

# Security Metadata, Service Accounts and Protecting Credentials

Lecture 4

# Security Metadata

Part 1

# Topics

- ▶ Security Principal Metadata
- ▶ Securable Metadata
- ▶ Audit Metadata
- ▶ Encryption Metadata
- ▶ Credentials Metadata
- ▶ Securing Metadata



# Introduction

- ▶ Most of the security metadata is available on SSMS, Some available through GUI, others only through T-SQL.
- ▶ This section provides some insights into interesting metadata objects and how to use them.

# Security Principal Metadata

- ▶ Use metadata to see database owners for each database under the sp\_MSHasdbaccess stored procedure or querying sys.databases catalog view

## Sp\_Mshasdbaccess

- does not accept parameters, returns name, owner and status.
- Only rows for databases that the caller has access to is returned.

## sys.databases

- return SID of each database and use SUSER\_SNAME() function

```
SELECT name, SUSER_SNAME(owner_sid)
FROM sys.databases ;
```

SUSER\_SNAME() only accepts SID as parameter and returns Login name, or Login name of caller if no input.

Listing 6-1 Retrieve Database Owners From sys.databases



# Security Principal Metadata

## Finding Effective User Permissions

- ▶ `sys.fn_my_permissions()` in Table 6-1 shows all permissions a user has.
- ▶ The function returns the columns detailed in Table 6-2
- ▶ Change the behavior of returning information about the caller using `EXECUTE AS`.
- ▶ Listing 6-2 shows how `EXECUTE AS` statement can be used to specify name of a Login or User as the identity used in the execution context.
- ▶ Use `REVERT` to change the session back.

Parameter	Description
securable	The name of the securable against which you wish to determine a user's permissions
securable_class	The type of securable that will be interrogated—for example, SERVER, DATABASE, or OBJECT

Table 6-1 `sys.fn_my_permissions` Parameters



# Security Principal Metadata

```
USE master
GO
CREATE LOGIN DemoLogin WITH
PASSWORD=N'Pa$$w0rd', CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF
ALTER SERVER ROLE sysadmin ADD MEMBER
DemoLogin
USE WideWorldImporters
GO
SELECT SUSER_SNAME() ;
EXECUTE AS LOGIN = 'DemoLogin' ;
SELECT SUSER_SNAME() ;
REVERT ;
SELECT SUSER_SNAME() ;
```

Listing 6-2 EXECUTE AS Example

Column	Description
entity_name	The name of the securable
subentity_name	If the securable has columns, then subentity_name contains the name of the column. Otherwise, it will be NULL.
permission_name	The name of the permission assigned to the security principal

Table 6-2 sys.fn\_my\_permissions



# Security Principal Metadata

- ▶ Listing 6-3 shows how `sys.fn_my_permissions` can be used with `EXECUTE AS` to find a user's effective permissions at the instance, database and object levels in a single query.
- ▶ If the database uses a different collation, use `COLLATE` statement to prevent issues with the script
- ▶ In theory, the script uses 3 separate queries and union the results. First is each object from `sys.objects` with the name, schema name. The second uses `sys.databases`. The third resolves the user's effective permissions against the instance. These query results are combined.



# Security Principal Metadata

```
EXECUTE AS LOGIN = 'DemoLogin'
    SELECT o.name
           , a.entity_name
           , a.subentity_name
           , a.permission_name
    FROM sys.objects o
    CROSS APPLY sys.fn_my_permissions(CONCAT(QUOTENAME(SCHEMA_NAME(schema_id)),
'.', QUOTENAME(o.name)), 'OBJECT') a
    UNION ALL
    SELECT d.name, a.entity_name, a.subentity_name, a.permission_name
    FROM sys.databases d
    CROSS APPLY fn_my_permissions(QUOTENAME(d.name), 'DATABASE') a
    UNION ALL
    SELECT @@SERVERNAME COLLATE Latin1_General_CI_AS, a.entity_name,
a.subentity_name, a.permission_name
    FROM fn_my_permissions(NULL, 'SERVER') a
    ORDER BY 1
REVERT
```

Listing 6-3 Find a User's Effective Permissions



# Securable Metadata

## Code Signing

- ▶ **Code injection can cause security breaches**, and you can protect against them (in part) by using **code signing**.
- ▶ The script in Listing 6-4 will report on which stored procedures in the database have been code signed and if the signature is valid.
- ▶ The script from Listing 6-4 uses two security metadata objects. The first is `sys.Certificates`. The columns returned by this catalog view are detailed in Table 6-3.

```
DECLARE @thumbprint VARBINARY(20) ;
SET @thumbprint =
(
SELECT thumbprint
FROM sys.certificates
WHERE name LIKE '%SchemaSigningCertificate%'
) ;
SELECT entity_id
      , SCHEMA_NAME(o.schema_id) + '.' + OBJECT_NAME(entity_id) AS
ProcedureName
      , is_signed
      , is_signature_valid
FROM sys.fn_check_object_signatures ('certificate', @thumbprint) cos
INNER JOIN sys.objects o
      ON cos.entity_id = o.object_id
WHERE cos.type = 'SQL_STORED_PROCEDURE' ;
GO
```

Listing 6-4 Check  
Objects' Signatures



# Securable Metadata

Column	Description
name	The name of the certificate
certificate_id	The ID of the certificate
principal_id	The ID of the database user that owns the certificate
pvt_key_encryption_type	The encryption method of the private key. Possible values are: <ul style="list-style-type: none"><li>• NA - Indicating that there is no private key associated with the certificate</li><li>• MK - Indicating that encryption is by the Database Master Key</li><li>• PW - Indicating that encryption is by password</li><li>• SK - Indicating that encryption is by the Service Master Key</li></ul>
pvt_key_encryption_type_desc	The textual description of the private key encryption type. Possible values are: <ul style="list-style-type: none"><li>• NO_PRIVATE_KEY</li><li>• ENCRYPTED_BY_MASTER_KEY</li><li>• ENCRYPTED_BY_PASSWORD</li><li>• ENCRYPTED_BY_SERVICE_MASTER_KEY</li></ul>
is_active_for_begin_dialog	Specifies if the certificate is allowed to be used to begin an encrypted Service Broker conversation. <ul style="list-style-type: none"><li>• 0 - Indicates that it is not allowed to start an encrypted Service Broker conversation</li><li>• 1 - Indicates that it is allowed to start an encrypted Service Broker conversation</li></ul>
issuer_name	The name of the authority that issued the certificate
cert_serial_number	The serial number of the certificate
sid	The Login SID of the certificate
string_sid	The name of the Login SID
subject	The subject associated with the certificate
expiry_date	The certificate's expiry date
start_date	The certificate's start date
thumbprint	The SHA-1 hash of the certificate
attested_by	Internal use
pvt_key_last_backup_date	The date and time that the certificate was last backed up

Table 6-3 sys.Certificates Columns



# Securable Metadata

- ▶ Also used in the script is **sys.fn\_check\_object\_signatures function** - used to return information regarding object signatures and their validity, based on the thumbprint of a certificate or asymmetric key.
- ▶ `sys.fn_check_object_signatures` accepts the parameters detailed in Table 6-4 and returns the columns detailed in Table 6-5.
- ▶ The first part of the script retrieves the thumbprint of the database's code signing certificate from the `sys.Certificates` catalog view.
- ▶ The second part of the script, passes this thumbprint into the `sys.fn_check_object_signatures` and joins the results to the `sys.objects` catalog view to retrieve the schema name of the procedure.



# Securable Metadata

Parameter	Description
@Class	The type of thumbprint that the function will check. Acceptable values are: <ul style="list-style-type: none"><li>• Certificate</li><li>• Asymmetric key</li></ul>
@Thumbprint	The thumbprint to be checked

Table 6-4 sys.fn\_check\_object\_signatures  
Parameters

Column	Description
type	They type description of the entity
entity_id	The object ID of the evaluated entity
is_signed	Denotes if the object is signed or not <ul style="list-style-type: none"><li>• 0 - Indicates that the object is not signed</li><li>• 1 - Indicates that the object is signed</li></ul>
is_signature_valid	Denotes if the object’s signature is valid. If the object is not signed, it returns 0. <ul style="list-style-type: none"><li>• 0 - Indicates that either the object is not signed or that the signature is not valid</li><li>• 1 - Indicates that the object’s signature is valid</li></ul>

Table 6-5  
sys.fn\_check\_object\_signatures  
Columns



# Securable Metadata

## Permissions Against a Specific Table

- ▶ Straightforward task using metadata to audit who permissions to a table and who assigned them.
- ▶ `sp_table_privileges` system stored procedure is used to identify all permissions that principals have against a specific table with who granted those permissions. The parameters accepted are detailed in Table 6-6.
- ▶ `sp_table_privileges` returns columns as per in Table 6-7.
- ▶ Listing 6-5 returns results for all tables in the current database.

```
EXEC sp_table_privileges @Table_name = '%' ;
```

Listing 6-5 `sp_table_privileges`



# Securable Metadata

Parameter	Description
@table_name	The name of the table to report on
@table_owner	The name of the schema to which the table belongs
@table_qualifier	The name of the database that hosts the table
@fUsePattern	Specifies if <code>_</code> , <code>%</code> , <code>[</code> and <code>]</code> should be treated as wildcard characters <ul style="list-style-type: none"><li>• 0 – Indicates that they should be treated as literals</li><li>• 1 – Indicates that they should be treated as wildcard characters</li></ul>

Table 6-6 `sp_table_privileges`  
Parameters

Column	Description
TABLE_QUALIFIER	The database in which the table resides
TABLE_OWNER	The schema in which the table resides
TABLE_NAME	The name of the table
GRANTOR	The security principal that granted the permission
GRANTEE	The security principal that has been assigned the permission
PRIVILEGE	The permission that has been assigned
IS_GRANTABLE	Specifies if the grantee has the <code>WITH GRANT*</code> assignment .

