CIS Microsoft SQL Server 2022 Database STIG Benchmark

v1.0.0 - 05-27-2025

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

For information on referencing and/or citing CIS Benchmarks in 3rd party documentation (including using portions of Benchmark Recommendations) please contact CIS Legal (legalnotices@cisecurity.org) and request guidance on copyright usage.

NOTE: It is ***NEVER*** acceptable to host a CIS Benchmark in ***ANY*** format (PDF, etc.) on a 3rd party (non-CIS owned) site.

Table of Contents

Overview

Target Technology Details

Microsoft SQL Server 2022 Database Secure Technical Implementation Guide (STIG)
Version: 1 Release: 1 Benchmark
Date: 27 May 2025

Intended Audience

This benchmark is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate Microsoft SQL Server 2022 Database and are looking to comply with the STIG guidance

Recommendation Definitions

The following defines the various components included in a CIS recommendation as applicable. If any of the components are not applicable it will be noted, or the component will not be included in the recommendation.

Title

Concise description for the recommendation's intended configuration.

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

Profile

A collection of recommendations (or rules) for securing a technology or a supporting platform. STIG Benchmark profiles are used to identify which Vulnerability Severity Category Code (CAT) each rule is associated with.

Description

The Rule Title from the STIG.

Rationale Statement

Detailed reasoning for the recommendation to provide the user a clear and concise understanding on the importance of the recommendation.

Audit Procedure

Systematic instructions for determining if the target system complies with the recommendation.

Remediation Procedure

Systematic instructions for applying recommendations to the target system to bring it into compliance according to the recommendation.

Additional Information

References from the STIG Rule if applicable.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **SEVERITY: CAT I**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be high severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

- **SEVERITY: CAT II**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be medium severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

- **SEVERITY: CAT III**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be low severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

Acknowledgements

The Recommendations in this Benchmark are a representation of the Rules in the unclassified DISA STIG for Microsoft SQL Server 2022 Database

Recommendations

1 STIG RULES

Microsoft SQL Server 2022

Microsoft SQL Server 2022 Database Secure Technical Implementation Guide (STIG)

Version: 1 Release: 1 Benchmark

Date: 27 May 2025

CLASSIFICATION unclassified

1.1 SQLD-22-000100 (Manual)

Profile Applicability:

- SEVERITY: CAT I

Description:

SQL Server must integrate with an organization-level authentication/access mechanism providing account management and automation for all users, groups, roles, and any other principals.

```
GROUP ID: V-271118
RULE ID: SV-271118r1107970
```

Rationale:

Enterprise environments make account management for applications and databases challenging and complex. A manual process for account management functions adds the risk of a potential oversight or other error. Managing accounts for the same person in multiple places is inefficient and prone to problems with consistency and synchronization.

A comprehensive application account management process that includes automation helps to ensure that accounts designated as requiring attention are consistently and promptly addressed.

Examples include using automation to act on multiple accounts designated as inactive, suspended, or terminated, or by disabling accounts located in noncentralized account stores, such as multiple servers. Account management functions can also include assignment of group or role membership; identifying account type; specifying user access authorizations (i.e., privileges); account removal, update, or termination; and administrative alerts. Using automated mechanisms can include, for example: using email or text messaging to notify account managers when users are terminated or transferred; using the information system to monitor account usage; and using automated telephone notification to report atypical system account usage.

SQL Server must be configured to automatically use organization-level account management functions, and these functions must immediately enforce the organization's current account policy.

Automation may comprise differing technologies that when placed together contain an overall mechanism supporting an organization's automated account management requirements.

Audit:

Determine if SQL Server is configured to allow the use of contained databases, if it is, take the appropriate precautions to limit their risk.

1. In the Object Explorer in SQL Server Management Studio (SSMS), right-click on the server instance, select "Properties", and then select the "Advanced" page.

If "Enabled Contained Databases" is "False", this is not a finding.

2. If "Enabled Contained Databases" is "True", then in a query interface such as the SSMS Transact-SQL editor, run the statement:

EXEC sp_configure 'contained database authentication'
If the returned value in the "config_value" or "run_value" column is "0", this is not a finding.

3. Determine whether SQL Server is configured to use only Windows authentication. In a query interface such as the SSMS Transact-SQL editor, run the statement:

SELECT CASE SERVERPROPERTY('IsIntegratedSecurityOnly')

WHEN 1 THEN 'Windows Authentication'

WHEN 0 THEN 'Windows and SQL Server Authentication'

END as [Authentication Mode]
If the returned value in the "Authentication Mode" column is "Windows Authentication", this is not a finding.
If mixed mode (both SQL Server authentication and Windows authentication) is in use, then it must be documented and approved.
From the documentation, obtain the list of accounts authorized to be managed by SQL Server. Determine the accounts (SQL Logins) actually managed by SQL Server.
Run the statement:
SELECT name
FROM sys.database_principals
WHERE type_desc = 'SQL_USER'
AND authentication_type_desc = 'DATABASE';
If any accounts listed by the query are not listed in the documentation, this is a finding.
Documentation must be approved by the information system security officer (ISSO)/information system security manager (ISSM).

