Fuente: Bing Chat AI

LABORATORIO 8-9. ENUMERACIÓN DE SERVICIOS

DEPARTAMENTO DE INFORMÁTICA. UNIVERSIDAD DE OVIEDO

Seguridad de Sistemas Informáticos | 2023 – 2024 (V4.0 *"S-81 Isaac Peral"*)

Contenido

🔙 **Infraestructura de este laboratorio** **2**

🗂️ **Bloque 1: Enumeración con *NMap* Nv1** **4**

⚒️ Ejercicio L89B1_NMAPSTART: Una simple “puesta en marcha” 4

⚒️ Ejercicio L89B1_PORTS: Tratar con puertos 6

⚒️ Ejercicio L89B1_BASICNSE: Uso del motor de *scripting* “de usar y tirar” (escaneo con *scripts* por defecto) 7

🗂️ **Bloque 2: Enumeración con *NMap* Nv2** **8**

⚒️ Ejercicio L89B2_TCP: Entender (un poco) el protocolo TCP 8

⚒️ Ejercicio L89B2_ADVDISC: Tipos de descubrimiento de hosts avanzados de *NMap* 10

⚒️ Ejercicio L89B2_SCAN: Tipos de escaneo *NMap* y evasión de *firewalls* básica 13

⚒️ Ejercicio L89B2_ADVSCAN: Ejemplos (más) avanzados de *NMap* 17

⚒️ Ejercicio L89B2_SILENTDETECT: Detección de tipo de sistema operativo sin "ruido” 18

⚒️ Ejercicio L89B2_SCANOUTPUT: Formatos de salida *NMap* 19

🗂️ **Bloque 3: Enumeración con *NMap* Nv3** **20**

⚒️ Ejercicio L89B3_NSE: *NMap Scripting Engine* (NSE) 20

⚒️ Ejercicio L89B3_INFOGAT: Técnicas de recopilación de información con NSE 22

⚒️ Ejercicio L89B3_HTTPINFOGAT: Recopilación de información HTTP con NSE 24

⚒️ Ejercicio L89B3_MALCVE: Detección de *malware* y vulnerabilidades con NSE 26

🗂️ **Bloque 4. Más allá de *NMap*. Otras técnicas de enumeración** **28**

⚒️ Ejercicio L89B4_WAZUHINV: *Wazuh* y el inventario de los agentes 28

⚒️ Ejercicio L89B4_SCRIPTSCAN: Escaneando en busca de archivos concretos 28

⚒️ Ejercicio L89B4_JSSCAN: *Endpoints* de servicios en archivos *JavaScript* 29

⚒️ Ejercicio L89B4_S3SCAN: *Buckets* de *Amazon S3* 30

⚒️ Ejercicio L89B4_GITHUBSCAN: Información confidencial en código en repositorios *GitHub* 30

⭐ **Anexo: *Cheatsheets*-resumen de *NMap*** **32**

✍️ **Insignias y Autoevaluación** **34**

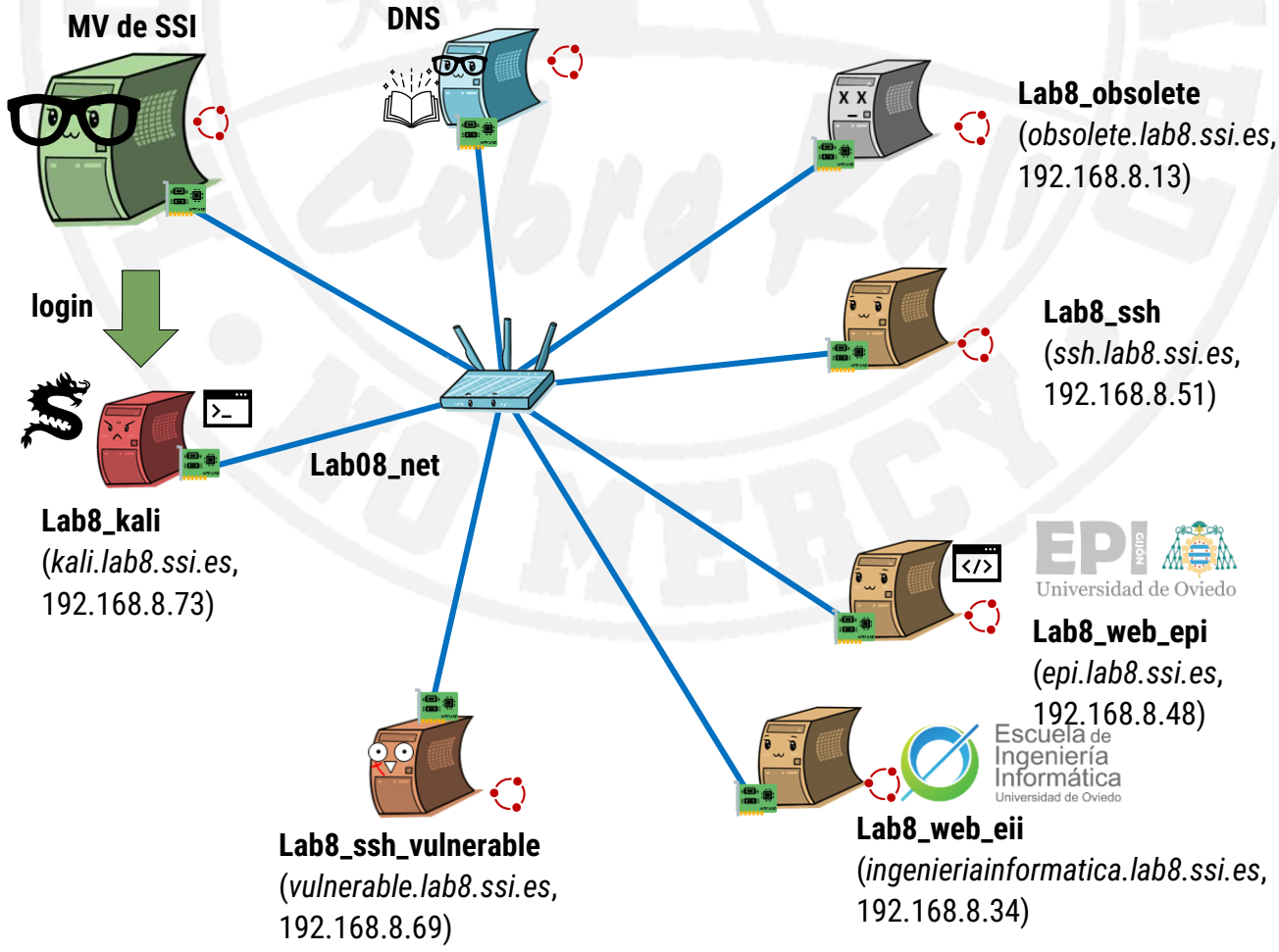
Infraestructura de este laboratorio

Este laboratorio contiene los siguientes contenedores:

- Un *Kali* "atacante", a donde se accede al ejecutar el script `enter_kali_lab8.sh`.
- Un servidor DNS.
- Dos servidores web (EII, EPI).
- Un contenedor con capacidades SSH.
- Un contenedor con capacidades SSH y más servicios potencialmente vulnerables.
- Un contenedor obsoleto (*Ubuntu* 14.04) ejecutando varios servicios.

NOTAS:

- Algunos de vosotros habéis experimentado problemas al construir el laboratorio debido a un problema temporal en la distribución del programa `theHarvester`. Mientras el proveedor del programa soluciona esto, vete a `<path donde descomprimiste los labs>\images\kalis\kali_exploiting\Dockerfile` y comenta (`#`) la línea que instala este programa en el comando `RUN apt-get` dentro de este archivo. Este es realmente un problema de la herramienta `TheHarvester`, por lo que tal vez el problema se corrija de inmediato, o incluso ya se haya corregido. No te preocupes si no te pasa. Por supuesto, el ejercicio de `theHarvester` no se puede hacer mientras exista este problema.



Por otro lado, también **tendrás que usar la "máquina *Wazuh*"** del laboratorio anterior para hacer un ejercicio. Recuerda que si no tienes en casa una máquina capaz de soportar la creación de una MV con estas características, te recomendamos que le **des prioridad a la actividad de *Wazuh* para hacerla en el laboratorio de la escuela**, donde sí tienes *hardware* que soporte esa MV. Puedes incluso empezar por ella sin problemas.

Bloque 1: Enumeración con *NMap* Nv1

Ejercicio L89B1_NMAPSTART: Una simple "puesta en marcha"

Descripción de la actividad / Aplicación práctica

Necesitas aprender usos comunes de `nmap`

Resultados Esperados

Esta tarea se terminará cuando puedas hacer estas operaciones

- Localizar y realizar un **análisis con parámetros por defecto** a todas las máquinas "vivas" en el rango de IPs de la red de la **infraestructura del Lab 8** proporcionada. Recuerda localizarlos primero, como te enseñamos en un laboratorio anterior.
- Puedas realizar un **análisis rápido** de cualquiera de las direcciones IP de la **infraestructura del Lab 8** que localizaste.
- Puedas realizar una **detección rápida del sistema operativo y sus servicios** sobre cualquiera de las direcciones IP de la **infraestructura del Lab 8** que localizaste.
- Puedas realizar un **análisis estándar de detección de servicios** sobre cualquiera de las direcciones IP de la **infraestructura del Lab 8** que localizaste.
- Puedas realizar un **análisis detallado lento** sobre cualquiera de las direcciones IP de la **infraestructura del Lab 8** que localizaste.
- Puedes contestar a estas preguntas: *¿Te das cuenta de que la información extraída por* `nmap` *es muy similar a la que extraen motores de búsqueda como Shodan (**Seminario 1**)? ¿Cuál es la principal diferencia desde el punto de vista de las máquinas que puedes alcanzar? ¿Y desde el del usuario que las hace?*

Otra información necesaria para su realización

NMap es el escáner de red de referencia, por lo que saber cómo usarlo es necesario para cualquier pentester / ingeniero de seguridad. Ya vimos cómo usarlo para realizar una detección básica de si una máquina está "viva", pero eso es sólo una mínima parte de su "potencia". Este laboratorio te da suficientes recursos para poder desbloquear una parte considerable del potencial de *NMap*. Practicar estas técnicas te convertirá en un digno "oponente" en cualquier red que quieras escanear, dándote acceso a técnicas avanzadas para utilizar esta herramienta. Dominar *NMap* no se puede hacer de inmediato, por lo que **hemos dividido este laboratorio en secciones o niveles**. Cada uno te dará una comprensión más profunda de la herramienta, dándote más posibilidades. **Este nivel es el más básico**, aunque sólo con estos contenidos podrás utilizar *NMap* de forma efectiva en la mayoría de las situaciones.

NMap es una herramienta de reconocimiento y enumeración de máquinas remotas capaz de localizar **máquinas**, sus **puertos** expuestos, los **servicios** en ejecución conectados a estos puertos, cada **tipo de servicio** concreto y su **versión**, el **tipo de sistema operativo** y mucho, mucho más. También se puede utilizar para realizar tareas de explotación y operaciones muy avanzadas gracias a su motor de *scripting*. Pero, por ahora, vamos a hacer un uso simple para sacar el máximo provecho de ella sin necesidad de mucho esfuerzo. Para ello, te presentaremos varios de los comandos de exploración más típicos (o útiles)).

Tipos de exploración típicos con todos los valores predeterminados

Estos escaneos de `nmap` son los más usados en escenarios comunes. Se pueden especificar rangos de puertos, de IPs y redes (CIDR) en los siguientes tipos de escaneo:

- **Escaneado de múltiples objetivos**: `nmap <objetivo 1> <objetivo 2> ... <objetivo N>`. P. ej.: `nmap 192.168.2.1 192.168.2.100 scanme.nmap.org`
- **Escaneado de un rango de IPs**: P. ej.: `nmap 192.168.2.1-192.168.2.100`
  Escaneado de una red completa: P. ej.: `nmap 192.168.0.0/24`
- **Escaneado de una red completa excluyendo ciertos hosts**: P. ej.: `nmap 192.168.0.0/24 --exclude 192.168.2.10`
- **Identificar el sistema operativo de las máquinas activas**: P. ej.: `nmap -O 192.168.0.0/24`
- **Escanear sólo un rango de puertos**: P. ej.: `nmap -p 22-80 192.168.0.0/24`

Análisis rápido del objetivo

Una vez que se conocen las IP de los objetivos, la mayoría de las veces la siguiente operación es realizar un **análisis rápido** de sus servicios en ejecución. Podemos hacer esto con el siguiente comando: `nmap --top-ports 20 --open <IP del objetivo>`.

