AVAHI, DHCP, DNS, IGMP, UPnP...). There is an extensive list of them, so we should give an opportunity to these scripts to see what happens. All can be found here: https://nmap.org/nsedoc/categories/broadcast.html

- **Advanced discovery techniques**: There are several machine discovery techniques that are described in the following table. This table expands the corresponding section of the cheatsheet. These techniques only work with TCP and IP (and related) protocols. For other protocols not included in the table we should rely on the scripting engine.

TABLE 1. HOST DISCOVERY OPTIONS (WAYS TO CHECK "ALIVE" MACHINES)

`nmap <option> <target or targets>`

Technique name	Nmap Option	Description	Notes
Host discovery using the TCP (and related) protocols			
TCP SYN	`-PS`	Uses a TCP connection init sequence to determine if a host is alive	Useful when firewalls block ICMP requests. Accepts port lists
TCP SYN/ACK	`-PA`	Emulates the ACK response of a (fake) TCP connection The response to this will determine if the target exist	Useful when firewalls block ICMP requests. Accepts port lists
UDP discovery	`-PY`	Send an UDP Ping to the target to see if there is some response	Accepts port lists
Host discovery using ICMP and IP protocols			
Do not ping	`-Pn`	Treat all provided hosts as online -- skip host discovery but scan ports	Useful when we know that there are firewalls blocking pings or we think that the targets are alive
ICMP echo	`-PE`	Uses the echo services of the *Internet Control Message Protocol* (ICMP) protocol to provoke a response	This is usually filtered, so it only works in (some) LANs
ICMP timestamp Ping	`-PP`	Uses services of the Internet Control Message Protocol (ICMP) protocol to provoke a response	May evade firewalls if they only filter ICMP Echo packages
IP Protocol Ping	`-PO[protocol list]`	Sends packages in an IP protocol	If none is specified ICMP, IGMP and IP-in-IP are used. May evade firewalls
PM Address Mask Ping	`-PM`	Uses services of the *Internet Control Message Protocol* (ICMP) protocol to provoke a response	May evade firewalls if they only filter ICMP Echo packages
Other host discovery techniques			
List	`-sL`	Simply list targets to scan	Can resolve the DNS of the passed IPs
ARP Ping discovery	`-PR`	Uses the *Address Resolution Protocol* (ARP) to discover targets	Much quicker than the other discovery methods, but only work in LANs (the only ones using ARP)
Ping discovery	`-sn`	Do not scan any port, just determine if a target is alive	Typical way of detecting alive hosts quickly
Traceroute discovery	`--traceroute`	Obtains a route to the host	If the route is successful, the host exist
Dealing with name resolution			
Use OS's DNS resolver	`--system-dns`	Perform target DNS name resolution using the default DNS configured in the OS	
Mandatory DNS resolution	`-R`	Always resolve [default: sometimes]	Default is to do some DNS resolutions

No DNS resolution	`-n`	Never do DNS resolution	This may speed up scans if DNS names are not important
Specify custom DNS servers	`--dns-servers <serv1[,serv2],...>`	Perform target DNS name resolution using the DNS server you want	
IP telephony and other SCTP services			
SCTP discovery	`-PU`	Uses the *Stream Control Transmission Protocol* (SCTP) to try to provoke responses in the targets and see if they exist	Accepts port lists

An initial basic discovery scan did not return any result? Few results? Do you suspect that there are far more devices available? Try some of this. If it is there, and you get the right protocol or firewall bypass technique, you will get a positive result: a machine is alive there. And once you do that, you can proceed to scan it. However, if a machine gave trouble to be located, it will give even more trouble been scanned. And hence, our scanning techniques have also to be upgraded too, as we will see in the next section.

Expected results: This task will be finished when you are able to check the "alive" machines in the Lab 8 infrastructure using some of the techniques of the Table 1, understanding the output they produce and the scan speed.

Nmap scan types and basic firewall evasion

Practical application: *You need to discover hosts that are firewalled, trying to circumvent the firewall isolation*

The following cheatsheet is like the previous one, but instead of host discovery techniques it shows us two **ways of performing advanced scan operations** (that require already located hosts).

- **(Again) the Nmap scripting engine (`--script=<script name and parameters>`)**. For pure scanning purposes, we should consider using scripts from the following categories, as described in the cheatsheet. Scripting usage for enumeration will be detailed on the next level:
  - `discovery`: https://nmap.org/nsedoc/categories/discovery.html
  - `external`: https://nmap.org/nsedoc/categories/external.html
  - `version`: https://nmap.org/nsedoc/categories/version.html
- **Advanced port scan techniques**: That are described into the following table. This table expands the corresponding section of the cheatsheet. These are based on the TCP protocol characteristics / package structure, and we should select those that we think will be most suited to our scenario, following the provided descriptions. Some may be filtered by IDS / firewalls, and some may not, depending on their configuration. Try to use the more stealth techniques first to avoid being prematurely blocked. If possible**, avoid using the `-O` option** in real environments, as it will be surely blocked by most defensive software.

  Also, keep in mind that is perfectly possible that an open port may suddenly appear in your scans only with a concrete scan technique, and not with the others, depending on the firewall configuration. This is common in CTFs, but in real environments too. Do not underestimate the power of the different scan techniques, they exist for a reason! 😊.

Nmap 7.80 Cheatsheet series (ingenieriainformatica.uniovi.es)

Part 2: Enumeration

https://nmap.org/

GENERAL USAGE

```
nmap [Scan Type(s)] [Options] {target specification}
```

NOTES

Scan techniques makes sense whan there is a firewall or similar solution preventing some types of scan (experimenting options may offer more information)

Increase service/version/OS detection "aggressiveness" if default methods do not return any useful result

TARGET SPECIFICATION

--exclude <host1[,host2][,host3],...>: Exclude hosts/networks

--excludefile <exclude_file>: Exclude list from file

-iL <inputfilename>: Input from list of hosts/networks

-iR <num hosts>: Choose random targets

SERVICE/VERSION DETECTION

-sV: Probe open ports to determine service/version info (see Other Options cheatsheet for a detailed explanation, typical fist option to try)

--version-all: Try every single probe (intensity 9)

--version-intensity <level>: Set from 0 (light) to 9 (try all probes)

--version-light: Limit to most likely probes (intensity 2)

--version-trace: Show detailed version scan activity (for debugging)

SCRIPT SCAN (https://nmap.org/book/man-nse.html)

-sC: equivalent to --script=default

--script=<NSE scripts>: <NSE scripts> is a comma separated list of **directories, script-files or script-categories**

--script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts (see each script documentation to consult argument names, number, and valid value types)

