

# Attacks on SQL Server



# Hijacking an Instance

Part 1

# Topics

- ▶ Hijacking an Instance
- ▶ Protecting Against Hijacking
  - ▶ Understanding Logon Triggers
  - ▶ Using Logon Triggers to Prevent Instance Hijacking
  - ▶ Understanding Server Agent
  - ▶ Putting it all Together
- ▶ Summary



# Hijacking an Instance

- ▶ Microsoft built a “**back-door**” for Windows administrator to regain admin access to “locked out” instances.
- ▶ Instances can be “locked out” if the only remaining DBA is no longer available before adding new administrators. (i.e., leaving the job, killed in accidents, forgets password)
- ▶ This method is **well documented and supported**, so if attacker gain local admin access, taking control of the database is easy.
- ▶ This method also introduces insider threat from the Windows admins.

# HOWTO

- ▶ To gain administrative control, open SQL Server Configuration Manager
- ▶ Navigate to SQL Server Services in the left-hand pane
- ▶ Select Properties from the context menu of the SQL Server service in the right-hand pane.
- ▶ Figure 11-1 shows the Properties window for the SQL Server service.
- ▶ Click on “Stop” button to stop the database service. This will also prompt the admin to stop the SQL Server Agent. Click “yes” in Figure 11-2.

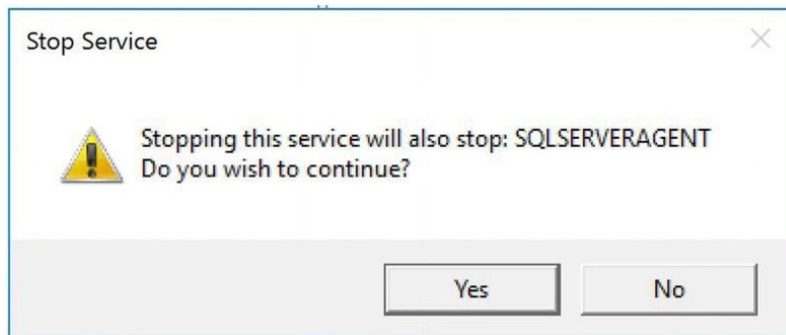


Figure 11-2 Stop SQL Server agent prompt

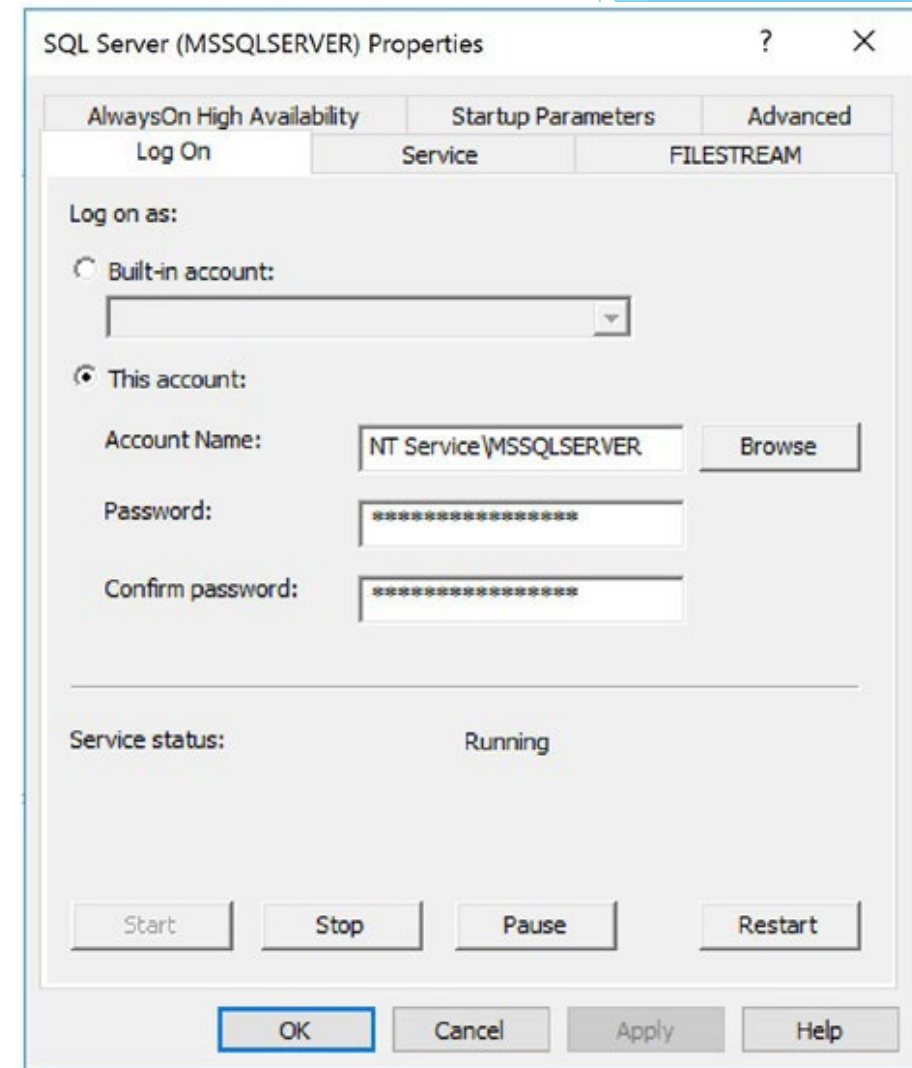


Figure 11-1 Service properties-log-on tab

# HOWTO

- ▶ Navigate to the Startup Parameters tab of the Properties dialog box, shown in Figure 11-3.
- ▶ Configure instance to start in single user mode by adding the '-m' option as a startup parameter.
- ▶ Navigate back to the Log On tab and use the "Start" button to start the database service in single user mode.

Note: Because the instance is in single user mode, do not start the SQL Server Agent service which will take the only available connection up. This will restrict logins from you.

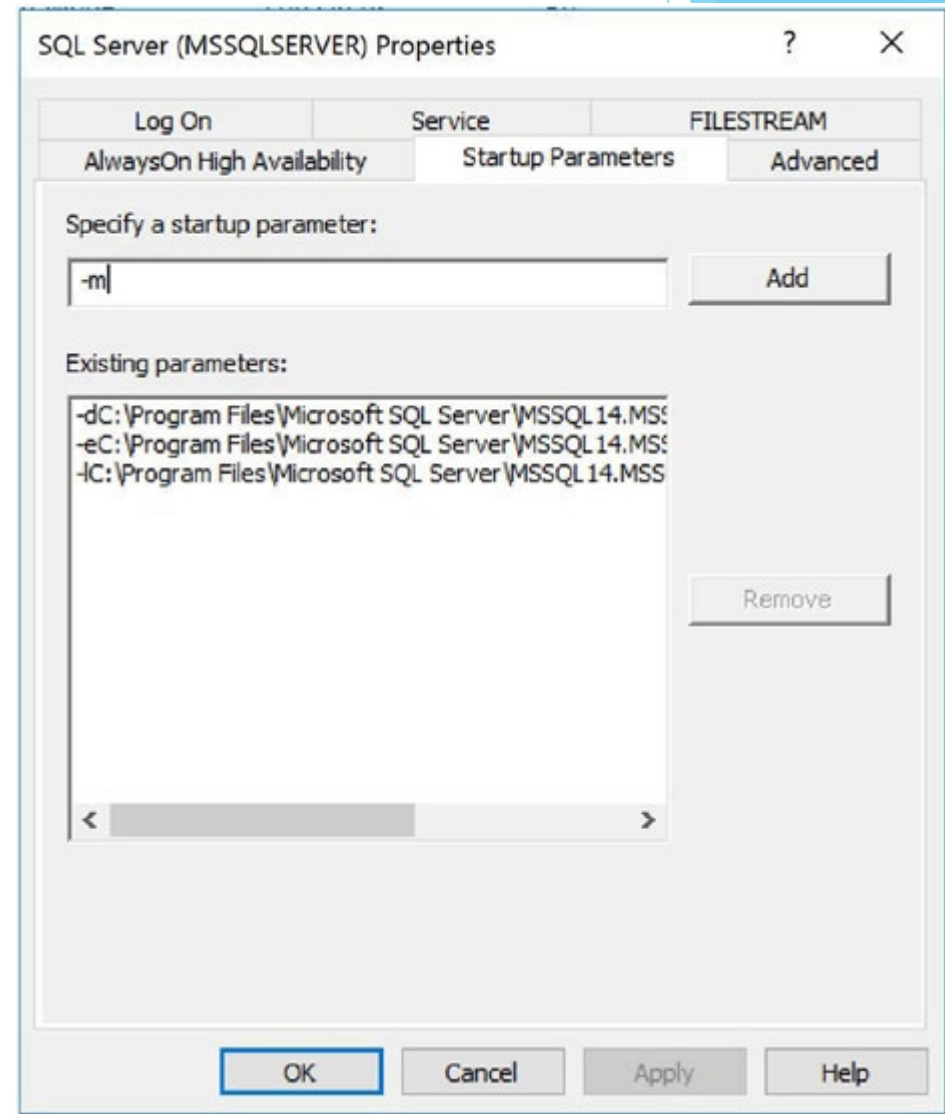


Figure 11-3 Service properties-startup parameters tab

# Startup Parameter Reference

Startup Parameter	Description
-d	Specifies the path to the master database .mdf file
-e	Specifies the path to the error log file
-E	Increases the number of extents allocated to each data file during the round-robin process
-l	Specifies the path to the master database log file. Usually .ldf
-c	Prevents the call to Service Control Manager when the instance is started from the command line, rather than as a service
-f	Starts the instance in minimal configuration mode. This allows DBAs to troubleshoot an instance that cannot start due to a misconfiguration.
-g	Specifies the amount of memory, in megabytes, that should be allocated to the instance but not allocated to the buffer pool
-m	Starts the instance in single-user mode
-mClientApplicationName	Starts the instance in what is essentially a “single application mode.” Only connections from the specified application are permitted.
-n	Prevents events generated by the instance being written to the Windows Application Log
-s	Allows the instance to start using the binaries of a different instance
-T	Specifies a global trace flag, which should be turned on when the instance starts. Multiple -T parameters can be specified.
-x	Disables some performance monitoring features. Specifically: <ul style="list-style-type: none"><li>• SQL Server-related performance monitor counters</li><li>• CPU time statistics</li><li>• Buffer cache hit ratio statistics</li><li>• Information collection for DBCC SQLPERF</li><li>• Information collection for a range of dynamic management objects</li><li>• A range of extended events event points</li></ul>

Table 11-1 Database Engine Startup Parameters

# HOWTO

- ▶ Launch the SQL Server Management Studio with administrator privileges. See Figure 11-4.
- ▶ Do not connect to Object Explorer. Select “Cancel” and in the Connect to Server dialog box.
- ▶ Setup a new query using the “New Query” button. Insert the query shown on Listing 11-1.
- ▶ Run the script and put the instance back into normal operating mode by removing the ‘-m’ startup parameter.
- ▶ Login with the credentials inserted in the query.

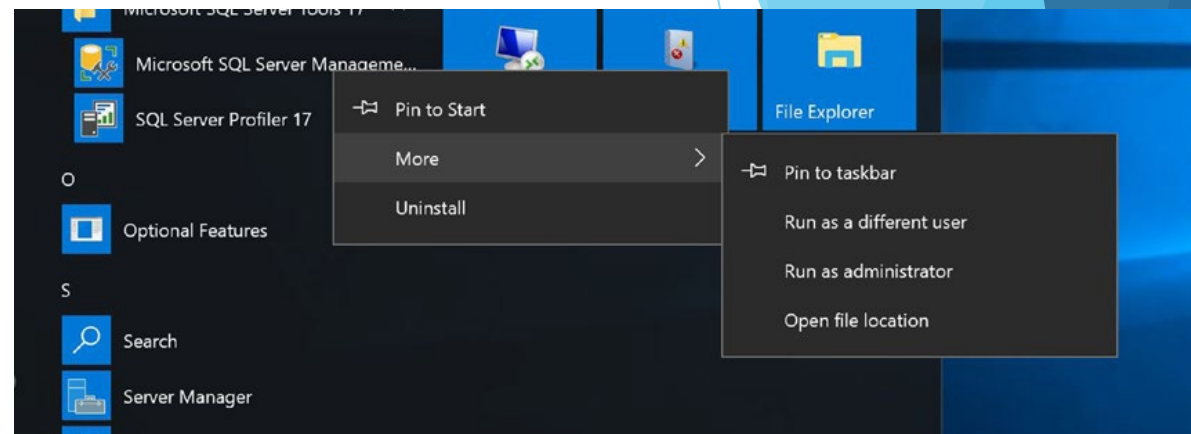


Figure 11-4 Run SSMS As Administrator

```
CREATE LOGIN Attacker
WITH PASSWORD =
'Pa$$w0rd12345678' ;
GO
ALTER SERVER ROLE sysadmin
ADD MEMBER Attacker ;
GO
```

Listing 11-1 Create a Login With Administrative Permissions





# Protecting Against Hijacking

- ▶ It's difficult, but not impossible to protect against hijacking.
- ▶ Logon triggers can be used as defence against instance hijacking.



# Understanding Logon Triggers

- ▶ Triggers are similar to stored procedures, they are both code routines.
- ▶ Triggers can be automatically “fired”, which runs the routines. Stored procedures must be run manually.
- ▶ **3 types of trigger is supported:**
  - ▶ DML Trigger (Fire upon INSERT, UPDATE or DELETE events).
  - ▶ DDL Trigger (Fire in response to a DDL event such as CREATE, ALTER, DROP or UPDATE STATISTICS).
  - ▶ Logon Trigger (Fire when a successful logon event occurs, corresponds to AUDIT\_LOGON extended event).



# Understanding Logon Triggers (cont'd)

- ▶ Logon Triggers fire synchronously, as opposed to asynchronously.
  - ▶ This means that if the triggers fail, the logon fails, too.
- ▶ Logon Triggers can be used many purposes. Examples are:
  - ▶ Auditing logons (write to table each time a logon occurs).
  - ▶ Limit number of concurrent connections to an instance.
  - ▶ Make sure no users logon during maintenance hours.
- ▶ Logon Triggers provide useful system functions to determine if logon is allowed.
  - ▶ `ORIGINAL_NAME()` → tells you the name of the SQL Login attempting to authenticate.
  - ▶ `IS_SRVROLEMEMBER()` → determine login's permission.



# Understanding Logon Triggers (cont'd)

Component	Description
ON	For Logon Triggers, the value supplied will always be <code>ALL SERVER</code> . DDL triggers may use <code>ALL SERVER</code> or <code>DATABASE</code> .
WITH	Allows for Logon Trigger options to be passed. These options are detailed in Table <a href="#">11-3</a> .
FOR   AFTER	<b>FOR</b> and <b>ALL</b> are interchangeable, and imply that the logon event will occur before the trigger is fired. The code within the trigger will occur within the same transaction, however, so <b>ROLLBACK</b> can be used to prevent the logon from occurring. DML triggers can use <b>INSTEAD OF</b> , which means that the original statement is never fired, but this option is not applicable to Logon Triggers or DDL triggers.

Table 11-2 CREATE TRIGGER Syntax Components



# Understanding Logon Triggers (cont'd)

Option	Description
ENCRYPTION	Obfuscates the definition of the Logon Trigger.
EXECUTE AS	Causes the code within the Logon Trigger to be executed under the security context of a different user. Please see Chapter <a href="#">13</a> for a broader discussion of the EXECUTE AS clause.

Table 11-3 Logon Trigger WITH Options



# Using Logon Triggers to Prevent Instance Hijacking

- ▶ Logon Trigger can be used to prevent users from logging in when instance is in single user mode, unless they are instance administrators.
- ▶ This can be done using the `sys.dm_server_registry` dynamic management view to check if instance is in single-user mode.
- ▶ Use `IS_SRVROLEMEMBER()` to then determine if the login is part of the sysadmin server role.

Column	Description
<code>registry_key</code>	The name of the registry key
<code>value_name</code>	The name of the registry key value
<code>value_data</code>	The value of the registry key

Table 11-4 Columns Returned by `sys.dm_server_registry`

# Using Logon Triggers to Prevent Instance Hijacking (cont'd)

- ▶ Listing 11-2 checks if the instance is in single user mode.
- ▶ The IS\_SRVROLEMEMBER() function accepts the parameters detailed in Table 11-5.
- ▶ Listing 11-3 checks if the login is part of the sysadmin role.

```
SELECT COUNT(*)  
FROM sys.dm_server_registry  
WHERE value_name LIKE 'SQLArg%' AND value_data = '-m' ;
```

Listing 11-2 Use the sys.dm\_server\_registry DMV to Check for Single-User Mode

Parameter	Description
Role	The name of the server role for which you wish to check membership
Login	The name of the login for which you wish to check membership

Table 11-5 Parameters Accepted by IS\_SRVROLEMEMBER()



# Using Logon Triggers to Prevent Instance Hijacking (cont'd)

```
CREATE TRIGGER PreventHijack
ON ALL SERVER WITH EXECUTE AS 'sa'
FOR LOGON
AS
BEGIN
    DECLARE @SingleUser INT ;
    SET @SingleUser =
    (
        SELECT COUNT(*)
        FROM sys.dm_server_registry
        WHERE value_name LIKE 'SQLArg%' AND value_data = '-m'
    ) ;
    IF @SingleUser <> 0
    BEGIN
        IF IS_SRVROLEMEMBER('sysadmin', ORIGINAL_LOGIN()) <> 1
        BEGIN
            ROLLBACK ;
        END
    END
END
```

Listing 11-3 Create a Logon Trigger





# PreventHijack1(Logon Triggers) - Doesn't Work!

- ▶ When SQL Server is in **single user mode**, when an administrator of the local server attempts login, SQL Server will **add them to the sysadmin server role**.
- ▶ The above action occurs BEFORE the logon trigger fires.
- ▶ This means IS\_SRVROLEMEMBER() will always resolve to true.
- ▶ What if we remove the IS\_SRVROLEMEMBER() check?
  - ▶ This works! BUT if the database needed to be rebuilt (which could only happen in single user mode), then logon is no longer possible.