En lugar de una sola IP, puedes dar **una lista de IPs en un archivo** mediante la opción `-iL`. Se trata de un **análisis rápido y eficaz** que explora los N puertos de red **estadísticamente más utilizados** (20 en el ejemplo), solo mostrando los que están abiertos o potencialmente abiertos.

Este análisis es recomendable para tener una idea inicial de los servicios de un dispositivo muy rápidamente, y así poder analizar cada sistema "interesante" más a fondo más adelante. Cómo de común es usar un puerto se obtiene del archivo `/usr/share/nmap/nmap-services`. Este archivo ordena el uso de cada puerto utilizando la **prevalencia** (relación de servidores observados que tienen ese puerto abierto, en una escala de probabilidad de 0 a 1; en otras palabras: la **probabilidad esperada de encontrar el puerto abierto en un análisis**).

Detección rápida del sistema operativo y de los servicios

El uso del parámetro `-A` te permite realizar **detección de SO y servicios con más detalle**. Al mismo tiempo, lo combinamos con `-T4` para hacer una ejecución más rápida. Por ejemplo: `nmap -A -T4 <IP objetivo>`. El tamaño de la salida es grande, por lo que es preferible volcarla a un archivo.

Detección estándar de versiones de servicios/*daemons*

Esto se puede hacer mediante el uso del parámetro `-sV`: `nmap -sV <IP objetivo>`

Exploración lenta y detallada

El comando `sudo nmap -sS -A -sV -O -p - <IP objetivo>` ejecuta un análisis detallado y lento, recomendable cuando **necesitamos explorar el objetivo elegido con mucho más detalle** (el que mencionamos antes), porque es interesante para nuestros propósitos.

- `-sS`: Utiliza un tipo de análisis TCP SYN, potencialmente más capaz de evadir *firewalls*.
- `-A`: Detecta las versiones de servicios y sistema operativo, realiza un `traceroute` y ejecuta algunos *scripts* de análisis para completar la información que obtiene.

- `-sV`: Investiga puertos abiertos para determinar qué servicio y versión están ejecutando.
- `-O`: Activa la detección de la versión y tipo del sistema operativo.
- `-p -`: Escanea un rango de puertos (`-` significa todos los puertos).

🛠 Ejercicio L89B1_PORTS: Tratar con puertos

🥷 Descripción de la actividad / Aplicación práctica

Necesitas entender **cómo escanear solamente rangos de puertos** determinados

🎖 Resultados Esperados

Esta tarea se terminará cuando:

- Puedas modificar cualquiera de los análisis anteriores **para restringirlo a intervalos de puertos específicos**, y compruebes si esto afecta a la velocidad de escaneo y/o modifica la salida `nmap` de alguna manera.
- Puedas localizar **los estados de los puertos** en la salida de cualquiera de los análisis anteriores

📔 Otra información necesaria para su realización

Los escaneos de *NMap* muestran información asociada a puertos, así que debemos entender la información que genera.

Limitaciones de puertos

NMap **escanea por defecto hasta 1000 de los puertos estadísticamente más utilizados en todo el mundo** (según el archivo mencionado anteriormente), en orden aleatorio. Esto equivale a hacer un `--top-ports 1000`. Esto da una cobertura sustancial sin el trabajo de escanear el rango de puertos 65k completo, algo que está muy desaconsejado por ser muy lento (a menos que tengas una muy buena razón para hacerlo 😊): https://www.scottbrownconsulting.com/2018/11/nmap-top-ports-frequencies-study/. Si se requiere más precisión dentro en un tiempo aceptable, **podemos probar hasta 5000**. También hay otros modificadores relacionados, y estos son ejemplos sobre cómo usarlos:

- `nmap -p 80,443 scanme.nmap.org`: Solo escanea los puertos 80 y 443 (útil si solo se quieren analizar los puertos típicos usados por un servidor web)
- `nmap -p 100-2000 scanme.nmap.org`: Escaneo aleatorio de puertos del 100 a 2000.
- `nmap -p -2000 scanme.nmap.org`: Ídem, pero del 1 a 2000.
- `nmap -p 100- scanme.nmap.org`: Ídem, pero del 100 a 65536 (nada recomendable).
- Puedes utilizar **protocolos concretos** asociados a cualquier número de puerto por utilizando letras (`T`: TCP, `U`: UDP, `S`: SCTP, `P`: IP). P. e.: `nmap -p U:53,11,13,T:22-25,80,443,8080 scanme.nmap.org`
- También hay dos modificadores adicionales

- `-F` **(Escaneo rápido):** En lugar de 1000, escanea los 100 puertos más utilizados, aleatoriamente
- `-r`: En lugar de utilizar un orden de puertos aleatorio, utiliza uno secuencial ascendente

Estados de los puertos

- ***Open***: Hay un servicio esperando una conexión en este puerto.
- ***Closed***: Ningún servicio parece estar esperando conexiones en este puerto.
- ***Filtered***: Los paquetes *NMap* no se reciben en este puerto, por lo que su estado actual es desconocido.
- ***Unfiltered***: Los paquetes *NMap* se reciben en este puerto, pero alguna razón impide que *NMap* sepa en qué estado está el puerto.
- ***Open/Filtered*****:** *NMap* no puede decidir en cuál de estos dos estados está el puerto.
- ***Closed/Filtered***: Ídem al anterior.

🛠 Ejercicio L89B1_BASICNSE: Uso del motor de *scripting* "de usar y tirar" (escaneo con *scripts* por defecto)

Descripción de la actividad / Aplicación práctica

Necesitas mejorar los resultados de un escaneo usando el motor de *scripting* NSE de `nmap`, pero **sin necesidad de conocer detalles** de los *scripts* que lanzas porque **usas un conjunto predeterminado**

Resultados Esperados

Esta tarea finalizará cuando puedas **lanzar un análisis con *scripts* predeterminados** en cualquiera de las direcciones IP de la **infraestructura del Lab 8** que localizaste anteriormente.

Otra *información* necesaria para su realización

Podemos lanzar una serie de *scripts* predeterminados contra los puertos escaneados que pueden mejorar en gran medida nuestros resultados de análisis, incluso sin saber exactamente lo que la herramienta está haciendo. Esto se puede lograr agregando la opción `-sC` a cualquiera de los escaneos anteriores y dejar que `nmap` haga el resto. Más adelante detallaremos los usos del motor de *scripting*.

🔍 ***El motor de scripting NMap mejora en gran medida las capacidades de la herramienta, pero usarlo está por encima de este nivel***

🗂️ Bloque 2: Enumeración con *NMap* Nv2

🛠️ Ejercicio L89B2_TCP: Entender (un poco) el protocolo TCP

🥷 Descripción de la actividad / Aplicación práctica

Consiste en **entender lo mínimo del protocolo TCP** para saber qué hacen ciertas opciones de `nmap` cuando las usamos

🏅 Resultados Esperados

Entiendes que una **conexión TCP** es un proceso que requiere tres pasos y que el paquete TCP tiene una serie de ***flags*** que permiten activar determinadas opciones del protocolo.

📖 Otra información necesaria para su realización

Aunque este no es un curso de redes, es necesario saber un poco sobre **cómo funciona** el protocolo **TCP subyacente** para dominar realmente `nmap`. Afortunadamente, no es mucho 😊. Simplemente definiremos un poco cómo se establece una conexión y cómo se estructura un paquete TCP. Esto se usará en explicaciones y ejercicios posteriores.

Según la *Wikipedia*: " *Las conexiones TCP se componen de tres etapas:*

- *Establecimiento de conexión (3-way handshake)*
- *Transferencia de datos*
- *Fin de la conexión.*

Para establecer la conexión se usa el procedimiento llamado "negociación en tres pasos" (3-way handshake).

Establecimiento de conexión

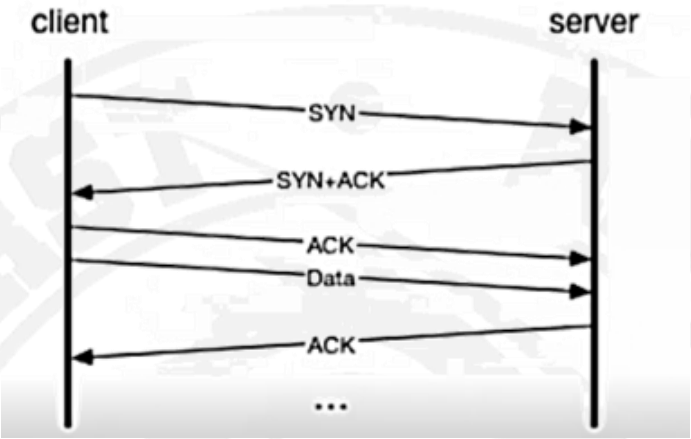
Según la *Wikipedia*: *"Establecimiento de la conexión (negociación en tres pasos):*

Aunque es posible que un par de entidades finales comiencen una conexión entre ellas simultáneamente, normalmente una de ellas abre un socket en un determinado puerto TCP y se queda a la escucha de nuevas conexiones. Es común referirse a esto como apertura pasiva, y determina el lado servidor de una conexión.

- *El lado cliente de una conexión realiza una apertura activa de un puerto enviando un paquete **SYN** inicial al servidor como parte de la negociación en tres pasos. En el lado del servidor (este receptor también puede ser una PC o alguna estación terminal) se comprueba si el puerto está abierto, es decir, si existe algún proceso escuchando en ese puerto, pues se debe verificar que el dispositivo de destino tenga este servicio activo y esté aceptando peticiones en el número de puerto que el cliente intenta usar para la sesión. En caso de no estarlo, se envía al cliente un paquete de respuesta con el bit RST activado, lo que significa el rechazo del intento de conexión.*
- *En caso de que sí se encuentre abierto el puerto, el lado servidor respondería a la petición **SYN** válida con un paquete **SYN/ACK**.*

- *Finalmente, el cliente debería responderle al servidor con un **ACK**, completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión. Es interesante notar que existe un número de secuencia generado por cada lado, ayudando de este modo a que no se puedan establecer conexiones falseadas (spoofing).*"

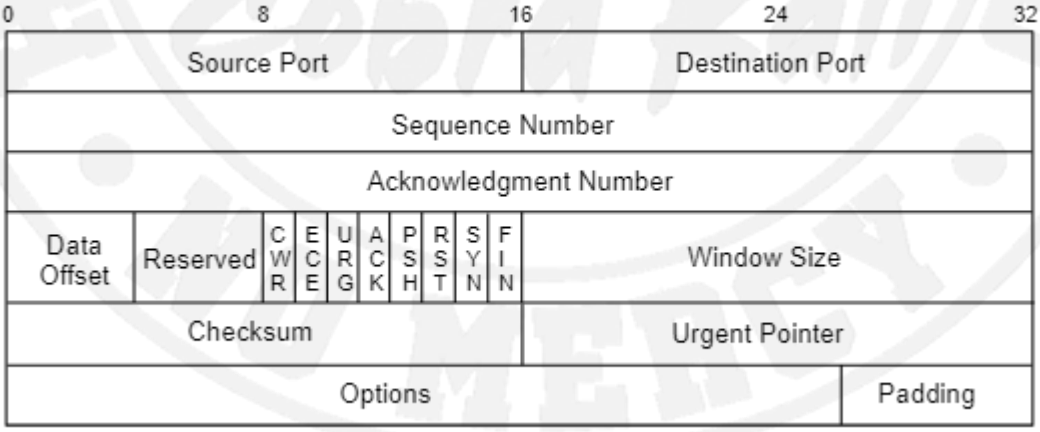
Comprender esto es importante, porque `nmap` manipula bastante el procedimiento de conexión para determinar los puertos activos y sus servicios en ejecución. La secuencia de conexión se representa en esta imagen



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

Estructura de un paquete TCP

Esta es la estructura de un paquete TCP. Aparte de los datos, otros campos importantes a recordar son los **puertos de origen y destino** (que indican qué puertos son el origen y el destino de la transmisión de datos), los seis **flags** (URG, ACK...) que se utilizan en algunos tipos de análisis, y el **tamaño de ventana,** que también se utiliza en otro tipo de análisis.