--script-args-file=filename: provide NSE script args in a file

--script-trace: Show all data sent and received

RECOMMENDED SCRIPT CATEGORIES FOR ENUMERATION (https://nmap.org/nsedoc/)

discovery: These scripts try to actively discover more about the network by querying public registries, SNMP-enabled devices, directory services, and the like. Examples include html-title (obtains the title of the root path of web sites), smb-enum-shares (enumerates Windows shares), and snmp-sysdescr (extracts system details via SNMP). See:

https://nmap.org/nsedoc/categories/discovery.html

external: Scripts in this category may send data to a third-party database or other network resource. An example of this is whois-ip, which makes a connection to whois servers to learn about the address of the target. There is always the possibility that operators of the third-party database will record anything you send to them, which in many cases will include your IP address and the address of the target. Most scripts involve traffic strictly between the scanning computer and the client; any that do not are placed in this category. Services include IP Geolocalization, Shodan, SMTP, DNS, Whois...very useful for enumeration. See:

https://nmap.org/nsedoc/categories/external.html

version: The scripts in this special category are an extension to the version detection feature and cannot be selected explicitly. They are selected to run only if version detection (-sV) was requested. Their output cannot be distinguished from version detection output and they do not produce service or host script results. Examples are skypev2-version, pptp-version, and iax2-version.

https://nmap.org/nsedoc/categories/version.html

```
operario@kali:~$ sudo nmap -sV --version-all 192.168.20.10
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-21 19:09 CEST
Nmap scan report for 192.168.20.10
Host is up (0.00019s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.14.0 (Ubuntu)
MAC Address: 08:00:27:67:7A:EF (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.76 seconds
operario@kali:~$ sudo nmap -sU -O --top-ports 10  192.168.20.10
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-21 19:11 CEST
Nmap scan report for 192.168.20.10
Host is up (0.00027s latency).

PORT     STATE  SERVICE
53/udp   closed domain
67/udp   closed dhcps
123/udp  closed ntp
135/udp  closed msrpc
137/udp  closed netbios-ns
138/udp  closed netbios-dgm
161/udp  closed snmp
445/udp  closed microsoft-ds
631/udp  closed ipp
1434/udp closed ms-sql-m
MAC Address: 08:00:27:67:7A:EF (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

SCAN TECHNIQUES (WAYS TO CHECK PORTS AND RUNNING SERVICES)

-b <FTP relay host>: FTP bounce scan

--scanflags <flags>: Customize TCP scan flags

-sI <zombie host[:probeport]>: Idle scan

-sN/sF/sX: TCP Null, FIN, and Xmas scans

-sO: IP protocol scan

-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans

-sU: UDP Scan

-sY/sZ: SCTP INIT/COOKIE-ECHO scans

PORT SPECIFICATION AND SCAN ORDER

--exclude-ports <port ranges>: Exclude the specified ports from scanning

-F: Fast mode - Scan fewer ports than the default scan

-p <port ranges>: Only scan specified ports. Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9

--port-ratio <ratio>: Scan ports more common than <ratio>

-r: Scan ports consecutively - don't randomize

--top-ports <number>: Scan <number> most common ports

OS DETECTION

-O: Enable OS detection

--osscan-guess: Guess OS more aggressively

--osscan-limit: Limit OS detection to promising targets

EXAMPLES

