Decrypting MSSQL Credential Passwords

A while ago I posted a blog on how to decrypt SQL Server link passwords (https://blog.netspi.com/decrypting-mssql-database-link-server-passwords/). By using the same technique it is possible to decrypt passwords for SQL Server Credentials as well. I modified the previously released password decryption script a little, namely by just changing the location where the encrypted passwords are stored, and released an updated PowerShell script for Credential decryption.

Similar remarks as with link password decryption… From the offensive point of view, this is pretty far into post exploitation as sysadmin privileges are needed on the SQL server and local administrator privileges are needed on the Windows server. From the defensive point of view, I guess this would be just another reminder that there is a way to disclose most saved passwords. So do not leave unnecessary credentials on database servers and do not grant excessive privileges for credentials used to access external resources.

SQL Server Credentials

Microsoft SQL Server allows users to add Credentials to a database. The credentials, typically Windows usernames and passwords, can be used to access resources outside SQL Server. A single credential can be used by multiple SQL logins for external access.

A simple example of credential use is the SQL Server proxy account. When xp_cmdshell is executed, by default it uses the permissions of the SQL Server service account. However, by configuring a proxy account for the server, it is possible to set xp_cmdshell to use a least privileged account for OS access rather than (quite often excessive) service account permissions.

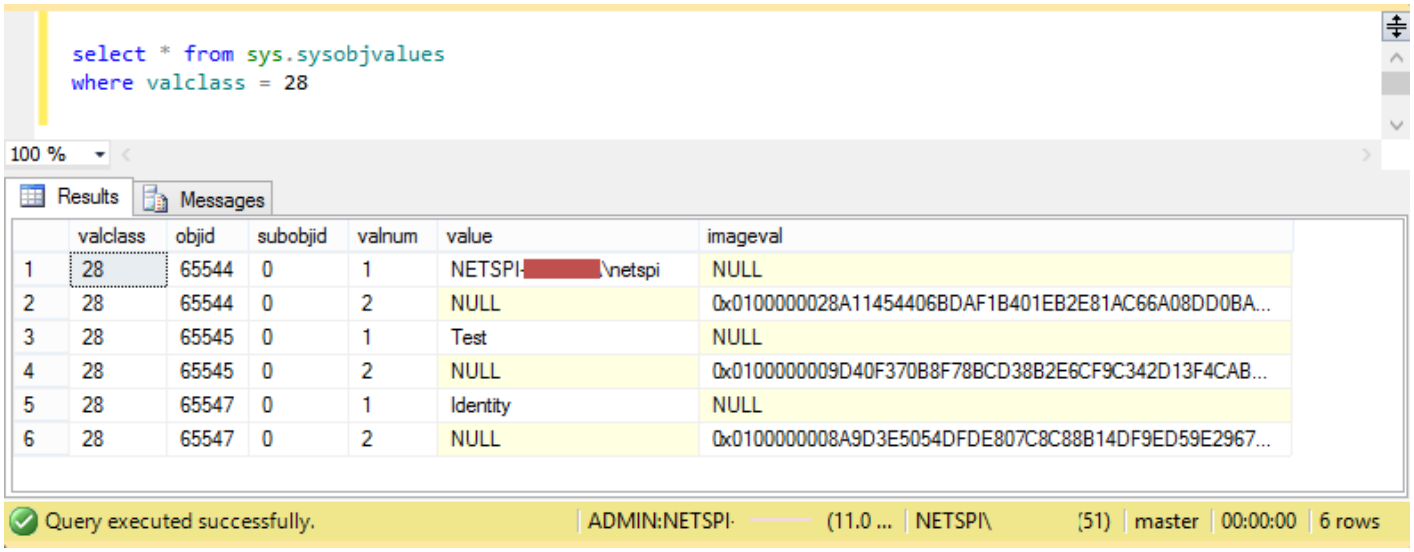
When credentials are added to a SQL Server, passwords have to be saved to the database using reversible encryption to allow for proper use of the credentials. It is possible to decrypt saved credentials password as explained in this blog.

Credential Password Storage

MSSQL stores credential passwords to the **master.sys.sysobjvalues** table. I was able to figure out the location of the encrypted passwords after looking at the definition of the **master.sys.credentials** view using the following query:

```
SELECT object_definition(OBJECT_ID('sys.credentials'))
```

Microsoft gives a pretty vague description for the table: “Exists in every database. Contains a row for each general value property of an entity.” **Master.sys.sysobjvalues** has a lot of data in it, but credential information appears to have valueclass 28. And encrypted passwords are stored in imageval column with valclass=28 and valnum=2. I could not find documentation about valclass and valnum but those values seemed to work on my test systems.