Table 6-7 `sp_table_privileges`  
Columns



# Audit Metadata

- ▶ In the last lecture, SQL Server Audit provides a granular and lightweight method of auditing users actions within SQL Server
- ▶ One advantage of SQL Server Audit is “auditing the audit”.
- ▶ SQL Server exposes many metadata objects to assist a DBA, primarily the `sys.fn_get_audit_file()` function which returns the contents of a SQL Server Audit file.
- ▶ This function accepts 3 parameters detailed in Table 6-8 and returns the columns in Table 6-9.
- ▶ Listing 6-6 will return all records from the audit files in the `c:\audit` folder.

```
SELECT *  
FROM sys.fn_get_audit_file('c:\audit\*',NULL,NULL) ;
```

Listing 6-6 Read an Audit File





# Audit Metadata

Parameter	Description
<code>file_pattern</code>	The name of the audit file that you wish to read. This path can contain the * wildcard to read multiple files. This is useful when you have rollover files.
<code>initial_file_name</code>	Specifies the path and name of a specific file in the audit file set where the file read should begin. If not required, pass <code>NULL</code>
<code>audit_record_offset</code>	Specifies a known location with the file specified for the <code>initial_file_name</code> parameter and begin the file read at this record. If not required, pass <code>NULL</code>

Table 6-8 `sys.fn_get_audit_file()` Parameters

Column	Description
event_time	The date and time at which the audited event occurred
sequence_number	A sequence number of records, within a single audit entry, where the entry was too large to fit inside a buffer and was broken down
action_id	The ID of the action
Succeeded	Specifies if the action that caused the audit event to fire was successful. <ul style="list-style-type: none"> <li>0 - Indicates that the action failed</li> <li>1 - Indicates that the action succeeded</li> </ul>
permission_bitmask	Where appropriate, specifies the permissions that were assigned or revoked
is_column_permission	Specifies if the permission (in the <code>permission_bitmask</code> column) was a column-level permission <ul style="list-style-type: none"> <li>0 - Indicates that it was not a column-level permission</li> <li>1 - Indicates that it was a column-level permission</li> </ul>
session_id	The ID of the session in which the event occurred
server_principal_id	The Principal ID of the Login that performed the action, which caused the audit event to fire
database_principal_id	The Principal ID of the database user that performed the action, which caused the audit event to fire
target_server_principal_id	Where applicable, returns the Principal ID of the Login that was subject to a permission assignment or revocation
target_database_principal_id	Where applicable, returns the Principal ID of the database user that was subject to a permission assignment or revocation
object_id	Where applicable, returns the Object ID of the target object that caused the audit event to fire
class_type	The type of auditable entity on which the auditable event occurred
session_server_principal_name	The name of the Login in which the session was executing. This will be blank if no session was established—for example, where a failed Login has been audited.
server_principal_name	The name of the Login that performed the action, which caused the audit event to fire
server_principal_sid	The SID of the Login that performed the action, which caused the audit event to fire
database_principal_name	The name of the database user that performed the action, which caused the audit event to fire
target_server_principal_name	Where applicable, returns the name of the Login that was subject to a permission assignment or revocation
target_server_principal_sid	Where applicable, returns the SID of the Login that was subject to a permission assignment or revocation
target_database_principal_name	Where applicable, returns the name of the database user that was subject to a permission assignment or revocation
server_instance_name	The server\instance name of the instance where the audit event occurred
database_name	The name of the database in which the audit event occurred
schema_name	The schema context in which the audit event occurred
object_name	The name of the object which was the subject of the auditable event
statement	The T-SQL statement that caused the audit event to fire
additional_information	For some events, an XML document is returned, containing additional information. For example, if a failed login is audited, the additional information will include the IP address from which the login attempt originated.
file_name	The fully qualified name of the audit file
audit_file_offset	The buffer offset of the audit record within the file
user_defined_event_id	When an audit event has been written using <code>sp_audit_write</code> , returns the user defined event ID
user_defined_information	When an audit event has been written using <code>sp_audit_write</code> , returns user defined additional information

**Table 6-9 sys.fn\_get\_audit\_file() Columns**



# Encryption Metadata

## Always Encrypted Metadata

- ▶ 1:M relationship between column master keys and column encryption keys, and 1:M between column encryption keys and encrypted columns.
- ▶ Metadata is invaluable to track how data is encrypted.
- ▶ Listing 6-7 shows the joins.
- ▶ The `sys.column_encryption_keys` view returns the columns detailed in Table 6-10.
- ▶ The `sys.column_encryption_key_values` view returns the columns detailed in Table 6-11.
- ▶ The `sys.column_master_keys` view returns the columns detailed in Table 6-12.

Column	Description
name	The name of the column encryption key
column_encryption_key_id	The ID of the column encryption key
create_date	The date and time that the key was created
modify_date	The date and time that the key was last modified

Table 6-10 `sys.column_encryption_keys` Columns



# Encryption Metadata

Column	Description
column_encryption_key_id	The ID of the column encryption key
column_master_key_id	The ID of the column master key that has been used to encrypt the column encryption key
encrypted_value	The value of the column encryption key, encrypted using the column master key
encryption_algorithm_name	The algorithm used to encrypt the column encryption key

**Table 6-11**  
**sys.column\_encryption\_key\_values Columns**

Column	Description
name	The name of the column master key
column_master_key_id	The ID of the column master key
create_date	The date and time that the key was created
modify_date	The date and time that the key was last modified
key_store_provider_name	The type of key store in which the column master key is stored
key_path	The path to the key within the key store

**Table 6-12**  
**sys.column\_master\_keys**

# Encryption Metadata

## Listing 6-7 Interrogate Always Encrypted Metadata

```
SELECT
    t.name AS TableName
  , c.name AS ColumnName
  , c.encrypted_type_desc
  , c.encrypted_algorithm_name
  , cek.name AS ColumnEncryptionKeyName
  , cev.encrypted_value
  , cev.encrypted_algorithm_name
  , cmk.name as ColumnMasterKeyName
  , cmk.key_store_provider_name AS column_master_key_store_provider_name
  , cmk.key_path
FROM sys.columns c
INNER JOIN sys.column_encryption_keys cek
    ON c.column_encryption_key_id = cek.column_encryption_key_id
INNER JOIN sys.tables t
    ON c.object_id = t.object_id
JOIN sys.column_encryption_key_values cev
    ON cek.column_encryption_key_id = cev.column_encryption_key_id
JOIN sys.column_master_keys cmk
    ON cev.column_master_key_id = cmk.column_master_key_id ;
```



# Encryption Metadata

## TDE Metadata

- ▶ TDE metadata is exposed through the `sys.databases`, `sys.certificates`, and `sys.database_encryption_keys` catalog views.
- ▶ The `sys.databases` catalog view contains a column called `is_encrypted` that returns 0 if the database is not encrypted and 1 otherwise.
- ▶ Details of the certificate used to encrypt the Database Encryption Key is exposed through `sys.certificates`.
- ▶ The `sys.database_encryption_keys` catalog view exposes details of the keys used to encrypt the databases - returns one row for each database that has a database encryption key associated with it.



# Encryption Metadata

Column	Description
database_id	The ID of the Database that is encrypted using the key
encryption_state	Specifies the current state of encryption, for the database indicated by the database_id column. Possible values are: <ul style="list-style-type: none"><li>• 0 - Indicates that no encryption key is present. You will not see this status under normal operations, because if no key exists, the catalog view does not return a row.</li><li>• 1 - Indicates that the database is not encrypted. You will see this status when TDE has been encrypted, but the database encryption key has not been dropped.</li><li>• 2 - Indicates that the database is currently being encrypted. You will see this status immediately after enabling TDE on a database, while the background encryption thread is still running.</li><li>• 3 - Indicates that the database is encrypted</li><li>• 4 - Indicates that a change to the database encryption key is currently in progress</li><li>• 5 - Indicates that the database is currently being decrypted. You will see this status immediately after turning off TDE for a database, before the background thread completes.</li><li>• 6- Indicates that a change to the database encryption key, or server certificate used to encrypt the database encryption key, is currently in progress</li></ul>
create_date	The date and time that the database encryption key was created
regenerate_date	The date and time that the database encryption key was regenerated
modify_date	The date and time that the database encryption key was last modified
set_date	The date and time that the database encryption key was associated with the database
opened_date	The date and time that the database encryption key was last opened
key_algorithm	The algorithm used to encrypt the database encryption key
key_length	The length of the key
encryptor_thumbprint	The encrypted value of the certificate used to encrypt the database encryption key
encryptor_type	Indicates the type of encryptor that was used to encrypt the database encryption key. Possible values are: <ul style="list-style-type: none"><li>• ASYMMETRIC KEY</li><li>• CERTIFICATE</li></ul>
percent_complete	If the encryption_state column indicates a status of 2 or 5, this column will indicate how far through the encryption or decryption process the background thread is. If the encryption_state column indicates a different status, then this column will return 0.