```
sudo nmap -sV --version-all 192.168.20.10
nmap -sS -A -sV -O -p - 192.168.20.10
sudo nmap -sU -O --top-ports 10  192.168.20.10
sudo nmap -p1-100 -sS 192.168.20.10
sudo nmap -p-100 --script=banner 192.168.20.10
sudo nmap --script=ip-geolocation-geoplugin www.ingenieriainformatica.uniovi.es
sudo nmap --script=http-server-header www.ingenieriainformatica.uniovi.es
```

The following table is an extension of the "*Scan techniques*" section of the cheatsheet. It is followed by an image with a more detailed description about what each scan does, but only for reference, not because you must understand them. **If you press intro while a scan is running, you can see the type of scan technique that this scan is using, and the estimated finish time**:

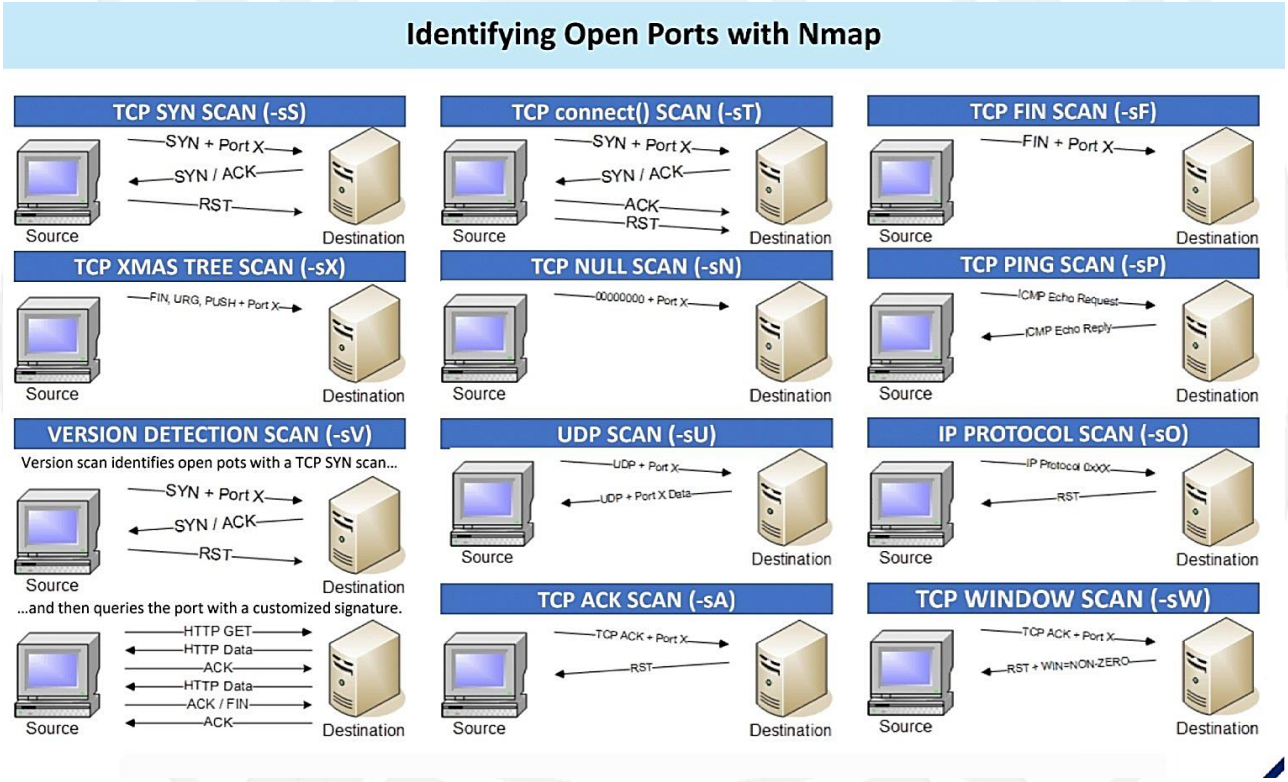
TABLE 2: SCAN TECHNIQUES (WAYS TO CHECK PORTS AND RUNNING SERVICES)

`nmap <option> <target or targets>`

Technique name	Nmap Option	Description	Notes
Main scan techniques			
ACK scan	-sA	• Used to know if there is a firewall active in the target. • Any port, no matter if it is open or closed will answer with an RST package • But if there is a firewall, there will be no answer	It does not determine if a port is open or closed

Idle scan	`-sI`	Very advanced and stealth scan mode	Seen on a later level, as it is too advanced for level 2
IP protocol scan	`-sO`	• Used to analyze which IP protocols are supported by the target • Targets answering ICMP unreachable packages for a port means that this port is closed	Very slow. Identifies supported protocols like ICMP, IGMP, etc.
TCP Connect	`-sT`	• Scanning computer send SYN • If scanned computer answers SYN/ACK: **Open port** • Nmap send ACK • If scanned computer does not answer, or answer RST/ACK: **Closed port**	Slow, highly detectable
TCP SYN	`-sS`	• Scanning computer sends SYN • If scanned computer answers SYN/ACK: **Open port** • If scanned computer answers RST: **Closed port**	**The most popular scan technique.** Fast, reasonably stealth, does not (normally) overload the target
UDP Scan	`-sU`	• 0-byte UDP packages are sent to each of the target ports • An ICMP "unreachable port" response means that the port is closed • However, no response does not mean that the port is filtered (UDP is connectionless)	• Very slow, but used by popular services such as DNS (port 53), SNMP (ports 161/162), DHCP (ports 67/68),… • The additional --data-length parameter sends random data of the indicated length
Window scan	`-sW`	Uses the TCP package "Window Size" (see picture) field to determine the type of OS (several types of OS have a characteristic value there)	This method is not very reliable. You can check different Window Sizes by OS here: https://www.infoq.com/articles/tcp-syn-security-unauthorized-os/
Techniques for IP telephony or other SCTP services			
COOKIE-ECHO Scan	`-sZ`	Advanced technique for determining open and closed ports in SCTP environments	Cannot really distinguish between open and filtered ports (they appear as `open|filtered`)
SCTP INIT Scan	`-sY`	Standard SCTP protocol scan	Used for environments that implement this protocol, it is a fast and stealth technique aimed to them
Techniques based on setting certain TCP packet flags (breaking the TCP standard)			
Customize TCP scan flags	`--scanflags <flags>`	Useful if you want to "play" with the TCP packet flag bits (see picture) to see what happens	Possible bit names to use: URG, ACK, PSH, RST, SYN y FIN (--scanflags URGACKPSHRSTSYNFIN activates everyone)
Maimon scan	`-sM`	Like a XMAS scan (see below), can be an alternative to it	
TCP FIN	`-sF`	The FIN flag of the TCP packet is enabled	
TCP Null	`-sN`	No flag of the TCP packet is enabled	
TCP XMAS	`-sX`	The FIN, PSH, and URG flags of the TCP packet are enabled	
Techniques for specific services			
FTP bounce scan	`-b user:password@<FTP server>:21 <target>`	• Connects to the indicated FTP server and send files to the target • It sends the files to each port to study on the target • The error message will determine if the port is open or closed	• User and password for the FTP server must be provided unless anonymous connections are allowed in the FTP server • May bypass firewalls if FTP is used

			• It is useful to look for FTP services on non-standard (21) ports
RPC scan	-sR	Determines which open ports belong to RPC calls, including the program listening requests and its version	Only if we expect to find RPC calls in the target system
Combined/detection techniques			
Aggressive scan	-A	Combined action of the -sC, -sV, -O and --traceroute options	Most likely filtered, avoid using it if possible
Default script scan	-sC	Launches the adequate scripts against each studied port from the "default" NSE script category (https://nmap.org/nsedoc/categories/default.html)	Equivalent to --script=default
Operating system detection scan	-O	Tries to determine the type and version of the OS of the target	Most likely filtered, avoid using it if possible
Version scan	-sV	Probe open ports to determine service/version info	



Fuente: @AllPentesting (*Twitter*)

Expected results: This task will be finished when you are able to scan the ports of one of the located machines in the Lab 8 infrastructure using some of the techniques of the Table 2. Additionally, the `epi` and `eii` web servers in the infrastructure are resistant to some of these scan techniques. *Are you able to find which ones? are you able to find a technique that work against the `eii` web server in this infrastructure, that apparently is firewalled and can resist default scan attempts?*. NOTE: If you see that a scan is reporting errors or problems and/or it is taking a lot of time, it is advisable to cancel it and try another technique (you have been busted! 😊)

Nmap (more) advanced examples

Practical application: *You need to precisely control nmap scan types and timing to perform advanced scans*

Apart from the examples of the previous sections and the new reconnaissance, and enumeration methods we can use now combined with them, we show additional examples to enhance our *Nmap* knowledge. The `-n` option can be added to any of them to enhance its speed (no DNS resolution is performed).

- **TCP SYN and UDP Scan (requires `root`)**: You can combine multiple scan techniques from the previous table. E. g.: `nmap -sS -sU -Pn 192.168.13.37`. This TCP SYN and UDP scan will take a while but is unobtrusive and stealthy. This command will check about 2000 common TCP and UDP ports (1000 TCP+1000 UDP most used ones) to see if they are responding. The `-Pn` flag skips the ping scan and assume the host is up. This can be useful when there is a firewall that might be preventing ICMP replies.
- **TCP SYN and UDP scan for all reserved ports (requires `root`)**: A variant of the previous one. E. g.: `nmap -sS -sU -PN -p 1-1024 192.168.13.37`
- **TCP Connect Scan (no `root`)**: E. g.: `nmap -sT 192.168.13.37`. Discouraged. As we said, try to use a more stealth scan first.
- **Fast Scan (no `root`)**: `nmap -T4 -F 192.168.13.37`. Uses predefined **timing options** to enhance the scan speed and scans only 100 ports (`-F`). The **Nmap timing options** appear in this table. By default, it uses `T3`. `T0` is very slow but also virtually undetectable (supposedly! 😊) This could be useful to know some potential hosts with open ports in a network. Timing options are just a predefined set of values for certain *Nmap* parameters, which can be also individually adjusted for more control (see next level).