# PreventHijack2: Using a Stale Cache

- ▶ The stale cache stores members of the sysadmin server roles.
- ▶ No a blanket ban on single user login anymore!
- ▶ Creating a sysadmin role stale cache as Listing 11-4:

```
USE MASTER
GO
CREATE TABLE dbo.SysadminMembers
(
    ID INT IDENTITY PRIMARY KEY NOT NULL,
    LoginName SYSNAME NOT NULL
);
```

Listing 11-4 Create the Administrators Stale Cache



# PreventHijack2: Using a Stale Cache (cont'd)

- ▶ Identify logins that should be in the SysadminMembers table using the sys.server\_role\_members and sys.server\_principals catalog views. The columns returned are described by Table 11-6:

Column	Description
role_principal_id	The internal ID of the server role
member_principal_id	The internal ID of the login

Table 11-6 Columns Returned by sys.server\_role\_members



# PreventHijack2: Using a Stale Cache (cont'd)

Column	Description
name	The name of the security principal
principal_id	The internal ID of the security principal
sid	The security identifier of the principal
type	A single character code, denoting the type of the principal. Possible values are: <ul style="list-style-type: none"><li>• S - Indicates an SQL login</li><li>• U - Indicates a Windows login</li><li>• G - Indicates a Windows group</li><li>• R - Indicates a Server role</li><li>• C - Indicates a login that is mapped to a certificate</li><li>• K - Indicates a login that is mapped to an Asymmetric key</li></ul>
type_desc	A textual description of the principal's type
is_disabled	Indicates if the principal's disabled state. <ul style="list-style-type: none"><li>• 0 - Indicates that the principal is enabled</li><li>• 1 - Indicates that the principal is disabled</li></ul>
create_date	The date and time that the principal was created
modify_date	The date and time that the principal was last modified
default_database_name	The name of the principal's default database
default_language_name	The default language configured for the principal
credential_id	If the principal has a credential associated with it, then this column returns the ID of the credential. A credential is a mechanism used by SQL Server to allow a login to interact with the operating system.
owning_principal_id	If the principal is a server role, the <code>owning_principal_id</code> column returns the ID of the principal who owns the server role.
is_fixed_role	If the principal's type is R, indicates if the server role is fixed, or user-defined. For principal's that are not server roles, this column always returns 0. For server roles, possible values are: <ul style="list-style-type: none"><li>• 0 - Indicates a user-defined server role</li><li>• 1 - Indicates a fixed server role</li></ul>

Table 11-7 Columns Returned by sys.server\_principals

# PreventHijack2: Using a Stale Cache (cont'd)

- ▶ The sys.server\_role\_members and sys.server\_principals catalog views can be joined together, to retrieve a lists of server roles, with all associated members.
- ▶ Results can then be filtered on the sysadmin server role.
- ▶ Listing 11-5 demonstrates how to do this and then merge the results into the SysadminMembers table.
- ▶ The MERGE statement will delete any members from the table who are no longer associated with the sysadmin server role and insert any new members.schedule the population to occur on a daily basis.
- ▶ The scheduling tool within SQL Server is SQL Server Agent.

```
USE master ;
GO
MERGE INTO dbo.sysadminMembers AS Target
USING (
    SELECT
        Roles.name AS RoleName
        , Members.name AS MemberName
    FROM sys.server_role_members RoleMembers
    INNER JOIN sys.server_principals Roles
        ON RoleMembers.role_principal_id =
Roles.principal_id
    INNER JOIN sys.server_principals AS Member
        ON RoleMembers.member_principal_id =
Members.principal_id
    WHERE Roles.name = 'sysadmin'
) AS Source
ON (Source.MemberName = Target.LoginName)
WHEN NOT MATCHED BY TARGET THEN
    INSERT (LoginName)
    VALUES (MemberName)
WHEN NOT MATCHED BY SOURCE THEN
    DELETE ;
```

Listing 11-5 Update SysadminMembers Table



# PreventHijack2: Understanding Server Agent

- ▶ The **stale cache should be populated periodically** to prevent it from being “too stale” and risk being unable to access the instance in single user mode.
- ▶ Using **SQL Server Agent functionality** to schedule cache updates.
- ▶ Server Agent is implemented using:
  - ▶ Jobs
  - ▶ Schedules
  - ▶ Alerts
  - ▶ Operators



# PreventHijack2: Understanding Server Agent

- ▶ Server Agent Schedule
  - ▶ One time → allows specification of date and time.
  - ▶ Start automatically when server agent starts → useful if there is a set of tasks that should run when the instance starts, assuming that the Server Agent service is configured to start automatically.
  - ▶ Start when CPU becomes idle → useful if resources intensive jobs are present but user activity needs to be unhindered.
  - ▶ Recurring → Allows definition of a complex schedule, with start and end dates and can reoccur daily, weekly or monthly.
- ▶ Individual schedules can be created for each job, or a schedule can be defined to be used to trigger multiple jobs.
- ▶ A job can have zero or more schedules.



# PreventHijack2: Understanding Server Agent

- ▶ **Server Agent Operators**
  - ▶ An operator is an individual or team that is configured to receive a notification of job status.
  - ▶ In the event that an alert is triggered, operators can be configured to be notified through e-mail, NET SEND or pager.
- ▶ For operators to be notified through e-mail, the database mail must also be configured, specifying the address and port of the SMTP Relay server that will deliver the messages.
- ▶ While NET SEND and pager notification still works, they are deprecated, and should be avoided.





# PreventHijack2: Understanding Server Agent

- ▶ Server Agent Jobs comprises of a series of actions that should be performed.
- ▶ Each action is known as a job step; each job step can be configured to perform an action with the following categories:
  - ▶ ActiveX scripts
  - ▶ Operating system commands
  - ▶ PowerShell scripts
  - ▶ Replication Distributor tasks
  - ▶ Replication Merge Agent tasks
  - ▶ Replication Queue Reader Agent tasks
  - ▶ Replication Snapshot Agent tasks
  - ▶ Replication Transaction Log Reader tasks
  - ▶ Analysis Services commands
  - ▶ Analysis Services queries
  - ▶ SSIS packages
  - ▶ T-SQL commands
- ▶ Each Job step can be configured to run under context of the service account using the Server Agent service OR under a proxy account, which runs under the context of a credential.
- ▶ Each Step can also be configured to retry a specific number of times with an interval between each retry.
- ▶ On Success and On Failure actions can be configured individually for each job step. This allows DBAs to implement decision-based logic and error handling.

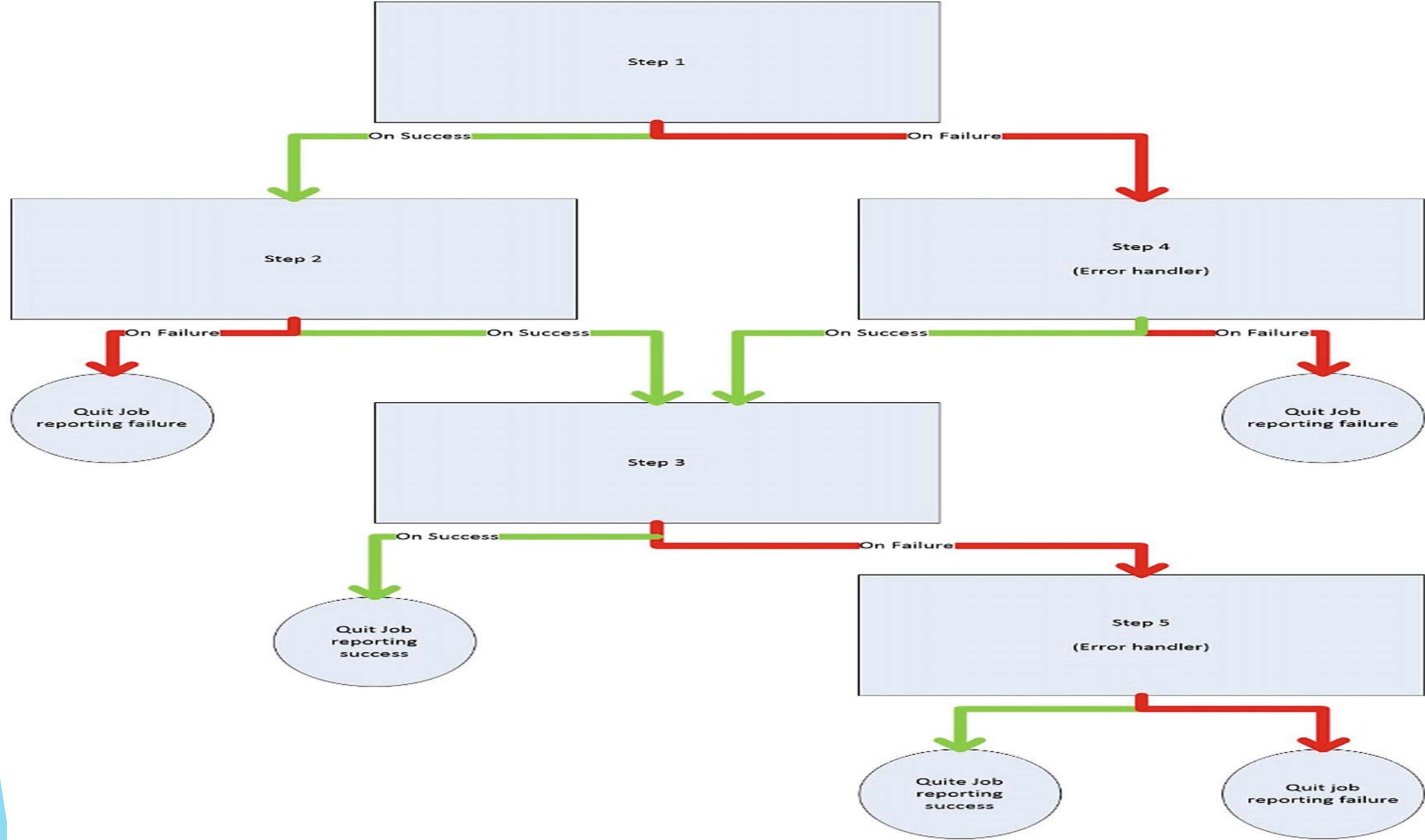


Figure 11-5 Decision Tree logic



# PreventHijack2: Understanding Server Agent

- ▶ Server Agent Alerts respond to the following categories of activity:
  - ▶ SQL Server events.
  - ▶ SQL Server performance conditions.
  - ▶ WMI events.
- ▶ Alert can be configured to respond to error message when created against SQL Server events category.
- ▶ Alerts can also be filtered so that they will only fire if the error or warning contains specific keywords.
- ▶ When created against SQL Server performance conditions, alerts are configured to trigger if a counter falls below, rises above or equals to a specified value. (Useful for monitoring performance)
- ▶ Example of performance objects available to be monitored:
  - ▶ Memory Manager
  - ▶ HTTP Storage
  - ▶ Transactions
  - ▶ SQL errors



# PreventHijack2: Understanding Server Agent

- ▶ Access to Server Agent is controlled via Database Roles
- ▶ Job steps can run under the context of either the Server Agent service account, or using a separate proxy accounts that map to credentials.
- ▶ The permissions provided by the roles are detailed in Table [11-8](#).
- ▶ Server Agent Database Roles:
  - ▶ SQLAgentUserRole
  - ▶ SQLAgentReaderRole
  - ▶ SQLAgentOperatorRole

Permission	SQLAgentUserRole	SQLAgentReaderRole	SQLAgentOperatorRole
CREATE/ALTER/DROP Operator	No	No	No
CREATE/ALTER/DROP Local Job	Yes (Owned only)	Yes (Owned only)	Yes (Owned only)
CREATE/ALTER/DROP multiserver Job	No	No	No
CREATE/ALTER/DROP Schedule	Yes (Owned only)	Yes (Owned only)	Yes (Owned only)
CREATE/ALTER/DROP Proxy	No	No	No
CREATE/ALTER/DROP Alerts	No	No	No
View list of Operators	Yes	Yes	Yes
View list of local Jobs	Yes	Yes	Yes
View list of multiserver Jobs	No	Yes	Yes
View list of Schedules	Yes (Owned only)	Yes	Yes
View list of Proxies	Yes	Yes	Yes
View list of Alerts	No	No	No
Enable/disable Operators	No	No	No
Enable/disable local Jobs	Yes (Owned only)	Yes (Owned only)	Yes
Enable/disable multiserver Jobs	No	No	No
Enable/disable Schedules	Yes (Owned only)	Yes (Owned only)	Yes
Enable/disable Alerts	No	No	No
View Operator properties	No	No	Yes
View local Job properties	Yes (Owned only)	Yes	Yes
View multiserver Job properties	No	Yes	Yes
View Schedule properties	Yes (Owned only)	Yes	Yes
View Proxy properties	No	No	Yes
View Alert properties	No	No	Yes
Edit Operator properties	No	No	No
Edit local Job properties	No	Yes (Owned only)	Yes (Owned only)
Edit multiserver job properties	No	No	No
Edit Schedule properties	No	Yes (Owned only)	Yes (Owned only)
Edit Proxy properties	No	No	No
Edit Alert properties	No	No	No
Start/stop local Jobs	Yes (Owned only)	Yes (Owned only)	Yes
Start/stop multiserver Jobs	No	No	No
View local Job history	Yes (Owned only)	Yes	Yes
View multiserver Job history	No	Yes	Yes
Delete local Job history	No	No	Yes
Delete multiserver Job history	No	No	No
Attach/detach Schedules	Yes (Owned only)	Yes (Owned only)	Yes (Owned only)

**Table 11-8**  
**Server Agent Permissions Matrix**



# PreventHijack2: Understanding Server Agent

- ▶ By default, all job steps will run under the context of the Server Agent service account.
- ▶ Large number of permissions need to be granted to the instance and objects within the Operating System, which poses a security risk.
- ▶ To mitigate this risk and follow the principle of least privilege, proxy accounts should be used.
- ▶ Proxy accounts are mapped to Credentials within the instance level.
- ▶ Proxy accounts can be configured to run only a subset of step types. Example:
  - ▶ Configure one Proxy to run OS commands
  - ▶ Configure another Proxy to run PowerShell scripts
- ▶ This minimizes the permissions that each Proxy requires.
- ▶ For job steps with the Transact-SQL script, it is not possible to select a Proxy account.
- ▶ The “Run as user” option is used, which allow the selection of a database user to use the security context to run the script. (Uses the EXECUTE AS functionality).

# PreventHijack2: Understanding Server Agent

- ▶ To create a SQL Server Agent Job to run daily and synchronize the SysadminMembers table (stale cache), first launch the SQL Server Agent in Object Explorer.
- ▶ Select New Job from the context menu of Jobs. Figure 11-6 should appear.
- ▶ Give the job a meaningful name “SynchronizeSysAdminsMemberTable” and set the owner to “sa”.
- ▶ Ensure the Enabled check box is ticked.
- ▶ On the Steps page of the New Job dialog box, use the “New” button to invoke the New Job Step dialog box as Figure 11-7.