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

Ejercicio L89B2_ADVDISC: Tipos de descubrimiento de hosts avanzados de *NMap*

Descripción de la actividad / Aplicación práctica

Necesitas descubrir *hosts* que **son resistentes a escaneos estándar de** `nmap`

Resultados Esperados

Esta tarea se terminará cuando **puedas encontrar las máquinas "vivas"** en la **infraestructura del laboratorio 8** utilizando algunas de las técnicas de la Tabla 1 vista posteriormente, entendiendo la salida que producen y la velocidad de escaneo.

Otra información necesaria para su realización

En un laboratorio anterior aprendimos conceptos básicos de reconocimiento de máquinas, pero ahora podemos ir mucho más lejos usando dos técnicas adicionales para descubrir cuando una máquina está "viva".

- **El motor de *scripting*** (`--script=<nombre de script y parámetros>`): Como muestra **este *cheatsheet* de reconocimiento** (*enumeration*), hay una categoría de *scripts* (***broadcast***) dedicada a descubrir diferentes tipos de servicios en una red (ATA a través de Ethernet, AVAHI, DHCP, DNS, IGMP, UPnP...). Hay una lista grande de ellos, por lo que deberíamos dar una oportunidad a estos *scripts* para ver lo que ocurre. Toda la información se puede encontrar aquí: https://nmap.org/nsedoc/categories/broadcast.html

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

- **Técnicas avanzadas de detección**: hay varias técnicas de detección de máquinas que se describen en la siguiente tabla. Esta tabla expande la sección correspondiente del *cheatsheet*. Estas técnicas funcionan solo con los protocolos TCP e IP (y relacionados). Para otros protocolos no incluidos en la tabla debemos confiar en el motor de *scripting*.

TABLA 1. OPCIONES PARA LA DETECCIÓN DE HOST (FORMAS DE COMPROBAR MÁQUINAS "VIVAS")

`nmap <opciones> <destino u objetivos>`

Nombre de la técnica	Opción de NMap	Descripción	Notas
Detección de hosts mediante los protocolos TCP (y relacionados)			
TCP SYN	-PS	Utiliza una secuencia de inicio de conexión TCP para determinar si un host está activo	Resulta útil cuando los firewalls bloquean las solicitudes ICMP. Acepta listas de puertos
TCP SYN/ACK	-PA	Emula la respuesta ACK de una (falsa) conexión TCP. La respuesta a esto determinará si el destino existe	Resulta útil cuando los firewalls bloquean las solicitudes ICMP. Acepta listas de puertos
Descubrimiento UDP	-PY	Envía un ping UDP al destino para ver si hay alguna respuesta	Acepta listas de puertos
Detección de hosts mediante protocolos ICMP e IP			
Don't ping	-Pn	Trata a todos los hosts especificados como “en línea”: omite la detección de hosts, pero analiza sus puertos	Es útil cuando sabemos que hay cortafuegos bloqueando pings o pensamos que los objetivos están “vivos”

ICMP echo	`-PE`	Utiliza los servicios de eco del protocolo *Internet Control Message Protocol* (ICMP) para provocar una respuesta	Esto generalmente se filtra, por lo que sólo funciona en (algunas) LAN
ICMP timestamp Ping	`-PP`	Utiliza los servicios del protocolo *Internet Control Message Protocol* (ICMP) para provocar una respuesta	Puede evadir los cortafuegos si solo filtran los paquetes ICMP Echo
IP Protocol Ping	`-PO[lista de protocolos]`	Envía paquetes en un protocolo IP	Si no se especifica ninguno, se usa ICMP, IGMP e IP-in-IP. Puede evadir cortafuegos
PM Address Mask Ping	`-PM`	Utiliza los servicios del protocolo *Internet Control Message Protocol* (ICMP) para provocar una respuesta	Puede evadir los cortafuegos si solo filtran los paquetes ICMP Echo
Otras técnicas de descubrimiento de host			
Lista	`-sL`	Simplemente enumera los objetivos a escanear	Puede resolver el DNS de las direcciones IP que se le pasan
Descubrimiento ARP Ping	`-PR`	Utiliza el *Address Resolution Protocol* (ARP) para detectar objetivos	Mucho más rápido que los otros métodos de detección, pero solo funcionan en LAN (las únicas que usan ARP)
Descubrimiento ping	`-sn`	No escanea ningún puerto, solo determina si un objetivo está "vivo"	Forma típica de detectar rápidamente hosts "vivos"
Descubrimiento Traceroute	`--traceroute`	Obtiene una ruta al host	Si la ruta es correcta, el host existe
Tratar con la resolución de nombres			
Usar el DNS del sistema operativo	`--system-dns`	Hace la resolución de nombres DNS del objetivo mediante el DNS por defecto configurado en el sistema operativo	
Resolución obligatoria de DNS	`-R`	Siempre resuelve el nombre [predeterminado: a veces]	El valor predeterminado es hacer algunas resoluciones DNS
Sin resolución DNS	`-n`	Nunca realiza la resolución de DNS	Esto puede acelerar los análisis si los nombres DNS no son importantes
Especificar servidores DNS personalizados	`--dns-servers <serv1[,serv2], ... >`	Realiza la resolución de nombres DNS de destino mediante el servidor DNS que se especifique	
Telefonía IP y otros servicios SCTP			
Descubrimiento SCTP	`-PU`	Utiliza el *Stream Control Transmission Protocol (SCTP)* para tratar de provocar respuestas en los objetivos y ver si existen	Acepta listas de puertos

¿Un análisis de detección básico inicial no devolvió ningún resultado? ¿Pocos resultados? ¿Sospechas que hay muchos más dispositivos disponibles? Prueba alguna de estas técnicas. Si la máquina está allí, y das con el protocolo correcto o la técnica de evasión del *firewall* adecuada, obtendrás un resultado: **una máquina detectada como "viva"**. Y una vez que lo hagas, puedes escanearla.

🔍 ***No obstante, una máquina que da problemas para ser encontrada también los dará a la hora de escanearla. Y, por tanto, nuestras técnicas de escaneo también tienen que ser actualizadas, como veremos en la siguiente sección.***

Ejercicio L89B2_SCAN: Tipos de escaneo *NMap* y evasión de *firewalls* básica

Descripción de la actividad / Aplicación práctica

Necesitas descubrir **hosts que están protegidos por un *firewall o IDS*** *(como el **Maltrail** del laboratorio anterior)*, tratando de saltarse la protección que les da el mismo (*firewall evasion*)

Resultados Esperados

Esta tarea finalizará cuando puedas **escanear los puertos** de una de las máquinas de la **infraestructura del Laboratorio 8** utilizando **algunas de las técnicas de la Tabla 2**. Además, los servidores web `epi` y `eii` en la infraestructura son resistentes a algunas de estas técnicas de escaneo. Puedes contestar a estas preguntas:

- *¿Eres capaz de encontrar qué servidor es resistente a qué técnica?*
- *¿Eres capaz de encontrar una técnica que funcione contra el* servidor web `eii` *en esta infraestructura, que aparentemente tiene **cortafuegos** y puede resistir intentos de escaneo por defecto?*
- *¿Entiendes que si sabes el IDS que alguien tiene instalado (Ej. **Maltrail** del **Laboratorio 7**), puedes instalarlo tú para ver qué técnicas detecta y cuales no en tu propio entorno de pruebas? (no hace falta que lo hagas, solo que lo recuerdes por si fuera útil en el futuro 😊)*

🔍 ***NOTA**: Si ves que un análisis está mostrando errores o problemas y/o está tardando mucho tiempo, es aconsejable cancelarlo y probar otra técnica (¡te han pillado! 😊)*

Otra información necesaria para su realización

El siguiente *cheatsheet* es como el anterior, pero en lugar de las técnicas de detección de host nos muestra dos formas de **realizar operaciones de escaneo avanzadas** (que requieren hosts ya localizados).

- **(Otra vez) el motor de *script* de *NMap*** (`--script=<nombre de script y parámetros>`). Para hacer escaneos debemos considerar el uso de *scripts* de las siguientes categorías, como se describe en el *cheatsheet*. El uso de *scripts* para enumeración se detallará en el siguiente nivel:
  - `discovery`: https://nmap.org/nsedoc/categories/discovery.html
  - `external`: https://nmap.org/nsedoc/categories/external.html
  - `version`: https://nmap.org/nsedoc/categories/version.html