Table 6-13  
sys.database\_encryption\_keys  
Columns



# Encryption Metadata

## TDE Metadata

- ▶ Listing 6-8 shows all encrypted databases on the instance.
- ▶ Listing 6-9 shows all certificates used in TDE that have not been backed up.
- ▶ Listing 6-10 shows how to use metadata to create a metadata-driven script to encrypt many databases on an instance.

```
SELECT name  
FROM sys.databases  
WHERE is_encrypted = 1 ;
```

**Listing 6-8 Return a List of Encrypted Databases**





# Encryption Metadata

```
SELECT
    DB_NAME(dek.database_id) AS DatabaseName
    ,c.name AS CertificateName
FROM WideWorldImporters.sys.dm_database_encryption_keys dek
INNER JOIN master.sys.certificates c
ON c.thumbprint = dek.encryptor_thumbprint
WHERE c.pvt_key_last_backup_date IS NULL ;
```

**Listing 6-9 Ensure that Certificates Have Been Backed Up**

```
USE master
GO
CREATE CERTIFICATE TDECert WITH SUBJECT = 'My DEK Certificate';
GO
EXEC sys.sp_MSforeachdb @command1 = 'USE ?
IF (SELECT DB_ID()) > 4
BEGIN
    IF (SELECT is_encrypted FROM sys.databases WHERE database_id = DB_ID()) = 0
    BEGIN
        CREATE DATABASE ENCRYPTION KEY
        WITH ALGORITHM = AES_128
        ENCRYPTION BY SERVER CERTIFICATE TDECert
    ALTER DATABASE ?
        SET ENCRYPTION ON
    END
END' ;
```

**Listing 6-10 Metadata-Driven Encryption Script**



# Credentials Metadata

- ▶ Credentials are security entities which map SQL Server security principals to Windows security principals to access resources outside the instances.
- ▶ Instance-level credentials can be viewed using the `sys.credentials` catalog.
- ▶ The columns are documented in Table 6-14.
- ▶ Table 6-15 shows columns returned by `sys.database_scoped_credentials` view. Scoped credentials are not mapped to a Login or User but an external entity.



# Credentials Metadata

Column	Description
Credential_id	The unique ID of the credential
Name	The name of the credential
Credential_identity	The name of the Windows user, or the key
Create_date	The date the credential was created
Modify_date	The date that the credential was last modified
Target_type	The type of credential. Possible values are: <ul style="list-style-type: none"><li>• NULL - Indicates a Windows user</li><li>• CRYPTOGRAPHIC PROVIDER - Indicates an external key</li></ul>
Target_id	The ID of the entity to which the credential is mapped. If the value is 0, it indicates that it is mapped to a Windows user. Non-0 values indicate the ID of a cryptographic provider, when the credential is mapped to a key.

Table 6-14 Columns Returned By sys.Credentials

Column	Description
Credential_id	The unique ID of the credential
Name	The name of the credential
Credential_identity	The name of the Windows user, or the key
Create_date	The date the credential was created
Modify_date	The date that the credential was last modified
Target_type	Always returns NULL for database scoped credentials
Target_id	Always returns 0 for database scoped credentials.

Table 6-15 Columns Returned By sys.database\_scoped\_credentials

```

CREATE TABLE ##DBCredentials
(
    credential_identity NVARCHAR(4000),
    name nvarchar(128),
    target_type nvarchar(100),
    target_id int
) ;
EXEC sp_msforeachdb '
INSERT INTO ##DBCredentials
SELECT
    credential_identity
    , name
    , target_type
    , target_id
FROM [?].sys.database_scoped_credentials' ;
SELECT
    'Instance Scoped'
    , credential_identity COLLATE Latin1_General_CI_AS
    , name COLLATE Latin1_General_CI_AS
    , target_type COLLATE Latin1_General_CI_AS
    , target_id
FROM sys.credentials
UNION
SELECT
    'Database Scoped'
    , credential_identity COLLATE Latin1_General_CI_AS
    , name COLLATE Latin1_General_CI_AS
    , target_type COLLATE Latin1_General_CI_AS
    , target_id
FROM ##DBCredentials ;
DROP TABLE ##DBCredentials ;

```

**Listing 6-11**  
**Viewing**  
**Credential**  
**Metadata**



# Securing Metadata

- ▶ Attackers can use metadata to gain information about the instance configuration.
- ▶ Most metadata should only become visible to users after permissions are granted.
- ▶ Use VIEW DEFINITION permission to allow users to see metadata about an object without other permissions. This can be at the scope of a database or entire instance.
- ▶ At instance level, this is achieved using VIEW ANY DEFINITION permission.
- ▶ Useful for creating metadata-driven automated scripts with principle of least privilege.
- ▶ For some metadata objects, VIEW DEFINITION does not apply since the Public role has the ability to view the associated metadata. Examples on the right.

•The metadata objects visible to the Public role:

- sys.partition\_functions
- sys.partition\_range\_values
- sys.partition\_schemes
- sys.data\_spaces
- sys.filegroups
- sys.destination\_data\_spaces
- sys.database\_files
- sys.allocation\_units
- sys.partitions
- sys.messages
- sys.schemas
- sys.configurations
- sys.sql\_dependencies
- sys.type\_assembly\_usages
- sys.parameter\_type\_usages
- sys.column\_type\_usages

# Securing Metadata

- ▶ Listing 6-12 shows an attempt to expose metadata if the overall security design of the application is weak.
- ▶ However, it should return the error message in Figure 6-1.
- ▶ The information has been provided to the attacker:
  - 1) There is a table in the database called Colors.
  - 2) The application is leaking metadata.
  - 3) The application is (probably) running through a highly privileged account.
- ▶ Advice is should always evaluate the security profile of an application holistically in order to minimize the risk of attack.
- ▶ A poorly designed application tier could leave you vulnerable.

```
SELECT 1 + name FROM sys.tables
```

Listing 6-12 Forced Information Disclosure

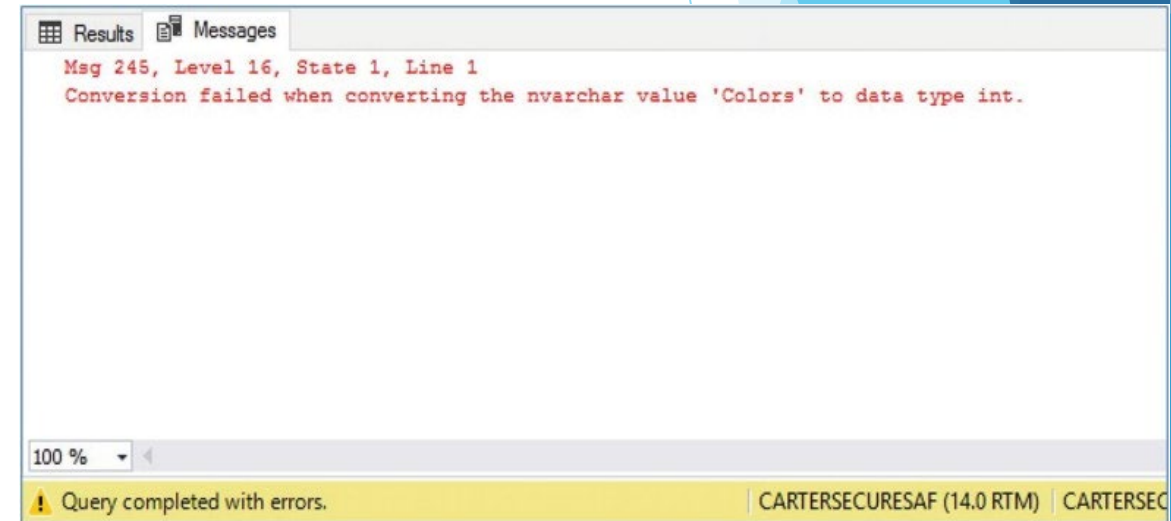


Figure 6-1 Forced error message



# Summary

- ▶ SQL Server exposes vast amounts of metadata
- ▶ This includes metadata on the security implementation of the instance.
- ▶ Security metadata can be used to assist a DBA in ensuring security policies are met.
- ▶ Useful metadata is also exposure about encryption artifacts and can be used for a variety of purposes such as auditing keys and automating a TDE.
- ▶ There are also risks of metadata exposure and subsequent attacks on SQL Server. Be mindful to take precautions, such as not use a single, highly privileged user to connect to the engine.