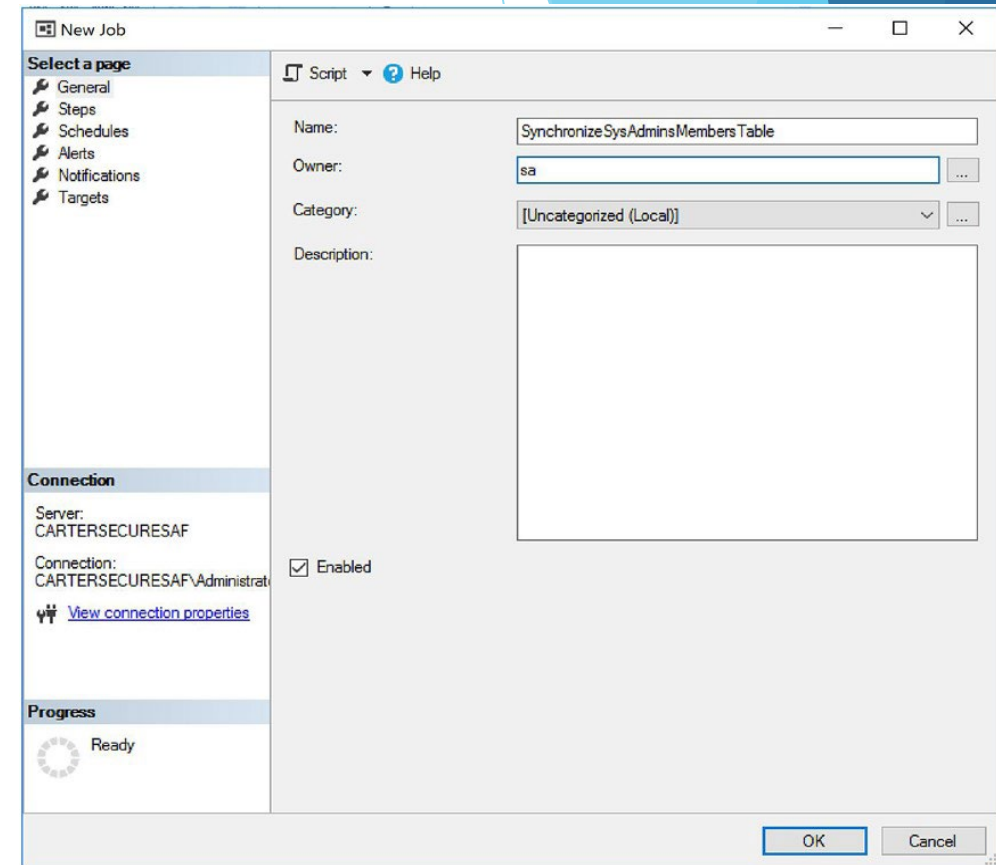


Figure 11-6 New Job dialog box-general page

# PreventHijack2: Understanding Server Agent

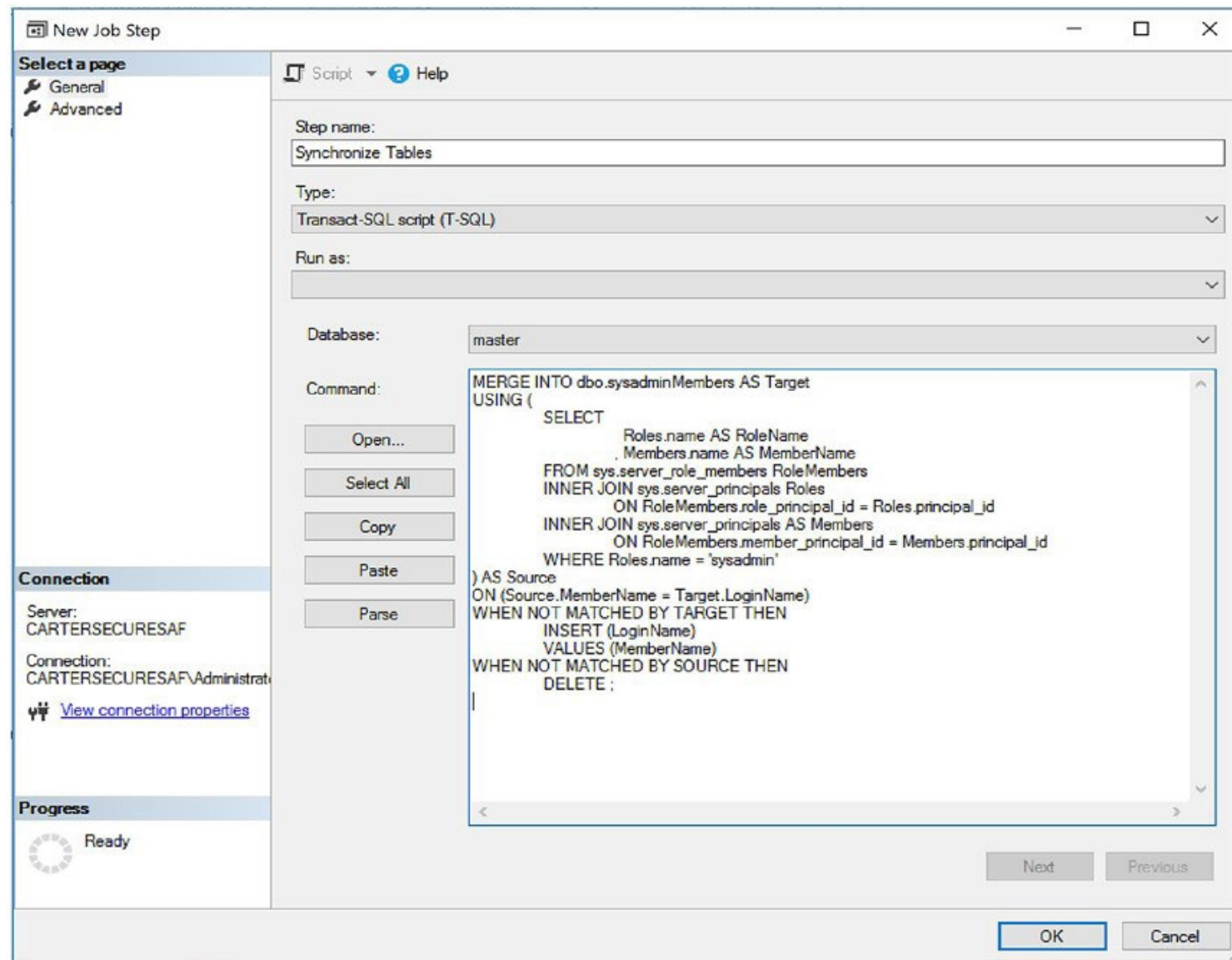


Figure 11-7 New Job Step dialog box-general page



# PreventHijack2: Understanding Server Agent

- ▶ The Job is given a name and specified as “T-SQL”.
- ▶ The step runs a T-SQL script.
- ▶ Select the master database in the Database drop-down.
- ▶ The command will synchronize the table.
- ▶ Click on “OK” button to return to the New Job dialog box’s step page in Figure 11-8
- ▶ On the schedules page of the New Job dialog box, use the “New” button to invoke the New Job Schedule dialog box as in Figure 11-9.

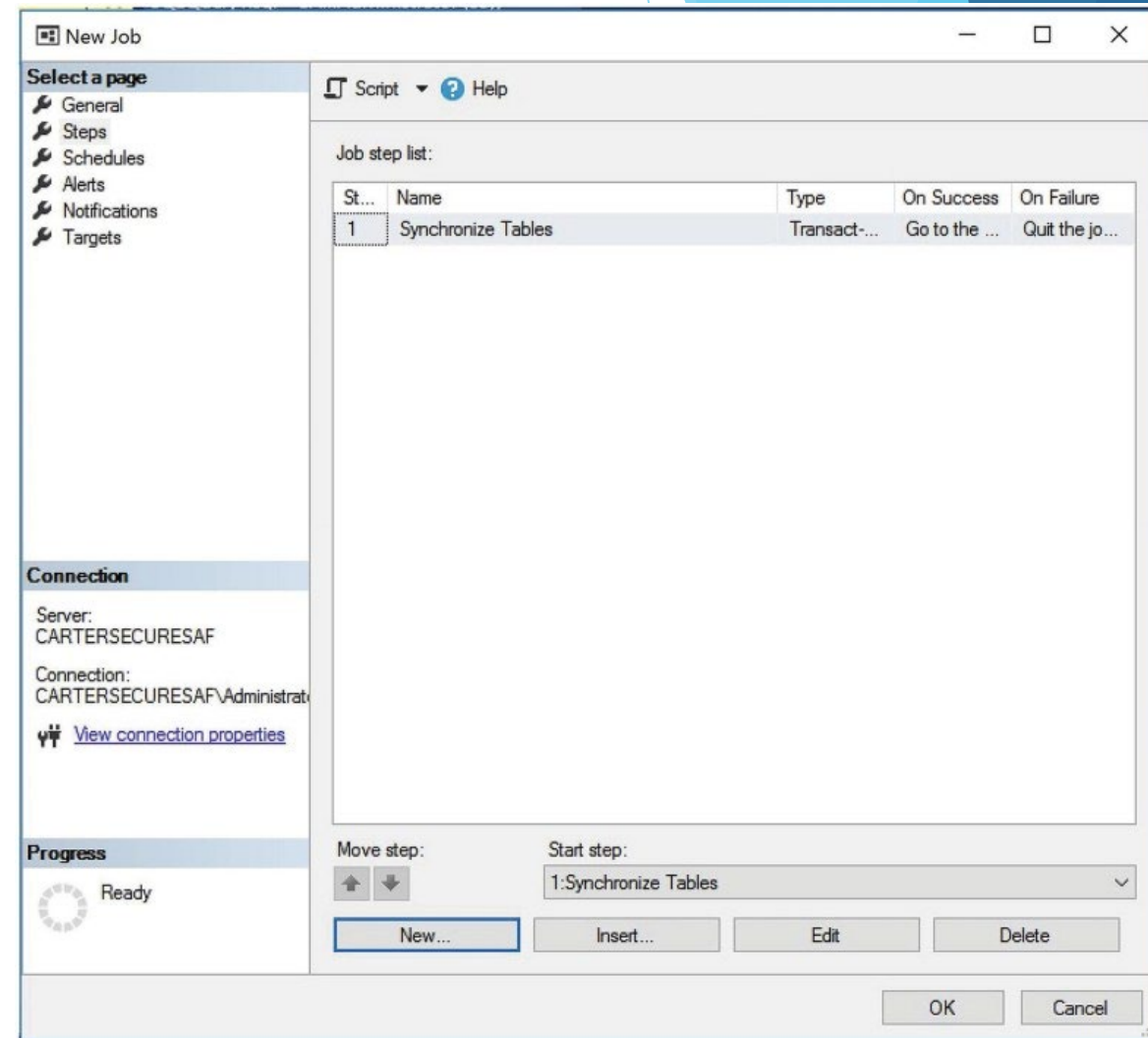


Figure 11-8 New Job dialog box-steps page

# PreventHijack2: Understanding Server Agent

- ▶ On the schedules page of the New Job dialog box, use the “New” button to invoke the New Job Schedule dialog box as in Figure 11-9.
- ▶ Select “Daily” frequency.
- ▶ The schedule is now configured to run every day.
- ▶ Press “OK” button to return.

The screenshot shows the 'New Job Schedule' dialog box with the following configuration:

- Name:** SynchronizeSysAdminsMembersSchedule
- Schedule type:** Recurring
- Enabled:** ☒
- One-time occurrence:** Date: 27/08/2018, Time: 17:09:55
- Frequency:** Occurs: Daily, Recurs every: 1 day(s)
- Daily frequency:** ☒ Occurs once at: 00:00:00, ☐ Occurs every: 1 hour(s)
- Starting at:** 00:00:00, **Ending at:** 23:59:59
- Duration:** Start date: 27/08/2018, ☐ End date: 27/08/2018, ☒ No end date
- Summary:** Description: Occurs every day at 00:00:00. Schedule will be used starting on 27/08/2018.
- Buttons:** OK, Cancel, Help

Figure 11-9 New Job Schedule dialog box



# PreventHijack2: Understanding Server Agent

## ► Creating the Final Logon Trigger:

```
CREATE TRIGGER PreventHijack2
ON ALL SERVER WITH EXECUTE AS 'sa'
FOR LOGON
AS
BEGIN
    DECLARE @SingleUser INT ;

    SET @SingleUser =
    (
        SELECT COUNT(*) FROM sys.dm_server_registry
        WHERE value_data = '-m'
    );

    IF @SingleUser <> 0
    BEGIN
        IF (
            SELECT COUNT(*)
            FROM dbo.sysadminmembers
            WHERE LoginName = ORIGINAL_LOGIN()
        ) = 0
        BEGIN
            ROLLBACK ;
        END
    END
END
```

Listing 11-6 Create the Final Logon Trigger



# Summary

- ▶ SQL Server instance is vulnerable to “Instance hijacking” due to the “back-door” feature by Microsoft.
- ▶ An attacker with system admin privilege can logon into the instance as a SQL user with sysadmin role, or even as the sysadmin itself.
- ▶ To prevent this, Logon triggers can be used.
- ▶ A naïve logon trigger will not work, because logging in as OS administrator privileges causes IS\_SRVROLEMEMBER() to return TRUE.
- ▶ If no check is done, logon trigger may disable single user logon, which hinders database rebuilt/maintenance work.
- ▶ The full solution is to use a stale-cache, which is updated by a SQL Server agent.
- ▶ The stale-cache is updated periodically by the SQL server agent.
- ▶ The Logon trigger reads from the stale-cache instead of using the IS\_SRVROLEMEMBER() function.
- ▶ This closes the back-door, which prevents “instance hijacking” by elevated users.



# Database Backup Theft

Part 2

# Topics

- ▶ Overview of Backups
  - ▶ Recovery Modes
  - ▶ Backup Types
  - ▶ Backup Media
- ▶ Securing Backup Media
  - ▶ Physical Security
  - ▶ Encrypting Backups
- ▶ Attempting to Steal an Encrypted Backup
- ▶ Administrative Consideration for Encrypted Backups
- ▶ Summary



# Overview of Backups



# Recovery Modes

- ▶ Database can be configured in one of 3 recovery modes:
  - ▶ SIMPLE
  - ▶ FULL
  - ▶ BULK LOGGED





# Recovery Modes (cont'd)

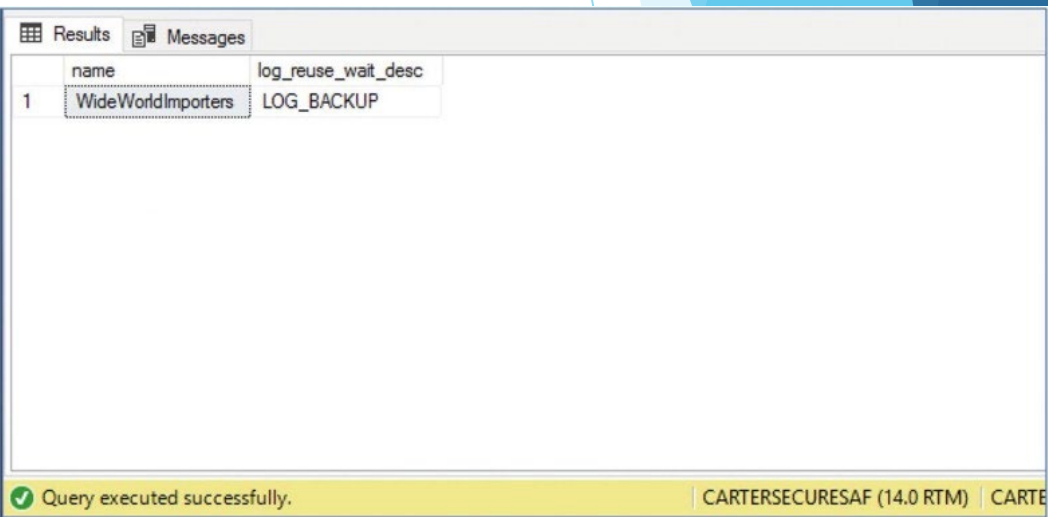
- ▶ SIMPLE Recovery Mode (Appropriate for Data Warehouse Style application)
  - ▶ Transaction logs contain transactions that are no longer required will be truncated after each checkpoint operation.
  - ▶ Little administration of the transaction log required.
  - ▶ Not possible to take transaction log backups.
  - ▶ Better performance, as transactions are minimally logged.
  - ▶ Stores just enough information to recover or rollback.
  - ▶ Benefits the following operations:
    - ▶ Bulk imports, SELECT INTO, UPDATE with large data types, WRITETEXT, UPDATETEXT, Index Creation, Index Rebuilds.
  - ▶ Cannot perform point-in-time recovery.
  - ▶ Incompatible with the following features:
    - ▶ AlwaysOn availability group, Database mirroring, Log shipping.

# Recovery Modes (cont'd)

- ▶ FULL Recovery Mode
  - ▶ Log truncation occurs AFTER transaction log backup, providing that a CHECKPOINT operation has occurred since the previous backup.
  - ▶ Point-in-time recovery is possible (Database can be restored to a point in the middle of a transaction log backup).
  - ▶ Compatible with **ALL** SQL Server functionality.
  - ▶ Many factors which might cause virtual log files (VLFs) within transaction log to not be truncated, known as delayed truncation. (reason can be found in the log\_reuse\_wait\_desc) column of sys.databases. Listing 12-1 shows this demonstration. The results are in Figure 12-1.
  - ▶ Transaction log backups must be scheduled to run on a frequent basis.
  - ▶ Failure to backup transaction logs will leave database risk of being unrecoverable.
  - ▶ Transaction log will continue to grow until it runs out of space.

```
SELECT
    name
    , log_reuse_wait_desc
FROM sys.databases
WHERE name = 'WideWorldImporters' ;
```

Listing 12-1 Finding the Reason for Delayed Log Truncation



	name	log_reuse_wait_desc
1	WideWorldImporters	LOG_BACKUP

Figure 12-1 Results of reason for delayed truncation



# Recovery Modes (cont'd)

## ▶ BULK LOGGED Recovery Mode

- ▶ Designed to be used on a short-term basis while a bulk import operation takes place.
- ▶ BULK LOGGED recovery model is used when bulk import takes place.
- ▶ After importing, switch back to FULL recovery model.
- ▶ Performance benefits: prevents transaction log from filling up. (bulk operation minimally logged)
- ▶ Good practice to take a transaction log backup when switching to and from FULL recovery to BULK LOGGED recovery.
- ▶ Logons should be disabled during BULK LOGGED recovery to ensure no other data modifications takes place.
- ▶ **BULK LOGGED recovery may not be faster than FULL recovery for bulk imports unless the system has very fast IO speeds.**
- ▶ **BULK LOGGED force data pages that have been updated with minimally logged pages to be flushed to disc as soon as operation completes instead of waiting for checkpoint operation.**



# Backup Types

- ▶ Full Backup:
  - ▶ Can be taken in any recovery mode.
  - ▶ SQL first issues a CHECKPOINT, causing dirty pages to be written to disc.
  - ▶ Backup every page within the database, then backup the transaction log.
  - ▶ Allows restoration to most recent point, including any transactions that took place during data read phase of the backup.
- ▶ Differential Backup:
  - ▶ Backup every page in the database that has been modified since the last full backup.
  - ▶ Changes are kept track using bitmap pages called DIFF pages. (occurs once every 64000 extents).
  - ▶ Useful if there is significant time between full backups, but log backups are taken frequently.
  - ▶ Drastically decreases the number of transaction log backups that need to be restored.



# Backup Types (cont'd)

## ▶ Log Backup

- ▶ Can only be taken in FULL or BULK LOGGED recovery mode.
- ▶ When issued in FULL recovery mode, it will backup all transaction log records since the last backup.
- ▶ When performed in BULK LOGGED recovery mode, it will backup any pages that include minimally logged transactions.
- ▶ Upon completion, SQL Server will truncate VLFs within transaction log until the first active VLF is reached.
- ▶ Important on databases that support Online Transaction Processing (OLTP).
  - ▶ Allow point-in-time recovery to the point immediately before the disaster occurred.
  - ▶ Least resource-intensive type backup (able to perform more frequently).