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
```

- **Técnicas avanzadas de análisis de puertos**: que se describen en la tabla siguiente. Se basan en las características del protocolo TCP / estructura de sus paquetes vistas antes, y debemos seleccionar los que creamos que serán más adecuados para nuestro escenario, siguiendo las descripciones dadas. Algunos pueden ser filtrados por IDS / *firewalls*, y otros no, dependiendo de su configuración. Trata de utilizar las técnicas más "sigilosas" primero para evitar ser bloqueado por un *firewall* a la primera de cambio. Si es posible, **evita utilizar la opción `-O`** en entornos reales, ya que seguramente serás bloqueado por la mayoría de los programas de tipo defensivo.

  Además, ten en cuenta que es perfectamente posible **que un puerto abierto pueda aparecer en escaneos sólo con una técnica de escaneo concreta** y no con las demás, dependiendo de la configuración del *firewall*. Esto es común en los CTF, pero en entornos reales también. No subestimes el poder de las diferentes técnicas de escaneo, ¡existen por una razón! 😊.

La siguiente tabla es **una extensión de la sección *"Scan techniques"* del *cheatsheet***. Después hay una imagen con una descripción más detallada de lo que hace cada escaneo, **pero sólo como referencia**, no

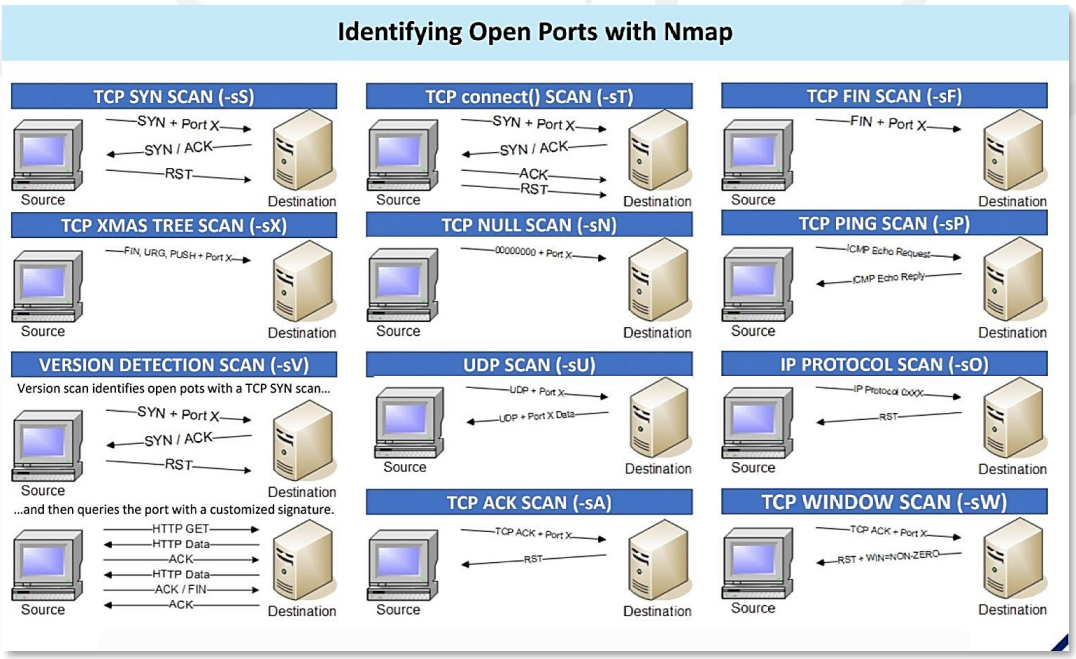
porque debas entenderlos. **Si pulsas intro mientras se ejecuta un análisis cualquiera, puedes ver el tipo de técnica de escaneado que está utilizando ese escaneo y el tiempo de finalización estimado:**

TABLA 2: TÉCNICAS DE ANÁLISIS (FORMAS DE COMPROBAR PUERTOS Y SERVICIOS EN EJECUCIÓN)

`nmap <opción> <destino u objetivos>`

Nombre de la técnica	Opción de NMap	Descripción	Notas
Principales técnicas de escaneo			
ACK scan	`-sA`	• Se utiliza para saber si hay un firewall activo en el destino. • Cualquier puerto, no importa si está abierto o cerrado responderá con un paquete RST • Pero si hay un cortafuegos, no habrá respuesta	No determina si un puerto está abierto o cerrado
Idle scan	`-sI`	Modo de escaneo muy avanzado y sigiloso	Visto en el nivel siguiente, ya que es demasiado avanzado para el nivel 2
IP protocol scan	`-sO`	• Se utiliza para analizar qué protocolos IP son compatibles con el objetivo • Los objetivos que responden paquetes *ICMP unreachable* para un puerto indican que este puerto está cerrado	Muy lento. Identifica protocolos compatibles como ICMP, IGMP, etc.
TCP Connect	`-sT`	• El equipo que escanea envía SYN • Si el ordenador escaneado responde SYN/ACK: **Puerto abierto** • NMap envía ACK • Si el ordenador escaneado no responde, o responde RST/ACK: **Puerto cerrado**	Lento, altamente detectable
TCP SYN	`-sS`	• El equipo que escanea envía SYN • Si el ordenador escaneado responde SYN/ACK: **Puerto abierto** • Si el ordenador escaneado responde RST: **Puerto cerrado**	**La técnica de escaneo más popular.** Rápido, razonablemente sigiloso, no sobrecarga (normalmente) el objetivo
UDP Scan	`-sU`	• Envía paquetes UDP de 0 bytes a cada uno de los puertos de destino • Una respuesta ICMP "unreachable port" significa que el puerto está cerrado • Sin embargo, que no haya respuesta no significa que el puerto esté filtrado (UDP no está orientado a conexión)	• Muy lento, pero utilizado por servicios populares como DNS (puerto 53), SNMP (puertos 161/162), DHCP (puertos 67/68), ... • El parámetro adicional --data-length envía datos aleatorios de la longitud indicada
Window scan	`-sW`	Utiliza el campo "Tamaño de ventana" del paquete TCP (ver imagen) para determinar el tipo de sistema operativo (los diferentes tipos de sistema operativo tienen un valor característico en este campo)	Este método no es muy fiable. Puedes ver diferentes tamaños de ventana por cada sistema operativo en: https://www.infoq.com/articles/tcp-syn-security-unauthorized-os/
Técnicas para la telefonía IP u otros servicios SCTP			
COOKIE-ECHO Scan	`-sZ`	Técnica avanzada para determinar puertos abiertos y cerrados en entornos SCTP	No se puede distinguir realmente entre puertos abiertos y filtrados (aparecen como open|filtered)
SCTP INIT Scan	`-sY`	Exploración estándar del protocolo SCTP	Utilizado para entornos que implementan este protocolo, es una técnica rápida y sigilosa especialmente dirigida a ellos
Técnicas basadas en la configuración de ciertos flags de los paquetes TCP (rompiendo el estándar TCP)			

Custom TCP Scan Flags	`--scanflags <flags>`	Es útil si desea "jugar" con los bits de los flags del paquete TCP (ver imagen) para ver qué sucede	Posibles nombres de bits a utilizar: URG, ACK, PSH, RST, SYN y FIN (--scanflags URGACKPSHRSTSYNFIN activa todos)
Maimon Scan	`-sM`	Al igual que un escaneo XMAS (ver más abajo), puede ser una alternativa al mismo	
TCP FIN	`-sF`	Se habilita el flag FIN del paquete TCP	
TCP Null	`-sN`	No se habilita ningún flag del paquete TCP	
TCP XMAS	`-sX`	Se habilitan los flag FIN, PSH y URG del paquete TCP	
Técnicas para servicios específicos			
Escaneo FTP bounce	`-b usuario:password@<Servidor FTP>:21 <destino>`	• Se conecta al servidor FTP indicado y envía archivos al destino • Envía los archivos a cada puerto para estudiar en el objetivo • El mensaje de error determinará si el puerto está abierto o cerrado	• Se debe proporcionar el usuario y la contraseña para el servidor FTP, a menos que se permitan conexiones anónimas en el servidor FTP • Puede evadir firewalls si se utiliza FTP • Es útil buscar servicios FTP en puertos no estándar (21)
Exploración RPC	`-sR`	Determina qué puertos abiertos pertenecen a llamadas RPC, incluyendo el programa que escucha las peticiones y su versión	Sólo si esperamos encontrar llamadas RPC en el sistema de destino
Técnicas combinadas/detección			
Escaneo agresivo	`-A`	Acción combinada de las opciones -sC, -sV, -O y --traceroute	Lo más probable es que se filtre, evita usarlo si es posible
Análisis de script predeterminado	`-sC`	Arranca los scripts adecuados para cada puerto estudiado pertenecientes a la categoría de script NSE "default" (https://nmap.org/nsedoc/categories/default.html)	Equivalente a --script-default
Análisis de detección del SO	`-O`	Intenta determinar el tipo y la versión del sistema operativo del destino	Lo más probable es que se filtre, evita usarlo si es posible
Análisis de versiones	`-sV`	Sondea puertos abiertos para determinar información del servicio/versión	



Fuente: @AllPentesting (*Twitter*)

Ejercicio L89B2_ADVSCAN: Ejemplos (más) avanzados de *NMap*

Descripción de la actividad / Aplicación práctica

Necesitas **controlar de manera precia los tipos de escaneo de** `nmap` y su *timing* para hacer escaneos avanzados

Resultados Esperados

Esta tarea finalizará cuando puedas **escanear los puertos** de una de las máquinas ubicadas en la **infraestructura del Lab 8** utilizando métodos de análisis combinados y diferentes opciones de temporización. También cuando puedas realizar una **búsqueda de vulnerabilidades** en los servicios y versiones que hayas localizado en una página web que de detalles de CVEs.

Puedes además contestar a esta pregunta: *¿Entiendes que si sabes el IDS que alguien tiene instalado (Ej.: **Maltrail** del **Laboratorio 7**), puedes instalarlo tú para ver qué técnicas de escaneo detecta y cuáles no en tu propio entorno de pruebas? (no hace falta que lo hagas, solo que lo recuerdes por si fuera útil en el futuro 😊)*

Otra información necesaria para su realización

Aparte de los ejemplos de las secciones anteriores y los nuevos métodos de reconocimiento y enumeración que podemos usar ahora combinados con ellos, mostraremos ahora ejemplos adicionales para mejorar nuestro conocimiento *de NMap*. La opción `-n` se puede añadir a cualquiera de ellos para mejorar su velocidad (no se realiza ninguna resolución DNS).

- **Escaneo TCP SYN y UDP (requiere tener privilegios de `root`)**: puedes combinar varias técnicas de análisis de la tabla anterior. E. g.: `nmap -sS -sU -Pn 192.168.13.37`. Este análisis TCP SYN y UDP tardará bastante tiempo, pero es discreto y sigiloso. Este comando comprobará alrededor de 2000 puertos TCP y UDP comunes (1000 TCP+1000 UDP de los más usados) para ver si están respondiendo. El parámetro `-Pn` se salta el escaneo vía *ping* y asume que el host está "vivo". Esto puede ser útil cuando hay un *firewall* que podría estar impidiendo las respuestas ICMP.
- **Escaneo TCP SYN y UDP para todos los puertos reservados (requiere tener privilegios de `root`)**: una variante del anterior. Ej.: `nmap -sS -sU -PN -p 1-1024 192.168.13.37`
- ***TCP Connect Scan* (no requiere tener privilegios de `root`)**: Ej.: `nmap -sT 192.168.13.37`. **Desaconsejado**. Como dijimos, trata de usar un escaneo más sigiloso primero.
- **Análisis rápido (no requiere tener privilegios de `root`)**: `nmap -T4 -F 192.168.13.37`. Usa **opciones de temporización** predefinidas para mejorar la velocidad de escaneo y escanea solo 100 puertos (`-F`). Las **opciones de temporización de *NMap*** aparecen en esta tabla. Por defecto se utiliza `T3`. `T0` es muy lento, pero también prácticamente indetectable (¡supuestamente! 😉). Las opciones de temporización solo son un conjunto predefinido de valores para ciertos parámetros *de NMap,* que también se pueden ajustar individualmente para obtener más control (visto en el siguiente nivel).

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

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

- **Verbose**: `nmap -T4 -A -v 192.168.13.37`. Con la opción `-v` añadida a la mayoría de los escaneos se obtiene una mejor visión de lo que está haciendo *NMap;* para algunos escaneos además da detalles adicionales que el informe de la salida no proporciona.

🔍 ***En cualquier caso, cada vez que localicemos un nombre y una versión de un servicio/SO, es fácil localizar vulnerabilidades potenciales (que podrían considerarse muy peligrosas o críticas) en los sistemas escaneados mediante los servicios de esta página web o similares:*** `https://www.cvedetails.com/` ***(Tema 1 de teoría)***

🛠 Ejercicio L89B2_SILENTDETECT: Detección de tipo de sistema operativo sin "ruido"

Descripción de la actividad / Aplicación práctica

Necesitas averiguar aproximadamente el tipo de SO que tiene un objetivo remoto **sin ser detectado**

Resultados Esperados

Esta tarea finalizará cuando puedas predecir el tipo de sistema operativo de una de las máquinas ubicadas en la **infraestructura del Lab 8** **usando el valor del parámetro TTL**.

Otra información necesaria para su realización

Como la opción `-O` es muy detectable y por lo tanto **esos intentos de escaneo pueden ser bloqueados en un entorno real por IDS,** como el ***Maltrail*** del laboratorio anterior, podemos utilizar un "truco" para tratar de adivinar el tipo de sistema operativo sin levantar sospechas: ***jugar con el TTL*** (tiempo de vida) de los paquetes que enviamos al sistema.

Esto no requiere usar *NMap*, sino hacer `ping` al sistema de la siguiente forma: `ping -c 1 <IP o nombre del destino>`. Como resultado, si hay conexión, recibiremos respuestas que indican un valor `ttl`. Este valor se puede consultar en esta tabla para adivinar el tipo de sistema operativo que el objetivo tiene:

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

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

Ejercicio L89B2_SCANOUTPUT: Formatos de salida *NMap*

Descripción de la actividad / Aplicación práctica

Necesitas obtener los resultados de los escaneos de `nmap` en un formato más adecuado para procesar con otros programas o tu código

Resultados Esperados

Esta tarea finalizará cuando puedas **extraer información de `nmap` en diferentes formatos** al realizar escaneos de una de las máquinas ubicadas en la **infraestructura del Lab 8**. Concretamente, debes ser capaz de obtener un escaneo de toda la red de máquinas en XML, ya que lo utilizaremos más adelante.

Otra información necesaria para su realización

Aparte de la salida estándar tradicional, *NMap* acepta **tres formatos de salida diferentes** añadiendo los siguientes parámetros a cualquiera de los escaneos que se pueden realizar:

- ***Formato NMap*** (`-oN <fichero>`): Equivalente a redirigir la salida *NMap* a un archivo, sigue la estructura de salida tradicional vista en teoría.
- **Formato "*Grepeable*"** (`-oG <fichero >`): Transforma la salida para que cada vez que un comando `grep` encuentre una coincidencia con la palabra que estamos buscando se muestre más información. También facilita su proceso mediante expresiones regulares.
- **Formato XML** (`-oX <fichero >`): la salida tradicional contenida en una estructura XML. Aparte de ser mucho más sencillos de procesar por programa, estos archivos pueden ser importados directamente por *Metasploit* para utilizar los resultados de un análisis para comenzar operaciones de explotación.

Bloque 3: Enumeración con *NMap* Nv3

Ejercicio L89B3_NSE: *NMap Scripting Engine* (NSE)

Descripción de la actividad / Aplicación práctica

Necesitas **entender como trabajar con el *NSE scripting engine* de** `nmap` para sacarle aún más partido a la herramienta

Resultados Esperados

Esta tarea finalizará cuando puedas **localizar *scripts* relacionados con un determinado servicio**, protocolo admitido o característica que quieras explorar.

Otra información necesaria para su realización

Como dijimos, la funcionalidad de *NMap* se puede ampliar en gran medida gracias a su **motor de *scripting* NSE**.

🔍 ***Hay más de 600 scripts diferentes cuya documentación se puede consultar aquí: https://nmap.org/nsedoc/.***

Los *scripts* se instalan en `/usr/share/nmap/scripts` y los que hay disponibles en un momento dado se pueden enumerar mediante `locate *.nse`. Como se dijo en el anterior nivel, para fines de enumeración se recomiendan las siguientes categorías:

- `discovery` (https://nmap.org/nsedoc/categories/discovery.html): estos scripts intentan **descubrir de forma activa** más información sobre la red consultando registros públicos, dispositivos que soportan el protocolo SNMP, servicios de directorio y similares. Entre los ejemplos se incluyen `html-title` (obtiene el título de la ruta raíz de los sitios web), `smb-enum-shares` (enumera los recursos compartidos con el protocolo SAMBA) y `snmp-sysdescr` (extrae detalles de sistemas remotos a través del protocolo SNMP).
- `external` (https://nmap.org/nsedoc/categories/external.html): Los *scripts* de esta categoría **pueden enviar datos a bases de datos de terceros u otros recursos de red**. Un ejemplo es `whois-ip`, que hace una conexión a los servidores WHOIS para saber más de la dirección del objetivo. Siempre existe la posibilidad de que los dueños de las base de datos de terceros registren todo lo que les envíe, que en muchos casos incluirá tu dirección IP y la dirección del objetivo.

🔍 ***La mayoría de los scripts de NMap causan tráfico estrictamente entre el equipo de análisis y el objetivo; pero los que no lo hacen e implican a un tercero pertenecen a esta categoría. Las funcionalidades de esta categoría incluyen geolocalización IP, Shodan, SMTP, DNS, WHOIS... u otros servicios útiles para enumeración.***

- `version` (https://nmap.org/nsedoc/categories/version.html): los scripts de esta categoría especial son **una extensión de la característica de detección de versiones de productos**. Se ejecutan

automáticamente solo si se solicitó la detección de versiones (`-sV`). Su salida no se puede distinguir de la salida de la detección de versiones, ya que está embebida en ella. Algunos ejemplos son `skypev2-version`, `pptp-version` e `iax2-version`

Si no queremos examinar el contenido de cada categoría, otra forma más sencilla de localizar *scripts* adecuados es **concatenar el comando `locate` anterior con un filtro `grep` utilizando un nombre de servicio** que esperamos encontrar (o sabemos que está presente) en el servidor de destino. Como nombre de servicio nos referimos **al que aparece justo al lado del nº de puerto** en un escaneo típico de `nmap`.

🔍 ***NOTA: Antes de usar `locate` por primera vez, por favor haz un `sudo updatedb` para actualizar la base de datos de scripts.***

De esta manera, **encontraremos cualquier posible *script* que se aplique a esos servicios**, y podemos consultar su documentación para probarlos y ver si se ajustan a nuestras necesidades. Las cuatro imágenes siguientes muestran cómo utilizar esta técnica de localización de *scripts* para cuatro servicios típicos diferentes.

🔍 ***Ten en cuenta no obstante que hacer esto localizará cualquier script asociado con un servicio, no importa si es utilizado para reconocimiento, enumeración o explotación. Por lo tanto, lo que encuentres puede exceder el propósito de este laboratorio, pero ser adecuados para los dos últimos, que son más de "ataque"*** 😊