Category	Initial_rtt_timeout	min_rtt_timeout	max_rtt_timeout	max_parallelism	scan_delay	max_scan_delay
T0 / Paranoid	5 min	Default (100 ms)	Default (10 sec)	Serial	5 min	Default (1 sec)
T1 / Sneaky	15 sec	Default (100 ms)	Default (10 sec)	Serial	15 sec	Default (1 sec)
T2 / Polite	Default (1 sec)	Default (100 ms)	Default (10 sec)	Serial	400 ms	Default (1 sec)
T3 / Normal	Default (1 sec)	Default (100 ms)	Default (10 sec)	Parallel	Default (0 sec)	Default (1 sec)
T4 / Aggressive	500ms	100ms	1,250ms	Parallel	Default (0 sec)	10ms
T5 / Insane	250ms	50ms	300ms	Parallel	Default (0 sec)	5ms

Source: https://kb.parallels.com/en/123568

- **Verbose**: `nmap -T4 -A -v 192.168.13.37`. By adding `verbose` to most of the scans you get a better insight into what *Nmap* is doing; for some scans, verbosity will provide additional details that the output report does not provide.

In either case, every time we locate a service/OS name and version, it is easy to locate potential vulnerabilities (that could be considered highly risky or critical) in the scanned systems by using the services of this webpage or similar ones: `https://www.cvedetails.com/`

Expected results: This task will be finished when you are able to scan the ports of one of the located machines in the Lab 8 infrastructure using combined scan methods and different timing options. You are also able to perform a vulnerability search in the CVE details web page of the service types and versions you located.

OS type detection without "noise"

Practical application: *You need to guess the remote target OS type without being detected*

As the `-O` option is highly discoverable and hence our scanning attempts may be blocked as a result on a real environment, we can use a trick to try to guess the OS type: *playing with the TTL* (time to live) of the packages we send to the system. This however does not require *Nmap* but pinging the system like this: `ping -c 1 <IP or target name>`. As a result, if there is connection, we will receive responses indicating a `ttl` value. This value can be consulted in this table to guess the type of OS that the target has:

Table 1. Popular OSs' time to live (TTL) and window size values.

OS	TTL	Window size (bytes)
Linux 2.4 and 2.6	64	5,840
Google customized Linux	64	5,720
Linux kernel 2.2	64	32,120
FreeBSD	64	65,535
OpenBSD, AIX 4.3	64	16,384
Windows 2000	128	16,384
Windows XP	128	65,535
Windows 7, Vista, and Server 8	128	8,192
Cisco Router IOS 12.4	255	4,128
Solaris 7	255	8,760
MAC	64	65,535

Source: https://www.semanticscholar.org/paper/Packet-Inspection-for-Unauthorized-OS-Detection-in-Tyagi-Paul/880095d7fe063c9553a3f958c050996682e5ec9e

Expected results: This task will be finished when you are able to predict the OS type of one of the located machines in the Lab 8 infrastructure using the TTL parameter value.

Nmap output formats

Practical application: *You need nmap scan results in a format more suitable to be processed by other programs or your code*

Apart from the traditional standard output, *Nmap* accepts three different output formats by adding the following parameters to any of the scans we may perform:

- ***Nmap* format** (`-oN <file>`): Equivalent to redirect *Nmap* output to a file, it follows the traditional output structure.
- **"Grepeable" format** (`-oG <file>`): Transforms the output so anytime a `grep` command finds a coincidence with the word we are searching for more information is shown. It also eases processing it using regular expressions.

- **XML format (`-oX <file>`)**: The traditional output contained in an XML structure. Additionally to being easier to process using code, these files can be imported directly by *Metasploit* to use the scan results to start exploiting operations.

Expected results: This task will be finished when you are able to extract `nmap` information in different formats when performing scans of one of the located machines in the Lab 8 infrastructure. Concretely, you should be able to obtain a scan of the whole network machines in XML, as we will use it later.

BLOCK 3: ENUMERATION WITH NMAP LV3

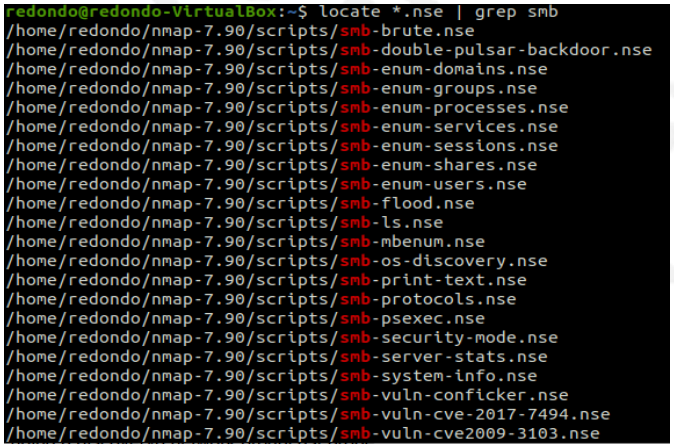
Nmap Scripting Engine

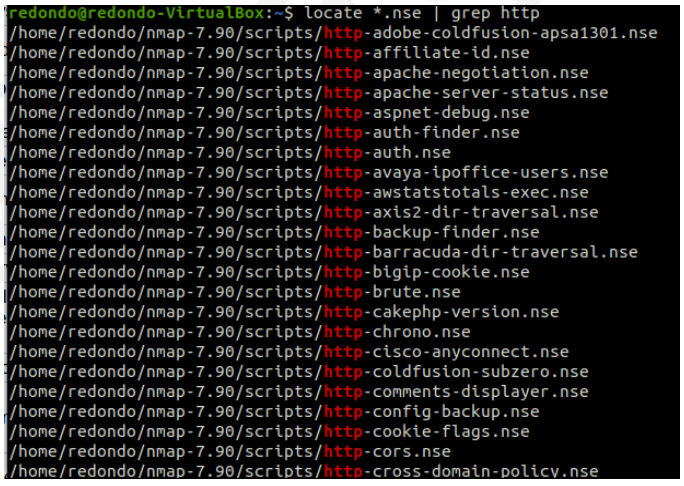
Practical application: *You need to understand how to work with the nmap NSE scripting engine*

As we said, the functionality of *Nmap* can be extended thanks to its **NSE scripting engine**. There are more than 600 different scripts whose documentation can be checked here: https://nmap.org/nsedoc/. Scripts are installed in `/usr/share/nmap/scripts` and available scripts can be listed using `locate *.nse`. As said in the previous level, for enumeration purposes the following categories of scripts are recommended:

- `discovery` (https://nmap.org/nsedoc/categories/discovery.html): These scripts try to actively discover more information about the network by querying public registries, SNMP-enabled devices, directory services, and the like. Examples include `html-title` (obtains the title of the root path of web sites), `smb-enum-shares` (enumerates *Windows* shares), and `snmp-sysdescr` (extracts system details via SNMP).
- `external` (https://nmap.org/nsedoc/categories/external.html): Scripts in this category may send data to a third-party database or other network resource. An example of this is `whois-ip`, which makes a connection to WHOIS servers to learn about the target address. There is always the possibility that operators of the third-party database will record anything you send to them, which in many cases will include your IP address and the address of the target. Most *Nmap* scripts involve traffic strictly between the scanning computer and the target; any that do not, and introduce a third party, is placed in this category. Services include IP *Geolocalization*, *Shodan*, SMTP, DNS, WHOIS…or other services useful for enumeration.
- `version` (https://nmap.org/nsedoc/categories/version.html): The scripts in this special category are an extension to the version detection feature and cannot be selected explicitly. They are automatically selected to run only if version detection (`-sV`) was requested. Their output cannot be distinguished from version detection output because it is embedded on it. Examples are `skypev2-version`, `pptp-version`, and `iax2-version`

If we do not want to examine each category contents, another more straightforward way of locating adequate scripts is to concatenate the previous `locate` command with a `grep` filter using a service name we expect to find (or we know it is present) in the target server. This way, we will find any possible script that applies to those services, and we could consult their documentation to try them to see if it fits our purposes. The following four images shows how to use this script location technique for four different typical services. Bear in mind though that doing this will locate any script associated with a service, no matter if it is used by reconnaissance, enumeration or exploiting. Therefore, what you find may well exceed the purpose of this laboratory.

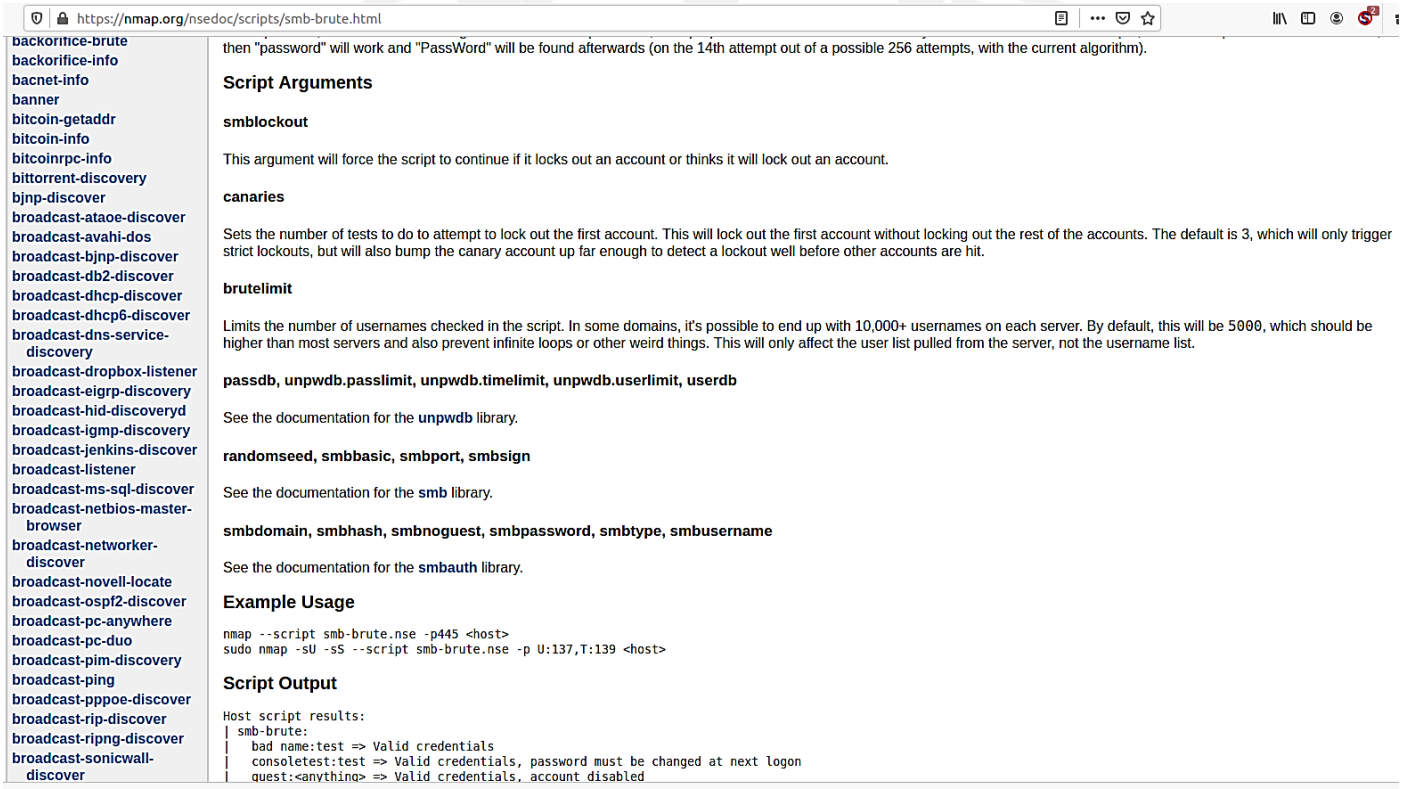