# Backup Media

- ▶ Databases can be backed-up to disc, tape, or Windows Azure Blob.
- ▶ Tape backups are deprecated
- ▶ The terminology around backup media consists of backup devices, logical backup devices, media sets, media families, and backup sets.
- ▶ The structure of a media set is depicted in Figure 12-2

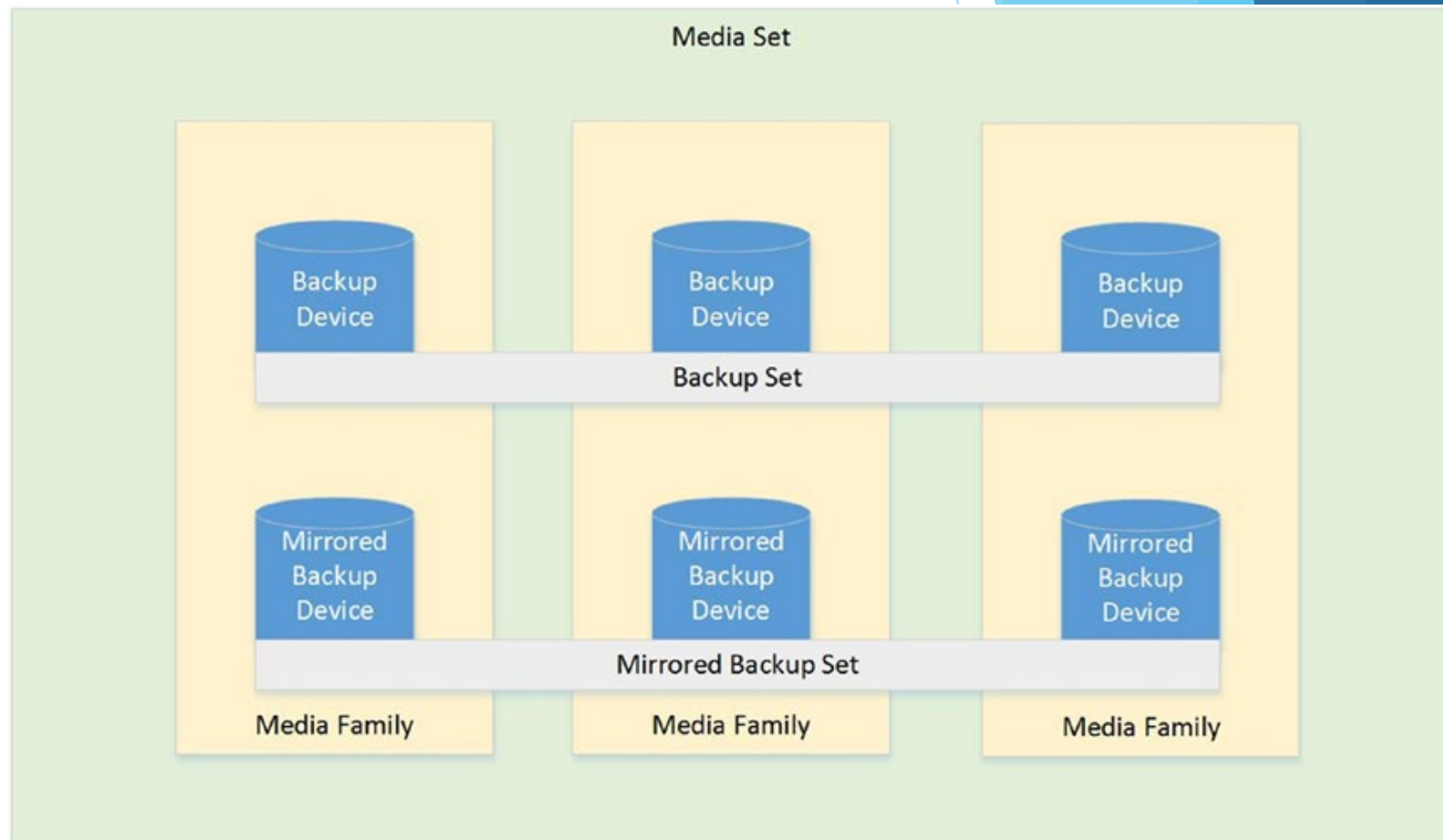


Figure 12-2 Backup media diagram

# Backup Media (cont'd)

## Backup Device

- ▶ A physical file on a disc, tape or Windows Azure Blob.
- ▶ When the device is a disc, the disc can be local or remote specified by a URL.
- ▶ A media set can contain max. 64 backup devices.
- ▶ Data can be striped across the backup devices and also mirrored (RAID configuration)
  - ▶ Striping the backup is useful for a large database (to increase throughput).
  - ▶ Mirroring is used to ensure the stripes are available at all times (redundancy).
- ▶ If one backup device in a media set is mirrored, then all devices within the media set must be mirrored.
- ▶ Each backup device or mirrored set of backup devices is known as a media family.
- ▶ Each device can have up to four mirrors.
- ▶ Devices within a media must be homogenous (all disc or all tape).
- ▶ Possibility of creating logical backup devices that abstract a physical backup device.
  - ▶ Can be created in SQL Server Management Studio by new backup device from the context menu of Server Objects | Backup Devices (Figure 12-3).

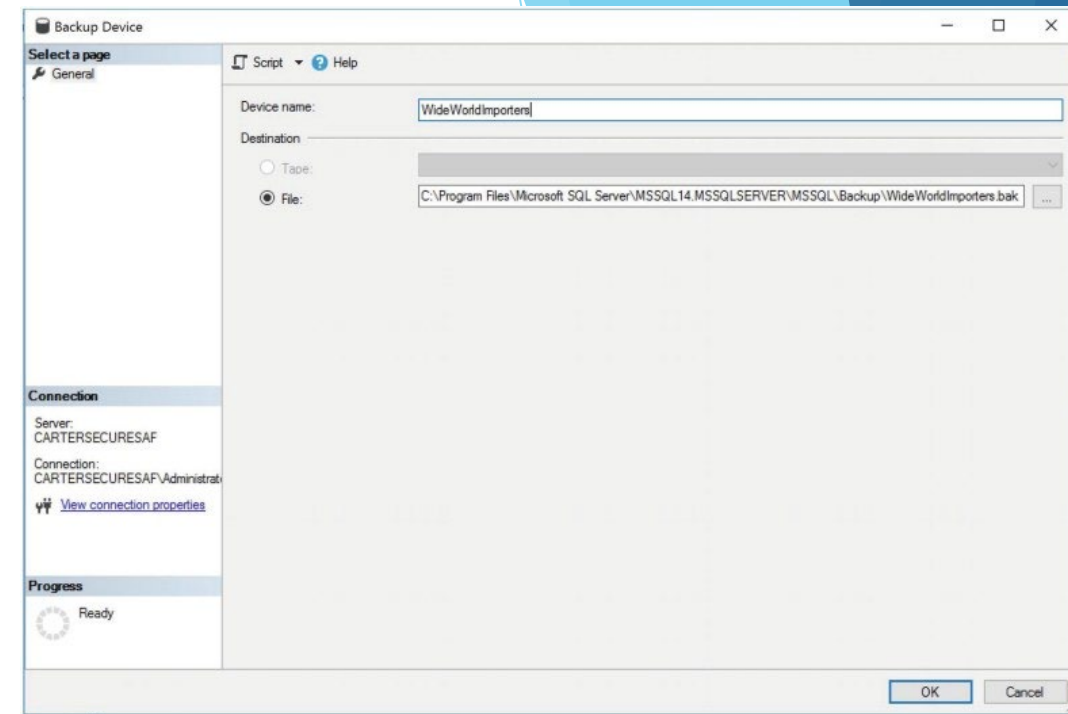


Figure 12-3 Backup Device dialog box

# Backup Media (cont'd)

- ▶ Alternatively, the same logical backup device can be created via T-SQL by using the `sp_addumpdevice` system stored procedure. Parameters are as in Table 12-1.

```
EXEC sp_addumpdevice
    @devtype = 'disk',
    @logicalname = 'WideWorldImporters',
    @physicalname =
        'C:\MSSQL\Backup\WideWorldImporters.bak' ;
GO
```

Listing 12-2 Create a Logical Backup Device

Parameter	Description
@devtype	Specifies the type of backup device. Possible values are: <ul style="list-style-type: none"><li>• Disc</li><li>• Tape</li></ul> Note, however, that tape backups are deprecated and will be removed in a future version of SQL Server.
@logicalname	Specifies the logical name of the backup device. This is the name that you will use when referring to it in <code>BACKUP</code> and <code>RESTORE</code> statements.
@physicalname	Specifies the fully qualified path and name of the backup device, on the backup media
@cntrltype	An obsolete parameter that exists for backup compatibility. This parameter should not be passed. Formally, it was used to specify the controller type.
@devstatus	An obsolete parameter that exists for backup compatibility. This parameter should not be passed.

Table 12-1 `sp_addumpdevice` Parameters





# Backup Media (cont'd)

## Media Sets

- ▶ Contains the backup devices to which the backup will be written.
- ▶ Each media family within a media set is assigned a sequential number based upon their position in the media set (known as family sequence number).
- ▶ Each physical device is allocated a physical sequence number to identify its physical position within the media set.
- ▶ When created, the backup devices are formatted and a media header is written to each device. The header consist of:
  - ▶ Name of media set.
  - ▶ GUID of media set.
  - ▶ Sequence numbers of media families
  - ▶ Number of mirrors in the set.
  - ▶ Date/Time the header was written.



# Backup Media (cont'd)

## Backup Sets

- ▶ Each time a backup is taken to the media set is known as a backup set.
- ▶ New backup sets can be appended to the media or be overwrite existing backup sets.
- ▶ If media set only has one media family, then the media family will contain the entire backup set.
- ▶ Each backup set within the media set will be given a sequential number if it is distributed across media families (to be able to select which backup set to restore).



# Securing Backup Media



# Physical Security

- ▶ Backup media should always be kept in a secure location.
- ▶ Generally, companies will temporarily store backup media in a cage within the data center (so they can be quickly accessed when needed).
- ▶ The backup media will usually be rotated to a secure offsite location where it is retained for a number of years (determined by regulatory reqs.)
- ▶ In the United States, several companies specialize in secure off-site storage, such as Iron Mountain and Saracen.
  - ▶ These companies manage the cycle of media.
  - ▶ provide an SLA for returning backup media, if and when it is required.



# Encrypting Backups

- ▶ SQL Server provides the ability to encrypt backups.
  - ▶ If an attacker were to steal the backup media, they cannot restore and access the database.
  - ▶ Unless if they too, were able to steal the keys used to encrypt the backup.
  - ▶ Encryption keys are stored in the Master database, and is encrypted using the database master key.
- ▶ To encrypt a database, first ensure that a database master key exists in the Master database. Listing 12-3 shows this query.

```
USE master
GO
SELECT *
FROM sys.symmetric_keys
WHERE name =
'##MS_DatabaseMasterKey##' ;
GO
```

Listing 12-3 Check if a Database Master Key Exists in the Master Database



# Encrypting Backups (cont'd)

- ▶ If Master key doesn't exist, it needs to be created first as in Listing 12-4.
- ▶ The key used to encrypt the backup is known as a “Certificate”. The certificate is protected by the Master key. Certificates can be imported using the CREATE CERTIFICATE statement.
- ▶ A trusted Certificate Authority (CA) issues certificates, or in some cases, self-signed certificates are created. This self-signed certificate can be created using Listing 12-5.

```
USE master
GO
CREATE MASTER KEY ENCRYPTION BY
PASSWORD = 'Pa$$wOrd' ;
GO
```

Listing 12-4 Create a Database Master Key

```
USE master
GO
CREATE CERTIFICATE BackupEncryptionCert
    WITH SUBJECT = 'Certificate used to
encrypt backups' ;
GO
```

Listing 12-5 Create a Self-Signed Certificate

# Encrypting Backups (cont'd)

- ▶ Creating a self-signed certificate, then create an encrypted backup. Figure 12-4 shows this dialog box.

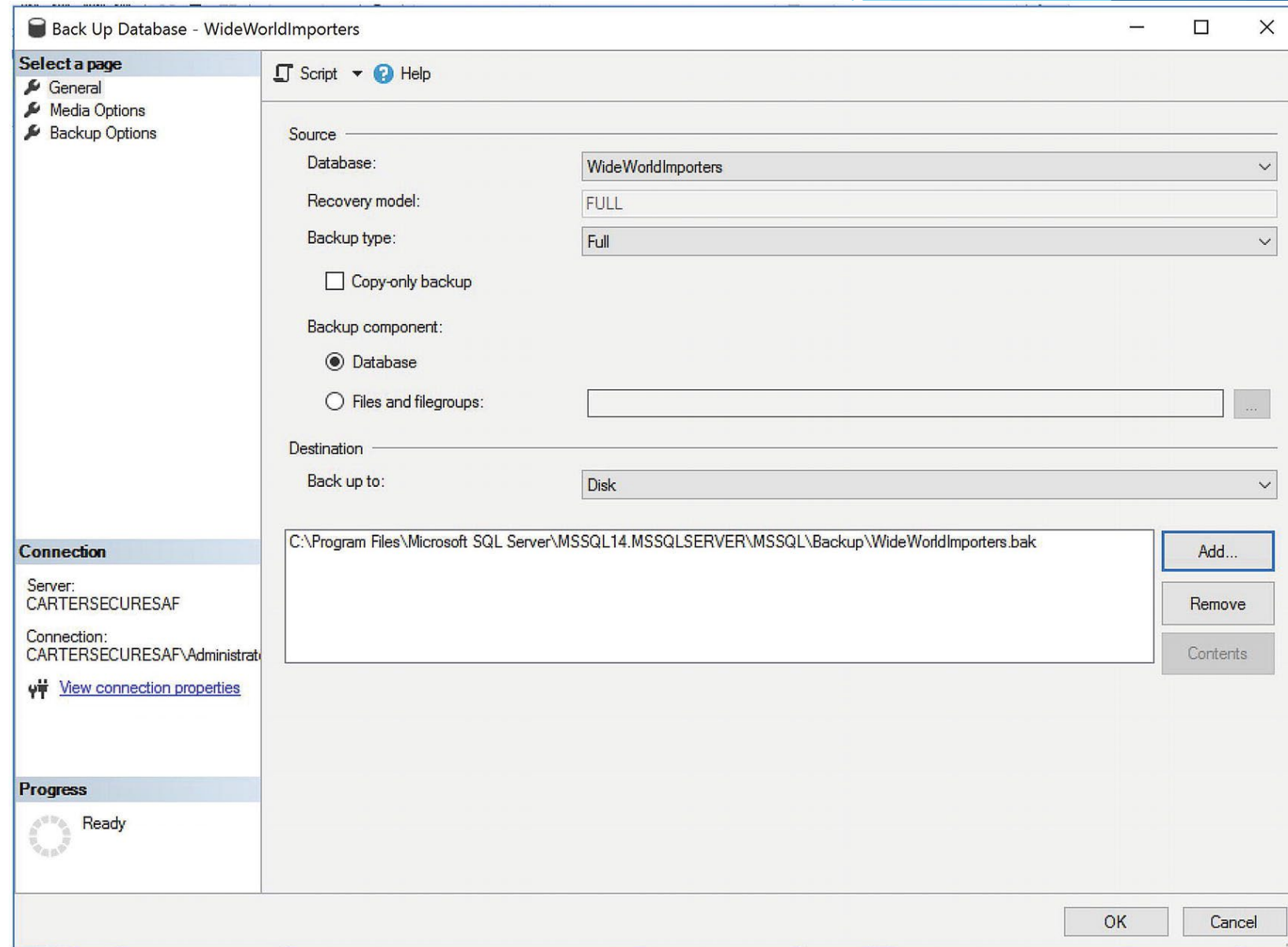


Figure 12-4 Backup Database dialog box-general tab

# Encrypting Backups (cont'd)

- ▶ Specify using an existing media or backup to a new media.
- ▶ Also note the option to overwrite if chose to backup to existing media.
- ▶ Choosing new media enables name specification.
- ▶ In order to take an encrypted backup, must create new media set.
- ▶ Good idea to check “Verify backup when finished” if backing up to Windows Azure Blob (ensure no corruption of backup)
- ▶ **Encrypted backups cannot be taken to a media set that contains unencrypted backups!**

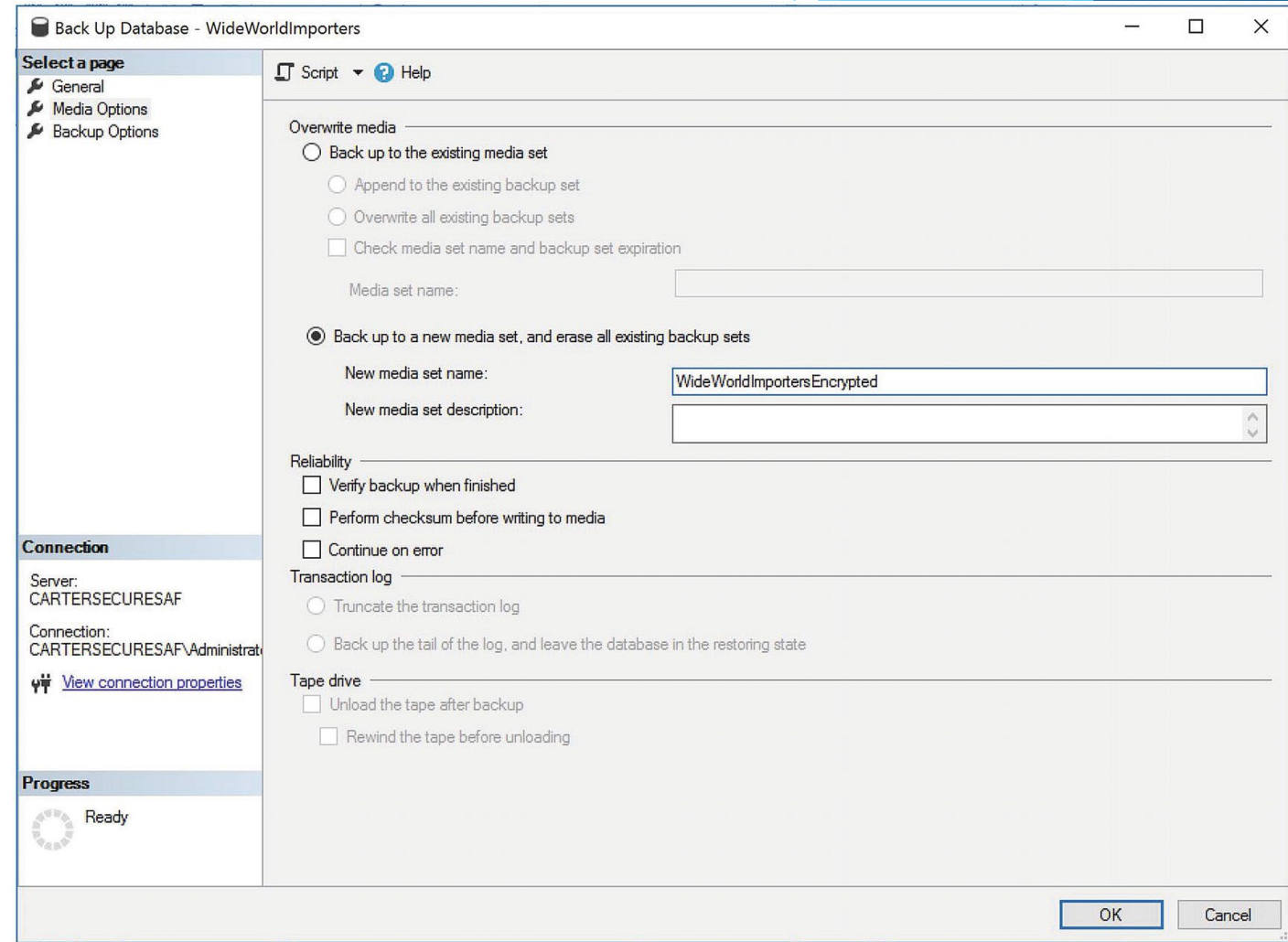


Figure 12-5 Backup dialog box-Media Options tab



# Encrypting Backups (cont'd)

- ▶ Under “Backup Options”, tick the Encrypt Backup checkbox as in Figure 12-6.
- ▶ Choose Symmetric algorithm (i.e., AES-256) and the Certificate to encrypt the backup.
  - ▶ Do not use 3DES as it is deprecated in future SQL Server versions.
  - ▶ AES-256 indicate that the key size is 256-bits. The longer the key the more secure the encryption, but this comes with some overhead (security vs. efficiency)
- ▶ Expiry date can also be configured.
- ▶ Similarly for Compression.

The screenshot shows the 'Back Up Database - WideWorldImporters' dialog box with the 'Backup Options' tab selected. The 'Backup set' section includes fields for 'Name' (WideWorldImporters-Full Database Backup), 'Description', and 'Backup set will expire' (set to 'After: 0 days'). The 'Compression' section has a dropdown for 'Set backup compression' set to 'Use the default server setting'. The 'Encryption' section has the 'Encrypt backup' checkbox checked, with 'Algorithm' set to 'AES 256' and 'Certificate or Asymmetric key' set to 'BackupEncryptionCert (Certificate)'. The 'Connection' section shows 'Server: CARTERSECURESAP' and 'Connection: CARTERSECURESAP\Administrat'. The 'Progress' section shows a 'Ready' status. The 'OK' and 'Cancel' buttons are at the bottom right.