# Service Accounts for Security

Part 2



# Topics

- ▶ Service Account Types
  - ▶ Virtual Accounts
  - ▶ Managed Service Accounts
- ▶ SQL Server Services
- ▶ How Service Accounts can be Compromised
- ▶ Designing a Pragmatic Service Account Strategy



# Service Account Types

- ▶ A Service Account is one that is used as the security context that Windows uses to run a service.
- ▶ Each SQL Server Service needs to be configured with one which will be granted the appropriate permissions to run the service.
- ▶ SQL Server 2017 supports the following types of accounts as Service Accounts:
  1. Local User - Windows user that has been created on the local server
  2. Domain User - an AD (Active Directory) user that has been created at the domain level
  3. Built-in Accounts - NETWORK SERVICE, LOCAL SERVICE, and LOCAL SYSTEM accounts that are in Windows environments
  4. Managed Service Accounts (MSAs) - more details to come
  5. Virtual Accounts - more details to come



# Service Account Types

## Virtual Accounts

- ▶ Introduced in Windows Server 2008R2 and Windows 7.
- ▶ Created locally on server but are able to access domain resources using credentials.
- ▶ Introduced to improve isolation between services and offer the benefit of being managed without registering an SPN and changing password – done by Windows.
- ▶ Cannot be used on multiple servers and are inappropriate for AlwaysOn Failover Clusters.
- ▶ Services on the right are defaults for a stand-alone installation of SQL Server
- ▶ Even for clustered SQL Server instances, SSIS, SSRS, FD Launcher will default to a Virtual Account as the services are not cluster aware.

The accounts for the following services will default to a Virtual Account: Database Engine

- SQL Server Agent
- SQL Server Analysis Services (SSAS)
- SQL Server Integration Services (SSIS)
- SQL Server Reporting Services (SSRS)
- SQL Server Distributed Relay Controller
- SQL Server Distributed Relay Client
- FD Launcher (Full-Text Daemon Launcher service)



# Service Account Types

## Managed Service Accounts

- ▶ Similar to virtual Accounts – managed automatically and no need for SPN or password management.
- ▶ Main difference is that they are domain-level rather than local.
- ▶ Should be used in preference to Virtual Accounts when need to interact with network resources.
- ▶ An MSA is still only assigned to a single machine on the network and also cannot be used for failover cluster instances.
- ▶ gMSAs were introduced in Windows Server 2012 R2 to run on multiple servers, and for cluster implementations, but with greater complexity of their own.
  - 1) gMSA must first be created in the domain using an AD group.
  - 2) Then install the service account on the server to be used
  - 3) The server may need to be restarted between permissions
  - 4) Between reboots, SQL Server may also fail to come online as gMSA has yet to authenticate with domain. Use Automatic (Delayed Start) to resolve



# SQL Server Services

- ▶ Consider only service account requirements with least privilege when installing SQL Server.
- ▶ Elevate permissions and reduce when complete if higher permissions are required for one-off tasks.
- ▶ Table 7-1 shows the details of the two services with minimum permissions and assignments required for basic functions. Full table in [https://link.springer.com/chapter/10.1007/978-1-4842-4161-5\\_7](https://link.springer.com/chapter/10.1007/978-1-4842-4161-5_7)



# SQL Server Services

Service	User Rights Assignments	Permissions
SQL Server Database Engine	<ul style="list-style-type: none"><li>• Log on as a service</li><li>• Replace a process-level token</li><li>• Bypass traverse checking</li><li>• Adjust memory quotas for a process</li></ul>	<ul style="list-style-type: none"><li>• Start SQL Writer</li><li>• Read the Event Log service</li><li>• Read the Remote Procedure Call service</li><li>• Instid\MSSQL\backup</li><li>• Full control</li><li>• Instid\MSSQL\binn</li><li>• Read</li><li>• Execute</li><li>• Instid\MSSQL\data</li><li>• Full control</li><li>• Instid\MSSQL\FTData</li><li>• Full control</li><li>• Instid\MSSQL\Install</li><li>• Read</li><li>• Execute</li><li>• Instid\MSSQL\Log</li><li>• Full control</li><li>• Instid\MSSQL\Repldata</li><li>• Full control</li><li>• 130\shared</li><li>• Read</li><li>• Execute</li></ul>
SQL Server Agent	<ul style="list-style-type: none"><li>• Log on as a service</li><li>• Replace a process-level token</li><li>• Bypass traverse checking</li><li>• Adjust memory quotas for a process</li></ul>	<ul style="list-style-type: none"><li>• Instid\MSSQL\binn</li><li>• Full control</li><li>• Instid\MSSQL\binn</li><li>• Full control</li><li>• Instid\MSSQL\Log</li><li>• Read</li><li>• Write</li><li>• Delete</li><li>• Execute</li><li>• 130\com</li><li>• Read</li><li>• Execute</li><li>• 130\shared</li><li>• Read</li><li>• Execute</li><li>• 130\shared\Errordumps</li><li>• Read</li><li>• Write</li><li>• ServerName\EventLog</li><li>• Full control</li></ul>

Table 7-1 Some Service Account Permission and Assignment Requirements



# How Service Accounts Can Become Compromised

- ▶ If attacker already has access to an instance, then can use SMB attack to compromise service account credentials:
- ▶ Extended stored procedures like undocumented `xp_dirtree` can be executed without permissions above the Public role.
- ▶ Upon revealing the folder contents with this tool, access can be gained through the SQL Server Database Engine service account.
- ▶ An SMB Capture from a tool like Metasploit can capture the authentication request which will capture the hash password and a nonce.

# Designing a Pragmatic Service Account Strategy

- ▶ Essentially a tradeoff between security and operational supportability when selecting the service account model.
- ▶ Even if OS and domain-level pre-requisites are met, DBA teams that rely heavily on automation to manage maintenance routines may argue VAs do not meet requirements.
- ▶ Recommended environment in Figure 7-1 for organizations that use domain accounts as service accounts. The design involves a common set of service account used by all instances but not shared with data-tier applications.
- ▶ In the example, each data-tier application would require a discrete set of service accounts for lower environments.