```
redondo@redondo-VirtualBox:~$ locate *.nse | grep smb
/home/redondo/nmap-7.90/scripts/smb-brute.nse
/home/redondo/nmap-7.90/scripts/smb-double-pulsar-backdoor.nse
/home/redondo/nmap-7.90/scripts/smb-enum-domains.nse
/home/redondo/nmap-7.90/scripts/smb-enum-groups.nse
/home/redondo/nmap-7.90/scripts/smb-enum-processes.nse
/home/redondo/nmap-7.90/scripts/smb-enum-services.nse
/home/redondo/nmap-7.90/scripts/smb-enum-sessions.nse
/home/redondo/nmap-7.90/scripts/smb-enum-shares.nse
/home/redondo/nmap-7.90/scripts/smb-enum-users.nse
/home/redondo/nmap-7.90/scripts/smb-flood.nse
/home/redondo/nmap-7.90/scripts/smb-ls.nse
/home/redondo/nmap-7.90/scripts/smb-mbenum.nse
/home/redondo/nmap-7.90/scripts/smb-os-discovery.nse
/home/redondo/nmap-7.90/scripts/smb-print-text.nse
/home/redondo/nmap-7.90/scripts/smb-protocols.nse
/home/redondo/nmap-7.90/scripts/smb-psexec.nse
/home/redondo/nmap-7.90/scripts/smb-security-mode.nse
/home/redondo/nmap-7.90/scripts/smb-server-stats.nse
/home/redondo/nmap-7.90/scripts/smb-system-info.nse
/home/redondo/nmap-7.90/scripts/smb-vuln-conficker.nse
/home/redondo/nmap-7.90/scripts/smb-vuln-cve-2017-7494.nse
/home/redondo/nmap-7.90/scripts/smb-vuln-cve2009-3103.nse
```

```
redondo@redondo-VirtualBox:~$ locate *.nse | grep http
/home/redondo/nmap-7.90/scripts/http-adobe-coldfusion-apsa1301.nse
/home/redondo/nmap-7.90/scripts/http-affiliate-id.nse
/home/redondo/nmap-7.90/scripts/http-apache-negotiation.nse
/home/redondo/nmap-7.90/scripts/http-apache-server-status.nse
/home/redondo/nmap-7.90/scripts/http-aspnet-debug.nse
/home/redondo/nmap-7.90/scripts/http-auth-finder.nse
/home/redondo/nmap-7.90/scripts/http-auth.nse
/home/redondo/nmap-7.90/scripts/http-avaya-ipoffice-users.nse
/home/redondo/nmap-7.90/scripts/http-awstatstotals-exec.nse
/home/redondo/nmap-7.90/scripts/http-axis2-dir-traversal.nse
/home/redondo/nmap-7.90/scripts/http-backup-finder.nse
/home/redondo/nmap-7.90/scripts/http-barracuda-dir-traversal.nse
/home/redondo/nmap-7.90/scripts/http-bigip-cookie.nse
/home/redondo/nmap-7.90/scripts/http-brute.nse
/home/redondo/nmap-7.90/scripts/http-cakephp-version.nse
/home/redondo/nmap-7.90/scripts/http-chrono.nse
/home/redondo/nmap-7.90/scripts/http-cisco-anyconnect.nse
/home/redondo/nmap-7.90/scripts/http-coldfusion-subzero.nse
/home/redondo/nmap-7.90/scripts/http-comments-displayer.nse
/home/redondo/nmap-7.90/scripts/http-config-backup.nse
/home/redondo/nmap-7.90/scripts/http-cookie-flags.nse
/home/redondo/nmap-7.90/scripts/http-cors.nse
/home/redondo/nmap-7.90/scripts/http-cross-domain-policy.nse
```

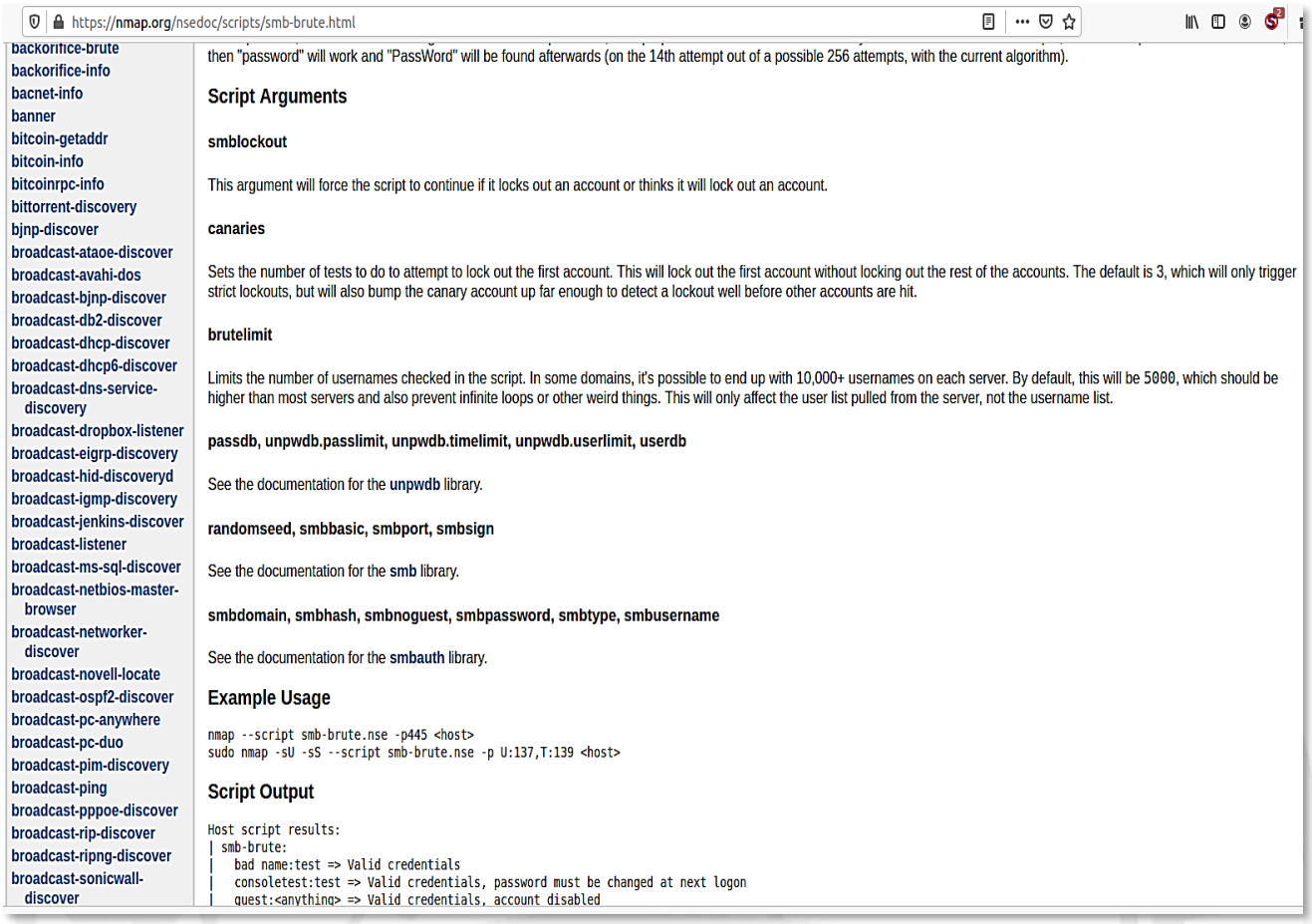
```
redondo@redondo-VirtualBox:~$ locate *.nse | grep ssh
/home/redondo/nmap-7.90/scripts/ssh-auth-methods.nse
/home/redondo/nmap-7.90/scripts/ssh-brute.nse
/home/redondo/nmap-7.90/scripts/ssh-hostkey.nse
/home/redondo/nmap-7.90/scripts/ssh-publickey-acceptance.nse
/home/redondo/nmap-7.90/scripts/ssh-run.nse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

***NMap* usa una sintaxis común independientemente del *script* que queramos utilizar**: `nmap --script <nombre script> --script-args <parámetros del script>`.

🔍 ***Cada script tiene su propio conjunto de parámetros que debe consultarse en su documentación. La documentación de cada script es muy completa, y contiene no sólo los parámetros aceptados, sino el significado de cada uno de ellos, una explicación de los propósitos del script, ejemplos de uso y la salida esperada.***

Para ilustrar el tipo de documentación de los parámetros de cada *script*, por ejemplo esto forma parte de la documentación del *script* `smb-brute` (https://nmap.org/nsedoc/scripts/smb-brute.html):



Aunque la forma recomendada de usar los scripts NSE es usando el comando `locate` para encontrar aquellos que pueden ser interesantes para cada caso específico (o examinando manualmente las categorías recomendadas), detallaremos en el resto de este bloque **varios *scripts* interesantes** para familiarizarse con las capacidades avanzadas de *NMap* que cubren usos típicos de NSE.