Figure 12-6 Backup Database dialog box-Backup Options tab



# Encrypting Backups (cont'd)

When backing up a database or log via T-SQL, there are many arguments that can be specified. These can be broken down into the following categories:

1. Backup options, which are described in Table 12-2
2. WITH options, which are described in Table 12-3
3. Backup set options, which are described in Table 12-4
4. Media set options, which are described in Table 12-5
5. Error management options, which are described in Table 12-6
6. Tape options, which are described in Table 12-7
7. Log specific options, which are described in Table 12-8
8. Miscellaneous options, which are described in Table 12-9

# Backup Options

Argument	Description
DATABASE/LOG	Specify DATABASE to perform a Full or Differential backup. Specify LOG to perform a transaction log backup.
database_name	The name of the database to perform the backup operation against. Can also be a variable, containing the name of the database.
file_or_filegroup	A comma-separated list of files or filegroups to backup, in the format FILE = logical file name or FILEGROUP = Logical filegroup name
READ_WRITE_FILEGROUPS	Perform a partial backup by backing up all read/write filegroups. Optionally use comma-separated FILEGROUP = syntax after this clause to add read-only filegroups.
TO	A comma-separated list of backup devices to stripe the backup set over, with the syntax DISK = physical device, TAPE = physical device or URL = physical device
MIRROR TO	A comma-separated list of backup devices to mirror the backup set too. If the MIRROR TO clause is used, the number of backup devices specified must equal the number of backup devices specified in the TO clause.

Table 12-2 Backup Options

# WITH Options

Argument	Description
CREDENTIAL	Used when backing up to a Windows Azure Blob.
FILE_SNAPSHOT	When a backup is made to a Windows Azure BLOB, the <code>FILE_SNAPSHOT</code> option will cause an Azure snapshot of the backup file to be taken.
DIFFERENTIAL	Specify that a \differential backup should be taken. If this option is omitted, then a full backup will be taken.
ENCRYPTION	Specify the algorithm to use for the encryption of the backup. If the backup is not to be encrypted, then <code>NO_ENCRYPTION</code> can be specified, which is the default option. Backup encryption is only available in Enterprise, Business Intelligence and Standard Editions of SQL Server.
encryptor_name	The name of the encryptor, in the format <code>SERVER CERTIFICATE = encryptor name</code> or <code>SERVER ASYMETRIC KEY = encryptor name</code>

Table 12-3 WITH Options



# BACKUP Set Options

Argument	Description
COPY_ONLY	Specifies that a copy_only backup of the database or log should be taken. This option is ignored if you perform a differential backup.
COMPRESSION/NO COMPRESSION	By default, SQL Server will decide if the backup should be compressed based on the instance-level setting. You can override this setting, however, by specifying COMPRESSION or NO COMPRESSION, as appropriate. Backup compression is only available in Enterprise, Business Intelligence, and Standard Editions of SQL Server.
NAME	Specifies a name for the backup set
DESCRIPTION	Adds a description to the backup set
EXPIRYDATE/RETAINEDDAYS	Use EXPIRYDATE = datetime to specify a precise date and time that the backup set expires. After this date, the backup set can be overwritten. Specify RETAINEDDAYS = int to specify a number of days before the backup set expires.

Table 12-4 Backup Set Options

# MEDIA Set Options

Argument	Description
INIT/NOINIT	INIT will attempt to overwrite the existing backup sets in the media set but leave the media header intact. It will first check the name and expiry date of the backup set, unless SKIP is specified. NOINIT will append the backup set to the media set, which is the default behavior.
SKIP/NOSKIP	SKIP will cause the INIT checks of backup set name and expiration date to be skipped. NOSKIP will enforce them, which is the default behavior.
FORMAT/NOFORMAT	FORMAT causes the media header to be overwritten, leaving any backup sets within the media set unusable. This essentially creates a new media set. The backup set names and expiry dates will not be checked. NOFORMAT will preserve the existing media header, which is the default behavior.
MEDIANAME	Specifies the name of the media set
MEDIADESCRIPTION	Adds a description of the media set
BLOCKSIZE	Specifies the block size in bytes that will be used for the backup. The BLOCKSIZE will default to 512 for disc and URL and will default to 65536 for tape.

Table 12-5 Media Set Options

# Error Management Options

Argument	Description
CHECKSUM/NO_CHECKSUM	Specifies if the page checksum of each page should be validated, before the page is written to the media set
CONTINUE_AFTER_ERROR/STOP_ON_ERROR	STOP_ON_ERROR is the default behavior and will cause the backup to fail if a bad checksum is discovered, when verifying the page checksum. CONTINUE_AFTER_ERROR will allow the backup to continue if a bad checksum is discovered.

Table 12-6 Error Management Options

# Tape-Specific Options

Argument	Description
UNLOAD/NOUNLOAD	NOUNLOAD specifies that the tape will remain loaded on the tape drive, after the backup operation completes. UNLOAD specifies that the tape will be rewound and unloaded, which is the default behavior.
REWIND/NOREWIND	NOREWIND can improve performance when you are performing multiple backup operations by keeping the tape open after the backup completes. NOREWIND implicitly implies NOUNLOAD as well. REWIND will release the tape and rewind it, which is the default behavior.

Table 12-7 Tape Options



# Log Specific Options

Argument	Description
NORECOVERY/STANDBY	NORECOVERY will cause the database to be left in a restoring state when the backup completes, making it inaccessible to users. STANDBY will leave the database in a read-only state when the backup completes. STANDBY requires that you specify the path and file name of the transaction undo file, so it should be used with the format STANDBY = transaction_undo_file. If neither option is specified, then the database will remain online when the backup completes.
NO_TRUNCATE	Specifies that the log backup should be attempted, even if the database is not in a healthy state. It will also not attempt to truncate in an inactive portion of the log. Taking a tail log backup involves backing up the log with NORECOVERY and NO_TRUNCATE specified.

Table 12-8 Log-Specific Options

# Miscellaneous Options

Argument	Description
BUFFERCOUNT	The total number of IO buffers used for the backup operation
MAXTRANSFERSIZE	The largest possible unit of transfer possible between SQL Server and the backup media, specified in bytes
STATS	Specifies how often progress messages should be displayed. The default is to display a progress message in 10% increments.

Table 12-9 Miscellaneous Options

# Encrypting Backups (cont'd)

- ▶ Performing an Encrypted Backup of the WideWorldImporters Database is as Listing 12-6

```
BACKUP DATABASE WideWorldImporters
  TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL14.
  MSSQLSERVER\MSSQL\Backup\WideWorldImporters.bak'
  WITH FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD,
  MEDIANAME = 'WideWorldImportersEncrypted',
  NAME = 'WideWorldImporters-Full Database Backup',
  ENCRYPTION(ALGORITHM = AES_256, SERVER CERTIFICATE =
  BackupEncryptionCert),
  STATS = 10 ;
GO
```

Listing 12-6 Perform an Encrypted Backup of the WideWorldImporters Database

# Attempting to steal an encrypted backup

- ▶ Restoring an encrypted backup with a valid Certificate.
- ▶ Use the restore database option as in Figure 12-7.

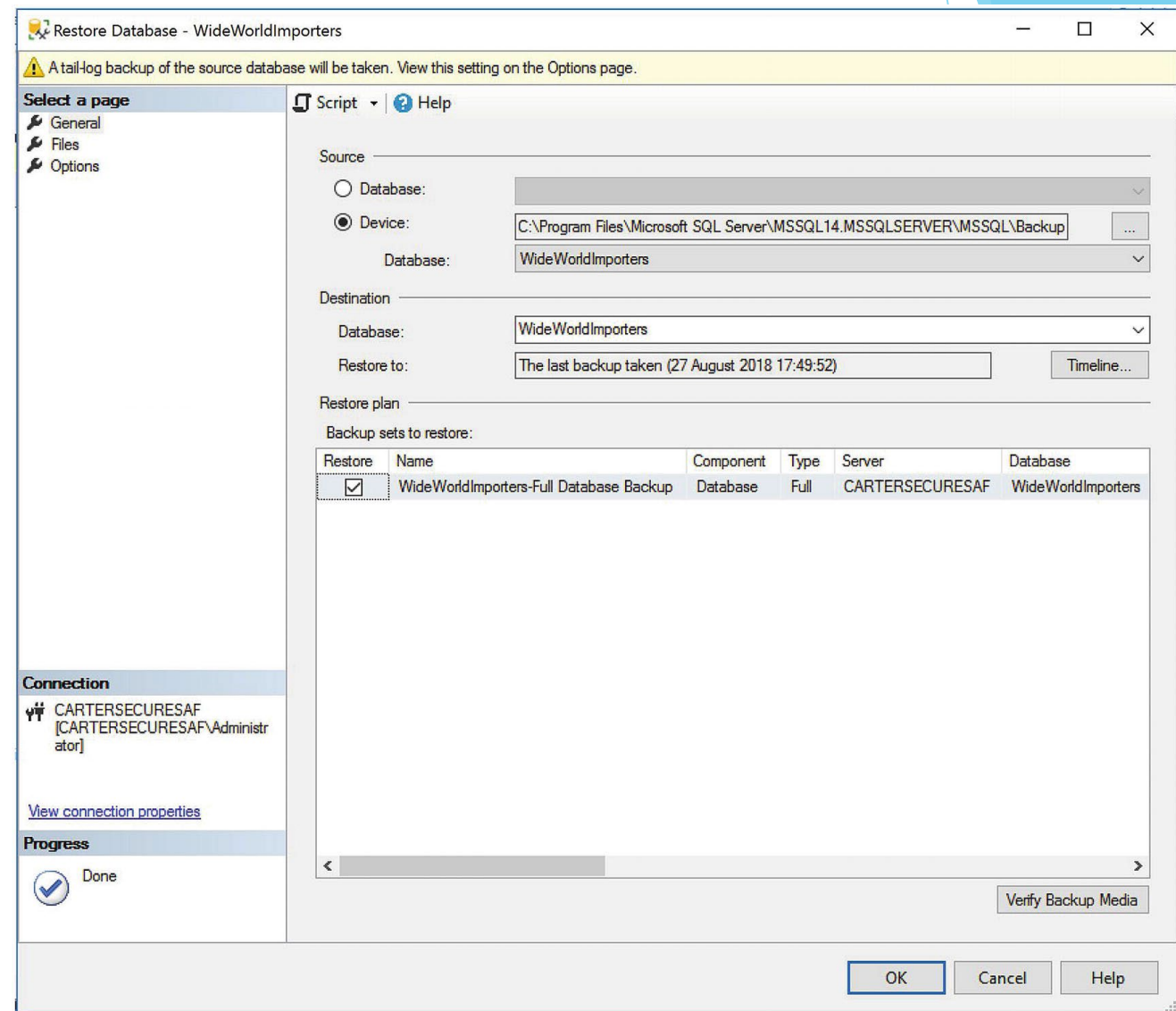


Figure 12-7 Restore Database dialog box

# Attempting to steal an encrypted backup (cont'd)

- ▶ Restoring an encrypted backup **without** a valid Certificate.
- ▶ This can be done as in Figure 12-8

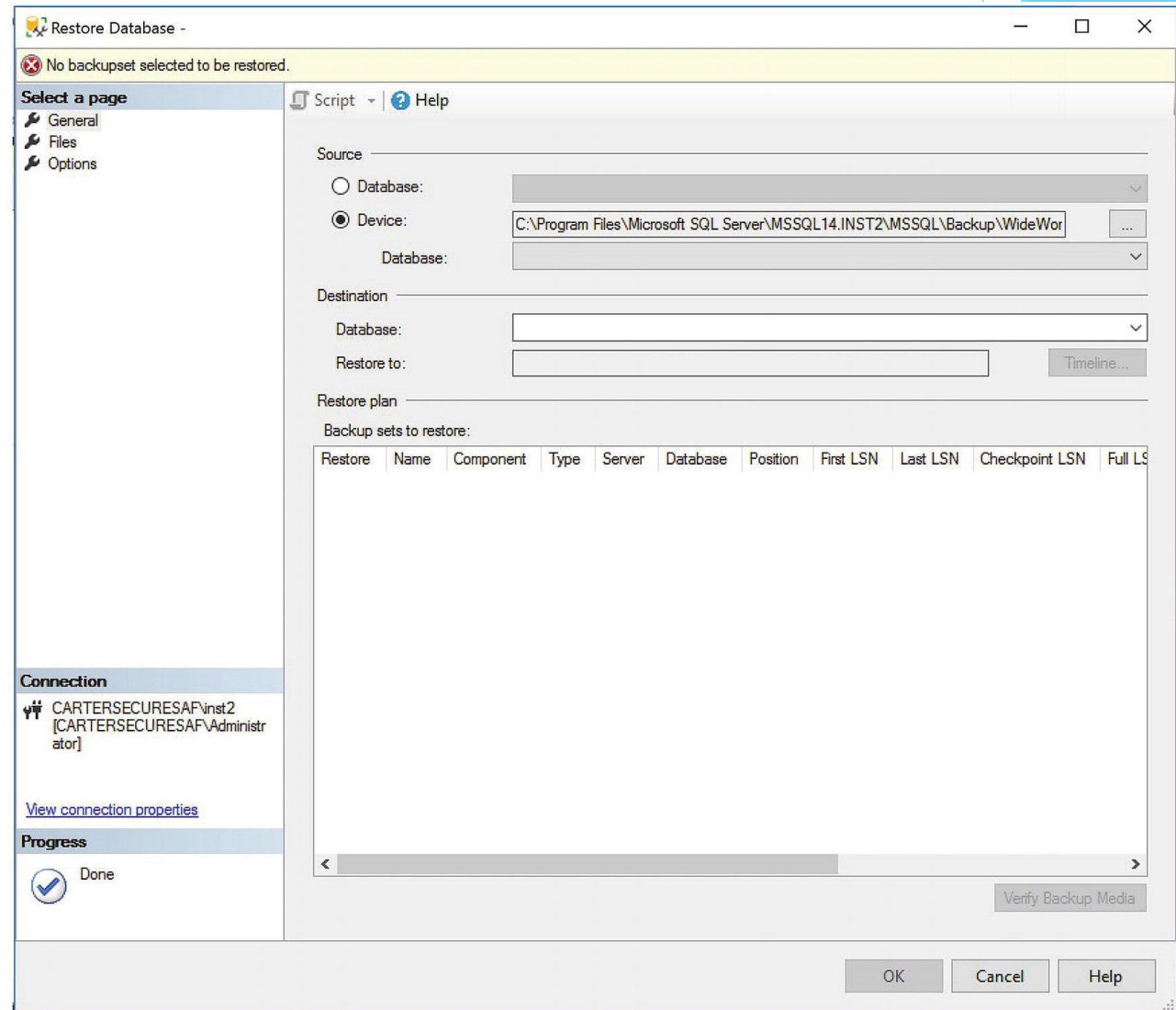


Figure 12-8 Restore dialog box without certificate present

# Attempting to steal an encrypted backup (cont'd)

If the attacker attempts to restore the encrypted database backup using T-SQL, then they would use the RESTORE command. When using the RESTORE command to perform a restore, there are many arguments that can be used to allow many restore scenarios to take place. These arguments can be categorized as follows:

- Restore arguments, which are described in Table 12-10
- WITH options, which are described in Table 12-11
- Backup set options, which are described in Table 12-12
- Media set options, which are described in Table 12-13
- Error management options, which are described in Table 12-14
- Tape options, which are described in Table 12-15
- Miscellaneous options, which are described in Table 12-16

# Restore Options

Argument	Description
DATABASE/LOG	Specify database to restore all or some of the files that constitute the database. Specify LOG to restore a transaction log backup.
DATABASE_NAME	Specify the name of the target database that will be restored.
FILE_OR_FILEGROUP_OR_PAGES	Specify a comma-separated list of the files, file groups, or pages to be restored. If restoring pages, then use the format page = fileid:pageid. In simple recovery mode, files and file groups can only be specified if they are read-only or if you are performing a partial restore, using with partial.
READ_WRITE_FILEGROUPS	Restores all read/write file groups, but no read-only file groups
FROM	A comma-separated list of backup devices that contain the backup set to restore, or the name of the database snapshot from which you wish to restore.

# WITH Options

Argument	Description
PARTIAL	Indicates that this is the first restore in a piecemeal restore
RECOVERY/NORECOVERY/ STANDBY	Specifies the state that the database should be left in when the restore operation completes. Recovery indicates that the database will be brought online. Norecovery indicates that the database will remain in a restoring state, so that subsequent restores can be applied. Standby indicates that the database will be brought online in read-only mode.
MOVE	Used to specify the file system location to which the files should be restored, if this is different from the original location
CREDENTIAL	Used when performing a restore from a windows azure blob
REPLACE	If a database already exists on the instance with the target database name that you have specified in the restore statement, or if the files already exist in the operating system, with the same name or location, then replace indicates that the database or files should be overwritten.
RESTART	Indicates that if the restore operation is interrupted, it should be restarted from that point
RESTRICTED_USER	Indicates that only administrators, and members of the db_owner and db_creator roles, should have access to the database after the restore operation completes



# BACKUP Set Options

Argument	Description
FILE	Indicates the sequential number of the backup set, within the media set, to be used.
PASSWORD	If you are restoring a backup that was taken in sql server 2008 or earlier, where a password was specified during the backup operation, then you will need to use this argument to be able to restore the backup.

# MEDIA Set Options

Argument	Description
MEDIANAME	If this argument is used, then the medianame must match the name of the media set that was allocated during the creation of the media set.
MEDIAPASSWORD	If you are restoring from a media set that was created using sql server 2008 or earlier, and a password was specified for the media set, then this argument must be used during the restore operation.
BLOCKSIZE	Specify the block size to use for the restore operation, in bytes, to override the default value of 65536 for tape and 512 for disc or url.

# Error Management Options

Argument	Description
CHECKSUM/NOCHECKSUM	If checksum was specified during the backup operation, then specifying checksum during the restore operation will verify page integrity during the restore operation. Specifying nochecksum will disable this verification.
CONTINUE_AFTER_ERROR/ STOP_ON_ERROR	Stop_on_error will cause the restore operation to terminate if any damaged pages are discovered. Continue_after_error will cause the restore operation to continue, even if damaged pages are discovered.