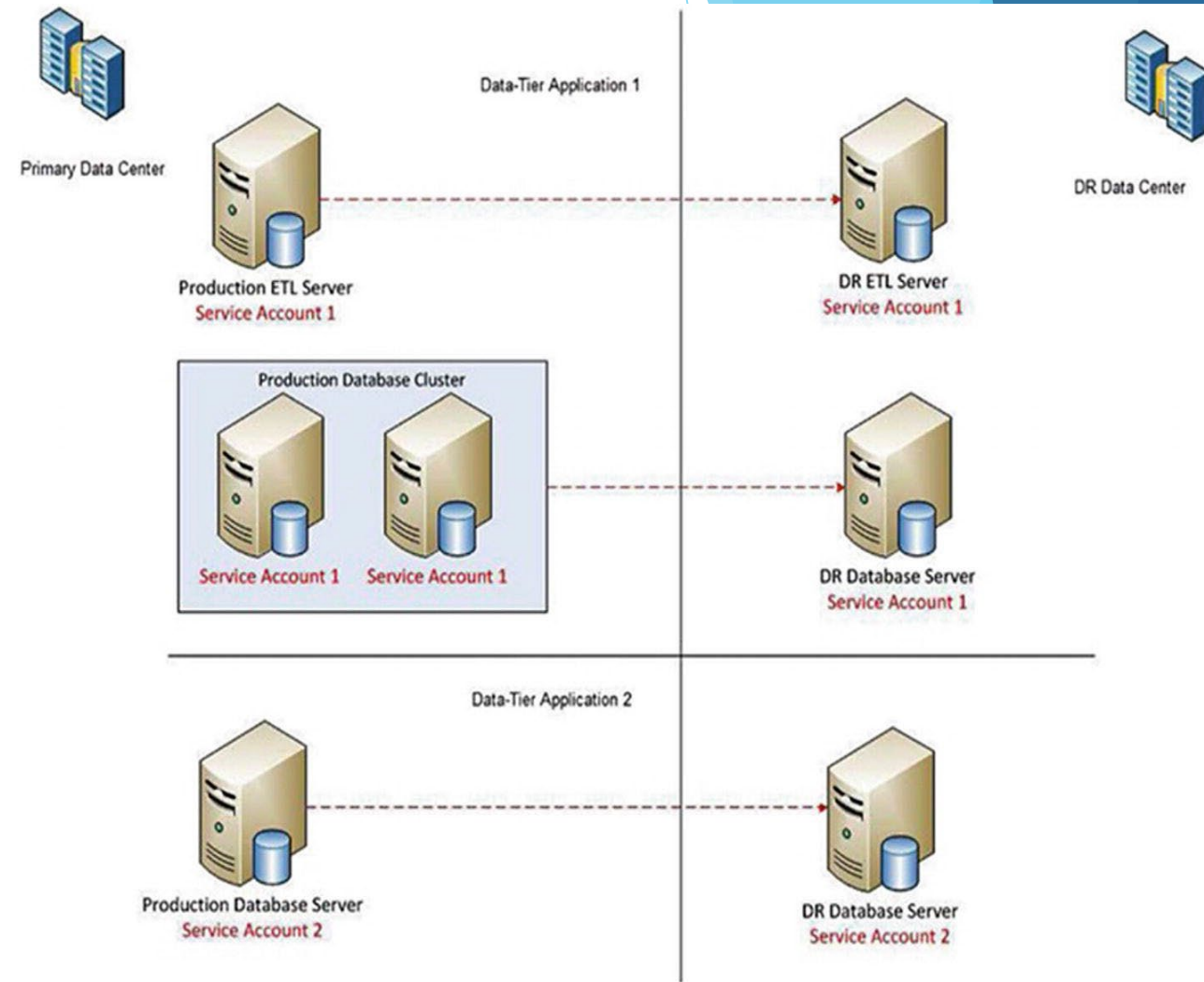


Figure 7-1 Service Account Model example





# Summary

- ▶ Depending on the components of SQL Server 2017, may have up to 11 service accounts in use across shared features and a single instance.
- ▶ Each service account has its own minimum permissions requirements.
- ▶ To mitigate the risks of service account exploits, follow least privilege principle.
- ▶ Complete isolation of services is difficult to manage and needs to be pragmatic into the overall security strategy.
- ▶ Services can run under different security context, but VAs and MSAs and gMSAs are considered better solutions provided with the correct domain functional level, OS and ability to support within DBA tooling.

# Protecting Credentials

## Part 3

# Topics

- ▶ Protecting the sa account
  - ▶ Disabling the sa account
  - ▶ Renaming the sa account
  - ▶ Ensuring reputability
  - ▶ Enforcing constant password changes
- ▶ Protecting user accounts
- ▶ Protecting windows accounts



# Protecting the SA Account

- ▶ Best practice is to use Windows Authentication, but many corporate instances still use Mixed Mode, usually due to technical reasons.
- ▶ The tools DBAs have built and grown internally relies on the use of the sa Account to run maintenance routines.
- ▶ Cost and operational risk are arguments against changing the approach.
- ▶ But it is high risk, as the sa Account if compromised has the same password on every instance.
- ▶ In Mixed Mode, the sa Account is particularly susceptible
  - 1) It is a highly privileged account and desirable
  - 2) Very well known account and is a prime target.



# Mitigating the Risks

- ▶ In Mixed mode, there are 4 steps the DBA can take to mitigate the risk and protect the sa Account:
  - 1) Disabling the sa Account - Listing 8-1
  - 2) Renaming the sa Account - Listing 8-2
  - 3) Ensuring reputability
  - 4) Enforcing password changes

```
ALTER LOGIN sa DISABLE ;  
GO
```

Listing 8-1 Disable the sa Account

```
ALTER LOGIN sa WITH NAME = AdminAccount ;  
GO
```

Listing 8-2 Rename the sa Account



# Ensuring Reputability

- ▶ Worst-case scenario is where there is a 3<sup>rd</sup> party or legacy application which the sa Account is hard-coded and cannot be modified.
- ▶ In this case, ensure the saAccount inherits the password complexity and expiry settings of the domain. Listing 8-3 enforces this.
- ▶ Some applications store the password in a configuration file, so make sure it is changed to follow policy and encrypted.
- ▶ Also set up an audit to trace any alterationis to this account. Figure 8-1 shows how to create an audit, which is similar to Lecture 2.
- ▶ You may check that the sa always has a server\_principal\_ID of 1 using Listing 8-4.

```
SELECT
    name
    ,server_principal_id
FROM sys.server_principals
WHERE name = 'sa' ;
```

Listing 8-4 Check server\_principal\_id of the sa Account

# Ensuring Reputability

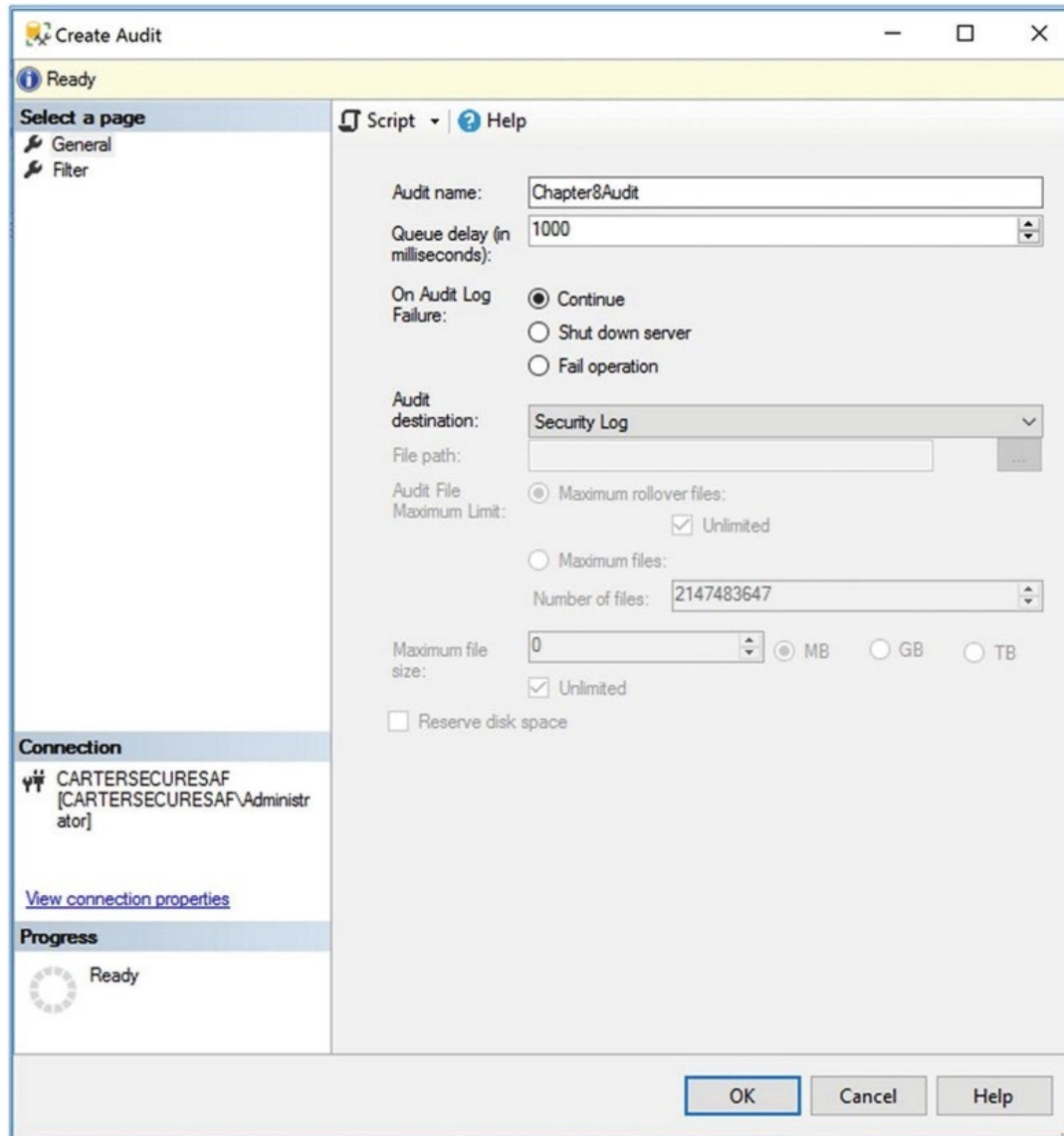


Figure 8-1 Create Audit dialog box—general page

```
ALTER LOGIN sa
WITH CHECK_EXPIRATION=ON
, CHECK_POLICY=ON ;
```

Listing 8-3 Force Password Policies for sa Account

# Ensuring Reputability

- ▶ Add a filter so only changes to the sa Account are audited, such as in Figure 8-2.
- ▶ Then create a Server Audit specification as in Figure 8-3.