```
redondo@redondo-VirtualBox:~$ locate *.nse | grep ssh
/home/redondo/nmap-7.90/scripts/ssh-auth-methods.nse
/home/redondo/nmap-7.90/scripts/ssh-brute.nse
/home/redondo/nmap-7.90/scripts/ssh-hostkey.nse
/home/redondo/nmap-7.90/scripts/ssh-publickey-acceptance.nse
/home/redondo/nmap-7.90/scripts/ssh-run.nse
/home/redondo/nmap-7.90/scripts/ssh2-enum-algos.nse
/home/redondo/nmap-7.90/scripts/sshv1.nse
```

```
redondo@redondo-VirtualBox:~$ locate *.nse | grep citrix
/home/redondo/nmap-7.90/scripts/citrix-brute-xml.nse
/home/redondo/nmap-7.90/scripts/citrix-enum-apps-xml.nse
/home/redondo/nmap-7.90/scripts/citrix-enum-apps.nse
/home/redondo/nmap-7.90/scripts/citrix-enum-servers-xml.nse
/home/redondo/nmap-7.90/scripts/citrix-enum-servers.nse
```

Nmap uses a common syntax no matter the script we want to use: `nmap --script <script name> --script-args <script arguments>`. **Every script has its own parameter set** that must be consulted in the documentation of each one. The documentation of each script is very complete, containing not only accepted parameters, but the meaning of each one of them, an explanation of the script purposes, example usages, and the expected output. For example, this is part of the documentation of the `smb-brute` script (https://nmap.org/nsedoc/scripts/smb-brute.html):



https://nmap.org/nsedoc/scripts/smb-brute.html

backorifice-brute
backorifice-info
bacnet-info
banner
bitcoin-getaddr
bitcoin-info
bitcoinrpc-info
bittorrent-discovery
bjnp-discover
broadcast-ataoe-discover
broadcast-avahi-dos
broadcast-bjnp-discover
broadcast-db2-discover
broadcast-dhcp-discover
broadcast-dhcp6-discover
broadcast-dns-service-discovery
broadcast-dropbox-listener
broadcast-eigrp-discovery
broadcast-hid-discoveryd
broadcast-igmp-discovery
broadcast-jenkins-discover
broadcast-listener
broadcast-ms-sql-discover
broadcast-netbios-master-browser
broadcast-networker-discover
broadcast-novell-locate
broadcast-ospf2-discover
broadcast-pc-anywhere
broadcast-pc-duo
broadcast-pim-discovery
broadcast-ping
broadcast-pppoe-discover
broadcast-rip-discover
broadcast-ripng-discover
broadcast-sonicwall-discover

then "password" will work and "PassWord" will be found afterwards (on the 14th attempt out of a possible 256 attempts, with the current algorithm).

Script Arguments

smblockout

This argument will force the script to continue if it locks out an account or thinks it will lock out an account.

canaries

Sets the number of tests to do to attempt to lock out the first account. This will lock out the first account without locking out the rest of the accounts. The default is 3, which will only trigger strict lockouts, but will also bump the canary account up far enough to detect a lockout well before other accounts are hit.

brutelimit

Limits the number of usernames checked in the script. In some domains, it's possible to end up with 10,000+ usernames on each server. By default, this will be 5000, which should be higher than most servers and also prevent infinite loops or other weird things. This will only affect the user list pulled from the server, not the username list.

passdb, unpwdb.passlimit, unpwdb.timelimit, unpwdb.userlimit, userdb

See the documentation for the **unpwdb** library.

randomseed, smbbasic, smbport, smbsign

See the documentation for the **smb** library.

smbdomain, smbhash, smbnoguest, smbpassword, smbtype, smbusername

See the documentation for the **smbauth** library.

Example Usage