# Miscellaneous Options

Argument	Description
BUFFERCOUNT	The total number of IO buffers used for the restore operation
MAXTRANSFERSIZE	The largest possible unit of transfer possible between sql server and the backup media, specified in bytes
STATS	Specifies how often progress messages should be displayed. The default is to display a progress message in 5% increments.
FILESTREAM (DIRECTORY_NAME)	Specifies the name of the folder to which filestream data should be restored.
KEEP_REPLICATION	Preserves the replication settings. This option should be used when configuring log shipping with replication.
KEEP_CDC	Preserves the change data capture settings of a database, when it is being restored. Only relevant if CDC was enabled at the time of the backup operation.
ENABLE_BROKER/ ERROR_BROKER_ CONVERSATIONS/NEW BROKER	Enable_broker specifies that service broker message delivery will be enabled after the restore operation completes, so that messages can immediately be sent. Error_broker_conversations specifies that all conversations will be terminated with an error message before message delivery is enabled. New_broker specifies that conversations will be removed without throwing an error and the database will be assigned a new service broker identifier. Only relevant if service broker was enabled when the backup was created.
STOPAT/STOPATMARK/ STOPBEFOREMARK	Used for point-in-time recovery and only supported in full recovery mode. Stopat specifies a datetime value, which will be time of the last transaction to be restored. Stopatmark specifies either an lsn (log sequence number) to restore to, or the name of a marked transaction, which will be the final transaction that is restored. Stopbeforemark will restore up to the transaction prior to the lsn or marked transaction specified.

# Restoring Database without Certificate

```
RESTORE DATABASE WideWorldImporters  
FROM DISK = 'F:\Backups\WideWorldImporters.bak'  
WITH FILE = 1, NOUNLOAD, STATS = 5 ;  
GO
```

Listing 12-7 Restore the WideWorldImporters Database



Figure 12-9 Error Received When Restoring Without a Certificate



# Administrative Consideration for Encrypted Backups

## Key Management

- ▶ Database Master key and Certificate used to encrypt the database should be backed up and stored in a secure location.
- ▶ Should not be stored on the same server as the SQL Server instance.
- ▶ Master key and certificate should NOT be stored in the same location as the backup media (Locking a door and leaving the key plugged in the key hole).
- ▶ Command to backup master key is as Listing 12-8.

```
USE master
GO
BACKUP MASTER KEY TO FILE = 'c:\keys\DBMasterKey_MasterDatabase'
ENCRYPTION BY PASSWORD = 'MySecurePa$$w0rd' ;
GO
```

Listing 12-8 Backup Master Key




# Administrative Consideration for Encrypted Backups (cont'd)

- ▶ Command to backup Certificate is as Listing 12-9.

```
USE master  
GO  
BACKUP CERTIFICATE BackupEncryptionCert  
TO FILE = 'C:\keys\BackupEncryptionCert' ;  
GO
```

Listing 12-9 Backup Certificate



# Administrative Consideration for Encrypted Backups (cont'd)

## Backup Size

- ▶ Tempting to think that encrypting a database backup may bloat the backup file.
- ▶ Encrypting backups does not bloat backup size!

Backup File	Size (KB)	Bloat
WideWorldImportersStandard.bak	693,118	N/A
WideWorldImportersEncrypted.bak	693,417	0.043%
WideWorldImportersCompressed.bak	164,024	N/A
WideWorldImportersEncryptedAndCompressed.bak	164,109	0.052%

Table 12-17 Size of Each Backup File



# Administrative Consideration for Encrypted Backups (cont'd)

Listing 12-10 shows how to back up the WideWorldImporters database with each possible combination of encryption and compression

```
--Standard
BACKUP DATABASE WideWorldImporters
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\WideWorldImporters.bak'
WITH NAME = 'WideWorldImporters-Full Database Backup - Standard',
NOFORMAT, NOINIT, SKIP, NOREWIND, NOUNLOAD, NO_COMPRESSION, STATS = 10 ;
GO

--Compressed Only
BACKUP DATABASE WideWorldImporters
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\WideWorldImportersCompressed.bak'
WITH NAME = 'WideWorldImporters-Full Database Backup - Compressed',
NOFORMAT, NOINIT, SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10 ;
GO

--Encrypted Only
BACKUP DATABASE WideWorldImporters
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\WideWorldImportersEncrypted.bak'
WITH MEDIANAME = 'WideWorldImporters-Encrypted',
NAME = 'WideWorldImporters-Full Database Backup - Encrypted',
FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, NO_COMPRESSION, STATS = 10,
ENCRYPTION(ALGORITHM = AES_128, SERVER CERTIFICATE = BackupEncryptionCert) ;
GO

--Encrypted and Compressed
BACKUP DATABASE WideWorldImporters
TO DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\WideWorldImportersEncryptedAndCompressed.bak'
WITH MEDIANAME = 'WideWorldImporters-Encrypted',
NAME = 'WideWorldImporters-Full Database Backup - Encrypted and Compressed',
FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, COMPRESSION,
ENCRYPTION(ALGORITHM = AES_128, SERVER CERTIFICATE = BackupEncryptionCert), STATS = 10 ;
GO
```

# Administrative Consideration for Encrypted Backups (cont'd)

- ▶ Figure 12-10 shows the bloat from backup encryption.

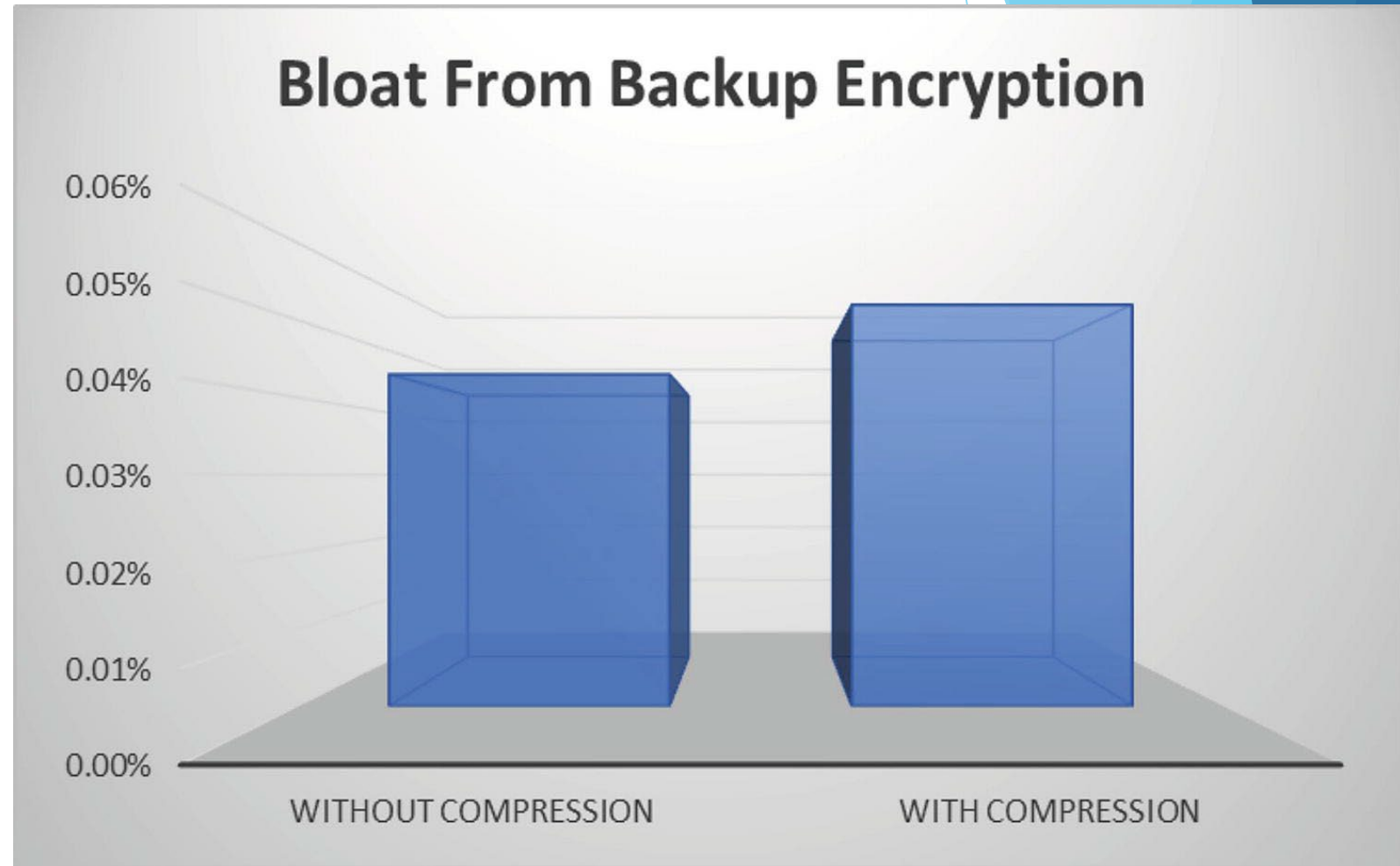


Figure 12-10 Backup sizes



# Summary

- ▶ Attackers can target theft of data without needing to access the instance by stealing the backups.
- ▶ This risk can be mitigated through physical security measures and encryption of backup files.
- ▶ Backup media should always be kept in a secure location, where access is limited.
- ▶ When Backup media is rotated, it should be moved to a secure, offsite location (e.g., Iron Mountain).
- ▶ When Backup is encrypted, it cannot be restored without the Certificate.
- ▶ It is important to store the master key and certificate in a secure different location to the location of the SQL Server instance.
- ▶ The master key and certificate should also not be stored on the same location as the encrypted backup.
- ▶ Enabling encryption on the backup DOES NOT bloat the backup files.



# Code Injection

Part 3

# Topics

- ▶ Understanding Code Injection
  - ▶ Understanding EXECUTE AS
  - ▶ Using EXECUTE AS to perform a code injection attack
  - ▶ Increasing the attack complexity with Obfuscation
- ▶ Protecting Against Code Injection
  - ▶ DevOps
  - ▶ Using Policy-Based Management to Protect Against Code
    - ▶ Creating Conditions
    - ▶ Creating the Policy
  - ▶ Code Signing
- ▶ Summary



# Code Injection

- ▶ Code injection can be thought of as a corporate equivalent to a Trojan Horse Virus.
- ▶ Seemingly innocent code is deployed, but the code contains a back-door through which individuals without authorization can elevate their privileges.
- ▶ Not to be confused with SQL Injection.
- ▶ *Typically* not designed to be malicious, although results can have vastly negative results.
- ▶ Usually happens when a lack of trust between SQL Server developers and DBAs. Developers insert code into deployment so they can have elevated permissions to the SQL Server.
  - ▶ Allows them to investigate or resolve issues without having to deal with the database administrators.
- ▶ Code Injection breaks company policy and compromises security. It may potentially break legal regulations as well (e.g, SOX).
- ▶ Attacks can be simple: Adding a line of code to a deployment script, creating a login using SQL authentication which has “sa” permissions.
- ▶ Attacks can also be sophisticated: “On-request” privileged user creation, and deletion afterwards when it is no longer needed.

# Understanding Code Injection

# Understanding EXECUTE AS

- ▶ Security feature of SQL Server that helps enforce the principle of least privilege.
- ▶ Enables possibility of **granting a user the permissions to execute a stored procedure.**
- ▶ Stored procedure is run in an elevated security context.
- ▶ Predecessor is the SETUSER statement, it is still available in SQL Server 2017 for backward compatibility.
- ▶ When using EXECUTE AS, the code can be run using the various contexts shown in Figure 13-1.
- ▶ It is possible to use this technique to impersonate:
  - ▶ A user based on SQL login
  - ▶ A user based on Windows login
- ▶ Not possible to impersonate a user based on a Windows group.

```
CREATE PROCEDURE dbo.ExecAsDemo
WITH EXECUTE AS 'PrivilegedUser'
AS
BEGIN
    SELECT USER_NAME()
    , SUSER_SNAME()
    , ORIGINAL_LOGIN()
END
GO
SELECT USER_NAME(), SUSER_SNAME(), ORIGINAL_LOGIN()
;
EXEC dbo.ExecAsDemo ;
```

Listing 13-1 Demonstrating EXECUTE AS



# Understanding EXECUTE AS (cont'd)

EXECUTE AS Context	Definition
CALLER	The stored procedure will execute under the security context of the user that invoked the stored procedure.
OWNER	The stored procedure will run under the security context of the user that owns the schema to which the procedure belongs.
SELF	The stored procedure will execute under the security context of the user that either created or last altered the procedure.
'UserName'	The stored procedure will run under the context of a named user.

Table 13-1 EXECUTE AS Contexts With a Stored Procedure

# Using EXECUTE AS to Perform Code Injection Attack

- ▶ Rogue developer can use EXECUTE AS approach to fire a stored procedure, which can create a privileged user account.
- ▶ The TRUSTWORTHY set to ON in the first line defines if the procedure can be used in a database and if UNSAFE CLR can be used within a database.
- ▶ Developer only need EXEC permissions on the procedure to execute.
- ▶ If value 1 is passed to the procedure, it will create the login with admin rights.
- ▶ If value 0, the login is deleted.

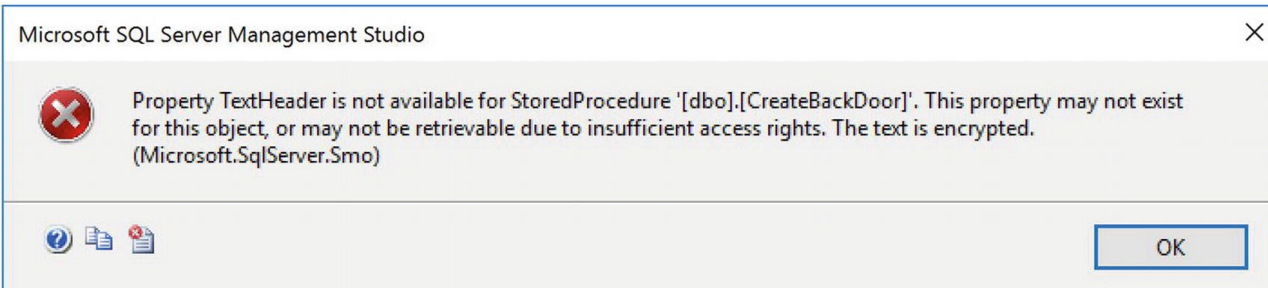
Listing 13-2 Creating a Stored Procedure to Create a User

```
ALTER DATABASE MyDatabase SET TRUSTWORTHY ON ;
GO
USE MyDatabase
GO
CREATE PROCEDURE dbo.CreateBackDoor @Usage INT
WITH EXECUTE AS 'dbo'
AS
BEGIN
    IF @Usage = 1
    BEGIN
        CREATE LOGIN Hack WITH PASSWORD =
        'Pa$$w0rd123' ;
        ALTER SERVER ROLE sysadmin ADD MEMBER [hack] ;
    END
    IF @Usage = 0
    BEGIN
        DROP LOGIN Hack ;
    END
END
GO
```

# Increasing the Attack Complexity with Obfuscation

- ▶ Reduce the chances of a DBA spotting what is happening.
- ▶ Prevents DBA from being able to view the code within the stored procedure.
- ▶ Obfuscation can be easily decrypted by tools such as SQL Compare.

```
ALTER PROCEDURE dbo.CreateBackDoor @Usage INT
WITH ENCRYPTION, EXECUTE AS 'dbo'
AS
BEGIN
    IF @Usage = 1
    BEGIN
        CREATE LOGIN Hack WITH PASSWORD = 'Pa$$w0rd123' ;
        ALTER SERVER ROLE sysadmin ADD MEMBER [hack] ;
    END
    IF @Usage = 0
    BEGIN
        DROP LOGIN Hack ;
    END
END
```



Listing 13-3 Obfuscate the Stored Procedure

Figure 13-1 Message Received When Attempting to View Definition

# Protecting Against Code Injection



# DevOps

- ▶ Non technical
- ▶ Code injection happens because of lack of co-operation and trust between developers and DBAs.
- ▶ DevOps is a set of processes that allows for a closer collaboration between development and operational teams.
- ▶ Creation of cross-functional team, where operations staff are hired as part of a development team to perform activities such as code deployments.
- ▶ Not possible in all circumstances, as some organization require clear separation of duties to comply with regulatory requirements.
- ▶ DevOps is a mindset



# Using Policy-Based Management to Protect Against Code Injection

- ▶ Rogue developer is relying on assumption that DBA will not evaluate their code before deploying.
  - ▶ With a large deployment consisting of thousands of lines of code, this is perfectly plausible
- ▶ DBA can implement simple checks using Policy-Based Management (PBM).
  - ▶ Stop the deployment from finishing
  - ▶ Allow DBA to easily evaluate the deployment, post-execution.



# Using Policy-Based Management to Protect Against Code Injection (cont'd)

- ▶ First step is to create a policy that will avoid the use of EXECUTE AS in stored procedures by
  - ▶ Checking the execution context of stored procedure
  - ▶ Ensuring the stored procedure are configured to execute under the context of the entity that executes the stored procedure.
- ▶ Achieved in SQL Server Management Studio (through Management | Policy Based Management) in Object Explorer as Figure 13-2
  - ▶ Select “New Condition” from context menu of Conditions
  - ▶ Define a name for condition and selected “Stored Procedure” facet.
  - ▶ Chose @ExecutionContext property and stated it must be Caller

# Using Policy-Based Management to Protect Against Code Injection (cont'd)

**Create New Condition - AvoidCodeInjection**

Ready

Select a page

- General
- Description

Connection

CARTERSECURESAP [CARTERSECURESAP\Administrator]

[View connection properties](#)

Progress

Ready

Script Help

Name: AvoidCodeInjection

Facet: Stored Procedure

Expression:

AndOr	Field	Operator	Value
▶	@ExecutionContext	=	Caller
* Click here to add a clause			

**ExecutionContext**  
Gets or sets the execution context for the stored procedure.

OK Cancel Help

Figure 13-2 New Condition dialog box





# Using Policy-Based Management to Protect Against Code Injection (cont'd)

- ▶ Can be alternatively achieved using the `sp_syspolicy_add_condition` system-stored procedure as Listing 13-4.
- ▶ Located in the `dbo` schema of the `MSDB` database. Parameters are as table 13-2.