🔍 ***Sin embargo, recuerda que estos son sólo ejemplos, y es mejor elegir lo que más te convenga tú mismo.***

🛠 Ejercicio L89B3_INFOGAT: Técnicas de recopilación de información con NSE

🕵 Descripción de la actividad / Aplicación práctica

Necesitas usar **técnicas de recopilación de información más avanzadas** gracias a la *NSE script library*

🎖 Resultados Esperados

Esta tarea finalizará cuando puedas **lanzar estos *scripts* sobre...**

- **Una máquina adecuada** de la **infraestructura del Lab 8** (autenticación SSH y fuerza bruta DNS)
- **Sobre la web de nuestra escuela** (www.ingenieriainformatica.uniovi.es) (resto de actividades)

🔍 ***Ten en cuenta que para geolocalizar algo necesitas una IP pública expuesta a Internet, por lo que los contenedores de nuestra infraestructura no sirven al tener IPs privadas***

Otra información necesaria para su realización

🔍 ***NOTA IMPORTANTE: Los escaneos a objetivos externos (como www.uniovi.es) también se pueden hacer con la instalación de `nmap` en la máquina virtual Ubuntu.***

Métodos de autenticación SSH

El script `ssh-auth-methods` (https://nmap.org/nsedoc/scripts/ssh-auth-methods.html) localiza los **métodos de autenticación aceptados por un servicio SSH** de un objetivo. Su uso recomendado es: `sudo nmap -p22 --script ssh-auth-methods <IP del objetivo>`. Usa esto para saber si es viable probar un ataque de fuerza bruta para averiguar una contraseña contra ese servicio o no, debido a que ese SSH no acepta combinaciones de usuario/contraseña.

Fuerza bruta a servidores DNS

El script `dns-brute.nse` encontrará registros **DNS** `A` válidos probando una lista de subdominios comunes y buscando los que se resuelven correctamente. Esto encuentra **subdominios asociados con un dominio principal** de una organización, que pueden revelar nuevos objetivos sobre los que realizar evaluaciones de seguridad (hicimos lo mismo con herramientas específicas en un laboratorio anterior). **Ejemplo:** `nmap -p 80 --script dns-brute.nse <TARGET URL o IP>`

Buscar hosts en una IP

Otra forma de expandir la superficie de ataque es encontrar **hosts virtuales** en una misma dirección IP (varios sitios web que están alojados en el mismo servidor web). Esto se puede hacer mediante los scripts `hostmap-*` de la colección de scripts NSE. El script `hostmap-bfk.nse` parece funcionar razonablemente bien (los servicios de IP a Host tienen una precisión variable). **Ejemplo:** `nmap -p 80 --script hostmap-bfk.nse <TARGET URL o IP>`

Geolocalización con *traceroute*

El script `traceroute-geolocation.nse` realiza un `traceroute` a la dirección IP de destino y proporciona **datos de geolocalización** de cada "salto" de esa ruta. Esto hace que se puedan correlacionar los nombres DNS inversos de los enrutadores en dicha ruta con ubicaciones físicas de forma mucho más fácil. **Ejemplo:** `sudo nmap --traceroute --script traceroute-geolocation.nse -p 80 <URL o IP objetivo>`

🔍 ***NOTA: Tu ISP puede impedir que se obtengan coordenadas de geolocalización. Si este script no muestra coordenadas, no te preocupes y continúa con los ejercicios***

Geolocalización con bases de datos/servicios externos

Hay una serie de *scripts* NSE que ayudan a **geolocalizar una IP** mediante bases de datos o servicios externos. Un ejemplo es `ipgeolocation-geoplugin` (https://nmap.org/nsedoc/scripts/ip-geolocation-geoplugin.html). Esto quiere decir que permiten identificar la ubicación física de una dirección IP mediante el servicio web de geolocalización *Geoplugin* (http://www.geoplugin.com/).

Ejercicio L89B3_HTTPINFOGAT: Recopilación de información HTTP con NSE

Descripción de la actividad / Aplicación práctica

Necesitas obtener **más información de servidores HTTP** remotos usando la *NSE script library*

Resultados Esperados

Esta tarea finalizará cuando puedas lanzar estos *script* sobre una máquina adecuada de la **infraestructura del Lab 8**. Las dos excepciones son el **WHOIS**, que debes lanzarlo sobre la web de nuestra universidad, y **la parte de los emails**, que debes usar un dominio de Internet, por ejemplo, `uniovi.es`. Para realizar las pruebas, limita los resultados a 50 elementos y usa `Google` como fuente de datos.

Otra información necesaria para su realización

NMap viene con una amplia gama de scripts NSE **para probar aplicaciones y servidores web**. Una ventaja de usar los scripts NSE para el reconocimiento HTTP es que puedes probar diferentes aspectos del servidor web en subredes grandes en una sola operación, proporcionando rápidamente información sobre los tipos de servidores y aplicaciones en esa subred.

Métodos HTTP

Descubre qué métodos HTTP admite un servidor enviando una petición `OPTIONS`. Enumera métodos HTTP potencialmente peligrosos. https://nmap.org/nsedoc/scripts/http-methods.html. Ej. `sudo nmap -p80 –script http-methods <IP objetivo>`

Captura del *banner* HTTP

Una de las formas típicas de determinar los tipos de servicios y sus versiones es **preguntarles directamente qué son**. Esto se hace leyendo la información del servicio que proporcionan cuando se conecta con ellos, lo que se conoce comúnmente como *"banner grabbing*". Hay varias técnicas para hacer *banner grabbing*, pero *NMap* tiene el script `banner` para usar estas técnicas: `nmap --script banner <target IP>`

Rutas comunes HTTP

El *script* `http-enum.nse` **encuentra rutas válidas** en un servidor web por bruta fuerza para descubrir aplicaciones web que estén en uso. Se realizarán intentos de encontrar rutas válidas en un servidor web que coincidan con una lista de rutas conocidas sacadas de aplicaciones web comunes. Una prueba estándar incluye más de 2000 rutas. **Ejemplos:** `nmap --script http-enum <URL o IP objetivo>` o especificando una ruta de acceso base: `nmap --script http-enum --script-args http-enum.basepath'pub/' <URL de destino o IP>`. El tamaño de salida podría ser grande, por lo que es preferible volcar la salida a un archivo.

Títulos de servicios HTTP

Este *script* **agrega los títulos de las páginas web a los resultados de un análisis** de *NMap* para que se pueda tener un mejor contexto de un host que ejecuta servicios HTTP. Esto permite identificar fácilmente el propósito principal del servidor web y si ese servidor es un destino de ataque potencial. **Ejemplo:** `nmap --script http-title -sV -p 80 <red objetivo>`

Exploración de registros WHOIS

Como vimos en un laboratorio anterior, estos registros pueden tener información interesante, como el nombre real del propietario de un dominio, datos de contacto... aunque con frecuencia estos datos pertenecen a una empresa de hosting. **Más información:** https://nmap.org/nsedoc/scripts/whois-ip.html. **Sintaxis**: `nmap --script whois-ip <URL o IP de destino>`

Este script **utiliza los datos de la IANA para encontrar una base de datos WHOIS** y localizar la información que queremos. También podemos especificar el orden de servicios WHOIS que queremos utilizar: `nmap --script whois-ip –script-args whois.whodb-arin+ripe+afrinic <target IP>`. Podemos obtener más información en: https://nmap.org/nsedoc/scripts/whois-ip.html

Cuentas de correo electrónico

NMap tenía la capacidad de recopilar direcciones de correo electrónico asociadas a un dominio como parte de su base de datos de scripts NSE oficial. Desafortunadamente, **este *script* se quedó obsoleto** y no se desarrolló oficialmente ningún reemplazo. Para cubrir estas necesidades utilizaremos ***The Harvester***, una herramienta de *Kali* que nos permite recopilar información OSINT de un dominio (complementando un laboratorio anterior) que también incluye cuentas de correo electrónico asociadas a un dominio. Usa este *cheatsheet* para realizar esta tarea.

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
```

Ejercicio L89B3_MALCVE: Detección de *malware* y vulnerabilidades con NSE

Descripción de la actividad / Aplicación práctica

Necesitas **detectar *malware* y vulnerabilidades** potenciales en hosts remotos

Resultados Esperados

Esta tarea se terminará cuando puedas **usar el *script* de detección de *malware*** sobre la máquina con el mayor nº de servicios de la **infraestructura del Lab 8**. Además, debes poder usar **los *scripts* de detección de CVE** sobre la máquina "***obsolete***" de la **infraestructura del Lab 8**.

> 🔍 ***Usa ambos scripts desde un `nmap` en la máquina virtual Ubuntu (tiene acceso a todas las máquinas en la infraestructura)***

Otra información necesaria para su realización

Malware

No vamos a hacer esta actividad en este laboratorio para no forzaros a pedir una clave de API, pero puedes probarla si ya tienes una.

Otra utilidad del motor de scripts *NSE* de *NMap* es saber si un host ha sido **identificado como fuente de *malware* o distribuidor de *phishing***. Esto es posible gracias a la **API de *Safe Browsing* de *Google***, ya que el *script* `http-google-malware` pide a ese servicio este tipo de información. Para ello, necesitamos registrarnos en *Google* para obtener una clave de API adecuada: http://code.google.com/apis/safebrowsing/key_signup.html. Una vez registrados podemos utilizarlo con: `nmap -p80 --script http-google-malware --script-args http-google-malware.api-<API key> <IP o dirección del objetivo>`.

> 🔍 ***Este servicio también es utilizado por Firefox o Chrome para proteger a los usuarios mientras navegan. Más información: https://nmap.org/nsedoc/scripts/http-google-malware.html.***

`nmap` también puede detectar *malware* y *backdoors* mediante la ejecución de varias pruebas a algunos servicios de sistema operativo populares como *Identd*, *Proftpd*, *Vsftpd*, *IRC*, *SMB* y *SMTP*. También tiene un módulo para comprobar si hay signos de *malware* conocidos en servidores remotos integrando las bases de datos *Safe Browsing de Google* y *VirusTotal* también. Ejemplo: `nmap -sV --script http-malware-host <URL o IP del objetivo>`. Este es el script que sí puedes probar.

Detección CVE mediante *NMap*. Scripts de terceros

El script `--script vuln` ejecuta una prueba de vulnerabilidades completa contra nuestro un objetivo. **Ejemplo:** `nmap -Pn --script vuln <URL o IP objetivo>`. Este *script* tarda una cantidad de tiempo considerable en terminar y el tamaño de salida es grande, por lo que es preferible volcar la salida a un archivo. Los detalles de CVE descubiertos se pueden consultar en bases de datos externas que ya hemos descrito.

Aunque el objetivo de este *script* es muy interesante, el resultado podría ser más completo. Por suerte, hay un script de terceros (`nmap-vulners`) que lo mejora.

> 🔍 ***Hay muchos scripts NSE de terceros disponibles, y a veces tendrás que usarlos porque el contenido de la biblioteca NSE no es suficiente o necesitas algo muy concreto.***

Para utilizar **cualquier script de terceros**, debes seguir este procedimiento:

- Descargar la carpeta o el archivo `.nse` del *script* en la carpeta de scripts NSE (`/usr/share/nmap/scripts`). En nuestro caso concreto sería hacer `git clone https://github.com/vulnersCom/nmap-vulners.git` una vez estemos en dicha carpeta.
- Actualizar la base de datos de scripts `nmap` con `nmap --script-updatedb`

Una vez hecho esto, puedes ejecutar el *script* como cualquier *script* estándar que ya hemos visto. Para el caso que nos ocupa, simplemente hay que hacer `sudo nmap -sV --script=nmap-vulners/ <ip>`. Nótese la información adicional que este script aporta y para qué puede servirnos.

Bloque 4. Más allá de *NMap*. Otras técnicas de enumeración

Ejercicio L89B4_WAZUHINV: *Wazuh* y el inventario de los agentes

Descripción de la actividad / Aplicación práctica

Necesitas saber si el *XDR Wazuh* también **incluye información sobre los puertos y servicios** de las máquinas que está monitorizando

Resultados Esperados

Puedes localizar **servicios y puertos abiertos de las máquinas monitorizadas** por el *XDR Wazuh*. Puedes contestar a estas preguntas:

- *¿Devuelven la misma clase de información los agentes de Wazuh y* `nmap`*?*
- *¿Cuál es la principal diferencia frente a usar* `nmap`*?*

Otra información necesaria para su realización

Arranca la MV *Wazuh* vista en laboratorios anteriores y busca alguna opción del agente que hemos puesto en marcha para ver si encuentras la información mencionada, para poder analizar qué devuelve.

Ejercicio L89B4_SCRIPTSCAN: Escaneando en busca de archivos concretos

Descripción de la actividad / Aplicación práctica

Necesitas una forma rápida, sencilla **y sin usar herramientas de terceros** de localizar ciertos ficheros en sitios remotos

Resultados Esperados

Esta tarea finalizará cuando puedas utilizar el *script* de ejemplo para intentar **descargar todos los archivos** `robots.txt` del rango IP `156.35.94.1` a `156.35.94.10`

Otra información necesaria para su realización

Hay archivos concretos que son útiles desde el punto de vista de enumeración, y por lo tanto descargarlos es muy importante para estudiar un objetivo. La parte buena es que ni siquiera necesitas un programa concreto para hacerlo, solo un simple *script* que puedes desarrollar con cualquier lenguaje de programación. El siguiente es un ejemplo para descargar el archivo `phpinfo.php` del rango IP `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&
```

🔍 ***Esto realiza una solicitud masiva para un determinado archivo para todas las direcciones de un intervalo, descargándolo en la carpeta actual si existe.***

Esta técnica se puede utilizar para automatizar búsquedas de archivos interesantes en un rango de IPs de cualquier empresa, desde archivos conocidos como `robots.txt`, `index.html`, etc. hasta archivos potencialmente más comprometedores como `backup.zip`, `backup.rar`, archivos conocidos de ciertos CMS...