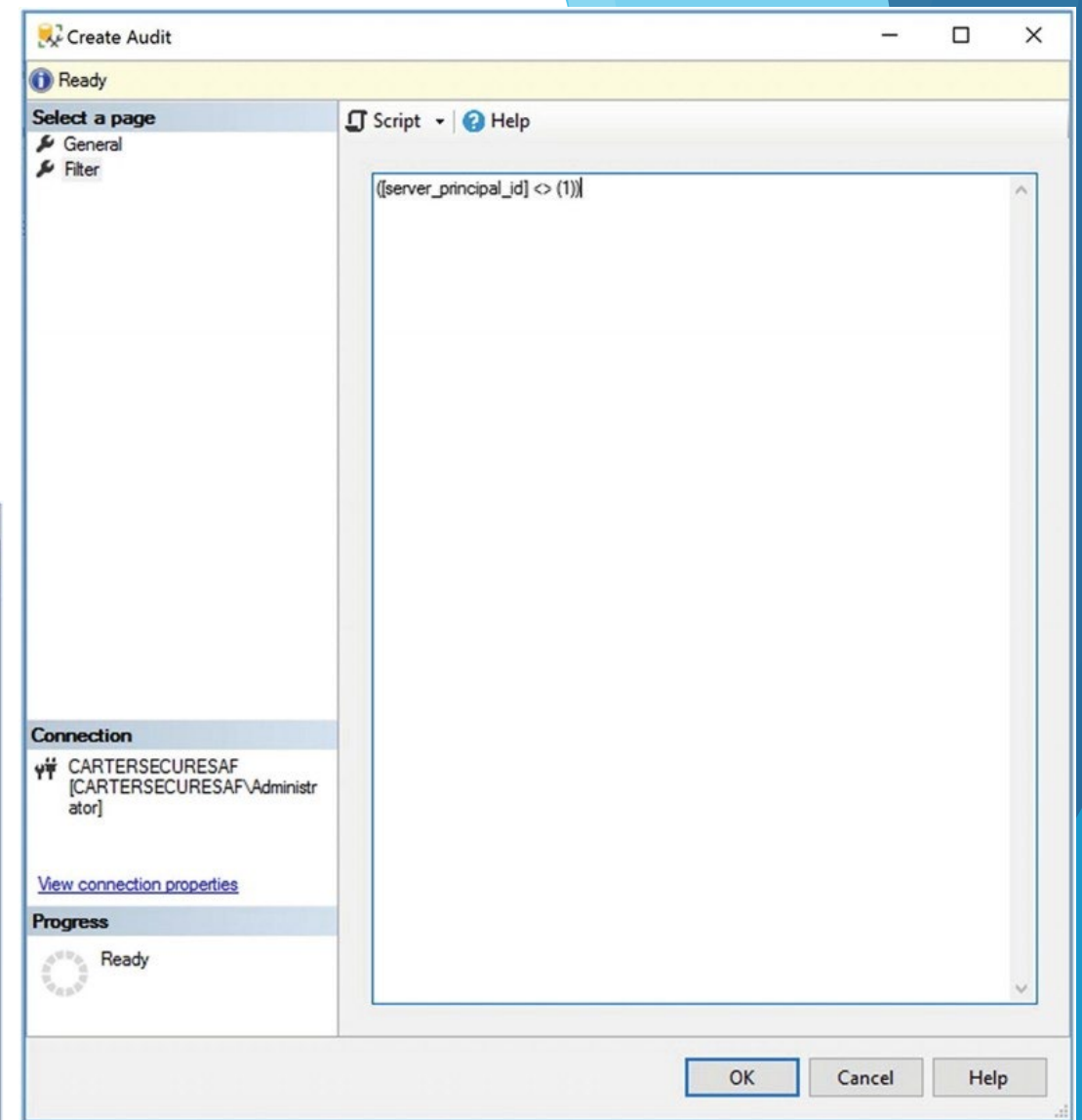


Figure 8-2 Create Audit dialog box—filter page

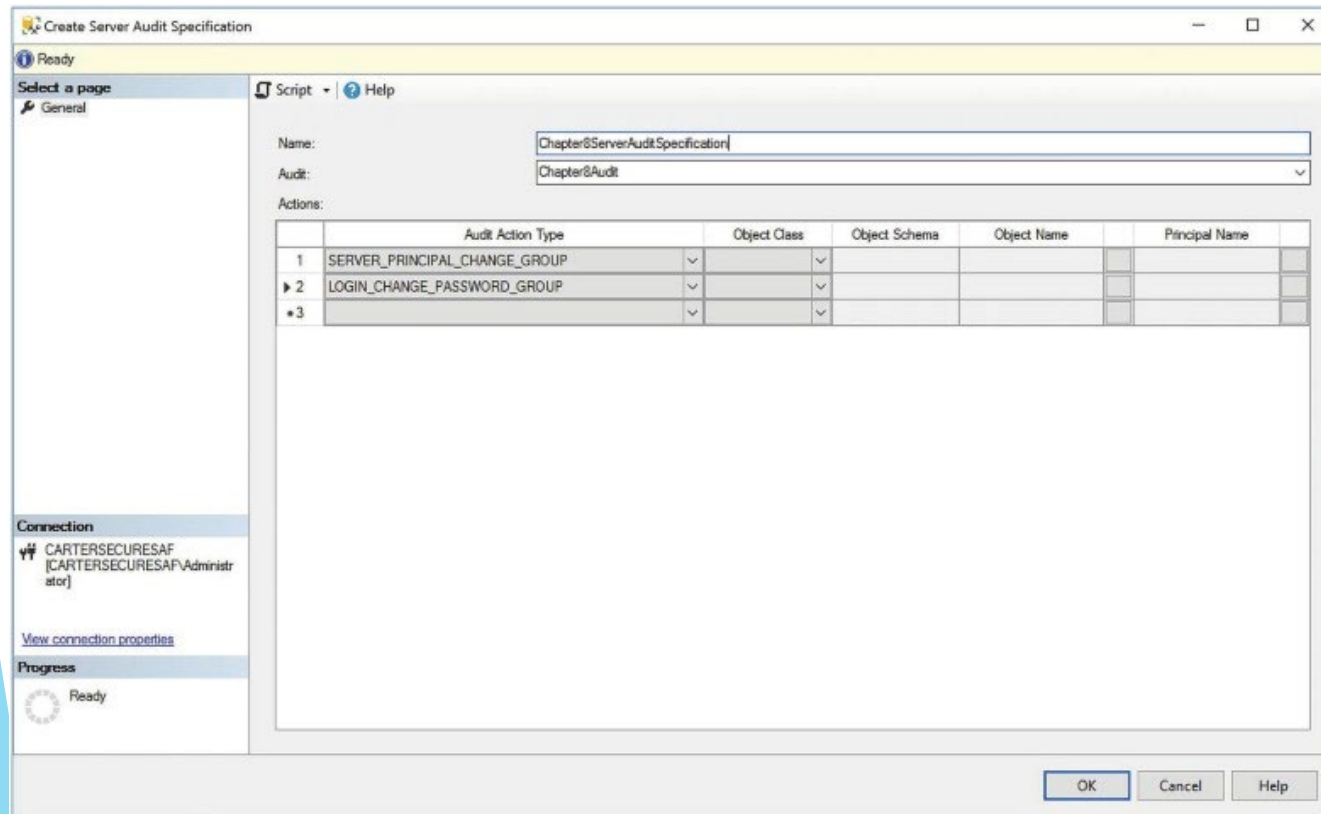


Figure 8-3 Create New Server Audit Specification dialog box





# Ensuring Reputability

- ▶ The query in Listing 8-5 will return a list of actions that will be audited by these two groups.
- ▶ If try to enable Audit at this point, it will fail as the OS is not configured to allow permissions to the Windows Security Log.
- ▶ Alter the audit policy to allow application-generated audit events using auditpol.exe. Run Listing 8-6 in the command prompt as Administrator.
- ▶ Then grant SQL Server service account the Generate Security Audits user rights assignment in the Local Security Policy Windows. Execute secpol.exe and enable as in Figure 8-4.
- ▶ Enable General Security Audits Users Rights Assignment to show as in Figure 8-5.
- ▶ Finally enable the Audit objects.



# Ensuring Reputability

```
SELECT
    name
    ,covering_parent_action_name
FROM sys.dm_audit_actions
WHERE covering_parent_action_name IN
('LOGIN_CHANGE_PASSWORD_GROUP','SERVER_PRINCIPAL_CHANGE_GROUP')
;
```

**Listing 8-5 List Audit Actions**

```
auditpol /set /subcategory:"application
generated" /success:enable /failure:enable
```