Parameter	Description
@name	The name to be assigned to the condition
@description	An optional description of the condition
@facet	The name of the facet to which the required properties belong
@expression	The @expression parameter has the data type <code>NVARCHAR (MAX)</code> , but accepts an XML representation of the required condition.
@is_name_condition	Indicates if the expression will evaluate the @name property of an object. Possible values are: <ul style="list-style-type: none"><li>• 0 - Indicates that the condition is not associated with the @name property</li><li>• 1 - Indicates that the condition is associated with the @name property</li></ul>
@obj_name	Specifies the name of the object that will be evaluated, in the event that @is_name_condition is 1
@condition_id	An output parameter, detailed the unique identifier (integer) that has been assigned to the new condition

Table 13-2 Parameters Accepted by `sp_syspolicy_add_condition`



# Using Policy-Based Management to Protect Against Code Injection (cont'd)

Listing 13-4 Create a Condition With sp\_syspolicy\_add\_condition

```
DECLARE @condition_id INT ;
EXEC msdb.dbo.sp_syspolicy_add_condition
    @name='AvoidCodeInjection',
    @description="",
    @facet='StoredProcedure',
    @expression='<Operator>
        <TypeClass>Bool</TypeClass>
        <OpType>EQ</OpType>
        <Count>2</Count>
        <Attribute>
            <TypeClass>Numeric</TypeClass>
            <Name>ExecutionContext</Name>
        </Attribute>
        <Function>
            <TypeClass>Numeric</TypeClass>
            <FunctionType>Enum</FunctionType>
            <ReturnType>Numeric</ReturnType>
            <Count>2</Count>
            <Constant>
                <TypeClass>String</TypeClass>
                <ObjType>System.String</ObjType>
                <Value>Microsoft.SqlServer.Management.Smo.
                ExecutionContext</Value>
            </Constant>
            <Constant>
                <TypeClass>String</TypeClass>
                <ObjType>System.String</ObjType>
                <Value>Caller</Value>
            </Constant>
        </Function>
    </Operator>',
    @is_name_condition=0,
    @obj_name="",
    @condition_id=@condition_id OUTPUT ;
```



# Using Policy-Based Management to Protect Against Code Injection (cont'd)

- ▶ Create the Policy object using the SQL Server Management Studio as in Figure 13-3.
  - ▶ Select “New Policy” from the context menu of policies.
  - ▶ Tick the “Enabled” checkbox.
  - ▶ Select “AvoidCodeInjection” with no filters (every stored procedure within every database on the instance will be assessed).
  - ▶ Configured Evaluation Mode to be “On change: prevent”
- ▶ This means the code deployment will fail should a stored procedure use the EXECUTE AS clause.
- ▶ DBA does not even need to check a log, post-deployment.

# Using Policy-Based Management to Protect Against Code Injection (cont'd)

- ▶ Can be alternatively achieved using the `sp_syspolicy_add_object_set`, `sp_syspolicy_add_target_set`, `sp_syspolicy_add_target_set_level`, `sp_syspolicy_add_policy` via T-SQL.
- ▶ The script in Listing 13-5 demonstrates how the stored procedures explained in Table 13-5 can be used to create the `AvoidCodeInjection` policy.

Listing 13-5 Create the `AvoidCodeInjection` Policy

```
DECLARE @object_set_id INT ;
DECLARE @target_set_id INT ;
DECLARE @policy_id INT ;
EXEC msdb.dbo.sp_syspolicy_add_object_set
EXEC msdb.dbo.sp_syspolicy_add_target_set
    @object_set_name='AvoidCodeInjection_ObjectSet',
    @type_skeleton='Server/Database/StoredProcedure',
    @type='PROCEDURE',
    @enabled=True,
    @target_set_id=@target_set_id OUTPUT ;
EXEC msdb.dbo.sp_syspolicy_add_target_set_level
    @target_set_id=@target_set_id,
    @type_skeleton='Server/Database/StoredProcedure',
    @level_name='StoredProcedure',
    @condition_name="",
    @target_set_level_id=0 ;
EXEC msdb.dbo.sp_syspolicy_add_target_set_level
    @target_set_id=@target_set_id,
    @type_skeleton='Server/Database',
    @level_name='Database',
    @condition_name="",
    @target_set_level_id=0 ;
EXEC msdb.dbo.sp_syspolicy_add_policy
    @name='AvoidCodeInjection',
    @condition_name='AvoidCodeInjection',
    @execution_mode=1,
    @is_enabled=1,
    @policy_id=@policy_id OUTPUT,
    @object_set='AvoidCodeInjection_ObjectSet' ;
```

GO

# sp\_syspolicy\_add\_object\_set and sp\_syspolicy\_add\_target\_set

Parameter	Description
@object_set_name	The object set name, derived from the policy name
@facet	The facet used by the condition associated with the policy
@object_set_id	An output parameter, supplying the unique ID associated with the object set

Table 13-3 sp\_syspolicy\_add\_object\_set Parameters

Parameter	Description
@object_set_id	The ID of the associated object set. Only pass if the @object_set_name parameter is not passed.
@object_set_name	The name of the associated object set. Only pass if the @object_set_id parameter is omitted.
@type_skeleton	The SQL Server object hierarchy path to the target set object type
@type	The type of the object set (which maps to a facet)
@enabled	Specifies if the target set is enabled. Possible values are: <ul style="list-style-type: none"><li>• 0 - Indicating that it is disabled</li><li>• 1 - Indicating that it is enabled</li></ul>
@target_set_id	An output parameter that passes the ID of the newly created target set

Table 13-4 sp\_syspolicy\_add\_target\_set Parameters

# sp\_syspolicy\_add\_policy

Parameter	Description
@name	The name of the Policy to be created
@condition_id	The ID of the condition that will be evaluated. This parameter should only be passed if the @condition_name parameter is omitted.
@condition_name	The name of the condition to be evaluated. This should only be supplied if the @condition_id parameter is omitted.
@schedule_uid	The GUID of the schedule on which the policy will be evaluated
@policy_category	The name of the category with which the policy will be associated
@description	A description of the policy
@help_text	Optional help text that can be displayed when the policy is evaluated. This is helpful to give a brief explanation of the policy.
@help_link	Optional hyperlink that can be displayed when the policy is evaluated. This is helpful if you wish to direct a DBA toward a hosting standards or policy document.
@execution_mode	An integer value, representing the evaluation mode of the policy
@is_enabled	Indicates if the policy will be enabled on creation. Possible values are: 0 – Indicates that the policy will be disabled (This is always the value if the evaluation mode is configured as On Demand.) 1 – Indicates that the policy will be enabled on creation
@root_condition_id	The ID of the condition that will be used to limit the policies evaluation range. Do not pass if the @root_condition_name parameter is passed.
@root_condition_name	The name of the condition that will be used to limit the policies evaluation range. Do not pass, if the @root_condition_id parameter is passed.
@object_set	The name of the object set that binds the policy
@policy_id	An output parameter supplying the ID of the newly created policy

Table 13-5 sp\_syspolicy\_add\_policy Parameters



# Code Signing

- ▶ Alternative to using the EXECUTE AS clause.
- ▶ Code signing does not change the context in which the code is run.
- ▶ The stored procedure is signed with a Certificate.
- ▶ The permission granted to the Certificate is combined with the permission granted to the caller of the stored procedure.
- ▶ Intended for scenarios where it is not possible to control permissions through ownership chains
- ▶ Also useful when it is insecure to use ownership chains, such as in multi-database applications.



# Code Signing (cont'd)

- ▶ First step is to create a certificate (using the CREATE CERTIFICATE statement) as Listing 13-6.
- ▶ Create a user associated with the certificate as Listing 13-7
- ▶ Grant user permissions based on requirements as Listing 13-8.

```
USE MyDatabase  
GO  
  
CREATE CERTIFICATE CodeSigning  
WITH SUBJECT = 'Code Signing Demo' ;
```

Listing 13-6 Create a Self-Signed Certificate

```
CREATE USER CodeSigning  
FROM CERTIFICATE CodeSigning ;
```

Listing 13-7 Create a User Associated with the Certificate

```
GRANT SELECT ON MySchema.MyTable TO CodeSigning ;
```

Listing 13-8 Grant Permissions to the CodeSigning User





# Code Signing (cont'd)

- ▶ Listing 13-9 shows how to create a stored procedure to access a table.
- ▶ Listing 13-10 shows how to add signature to stored procedure.

```
CREATE PROCEDURE dbo.MyProc  
AS  
BEGIN  
    SELECT *  
    FROM dbo.MyTable ;  
END
```

Listing 13-9 Create the Stored Procedure to Access the Table

```
ADD SIGNATURE TO dbo.MyProc BY CERTIFICATE CodeSigning ;
```

Listing 13-10 Sign the Stored Procedure



# Code Signing (cont'd)

- ▶ When the stored procedure is executed by a user, it will return the desired results
  - ▶ even if they don't have permission to the underlying table.
  - ▶ Only permission required by caller is the EXECUTE permission on the stored procedure.
- ▶ Note that if the user has been denied permission to the table, then
  - ▶ The procedure will fail to return results
  - ▶ This is because the 2 permission sets are combined
  - ▶ The DENY associated with the caller will override the GRANT associated with the certificate.



# Summary

- ▶ Code Injection may have non malicious intentions (perceived as a “means of getting things done”), but it is dangerous to the SQL Server.
- ▶ Allows unauthorized changes to be made without the DBA being aware.
- ▶ Good way to avoid is through collaboration (i.e., with DevOps mentality).
- ▶ Policy-Based Management is another tool used to avoid code injection attacks by constraining the deployment code to not use EXECUTE AS clauses.
- ▶ Code signing is used when ownership chains cannot be implemented (insecure when multi-database applications).

# Whole Value Substitution Attack

Part 4



# Whole Value Substitution Attacks

- ▶ Instead of attempting to decrypt or stealing an encrypted value, **attacker replaces** the encrypted value with a different encrypted value.
- ▶ This may benefit the attacker (e.g., replacing their balance amount).

# Topics

- ▶ Understanding Whole Value Substitution Attacks
  - ▶ Salary Manipulation
  - ▶ Credit Card Fraud
- ▶ Protecting Against Whole Value Substitution Attacks
- ▶ Performance Consideration
- ▶ Summary

The background features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the right side of the frame, creating a modern, layered effect. The left side of the image is a solid, very light blue, providing a clean backdrop for the text.

# Understanding Whole Value Substitution Attacks



# Understanding Whole Value Substitution Attacks

```
USE WideWorldImporters
GO
CREATE TABLE Application.Salary
(
    SalaryID INT NOT NULL PRIMARY KEY
    IDENTITY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Posistion NVARCHAR(50) NOT NULL,
    Salary INT NOT NULL,
    SalaryEncrypted VARBINARY(256) NULL
) ;
GO
INSERT INTO Application.Salary (FirstName, LastName, Position,
Salary)
VALUES ('Simon', 'Cutler', 'Warhouse Manager', 50000),
('Mark', 'Walsh', 'Sales Manager', 65000),
('Gerrard', 'Long', 'IT Manager', 54000),
('Oviler', 'Stoneman', 'DBA', 38000),
('Grant', 'Culberston', 'HR Administrator', 20000),
('Michael', 'Ramsdon', 'CEO', 90000) ;
GO
UPDATE Application.Salary
    SET SalaryEncrypted = ENCRYPTBYPASSPHRASE('Pa$$w0rd',CAST
    (Salary AS NVARCHAR)) ;
GO
```

**Listing 14-1 Create the  
Application.Salary Table**





# Understanding Whole Value Substitution Attacks

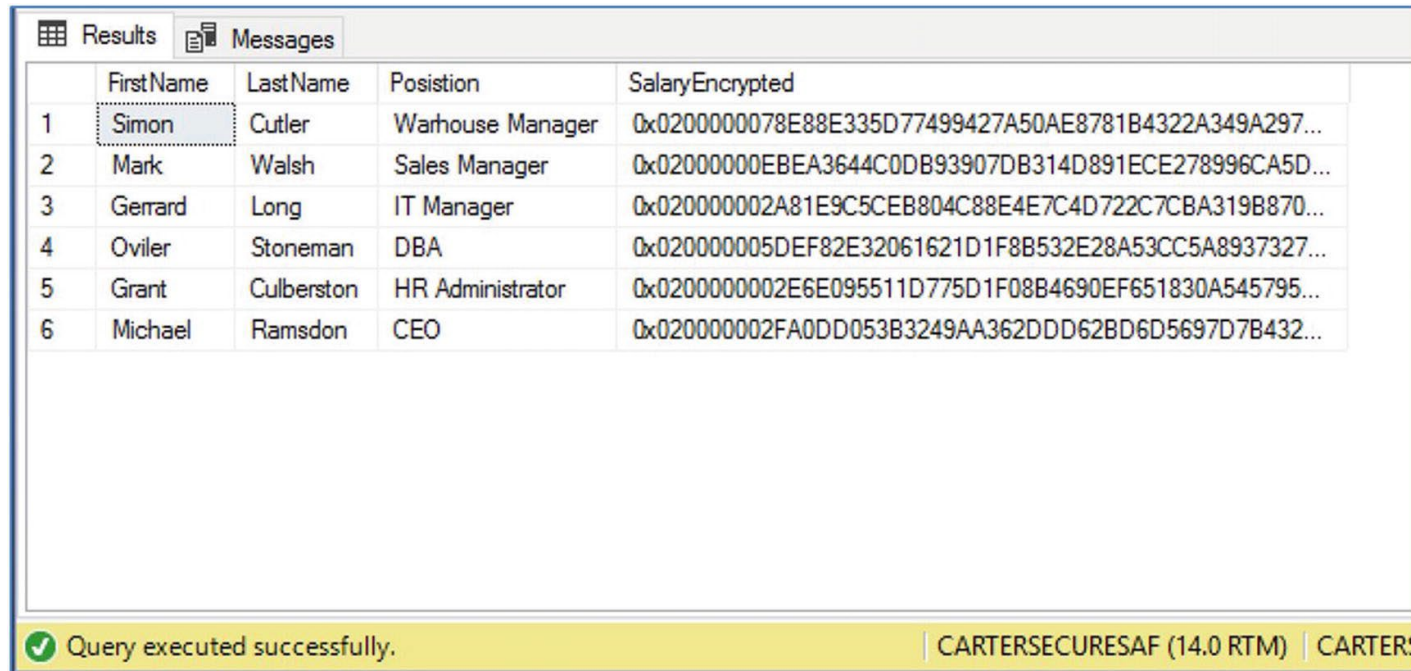
- ▶ Example: Salary Manipulation.
- ▶ Imagine an employee “Grant Culberston” in WorldWideImporters is dishonest
- ▶ He is an HR administrator with SELECT and UPDATE permissions granted against the Salary Table.
- ▶ The plaintext table is used for reference (it would be dropped in real production environment).

```
SELECT    FirstName
          , LastName
          , Posistion
          , SalaryEncrypted
FROM Application.Salary ;
```

Listing 14-2 Viewing the Application.Salary Table

# Understanding Whole Value Substitution Attacks

- ▶ Figure 14-1 is what Grant would see if he display the table using Listing 14-2:



	FirstName	LastName	Posistion	SalaryEncrypted
1	Simon	Cutler	Warhouse Manager	0x0200000078E88E335D77499427A50AE8781B4322A349A297...
2	Mark	Walsh	Sales Manager	0x02000000EBEA3644C0DB93907DB314D891ECE278996CA5D...
3	Gerrard	Long	IT Manager	0x020000002A81E9C5CEB804C88E4E7C4D722C7CBA319B870...
4	Oviler	Stoneman	DBA	0x020000005DEF82E32061621D1F8B532E28A53CC5A8937327...
5	Grant	Culberston	HR Administrator	0x0200000002E6E095511D775D1F08B4690EF651830A545795...
6	Michael	Ramsdon	CEO	0x020000002FA0DD053B3249AA362DDD62BD6D5697D7B432...

Query executed successfully. CARTERSECURESAP (14.0 RTM) CARTERS

Figure 14-1 Data visible to Grant

# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ Scenario: Grant knows “Michael Ramsdon” is CEO, and therefore has higher salary than he does.
- ▶ Grant could UPDATE his encrypted salary field to be equal to that of Michael using Listing 14-3. This would make him have the same salary as the CEO!

```
UPDATE Application.Salary
SET SalaryEncrypted =
(
    SELECT SalaryEncrypted
    FROM Application.Salary
    WHERE FirstName = 'Michael'
        AND LastName = 'Ramsdon'
)
WHERE FirstName = 'Grant'
AND LastName = 'Culberston' ;
```

**Listing 14-3 Perform a Whole Value Substitution Attack**

# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ One can view the results after the update in Listing 14-4

```
UPDATE Application.Salary
SET SalaryEncrypted =
(
    SELECT SalaryEncrypted
    FROM Application.Salary
    WHERE FirstName = 'Michael'
        AND LastName = 'Ramsdon'
)
WHERE FirstName = 'Grant'
AND LastName = 'Culberston' ;
```

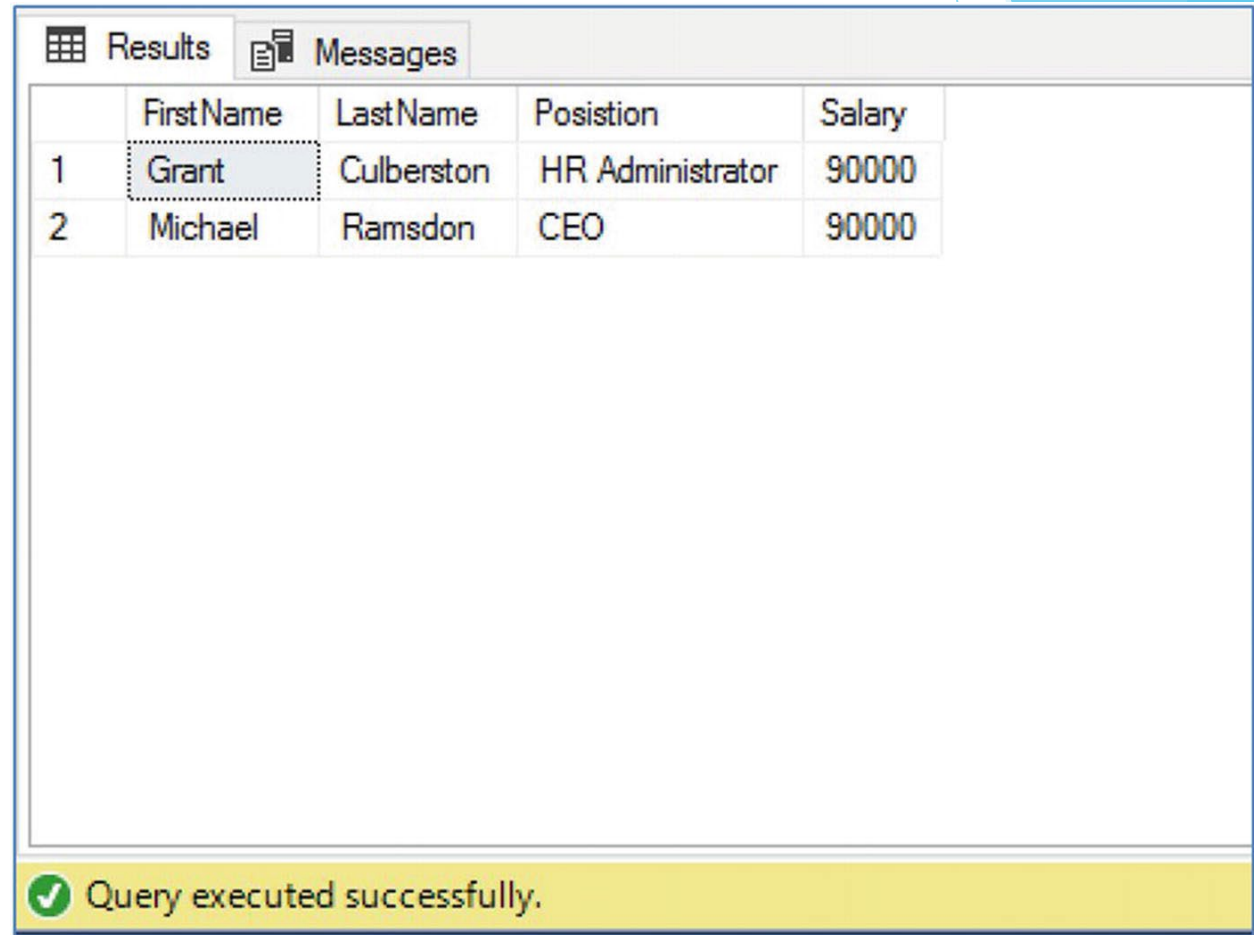
Listing 14-3 Perform a Whole Value Substitution Attack

```
SELECT FirstName
    , LastName
    , Posistion
    ,
CAST(DECRYPTBYPASSPHRASE('Pa$$w0rd',SalaryEncryp
ted) AS NVARCHAR) AS Salary
FROM Application.Salary
WHERE (FirstName = 'Michael' AND LastName =
'Ramsdon')
    OR (FirstName = 'Grant' AND LastName =
'Culberston') ;
```

Listing 14-4 Assessing the Impact of the Attack

# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ After Grant's attack, here's what the salary of Grant and Michael looks like:



	FirstName	LastName	Posistion	Salary
1	Grant	Culberston	HR Administrator	90000
2	Michael	Ramsdon	CEO	90000

✓ Query executed successfully.