Remediation:

If mixed mode is required, document the need and justification; describe the measures taken to ensure the use of SQL Server authentication is kept to a minimum; describe the measures taken to safeguard passwords; list or describe the SQL Logins used; and obtain official approval.
If mixed mode is not required, disable it as follows:
In the SSMS Object Explorer, right-click on the server instance, select Properties >> Security page. Click the radio button for "Windows Authentication Mode", and then click "OK".
Restart the SQL Server instance.
OR
Run the statement:
USE [master]
EXEC xp_instance_regwrite N'HKEY_LOCAL_MACHINE', N'Software\Microsoft\MSSQLServer\MSSQLServer', N'LoginMode', REG_DWORD, 2
GO
Restart the SQL Server instance.
For each account being managed by SQL Server but not requiring it, drop or disable the SQL Database user. Replace it with an appropriately configured account, as needed.
To drop a User in the SSMS Object Explorer:
Navigate to Databases >> Security Users. Right-click on the username and then click "Delete".
To drop a User via a query:
USE database_name;
DROP USER <user_name>;

Additional Information:

CCI-000015 Support the management of system accounts using organization-defined automated mechanisms.

- NIST SP 800-53::AC-2 (1)
- NIST SP 800-53A::AC-2 (1).1
- NIST SP 800-53 Revision 4::AC-2 (1)
- NIST SP 800-53 Revision 5::AC-2 (1)

1.2 SQLD-22-000300 (Manual)

Profile Applicability:

- SEVERITY: CAT I

Description:

SQL Server must enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies.

```
GROUP ID: V-271119
RULE ID: SV-271119r1107973
```

Rationale:

Authentication with a DOD-approved PKI certificate does not necessarily imply authorization to access the DBMS. To mitigate the risk of unauthorized access to sensitive information by entities that have been issued certificates by DOD-approved PKIs, all DOD systems, including databases, must be properly configured to implement access control policies.

Successful authentication must not automatically give an entity access to an asset or security boundary. Authorization procedures and controls must be implemented to ensure each authenticated entity also has a validated and current authorization. Authorization is the process of determining whether an entity, once authenticated, is permitted to access a specific asset. Information systems use access control policies and enforcement mechanisms to implement this requirement.

Access control policies include identity-based policies, role-based policies, and attribute-based policies. Access enforcement mechanisms include access control lists, access control matrices, and cryptography. These policies and mechanisms must be employed by the application to control access between users (or processes acting on behalf of users) and objects (e.g., devices, files, records, processes, programs, and domains) in the information system.

This requirement is applicable to access control enforcement applications, a category that includes database management systems. If SQL Server does not follow applicable policy when approving access, it may be in conflict with networks or other applications in the information system. This may result in users either gaining or being denied access inappropriately and in conflict with applicable policy.

Audit:

If the database is tempdb, this is Not Applicable.
Check SQL Server settings to determine whether users are restricted from accessing objects and data they are not authorized to access.
Review the system documentation to determine the required levels of protection for securables in the database by type of user.
Review the permissions in place in the database.
If the permissions do not match the documented requirements, this is a finding.
Use the supplemental file "Database permission assignments to users and roles.sql".

Remediation:

Configure SQL Server settings and access controls to permit user access only to objects and data that the user is authorized to view or interact with, and to prevent access to all other objects and data.
Use GRANT, REVOKE, DENY, ALTER ROLE … ADD MEMBER … and/or ALTER ROLE …. DROP MEMBER statements to add and remove permissions on database-level securables, bringing them into line with the documented requirements.

Additional Information:

CCI-000213 Enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies.

- NIST SP 800-53::AC-3
- NIST SP 800-53A::AC-3.1
- NIST SP 800-53 Revision 4::AC-3
- NIST SP 800-53 Revision 5::AC-3

1.3 SQLD-22-000500 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must protect against a user falsely repudiating by using system-versioned tables (Temporal Tables).

```
GROUP ID: V-271121
RULE ID: SV-271121r1109177
```

Rationale:

Nonrepudiation of actions taken is required to maintain data integrity. Examples of particular actions taken by individuals include creating information, sending a message, approving information (e.g., indicating concurrence or signing a contract), and receiving a message.