**Listing 8-6 Enable Application-Generated Audit Events**

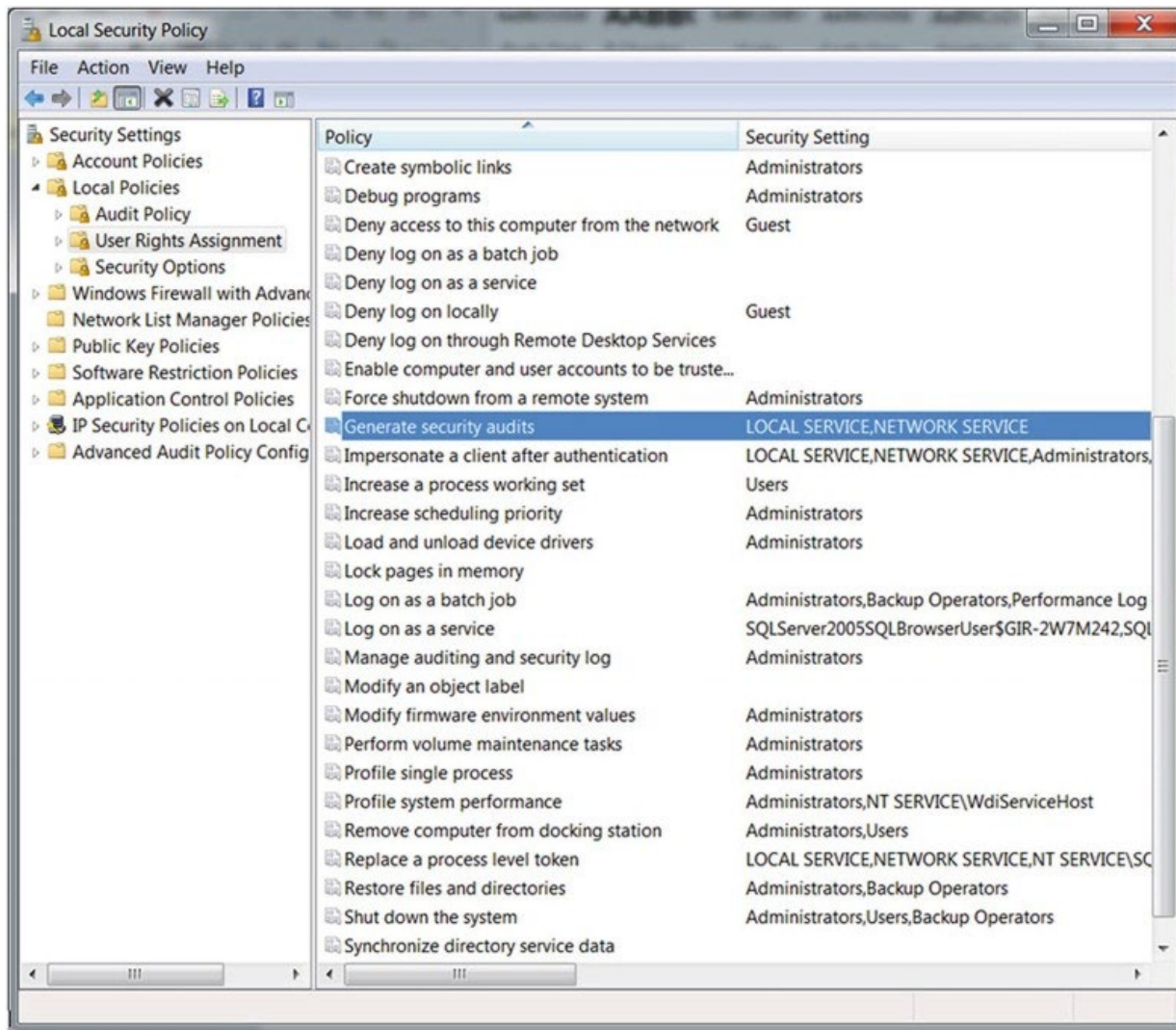


Figure 8-4 Local Security Policy console

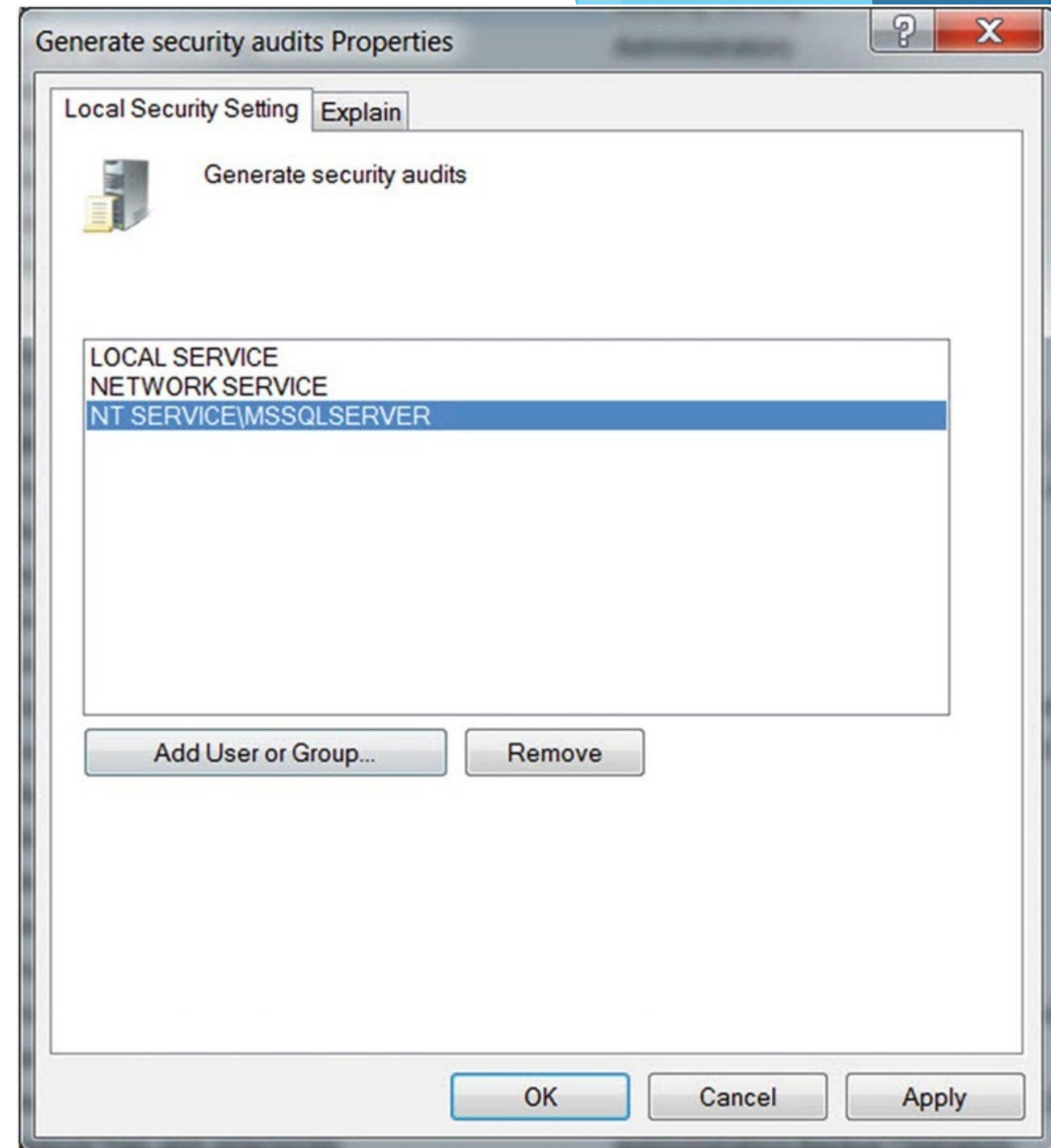


Figure 8-5 Generate Security Audits Properties dialog box

# Enforcing Constant Password Changes

- ▶ Can enforce password rotation using a routine on SQL Server
- ▶ Create an SQL Server Agent job - go to the SQL Server Agent in the Object Explorer and select new job. The dialog box will show as in Figure 8-6.

The screenshot shows the 'New Job' dialog box with the 'General' page selected. The left sidebar contains a 'Select a page' section with links to General, Steps, Schedules, Alerts, Notifications, and Targets. Below this are sections for 'Connection' and 'Progress'. The 'Connection' section shows 'Server: CARTERSECURESAF' and 'Connection: CARTERSECURESAF\Administrat' with a link to 'View connection properties'. The 'Progress' section shows a circular progress indicator and the word 'Ready'. The main area of the dialog has a title bar 'New Job' and a menu bar with 'Script' and 'Help'. It contains fields for 'Name' (ChangeSAPassword), 'Owner' (NT Service\MSSQLSERVER), 'Category' ([Uncategorized (Local)]), and 'Description' (a large empty text box). There is a checkbox labeled 'Enabled' which is checked. At the bottom right are 'OK' and 'Cancel' buttons.