🛠 Ejercicio L89B4_JSSCAN: *Endpoints* de servicios en archivos *JavaScript*

🥷 Descripción de la actividad / Aplicación práctica

Necesitas localizar **servicios a los que se referencia desde el código *JavaScript*** de una página web

🏅 Resultados Esperados

Esta tarea finalizará cuando puedas utilizar la herramienta `photon` para inspeccionar la página web de nuestra escuela para buscar **posibles puntos de conexión vulnerables en archivos *JavaScript***.

📔 Otra información necesaria para su realización

Los *endpoints* de servicios en archivos *JavaScript* son sitios típicos en los que **la seguridad se relaja** debido a que los desarrolladores no cuidan adecuadamente ese aspecto en ellos pensando que solo se accederá a ellos desde los *scripts* cliente que programan.

🔍 ***Además, las herramientas de análisis automatizados generalmente no analizan estos puntos de conexión, dejando una vulnerabilidad de seguridad potencial muy interesante sin procesar.***

Podemos utilizar una herramienta de *Kali Linux*, `photon`, para extraer esta información y mucho más, y todo se guardará en un modo organizado:

- URLs tanto dentro y fuera del ámbito, así como direcciones URL con parámetros (`example.com/gallery.php?id=2`)
- Archivos *JavaScript* y *endpoints* presentes en ellos
- Cadenas basadas en un patrón *Ne* (expresión regular) personalizado
- **Datos de OSINT**: correos electrónicos, cuentas de redes sociales, *buckets* de Amazon, etc.
- Ficheros extraídos: .pdf, .png, .xml, etc.

Ejercicio L89B4_S3SCAN: *Buckets* de *Amazon S3*

Descripción de la actividad / Aplicación práctica

Necesitas **localizar *buckets* de *Amazon S3*** para inspeccionarlos

Resultados Esperados

Esta tarea finalizará cuando uses una búsqueda de *Google* para encontrar URL potenciales de Uniovi que **estén usando *buckets* de S3**. **No debes navegar por ellos, sólo localizarlos.**

Otra información necesaria para su realización

AWS Simple Storage Service (también conocido como *Amazon S3*) es utilizado por empresas que no desean instalar y administrar sus propios repositorios de almacenamiento para determinados datos.

🔍 ***Este servicio puede almacenar objetos y archivos de diferentes tipos en la nube (servidor virtual) en lugar de en una máquina física.***

Los espacios de almacenamiento de esta tecnología se denominan ***buckets***. El creador de un *bucket* puede almacenar en él código fuente, certificados, contraseñas, bases de datos...cualquier tipo de contenido. Desafortunadamente, **a veces no están adecuadamente protegidos**, y si obtienes acceso completo a un *bucket* de S3 puedes descargar, cargar o modificar cualquiera de los archivos que encuentres. Como estos **suelen ser archivos con contenido muy sensible**, los efectos suelen ser terribles y con frecuencia se producen noticias relacionadas con *buckets* S3 como la origen de ataques.

Hay maneras sencillas de encontrar cuándo un sitio web utiliza *buckets* de S3:

- Mediante el uso de una herramienta que explore la URLs de un sitio web, como la herramienta `photon` anterior o la famosa *Burp Suite*. Estas herramientas pueden encontrar URLs que terminan en `s3.amazonaws.com` que pueden llevarnos a encontrar un potencial *bucket* sin protección.
- Mediante el uso de la base de datos de *Google Hacking* del **laboratorio 2**, buscando "S3" para obtener una lista de búsquedas de *Google* que puedan revelar esta información
- Al buscar directamente estas URL tú mismo en *Google*: `site:amazonaws.com inurl:<URL>` (Ej. `site:amazonaws.com inurl:uniovi`)

Una vez encontrados, se puede utilizar el paquete `awscli` en *Kali* para interactuar con ellos si es posible.

Ejercicio L89B4_GITHUBSCAN: Información confidencial en código en repositorios *GitHub*

Descripción de la actividad / Aplicación práctica

Necesitas **localizar secretos en repositorios públicos** de *GitHub*

Resultados Esperados

Esta tarea se terminará cuando **comprendas los problemas de dejar información comprometedora en *GitHub***. Basta con probar a hacer alguna de estas búsquedas (aunque no encuentren nada) en un repositorio de código que tengas de alguna otra asignatura, tu TFG o de un compañero que te haya dado permiso para ello. **No hagas búsquedas en repositorios ajenos.**

Otra información necesaria para su realización

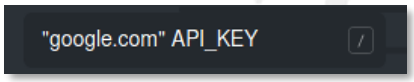
GitHub es un sitio popular para cargar código de proyectos de ingeniería de *software* que también se utiliza en otros cursos. Sin embargo, lo que subas a *GitHub* debe ser tratado con cuidado, ya que es muy común encontrar cosas privadas, generalmente llamados "secretos", como:

- Credenciales FTP/SSH.
- Claves secretas (`API_KEY`, `Aws_secretkey`, etc.).
- Credenciales internas, como usuarios o empleados.
- Puntos de conexión a APIs.
- Patrones de nombres de dominio (para realizar más búsquedas en subdominios de un sitio web).

🔍 ***En general, cualquiera de las cosas clasificadas como tal en el tema 6 de la teoría. Esta es una de las razones por las que se menciona tan insistentemente en dicho tema lo de "no hardcodear" información.***

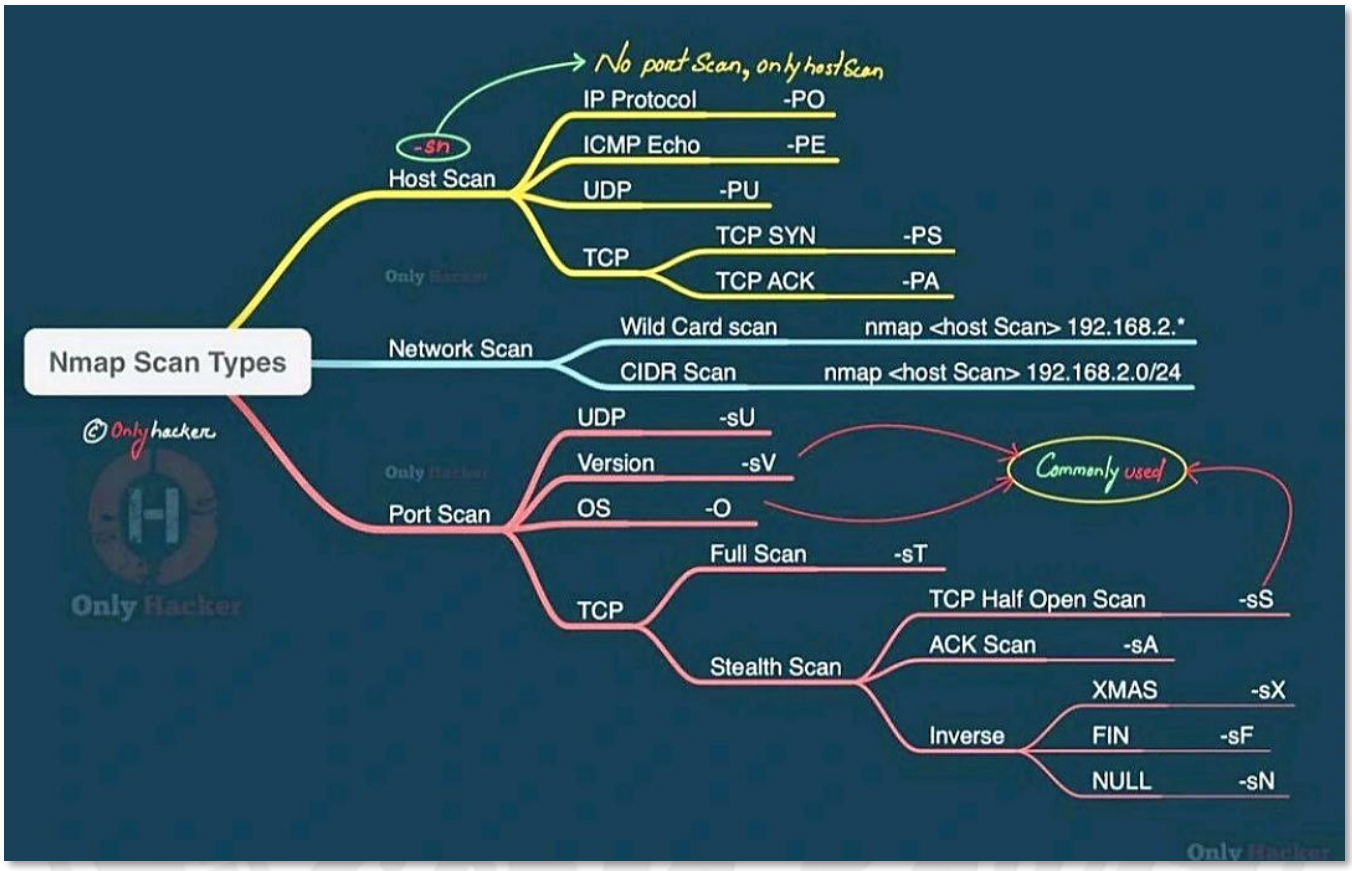
Encontrar estos datos también es muy simple, ya que se puede hacer con búsquedas en la herramienta web de búsqueda integrada *de GitHub*. Ejemplos de búsquedas son (suponiendo `target.com` es el sitio para el que queremos probar):