```
nmap --script smb-brute.nse -p445 <host>
sudo nmap -sU -sS --script smb-brute.nse -p U:137,T:139 <host>
```

Script Output

```
Host script results:
| smb-brute:
|   bad name:test => Valid credentials
|   consoletest:test => Valid credentials, password must be changed at next logon
|   guest:<anything> => Valid credentials, account disabled
```

Although the recommended way of using the NSE scripts is using the `locate` command to find those that may be interesting for each specific case (or manually examining the recommended categories), we will review several interesting scripts to familiarize with the advanced capabilities of *Nmap* that covers typical NSE usages. However, remember that these are only examples, and it is better to choose whichever suits you the best on your own.

Expected results: This task will be finished when you are able to locate scripts related to a certain service, supported protocol or characteristic you want to explore.

Information Gathering Techniques

Practical application: *You need to use typical information gathering techniques using the NSE script library*

IMPORTANT NOTE: Scans to external targets (such as www.uniovi.es) can also be done with the `nmap` installation of the *Ubuntu VM*.

SSH Auth methods

The `ssh-auth-methods` script (https://nmap.org/nsedoc/scripts/ssh-auth-methods.html) locates the accepted authentication methods of a SSH service located in a target. Its recommended usage is: `sudo nmap -p22 --script ssh-auth-methods <target IP>`. Use this to know if it is worth to try a brute-force password guessing attack against that SSH or not because it does not accept user/password combinations.

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure.

DNS Brute Force

The `dns-brute.nse` script will find valid *DNS A* records by trying a list of common sub-domains and finding those that successfully resolve. This effectively finds sub-domains associated with an organization main domain, which can reveal additional targets to perform security assessments (we did the same with specific tools in a previous lab). **Example:** `nmap -p 80 --script dns-brute.nse <target URL or IP>`

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure and www.uniovi.es (do it from the VM)

Find Hosts on an IP

Another tactic for expanding the attack surface is to find virtual hosts on an IP address (multiple websites that are hosted on the same web server). This can be done by using the `hostmap-*` scripts in the NSE script collection. The `hostmap-bfk.nse` script seems to work reasonably well (IP to Host services vary in accuracy). **Example:** `nmap -p 80 --script hostmap-bfk.nse <target URL or IP>`

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure and www.ingenieriainformatica.uniovi.es

Traceroute Geolocation

The `traceroute-geolocation.nse` script performs a *traceroute* to your target IP address and have geolocation data plotted for each hop along the way. This makes correlating the reverse DNS names of routers in your path with physical locations much easier. **Example:** `sudo nmap --traceroute --script traceroute-geolocation.nse -p 80 <target URL or IP>`

*(**NOTE**: Your ISP may prevent geolocation coordinates to be obtained. If this script does not show coordinates, do not worry, and carry on with the exercise.)*

Expected results: This task will be finished when you are able to launch this script over the web page of our school.

Database/external service geolocation

There are a series of NSE scripts that help to geolocate an IP using external databases or services. An example is `ip-geolocation-geoplugin` (https://nmap.org/nsedoc/scripts/ip-geolocation-geoplugin.html). It tries to

identify the physical location of an IP address using the *Geoplugin* geolocation web service (http://www.geoplugin.com/).

Expected results: This task will be finished when you are able to explore how to use and launch this script over the web page of our school and use *Google Maps* to know the physical site of the server of a domain.

HTTP Information Gathering

Practical application: *You need to gather information from remote HTTP servers using the NSE script library*

Nmap comes with a wide range of NSE scripts for testing web servers and applications. An advantage of using the NSE scripts for HTTP reconnaissance is that you can test several aspects of web servers in large subnets with a single operation, quickly providing information about the types of servers and applications in the subnet.

HTTP Methods

Finds out what options are supported by an HTTP server by sending an OPTIONS request. Lists potentially risky HTTP methods. https://nmap.org/nsedoc/scripts/http-methods.html. E. g. `sudo nmap -p80 -script http-methods <target IP>`

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure

HTTP Banner grabbing

One of the typical ways to determine service types and versions is just to ask them who they are. This is done reading the service information they provide upon connecting to them, which is commonly known as "*banner grabbing*". There are multiple techniques to perform banner grabbing, but *Nmap* has the `banner` script to perform these techniques: `nmap --script banner <target IP>`

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure

HTTP Common paths

The `http-enum.nse` script effectively brute forces paths on a web server to discover web applications in use. Attempts will be made to find valid paths on the web server that match a list of known paths extracted from common web applications. The standard test includes testing of over 2000 paths. **Examples**: `nmap --script http-enum <target URL or IP>` or specifying a base path: `nmap --script http-enum --script-args http-enum.basepath='pub/' <target URL or IP>`. The output size could be large, so is preferable to dump the output to a file.

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure

HTTP Title

This script adds web page tittles to *Nmap* scan results so they can provide context to a host running HTTP services; this may easily identify the primary purpose of the web server and whether that server is a potential attack target. **Example**: `nmap --script http-title -sV -p 80 <target network>`

Expected results: This task will be finished when you are able to launch this script over an appropriate machine of the Lab 8 infrastructure

WHOIS Records scan

As we saw in a previous laboratory, these records have interesting information, like the domain owner real name, contact data…although frequently this data belong to the hosting company. More **information:** https://nmap.org/nsedoc/scripts/whois-ip.html. Syntax: `nmap --script whois-ip <target URL or IP>`

This script uses the IANA data to find an adequate WHOIS database to locate the information we want. We can also specify the WHOIS service order we want to use: `nmap --script whois-ip -script-args whois.whodb=arin+ripe+afrinic <target IP>`.

Expected results: This task will be finished when you are able to launch this script over the webpage of our university

E-mail accounts

Nmap had the capability to gather email addresses associated with a domain as part of its official NSE script database. Unfortunately, these scripts became obsolete, and no replacement was officially developed. To cover these needs, we will use *The Harvester*, a *Kali* tool that allows us to gather OSINT information of a domain (complementing a previous laboratory) that also includes email accounts associated to a domain. Use this cheatsheet to do this task.

theHarvester 3.0.0 Cheatsheet (ingenieriainformatica.uniovi.es)
Website OSINT tool

https://tools.kali.org/information-gathering/theharvester

GENERAL USAGE