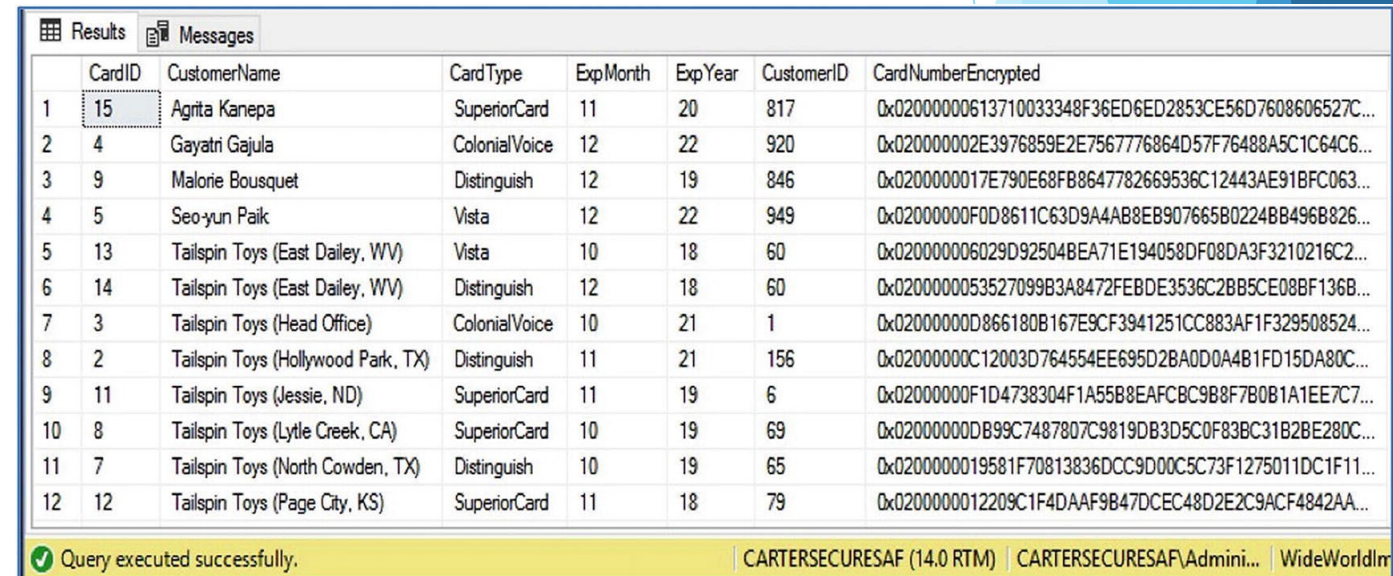
Figure 14-2 Results of assessment

# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ Can external attackers carry out this kind of attacks? **YES!**
- ▶ Imagine a customer of WideWorldImporters, “Valter Viiding” has gained access to the database using an SQL injection attack.
- ▶ Valter managed to display the credit card details as Figure 14-3 using Listing 14-5 :

```
SELECT
    CardID
  , Cust.CustomerName
  , CardType
  , ExpMonth
  , ExpYear
  , cc.CustomerID
  , CardNumberEncrypted
FROM Application.CreditCards cc
INNER JOIN Sales.Customers Cust
    ON Cust.CustomerID =
cc.CustomerID ;
```

Listing 14-5 Returning Credit Card Details



	CardID	CustomerName	CardType	ExpMonth	ExpYear	CustomerID	CardNumberEncrypted
1	15	Agrita Kanepa	SuperiorCard	11	20	817	0x02000000613710033348F36ED6ED2853CE56D7608606527C...
2	4	Gayatri Gajula	ColonialVoice	12	22	920	0x020000002E3976859E2E7567776864D57F76488A5C1C64C6...
3	9	Malorie Bousquet	Distinguish	12	19	846	0x0200000017E790E68FB8647782669536C12443AE91BFC063...
4	5	Seo-yun Paik	Vista	12	22	949	0x02000000F0D8611C63D9A4AB8EB907665B02248B496B826...
5	13	Tailspin Toys (East Dailey, WV)	Vista	10	18	60	0x020000006029D92504BEA71E194058DF08DA3F3210216C2...
6	14	Tailspin Toys (East Dailey, WV)	Distinguish	12	18	60	0x0200000053527099B3A8472FEBDE3536C2BB5CE08BF136B...
7	3	Tailspin Toys (Head Office)	ColonialVoice	10	21	1	0x02000000D866180B167E9CF3941251CC883AF1F329508524...
8	2	Tailspin Toys (Hollywood Park, TX)	Distinguish	11	21	156	0x02000000C12003D764554EE695D2BA0D0A4B1FD15DA80C...
9	11	Tailspin Toys (Jessie, ND)	SuperiorCard	11	19	6	0x02000000F1D4738304F1A55B8EAFBCB9B8F7B0B1A1EE7C7...
10	8	Tailspin Toys (Lytle Creek, CA)	SuperiorCard	10	19	69	0x02000000DB99C7487807C9819DB3D5C0F83BC31B2BE280C...
11	7	Tailspin Toys (North Cowden, TX)	Distinguish	10	19	65	0x0200000019581F70813836DCC9D00C5C73F1275011DC1F11...
12	12	Tailspin Toys (Page City, KS)	SuperiorCard	11	18	79	0x0200000012209C1F4DAAF9B47DCEC48D2E2C9ACF4842AA...

Query executed successfully. | CARTERSECURESAT (14.0 RTM) | CARTERSECURESAT\Admini... | WideWorldIm

Figure 14-3 Credit card details

# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ Valter can replace his own CardNumberEncrypted with the CardNumberEncrypted value of “Agrita Kanepa”.
- ▶ Valter can do this because the web application which was vulnerable to the SQL injection most likely has UPDATE permissions to allow customers to update their credit card information.
- ▶ Now when Valter buys something, Agrita has to pay for it.

```
SELECT
    CardID
    , Cust.CustomerName
    , CardType
    , ExpMonth
    , ExpYear
    , cc.CustomerID
    , CONVERT(NVARCHAR(25), DECRYPTBYPASSPHRASE('Pa$$w0rd', cc.CardN
umberEncrypted, 0)) AS CardNumber
FROM Application.CreditCards cc
INNER JOIN Sales.Customers Cust
    ON Cust.CustomerID = cc.CustomerID
WHERE CardID IN (15,1) ;
```

Listing 14-7 Assessing the Results of the Attack

```
UPDATE Application.CreditCards
SET CardNumberEncrypted =
(
    SELECT CardNumberEncrypted
    FROM Application.CreditCards
    WHERE CardID = 15
),
ExpMonth =
(
    SELECT ExpMonth
    FROM Application.CreditCards
    WHERE CardID = 15
),
ExpYear =
(
    SELECT ExpYear
    FROM Application.CreditCards
    WHERE CardID = 15
)
WHERE CardID = 1 ;
```

Listing 14-6 Credit Card Whole Value Substitution Attack



# Understanding Whole Value Substitution Attacks (cont'd)

- ▶ Consider if the application allows customer to view their OWN credit card information.
- ▶ This means Valter can set his CardNumberEncrypted value to that of his victim, then he can effectively “Decrypt” and obtain the credit card information of the users in the database!
- ▶ Valter can sell these information on the darkweb.

	CardID	CustomerName	CardType	CardNumber	ExpMonth	ExpYear	CustomerID	CardNumber
1	1	Valter Viding	SuperiorCard	33332664695310	11	20	991	33336866065599
2	15	Agrita Kanepa	SuperiorCard	33336866065599	11	20	817	33336866065599

Query executed successfully. CARTERSECURESAP (14.0 RTM) CART

Figure 14-4 Attack assessment results





# Protecting Against Whole Value Substitution Attacks

- ▶ Using `ENCRYPTBYPASSPHRASE()`, contextual information can be used to encrypt important information. Table 14-1 shows the parameters.
- ▶ Utilizing the “add\_authenticator” parameter, an additional keyword is used to encrypt/decrypt the values.
- ▶ For example, the primary key can be used as contextual information (since it is unique).
- ▶ This means without the same contextual information, the decryption would fail.
- ▶ **The authenticator should be static. If it changes, decryption will result in the wrong plaintext!**
- ▶ **The authenticator should also be unique. If there the same authenticator is used between 2 entries, either of them is vulnerable to whole value substitution.**

Parameter	Description
Passphrase	The passphrase that will be used to encrypt the data
cleartext	The value to be encrypted
add_authenticator	Specifies if an authenticator should be used
authenticator	The value to be used to derive an authenticator

Table 14-1 ENCRYPTBYPASSPHRASE() Parameters



# Protecting Against Whole Value Substitution Attacks (cont'd)

- ▶ Additionally, DECRYPTBYPASSPHRASE() parameters are given in Table 14-2.
- ▶ Using an authenticator to add contextual information when encrypting:

```
UPDATE Application.CreditCards
    SET CardNumberEncrypted =
    ENCRYPTBYPASSPHRASE('Pa$$w0rd',CardNumber,1,
        CONVERT(VARBINARY, CardID)) ;
```

Listing 14-8 Re-encrypt Credit Card Details Using an Authenticator

Parameter	Description
Passphrase	The passphrase that will be used to decrypt the data
cipher text	The value to be decrypted
add_authenticator	Specifies if an authenticator will be required to decrypt the data
authenticator	The authenticator data

Table 14-2 DECRYPTBYPASSPHRASE() Parameters

# Protecting Against Whole Value Substitution Attacks (cont'd)

- Now when Valter tries to read his substituted value using Listing 14-9, it would be NULL because decryption would fail as shown in Figure 14-5.

	CardID	CustomerName	CardType	ExpMonth	ExpYear	CustomerID	CardNumber
1	1	Valter Viding	SuperiorCard	11	20	991	NULL
2	15	Agrita Kanepa	SuperiorCard	11	20	817	33336866065599

Query executed successfully. CARTERSECURESAT (14.0)

Figure 14-5 Results of repeating the whole value substitution attack

```
SELECT
    CardID
    , Cust.CustomerName
    , CardType
    , ExpMonth
    , ExpYear
    , cc.CustomerID
    , CONVERT(NVARCHAR(25), DECRYPTBYPASSPHRASE('Pa$$w0rd', cc.CardNumberEncrypted,
1, CONVERT(VARBINARY, CardID))) AS CardNumber
FROM Application.CreditCards cc
INNER JOIN Sales.Customers Cust
    ON Cust.CustomerID = cc.CustomerID
WHERE CardID IN (15,1) ;
```

Listing 14-9 Assess Results of Attack With Authenticator

# Performance Considerations

- ▶ Does encryption with/without authenticator affects performance? Use Listing 14-10 to test.
- ▶ Benchmarking on both cases is performed to ascertain the impact on performance:

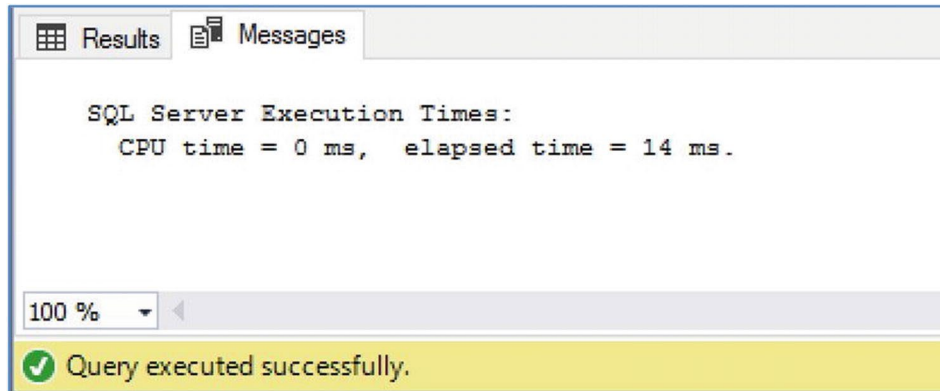


Figure 14-6 Results of first benchmark

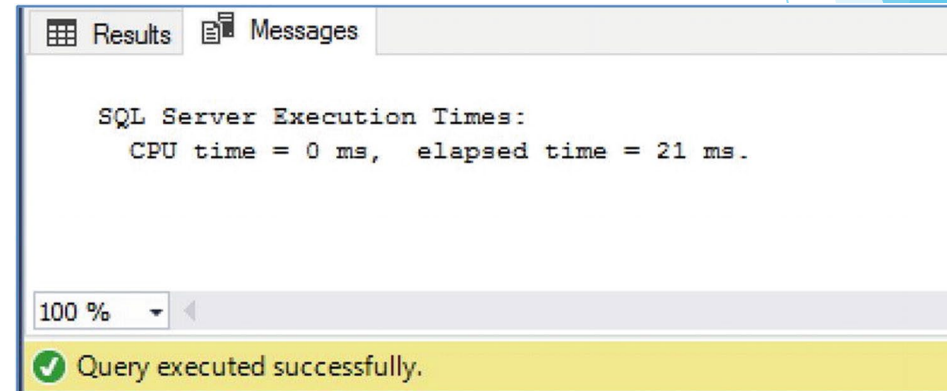


Figure 14-7 Results of second benchmark

- ▶ A simple run shows with authenticator took 33% longer. If this is called up to a table with millions of rows, the impact is significant.
- ▶ trade-off between security and performance.

```
--Encrypt with no authenticator
UPDATE Application.CreditCards
    SET CardNumberEncrypted = ENCRYPTBYPASSPHRASE('Pa$$w0rd',CardNumber) ;
-- Tear down the buffer and plan caches to ensure a fair test and turn on IO
statistics
DBCC FREEPROCCACHE
DBCC DROPCLEANBUFFERS
GO
SET STATISTICS TIME ON
GO
```

## Listing 14-10a Benchmarking Performance

```
--Run first benchmark
SELECT
    CardID
    , Cust.CustomerName
    , CardType
    , ExpMonth
    , ExpYear
    , cc.CustomerID
    , CONVERT(NVARCHAR(25),DECRYPTBYPASSPHRASE('Pa$$w0rd',cc.CardNumberEncrypted, 1,CONVERT(VARBINARY,
CardID))) AS CardNumber
FROM Application.CreditCards cc
INNER JOIN Sales.Customers Cust
    ON Cust.CustomerID = cc.CustomerID ;
--Encrypt with authenticator
UPDATE Application.CreditCards
    SET CardNumberEncrypted = ENCRYPTBYPASSPHRASE('Pa$$w0rd',CardNumber,1,CONVERT(VARBINARY, CardID)) ;
-- Tear down the buffer and plan caches to ensure a fair test and turn on IO statistics
DBCC FREEPROCCACHE
DBCC DROPCLEANBUFFERS
GO
```

## Listing 14-10b Benchmarking Performance

```
--Run second benchmark
SELECT
    CardID
    , Cust.CustomerName
    , CardType
    , ExpMonth
    , ExpYear
    , cc.CustomerID
    , CONVERT(NVARCHAR(25), DECRYPTBYPASSPHRASE('Pa$$w0rd', cc.CardNumberEncrypted,
1, CONVERT(VARBINARY, CardID))) ) AS CardNumber
FROM Application.CreditCards cc
INNER JOIN Sales.Customers Cust
    ON Cust.CustomerID = cc.CustomerID ;
```

### Listing 14-10c Benchmarking Performance



# Performance Considerations

- ▶ Performance depends on a number of factors
  - ▶ Machine specification of server
  - ▶ Concurrent activities on server
- ▶ If data is encrypted with authenticator, processor utilization should be monitored during capacity planning.
- ▶ As the encryption of the authenticator is used, the bloat will increase.
  - ▶ For a large dataset, more data pages will need to be read.
  - ▶ Forces pages out of the buffer cache, meaning more data will need to be retrieved from disc



# Summary

- ▶ Whole value substitution attacks can be used by attackers to manipulate data WITHOUT needing decryption.
- ▶ The encrypted value is simply replaced with another encrypted value.
- ▶ This attack can be deferred using an authenticator during the value encryption process.
- ▶ The authenticator should be unique and static.
- ▶ Changing the encrypted value without changing the authenticator results in the decryption returning a NULL value.
- ▶ Authenticator does not stop encrypted data from being moved, but prevents reveal/utilization of encrypted data to the hacker's advantage.
- ▶ Using an authenticator comes at the expense of performance degradation.