The **master.sys.sysobjvalues** table cannot be accessed using a normal SQL connection, but rather a Dedicated Administrative Connection (DAC) is needed (more information about DAC at http://technet.microsoft.com/en-us/library/ms178068%28v=sql.105%29.aspx).

MSSQL Encryption

MSSQL encryption basics were detailed in my previous blog (https://blog.netspi.com/decrypting-mssql-database-link-server-passwords/). In a nutshell, the credential passwords are encrypted using Service Master Key (SMK) which can be obtained from the server using DPAPI.

Decrypting Credential Passwords

Depending on the version of the MSSQL server, the credential passwords are encrypted using AES (MSSQL 2012+) or 3DES (MSSQL 2008 and older). Passwords stored in sys.sysobjvalues imageval column must be parsed a little prior to decryption (luckily exactly the same way as link server passwords). After the parsing credential passwords can be decrypted using the SMK.

Decrypting Credential Passwords with PowerShell – Get-MSSQLCredentialPasswords.psm1

A little modified version of "Get-MSSQLLinkPasswords.psm1", unsurprisingly named "Get-MSSQLCredentialPasswords.psm1", automates credential password decryption. The script can be downloaded from GitHub here: https://github.com/NetSPI/Powershell-Modules/blob/master/Get-MSSQLCredentialPasswords.psm1

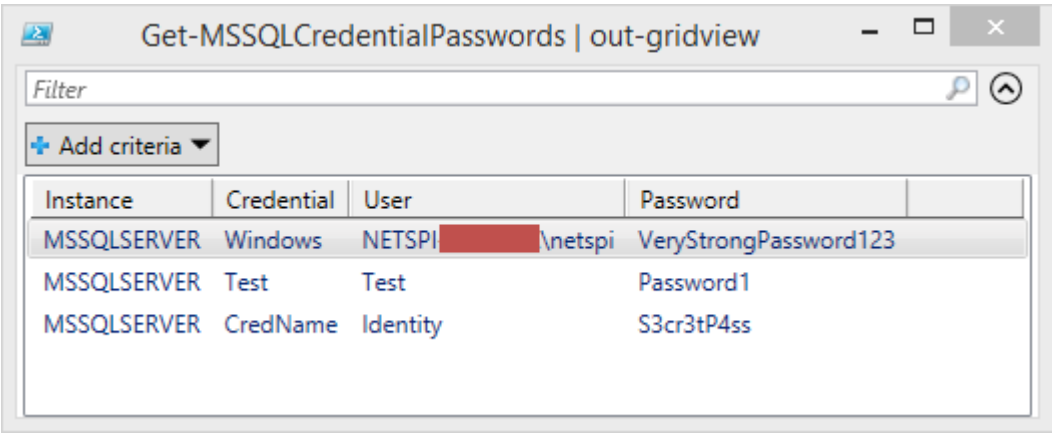
The script must be run locally on the MSSQL server (as DPAPI requires access to the local machine key). The user executing the script must also have sysadmin access to all the database instances (for the DAC connection) and local admin privileges on the Windows server (to access the entropy bytes in registry). In addition, if UAC is enabled, the script must be ran as an administrator. Below is a summary of the process used by the script.

1. Identify all of the MSSQL instances on the server.
2. Attempt to create a DAC connection to each instance.
3. Select the encrypted credential passwords from the "imageval" column of the

"master.sys.sysobjvalues" table for each instance.

4. Select the encrypted Service Master Key (SMK) from the "master.sys.key_encryptions" table of each instance where the "key_id" column is equal to 102. Select the version that has been encrypted as LocalMachine based on the "thumbprint" column.
5. Extract the entropy value from the registry location HKLM:\\SOFTWARE\Microsoft\Microsoft SQL Server\[instancename]\Security\Entropy.
6. Use the information to decrypt the SMK.
7. The script determines the encryption algorithm (AES or 3DES) used to encrypt the SMK based on SQL Server version and SMK key length.
8. Use the SMK to decrypt the credential passwords.
9. If successful, the script displays the cleartext credential passwords. Below is an example of the end result:

```
PS C:\> Get-MSSQLCredentialPasswords | out-gridview
```



I've tested the script with MSSQL 2008 and 2012. There might be some bugs, but it appears to work reliably. Please let me know if you notice any errors or if I did not account for certain situations etc.