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

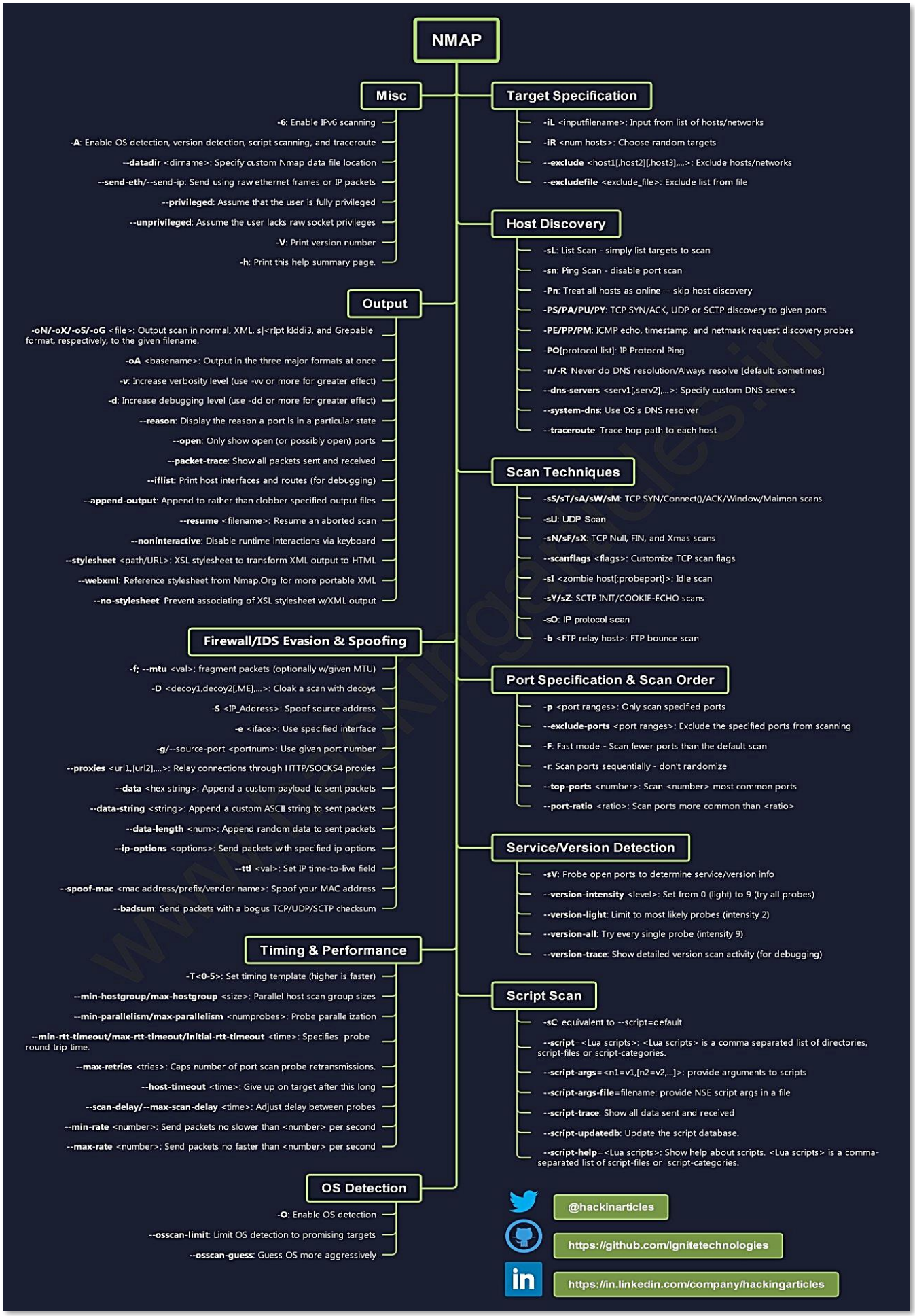


Anexo: *Cheatsheets*-resumen de *NMap*

Una vez que has visto la mayor parte de opciones importantes de `nmap`, ya estás en condiciones de usar estos *cheatsheets* cuando tengas que usar la herramienta para alguna labor de escaneo rápido entendiendo su significado, de manera que tengas a tu disposición las opciones más relevantes de un vistazo.



***Fuente**: OnlyHacker. https://twitter.com/TheOnlyHackers*


NMAP
Misc
-6: Enable IPv6 scanning
-A: Enable OS detection, version detection, script scanning, and traceroute
--datadir <dirname>: Specify custom Nmap data file location
--send-eth/--send-ip: Send using raw ethernet frames or IP packets
--privileged: Assume that the user is fully privileged
--unprivileged: Assume the user lacks raw socket privileges
-V: Print version number
-h: Print this help summary page.
Output
-oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<r|pt kiddi3, and Grepable format, respectively, to the given filename.
-oA <basename>: Output in the three major formats at once
-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
--open: Only show open (or possibly open) ports
--packet-trace: Show all packets sent and received
--iflist: Print host interfaces and routes (for debugging)
--append-output: Append to rather than clobber specified output files
--resume <filename>: Resume an aborted scan
--noninteractive: Disable runtime interactions via keyboard
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--webxml: Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet: Prevent associating of XSL stylesheet w/XML output
Firewall/IDS Evasion & Spoofing
-f; --mtu <val>: fragment packets (optionally w/given MTU)
-D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
-S <IP_Address>: Spoof source address
-e <iface>: Use specified interface
-g/--source-port <portnum>: Use given port number
--proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
--data <hex string>: Append a custom payload to sent packets
--data-string <string>: Append a custom ASCII string to sent packets
--data-length <num>: Append random data to sent packets
--ip-options <options>: Send packets with specified ip options
--ttl <val>: Set IP time-to-live field
--spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
--badsum: Send packets with a bogus TCP/UDP/SCTP checksum
Timing & Performance
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <numprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
--host-timeout <time>: Give up on target after this long
--scan-delay/--max-scan-delay <time>: Adjust delay between probes
--min-rate <number>: Send packets no slower than <number> per second
--max-rate <number>: Send packets no faster than <number> per second
OS Detection
-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
Target Specification
-iL <inputfilename>: Input from list of hosts/networks
-iR <num hosts>: Choose random targets
--exclude <host1[,host2][,host3],...>: Exclude hosts/networks
--excludefile <exclude_file>: Exclude list from file
Host Discovery
-sL: List Scan - simply list targets to scan
-sn: Ping Scan - disable port scan
-Pn: Treat all hosts as online -- skip host discovery
-PS/PA/PU/PY: TCP SYN/ACK, UDP or SCTP discovery to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
-PO[protocol list]: IP Protocol Ping
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
--dns-servers <serv1[,serv2],...>: Specify custom DNS servers
--system-dns: Use OS's DNS resolver
--traceroute: Trace hop path to each host
Scan Techniques
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sU: UDP Scan
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idle scan
-sY/sZ: SCTP INIT/COOKIE-ECHO scans
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
Port Specification & Scan Order
-p <port ranges>: Only scan specified ports
--exclude-ports <port ranges>: Exclude the specified ports from scanning
-F: Fast mode - Scan fewer ports than the default scan
-r: Scan ports sequentially - don't randomize
--top-ports <number>: Scan <number> most common ports
--port-ratio <ratio>: Scan ports more common than <ratio>
Service/Version Detection
-sV: Probe open ports to determine service/version info
--version-intensity <level>: Set from 0 (light) to 9 (try all probes)
--version-light: Limit to most likely probes (intensity 2)
--version-all: Try every single probe (intensity 9)
--version-trace: Show detailed version scan activity (for debugging)
Script Scan
-sC: equivalent to --script=default
--script=<Lua scripts>: <Lua scripts> is a comma separated list of directories, script-files or script-categories.
--script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
--script-args-file=filename: provide NSE script args in a file
--script-trace: Show all data sent and received
--script-updatedb: Update the script database.
--script-help=<Lua scripts>: Show help about scripts. <Lua scripts> is a comma-separated list of script-files or script-categories.
@hackinarticles
https://github.com/Ignitetechnologies
https://in.linkedin.com/company/hackingarticles

✍ Insignias y Autoevaluación

NOTA: Tienes una versión de esta tabla de insignias en formato editable disponible en el *Campus Virtual*. Puedes usar este archivo para crear un documento en el formato que desees y tomar notas extendidas de tus actividades para crear un log de lo que has hecho. Recuerda que el material que elabores se puede llevar a los exámenes de laboratorio.

Nivel de Insignia	Desbloqueado cuando	¿Desbloqueado?
	Puedes realizar un análisis por defecto de un intervalo de IPs o una IP	
	Puedes realizar un análisis rápido con `nmap`. Puedes responder a esta pregunta: *¿es más rápido que un análisis por defecto?*	
	Puedes realizar un análisis rápido del sistema operativo y hacer una detección de servicios. Puedes responder a estas preguntas: *¿Qué información obtienes? ¿Es más o menos información que el escaneo rápido?*	
	Puedes realizar un análisis estándar del sistema operativo y hacer una detección de servicios. Puedes responder a estas preguntas: *¿Qué información obtienes? ¿Es más o menos información que el escaneo rápido de sistema operativo y servicios?*	
	Puedes identificar claramente los estados de los puertos en una salida de un escaneo	
	Entiendes los conceptos básicos de cómo se hace una conexión TCP y la estructura del paquete TCP (suficiente para entender las operaciones de `nmap`)	
	Entiendes la "peligrosidad" de usar la opción `-o`	
	Puedes localizar *scripts* NSE relacionados con un servicio determinado	
	Puedes extraer cuentas de correo electrónico asociadas a un dominio	
	Entiendes los diferentes tipos de escaneos de `nmap` y la cantidad de información que muestran, así como el tiempo que tardan en función de la información que obtienen	
	Entiendes cómo especificar rangos de puertos cambia la salida de `nmap`	
	Puedes ver el tipo de escaneo y el tiempo de finalización estimado de cualquier análisis en curso	
	Puedes realizar un análisis detallado lento. Puedes responder a estas preguntas: *¿Tarda mucho tiempo? ¿Qué recomiendas para disminuirlo?*	
	Puedes realizar un análisis con scripts predeterminado. Puedes responder a estas preguntas: *¿Qué información obtienes? ¿Es más o menos información que otros escaneos del primer nivel?*	

	Puedes utilizar técnicas avanzadas de reconocimiento contra las máquinas de una red y comprender cómo puede variar la velocidad de escaneo dependiendo de ellas.	
	Puedes utilizar técnicas avanzadas de escaneo contra una máquina de una red y comprender cómo la velocidad de escaneo puede variar dependiendo de ellas.	
	Puedes combinar técnicas de análisis avanzadas contra una máquina de una red.	
	Puedes utilizar repositorios CVE públicos y la información que se obtiene mediante `nmap` para localizar vulnerabilidades conocidas de productos y versiones.	
	Puedes responder a esta pregunta: *¿Qué formato de salida* `nmap` *crees más útil para procesar los resultados de* `nmap` *mediante código?*	
	Puedes ejecutar *scripts* NSE contra servicios SSH	
	Puedes ejecutar *scripts* NSE contra servidores web	
	Puedes geolocalizar servidores mediante `traceroute`. Puedes responder a esta pregunta: *¿Crees que esta geolocalización es exacta?*	
	Puede ejecutar *scripts* NSE contra un objetivo usando servicios WHOIS	
	Puedes detectar malware en objetivos remotos mediante *scripts* NSE	
	Puedes identificar CVEs potenciales con vulnerabilidades en objetivos remotos mediante scripts NSE	
	Puedes responder a estas preguntas: *¿Qué otra información obtiene la the Harvester? ¿Por qué es interesante desde el punto de vista de la enumeración?*	
	Comprendes la peligrosidad de encontrar ciertos archivos en un rango de IPs y lo fácil que es hacerlo con scripts simples. Puedes responder a esta pregunta: *¿Qué más significa que un archivo se pueda descargar desde una IP o que se muestre un error 404 Not Found al hacerlo?*	
	Puedes localizar *endpoints* en archivos *JavaScript* y entender por qué pueden ser peligrosos	
	Puedes determinar el algoritmo de cifrado utilizado por el servicio `ssh` de las máquinas en la infraestructura del Lab 8. Puedes responder a esta pregunta: *¿Es fuerte?* (tipo, longitud de la clave)	
	Puedes utilizar diferentes tipos de análisis para evadir bloqueos de *firewall* básicos	
	Puedes geolocalizar servidores mediante servicios externos, leer y comprender la documentación de los scripts NSE y usarla para encontrar lo que necesitas.	
	Puedes obtener e inferir información del valor TTL de una máquina remota.	

	Puedes responder a esta pregunta: *¿Qué tipo de análisis típico (nivel 1) crees que es más probable que sea identificado (y bloqueado) por un firewall o un sistema de detección de intrusiones de red (IDS)?*	
	Escanea la máquina que tiene más servicios de la infraestructura que te dimos con las diferentes opciones de *timing*. Puedes responder a estas preguntas: *¿Cambia sustancialmente el tiempo de escaneado? Si no, ¿crees que cambiará en un escenario real? ¿Crees que esto afecta a la precisión del escaneo?*	
	Sabes la importancia de encontrar *buckets* de S3 en una página web, cómo utilizar técnicas simples para hacerlo y la importancia crítica que tiene encontrarlos desprotegidos.	
	Conoces la importancia de dejar datos comprometidos en un repositorio *GitHub*. Puedes responder a esta pregunta: *¿Qué famosa aplicación tuvo este problema en 2020?*	
	**El mundo temblará con el sonido de nuestro silencio**: Tu dominio `nmap` te permite escanear redes de máquinas utilizando técnicas básicas y avanzadas, investigando diferentes tipos de servicios, aprovechando gran parte del potencial de `nmap`, y dándote la capacidad de obtener información del objetivo en diversas situaciones y entornos. Además, puedes utilizar otras herramientas para enumerar información adicional en servicios comunes que pueden ser la vía para encontrar una vulnerabilidad	