```
theharvester options
```

NOTES

Installable directly from Kali Linux repositories

OPTIONS

-b: data source: baidu, bing, bingapi, dogpile, google, googleCSE, googleplus, google-profiles, linkedin, pgp, twitter, vhost, virustotal, threatcrowd, crtsh, netcraft, yahoo, all
-c: perform a DNS brute force for the domain name
-d: Domain to search or company name
-e: use this DNS server
-f: save the results into an HTML and XML file (both)
-h: use SHODAN database to query discovered hosts
-l: limit the number of results to work with(bing goes from 50 to 50 results, google 100 to 100, and pgp doesn't use this option)
-n: perform a DNS reverse query on all ranges discovered
-p: port scan the detected hosts and check for Takeovers (80,443,22,21,8080)
-s: start in result number X (default: 0)
-t: perform a DNS TLD expansion discovery
-v: verify host name via dns resolution and search for virtual hosts

EXAMPLES

```
theharvester -d uniovi.es -l 50 -b google
theharvester -d uniovi.es -b pgp
theharvester -d uniovi -l 200 -b linkedin
theharvester -d uniovi.es -b googleCSE -l 100 -s 300
```

Expected results: This task will be finished when you are able to gather email addresses associated with a domain, for example `uniovi.es`. To make the tests, limit your results to 50 elements and use `google` as a data source.

Malware and vulnerabilities

Practical application: *You need to detect potential malware and vulnerabilities on remote hosts*

Malware

Another utility of the *Nmap NSE script engine* is to know if a host has been identified as a malware or phising attack source / distributor. This is possible thanks to the *Google Safe Browsing API*, as the script `http-google-malware` ask this service for that kind of information. To use that, we need to register with *Google* for an adequate API key: http://code.google.com/apis/safebrowsing/key_signup.html . Once registered we can use this with: `nmap -p80 --script http-google-malware --script-args http-google-malware.api=<API key> <target IP or URL>`. *Firefox* or *Chrome* also use this service to protect users while browsing. More information: https://nmap.org/nsedoc/scripts/http-google-malware.html . **We are not doing this in this lab to not to force you to ask for an API key.**

`nmap` can also detect malware and backdoors by running extensive tests on a few popular OS services like on *Identd*, *Proftpd*, *Vsftpd*, *IRC*, *SMB*, and *SMTP*. It also has a module to check for popular malware signs in remote servers and integrates *Google's Safe Browsing* and *VirusTotal* databases as well. Example: `nmap -sV --script http-malware-host <target URL or IP>`

Expected results: This task will be finished when you are able to launch this script over the machine with most services of the Lab 8 infrastructure. Launch this from the `nmap` in the Ubuntu VM (you have access to the machines in the infrastructure from there)

CVE detection using Nmap

The `--script vuln` script runs a full vulnerability test against our target. **Example**: `nmap -Pn --script vuln <target URL or IP>` . This script takes a sizable amount of time to complete, and the output size is large, so it is preferable to dump the output to a file. The discovered CVE details can be consulted in external databases we already described.

Even if the purpose of this script is very interesting, its result could be more complete. Luckily, there is a third-party script (`nmap-vulners`) that improves it. There are plenty of third-party NSE scripts available and sometimes you will need to use them because the contents of the NSE library are not enough, or you have very particular requirements. To use **any third-party script**, you must follow this procedure:

- Download the script `.nse` file or folder to the NSE script folder (`/usr/share/nmap/scripts`). In our case we need to do `git clone https://github.com/vulnersCom/nmap-vulners.git` once we are in that folder.
- Update the `nmap` script database with `nmap --script-updatedb`

Once this is done, we can run the script like any standard one we saw. For our case, we only need to do `sudo nmap -sV --script=nmap-vulners/ <ip>`. Observe the additional information it outputs and why it could be useful.

Expected results: This task will be finished when you are able to launch these scripts over the machine "`obsolete`" of the Lab 8 infrastructure. Launch this from the `nmap` in the Ubuntu VM (you have access to the machines in the infrastructure from there)

Nmap and Metasploit Framework (MSF)

Practical application: *You need to integrate nmap scan results with Metasploit*

Nmap is also fully integrated inside the *Metasploit Framework (MSF)*. It can be used to scan targets and their results will be kept inside an MSF session, so the discovered hosts can be used to perform further research or exploiting attacks. This links with future labs that will use MSF features to run exploiting techniques, but for now we will only use MSF as a `nmap` scan analysis tool.

The integration of both products is managed using a series of commands that `msf` implement and will be described now. Also, it is possible to load the result of any scan we made with `nmap` provided we saved it on `.xml` format, as indicated in a previous part of this laboratory. To do this:

- Go to the directory with the `.xml` file scan results you saved earlier
- Run *Metaspolit Framework* running these three commands (ignore warnings)
  - Start MSF database: `service postgresql start`
  - Init MSF: `msfdb init`
  - Enter the MSF console: `msfconsole -q`

Now you can use the following commands

- `db_import`: Imports a `nmap` previous scan output in XML format, updating the known hosts of an MSF session.
- `db_nmap` is `nmap` itself (we can use it with all the scripts, parameters and features we saw). If used like this, its scan results update any previous scan result we have (even loaded via an XML file), so new discovered information about hosts, IPs, ports, versions, etc. will be stored.
- `hosts`: shows the identified hosts (machines).
- `hosts -c address,purpose` (or other column names): Identified hosts are shown with a series of data columns and this command only show those that may be interesting to us.
- `hosts -u`: Show only "alive" hosts in the last scan performed.
- `services`: Show all services identified in all scanned hosts.
- `services -p 80`: Identical to the previous, but only for the indicated ports.
- `services -p tcp`: Idem, but for the indicated protocols.

Expected results: This task will be finished when you can load the XML results of a `nmap` scan you made previously and inspect it with MSF

On the other hand, if you need to use `nmap` programmatically, one of the easiest ways is to use *Python*, as it has a `nmap` library that gives you access to the options we saw on this laboratory from your programs.

BLOCK 4. BEYOND NMAP. OTHER ENUMERATION TECHNIQUES

Scanning for concrete files

Practical application: *You need a quick and uncomplicated way of locating certain files in remote endpoints*

There are concrete files that are useful from an enumeration point of view, and therefore downloading them is very important to study a target. The good part is that you do not even need a custom program to do that, just a simple script you can develop with any programming language. The following is an example to download the `phpinfo.php` file from the IP range `156.35.90.0` -`156.35.99.255`)