Figure 8-6 New Job dialog box –general page



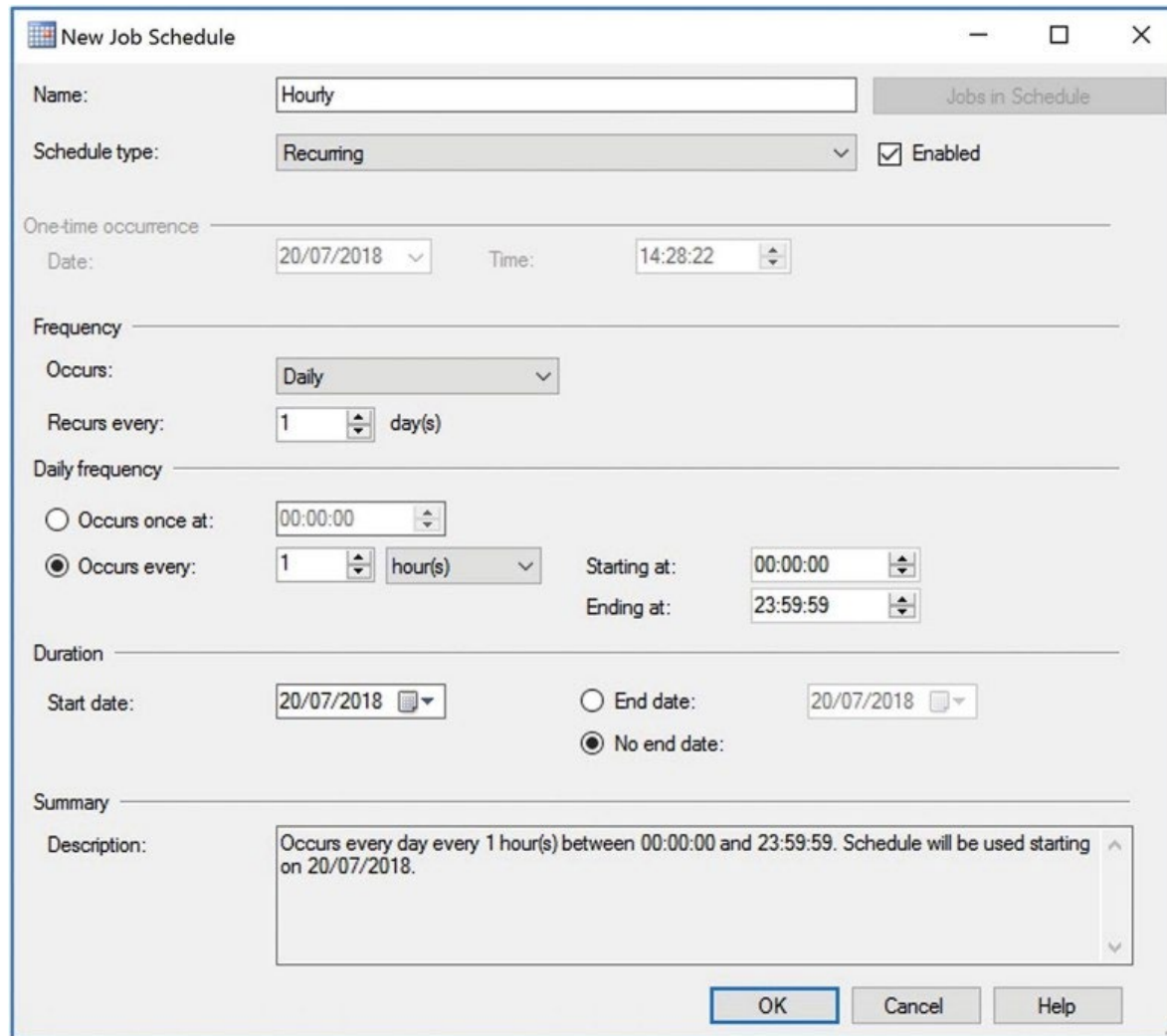
# Enforcing Constant Password Changes

- ▶ Use the New button to invoke the New Job Step dialog box as in Figure 8-7. Specify the name and use the T-SQL script. Enter Listing 8-7's script.
- ▶ Upon exiting the New Job Step dialog box, go to the Schedule page and use the New button to show the New Schedule dialog box as in Figure 8-8.
- ▶ Specify the name of the schedule and the frequency. Once done, this job will be created.

```
DECLARE @Password NVARCHAR(16) ;  
DECLARE @SQL NVARCHAR(MAX) ;  
SELECT @Password = CAST(CHECKSUM(GETDATE()) AS NVARCHAR(16)) ;  
SET @SQL = 'ALTER LOGIN sa WITH PASSWORD = ''' + @password + '''' ;  
EXEC (@SQL) ;
```

Listing 8-7 Change sa Account Password to a Dynamic Value

# Enforcing Constant Password Changes



The image shows a 'New Job Schedule' dialog box with the following fields and options:

- Name:** Hourly
- Schedule type:** Recurring (dropdown menu)
- Enabled:** ☒ Enabled
- One-time occurrence:**
  - Date:** 20/07/2018 (dropdown menu)
  - Time:** 14:28:22 (spinners)
- Frequency:**
  - Occurs:** Daily (dropdown menu)
  - Recurs every:** 1 (spinners) day(s)
- Daily frequency:**
  - ☐ Occurs once at: 00:00:00 (spinners)
  - ☒ Occurs every: 1 (spinners) hour(s) (dropdown menu)
  - Starting at:** 00:00:00 (spinners)
  - Ending at:** 23:59:59 (spinners)
- Duration:**
  - Start date:** 20/07/2018 (calendar icon)
  - ☐ End date: 20/07/2018 (calendar icon)
  - ☒ No end date:
- Summary:**
  - Description:** Occurs every day every 1 hour(s) between 00:00:00 and 23:59:59. Schedule will be used starting on 20/07/2018.

Buttons: OK, Cancel, Help

Figure 8-8 New Schedule dialog box



# Protecting User Accounts

- ▶ To protect Logins and User accounts, ensure that passwords are complex, changed frequently and not in common words list.
- ▶ A brute force attack is where an attacker attempts every possible combination of words and numbers.
- ▶ For a word list attack, the hacker uses a password dictionary using a list of common passwords.
- ▶ Best way to ensure passwords are complex and frequently changed is to enforce the domain-level password policy.
- ▶ Also perform spot checks to ensure Logins are not using common passwords.



# Protecting User Accounts

- Common passwords are first passwords that are used in a word list attack. The full list of common passwords can be found at

<http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time>

NO	Top 1-100	Top 101-200	Top 201-300	Top 301-400	Top 401-500
1	123456	porsche	firebird	prince	rosebud
2	password	guitar	butter	beach	jaguar
3	12345678	chelsea	united	amateur	great
4	1234	black	turtle	7777777	cool
5	pussy	diamond	steelers	muffin	cooper
6	12345	nascar	tiffany	redsox	1313
7	dragon	jackson	zxcvbn	star	scorpio
8	qwerty	cameron	tomcat	testing	mountain
9	696969	654321	golf	shannon	madison
10	mustang	computer	bond007	murphy	987654
11	letmein	amanda	bear	frank	brazil
12	baseball	wizard	tiger	hannah	lauren
13	master	xxxxxxxx	doctor	dave	japan
14	michael	money	gateway	eagle1	naked
15	football	phoenix	gators	11111	squirt
16	shadow	mickey	angel	mother	stars
17	monkey	bailey	junior	nathan	apple
18	abc123	knight	thx1138	raiders	alexis
19	pass	iceman	porno	steve	aaaa
20	fuckme	tigers	badboy	forever	bonnie
21	6969	purple	debbie	angela	peaches
22	jordan	andrea	spider	viper	jasmine
23	harley	horny	melissa	ou812	kevin
24	ranger	dakota	booger	jake	matt
25	iwantu	aaaaaa	1212	lovers	qwertyui





# Protecting User Accounts

- ▶ To audit passwords, use the PWDCOMPARE() function. It returns 1 if the cleartext matches the password hash and 0 otherwise.
- ▶ Table 8-1 shows the acceptable parameters.
- ▶ Listing 8-8 shows how to use the function to check if any Logins use some of the listed common passwords.

Parameter	Description
<code>clear_text_password</code>	Specify the clear text version of a password that you wish to compare
<code>password_hash</code>	The hashed password value that you wish to audit
<code>version</code>	This parameter is obsolete and should not be used

Table 8-1 PWDCOMPARE Parameters



# Protecting User Accounts

## Listing 8-8 Audit Common Passwords

```
CREATE TABLE ##Passwords
(
  [Password]      NVARCHAR(128)
) ;
INSERT INTO ##Passwords
VALUES ('123456'),
('password'),
('12345678'),
('1234'),
('palssy'),
('12345'),
('dragon'),
('qwerty'),
('696969'),
('mustang'),
('letmein'),
('baseball'),
('master'),
('michael'),
('football'),
('shadow'),
('monkey'),
('abc123'),
('pass'),
('fuckme'),
('6969'),
('jordan'),
('harley'),
('ranger'),
('iwantu') ;
SELECT l.name,
       p.[password]
FROM sys.sql_logins l
CROSS JOIN ##Passwords p
WHERE PWDCOMPARE(p.Password,l.password_hash) = 1 ;
DROP TABLE ##Passwords ;
```



# Protecting Windows Accounts

- ▶ DBAs can also consider security for windows users that have highly privileged access to SQL Server Instances.
- ▶ Although not directly responsible, the DBA can make recommendations.
- ▶ There are many mechanisms for protecting privileged Windows accounts. These fall into two main categories:
  - 1) Secure storage and rotation.
  - 2) Two factor authentication.



# Summary

- ▶ Important to enforce password policies to reduce likelihood of successful attack.
- ▶ Sa Account is a particular target, so use Windows Authentication wherever possible.
- ▶ If not, then disable or rename the sa Account.
- ▶ If still not, then use auditing, enforce password checks and rotation.
- ▶ To mitigate against other Logins being victims of brute force or word list attacks, enforce a domain-level password policy.
- ▶ Use spot checks to check for Logins that have common passwords using the PWDCOMPARE() function.