```
#!/bin/bash
for ip_address in 156.35.9{0..9}.{0..255}; do
    wget -t 1 -T 5 http://${ip_address}/phpinfo.php;
done&
```

This performs a mass request for a certain file for all the addresses in a range, downloading them to the current folder if they exist. This technique can be used to automate searches for interesting files in an IP range of any company, from known files such as `robots.txt`, `index.html`, etc. to potentially more compromising files such as `backup.zip`, `backup.rar`, known files from certain CMSs...

Expected results: This task will be finished when you can use the script example to attempt to download all the `robots.txt` files from the IP range `156.35.94.1` to `156.35.94.10`

Service endpoints in JavaScript files

Practical application: *You need to locate services pointed out by JavaScript code in a web page*

Service endpoints in *JavaScript* files are typical places in which security is relaxed due to developers not taking proper care of them. Also, automated scanning tools usually do not scan these endpoints, leaving a very interesting potential security vulnerability without processing. We can use a *Kali Linux* tool, `photon`, to extract this information and much more, and all will be saved in an organized mode:

- URLs both in-scope & out-of-scope, as well as URLs with parameters (`example.com/gallery.php?id=2`)
- *JavaScript* files y endpoints present in them
- Strings based on custom regex pattern
- **OSINT data**: emails, social media accounts, Amazon buckets etc.
- Extracted files: .pdf, .png, .xml etc.

Expected results: This task will be finished when you can use the `photon` tool to inspect our school webpage to search for potential vulnerable endpoints in *JavaScript* files.

Amazon S3 Buckets

Practical application: *You need to locate Amazon S3 buckets to inspect them*

AWS Simple Storage Service (aka *Amazon S3*) is used by companies that does not want to install and manage their own storage repositories for certain data. This service can store objects and files of several types in the cloud (virtual server) instead of a physical machine. Storage spaces in this technology are called *buckets*. The

creator of a bucket can store in it source code, certificates, passwords, databases...any type of content. Unfortunately, sometimes they are not adequately protected, and if you get full access to an S3 bucket you can download, upload, or modify any of the files you find. As these usually are files with very sensitive content, the effects are usually terrible and frequently you can hear about S3 buckets as the source of attacks. There are simple ways of finding when a website is using S3 buckets:

- By using a tool that explore the URLs of a website, like the previous `photon` tool or the famous *Burp Suite*. These tools can find URLs ending in `s3.amazonaws.com` that may lead to an unprotected bucket.
- By using the *Google Hacking Database* of the laboratory 2, searching for "S3" to obtain a list of *Google Search* queries that may reveal this information
- By directly search for these URLs yourself in *Google*: `site:amazonaws.com inurl:<URL word>` (i. e. `site:amazonaws.com inurl:uniovi`)

Once found, you can use the `awscli` package in *Kali* to interact with it if possible.

Expected results: This task will be finished when you use a *Google Search* to find potential URLs from *Uniovi* using S3 buckets. You do not have to browse them, just locate them.

GitHub

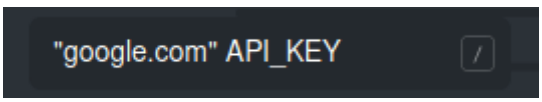
Practical application: *You need to locate secrets in public GitHub repositories*

GitHub is a popular site to upload code of software engineering projects that you also use in other courses. However, what you upload to *GitHub* must be treated with care, as it is very common to find private things like:

- FTP/SSH Credentials.
- Secret Keys (`API_KEY`, `Aws_secretkey`, etc.).
- Internal credentials, like users or employees.
- API Endpoints.
- Domain Patterns (to further search in a web site subdomains).

Finding this data is also extremely simple, as it can be found using simple searches in the *GitHub* integrated web search tool. Search examples are (assuming `target.com` is the site that we want to check the credentials for):

```
• "target.com" "dev"
• "dev.target.com"
• "target.com" API_KEY
• "target.com" password
• "api.target.com"
```



Expected results: This task will be finished when you understand the problems of leaving compromising information in *GitHub*. It does not require to do any practical activity.

BADGES AND SELF-ASSESSMENT

NOTE: You have a version of this badge table in an editable format available in the *Virtual Campus*. You can use this file to easily create a document in the format you want and take extended notes of your activities to create a log of what you did. Remember that this material is key to keep track of your self-evaluation.

Badge Level	Unlocked when	Unlocked?
	You can perform an all-default scan of an IP or IP range	
	You can perform a quick scan. You can answer this question: *Is it faster than a default one?*	
	You can perform a fast OS and service detection scan. You can answer these questions: *What information do you obtain? Is it more or less than the quick scan?*	
	You can perform a standard OS and service detection scan. You can answer these questions: *What information do you obtain? Is it more or less than the fast OS and service detection scan?*	
	You can clearly identify port states in a scan output	
	You understand the basics of how a TCP connection is made and the TCP packet structure (enough to understand `nmap` operations)	
	You understand the "dangerousness" of using the `-O` option.	
	You can locate NSE scripts related to a particular service	
	You can extract email accounts associated to a domain	
	You understand the different types and amounts of information that every possible `nmap` scan shows, and their time-information tradeoffs	
	You understand how specifying port ranges changes the `nmap` output	
	You can see the scan type and the estimated finish time of any ongoing scan	
	You can perform a slow detailed scan. You can answer these questions: *Does it take a long time? What do you recommend to decrease it?*	
	You can perform a default script scan. You can answer these questions: *What information do you obtain? Is it more or less than the others of the first level?*	
	You can use advanced reconnaissance techniques against the machines of a network and understand how the scan speed may vary depending on them.	
	You can use advanced scan techniques against a machine of a network and understand how the scan speed may vary depending on them.	
	You can combine advanced scan techniques against a machine of a network.	

	You can use public CVE repositories and the information you obtain using `nmap` to locate known product and version vulnerabilities.	
	You can answer this question: *What* `nmap` *output format do you think it is most useful to process* `nmap` *results with program code?*	
	You can run NSE scripts against SSH services	
	You can run NSE scripts against web servers	
	You can geolocate servers using `traceroute`. You can answer this question: *Do you think this geolocalization is accurate?*	
	You can run NSE scripts against a target using the WHOIS services	
	You can detect malware on remote targets using NSE scripting	
	You can identify potential CVEs with vulnerabilities on remote targets using NSE scripting	
	You can answer these questions: *What other information does the Harvester obtain? Why is it interesting from an enumeration point of view?*	
	Understand the dangerousness of finding certain files in an IP range and how easy is to do that with simple scripting. You can answer this question: *What else means that a file can be downloaded from an IP or that a 404 Not Found error is displayed when you try to do it?*	
	You can locate endpoints in *JavaScript* files and understand why these may be dangerous	
	You can determine the encryption algorithm used by the `ssh` service of the machines in Lab 8 infrastructure. You can answer this question: *Is it a strong one?* (type, key length)	
	You understand how MSF can help processing the information of a `nmap` scan you made	
	You can use different scan types to evade basic firewall blocks	
	You can inspect `nmap` scan results with MSF	
	You can geolocate servers using external services reading, and understanding, NSE script documentation, and using it to find what you want.	
	You can obtain and infer information from the TTL value of a remote machine.	
	You can answer this question: *Which type of typical scan (level 1 ones) do you think it is the most likely identified (and blocked) by a firewall or network Intrusion Detection System?*	
	Scan the machine with most services of the infrastructure we gave to you with the different timing options. You can answer these questions: *Does the scan time*	

	substantially change? are the results altered? If not, do you think they will change in a real scenario? Do you think this affects scan precision?	
	You know the importance of finding S3 buckets in a webpage, how to use simple techniques to do that, and the criticalness of finding them unprotected.	
	You know the importance of leaving compromised data on a *GitHub* repository. You can answer this question: *What famous application had this problem on 2020?*	
	**The world will tremble with the sound of our silence**: Your `nmap` proficiency allows you to scan machine networks using both basic and advanced techniques, investigating several types of services, taking advantage of a large amount of `nmap` potential, and giving you the ability of obtaining target information in various situations and environments. Additionally, you can use other tools to enumerate extra information in common services that may be the path to find a vulnerability	