Nonrepudiation protects against later claims by a user of not having created, modified, or deleted a particular data item or collection of data in the database.

In designing a database, the organization must define the types of data and the user actions that must be protected from repudiation. The implementation must then include building audit features into the application data tables and configuring SQL servers' audit tools to capture the necessary audit trail. Design and implementation also must ensure that applications pass individual user identification to SQL Server, even where the application connects to the DBMS with a standard, shared account.

Applications should use temporal tables to track the changes and history of sensitive data.

Audit:

Check the server documentation to determine if collecting and keeping historical versions of a table is required.
If collecting and keeping historical versions of a table is NOT required, this is not a finding.
Find all of the temporal tables in the database using the following query:

```
SELECT SCHEMA_NAME(T.schema_id) AS schema_name, T.name AS table_name,
T.temporal_type_desc, SCHEMA_NAME(H.schema_id) + '.' + H.name AS
history_table
FROM sys.tables T
JOIN sys.tables H ON T.history_table_id = H.object_id
WHERE T.temporal_type != 0
ORDER BY schema_name, table_name
```

Using the system documentation, determine which tables are required to be temporal tables.
If any tables listed in the documentation are not in the list created by running the above statement, this is a finding.
Verify that a field exists documenting the login and/or user who last modified the record.
If this does not exist, this is a finding.
Review the system documentation to determine the history retention period.
Navigate to the table in Object Explorer. Right-click on the table and then select Script Table As >> CREATE To >> New Query Editor Window.
Locate the line that contains "SYSTEM_VERSIONING".
Locate the text that states "HISTORY_RETENTION_PERIOD".
If this text is missing or is set to a value less than the documented history retention period, this is a finding.

Remediation:

Alter sensitive tables to use system versioning.

```
--Alter non-temporal table to define periods for system versioning
ALTER TABLE <MyTableName>
ADD PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
SysStartTime datetime2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL
    DEFAULT SYSUTCDATETIME(),
SysEndTime datetime2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL
    DEFAULT CONVERT(DATETIME2, '9999-12-31 23:59:59.99999999') ;

--Enable system versioning with 1 year retention for historical data
ALTER TABLE <MyTableName>
SET (SYSTEM_VERSIONING = ON (HISTORY_RETENTION_PERIOD = 1 YEAR)) ;
```

https://docs.microsoft.com/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-2022#system_versionin

Additional Information:

CCI-000166 Provide irrefutable evidence that an individual (or process acting on behalf of an individual) falsely denying having performed organization-defined actions to be covered by non-repudiation.

- NIST SP 800-53::AU-10
- NIST SP 800-53A::AU-10.1
- NIST SP 800-53 Revision 4::AU-10
- NIST SP 800-53 Revision 5::AU-10

1.4 SQLD-22-000600 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must protect against a user falsely repudiating by ensuring databases are not in a trust relationship.

```
GROUP ID: V-271122
RULE ID: SV-271122r1109180
```

Rationale:

Nonrepudiation of actions taken is required to maintain data integrity. Examples of particular actions taken by individuals include creating information, sending a message, approving information (e.g., indicating concurrence or signing a contract), and receiving a message.

Nonrepudiation protects against later claims by a user of not having created, modified, or deleted a particular data item or collection of data in the database.

In designing a database, the organization must define the types of data and the user actions that must be protected from repudiation. The implementation must then include building audit features into the application data tables and configuring the DBMS's audit tools to capture the necessary audit trail. Design and implementation also must ensure that applications pass individual user identification to the DBMS, even where the application connects to the DBMS with a standard, shared account.

SQL Server provides the ability for high privileged accounts to impersonate users in a database using the TRUSTWORTHY feature. This will allow members of the fixed database role to impersonate any user within the database.

Audit:

If the database being reviewed is MSDB, trustworthy is required to be enabled, and therefore this is not a finding.
Execute the following query:

```
SELECT
[DatabaseName] = d.name
,[DatabaseOwner] = login.name
,[IsTrustworthy] = CASE
WHEN d.is_trustworthy_on = 0 THEN 'No'
WHEN d.is_trustworthy_on = 1 THEN 'Yes'
END
,[IsOwnerPrivilege] = CASE
WHEN role.name IN ('sysadmin','securityadmin')
OR permission.permission_name = 'CONTROL SERVER'
THEN 'YES'
ELSE 'No'
END
FROM sys.databases d
LEFT JOIN sys.server_principals login ON d.owner_sid = login.sid
LEFT JOIN sys.server_role_members rm ON login.principal_id =
rm.member_principal_id
LEFT JOIN sys.server_principals role ON rm.role_principal_id =
role.principal_id
LEFT JOIN sys.server_permissions permission ON login.principal_id =
permission.grantee_principal_id
WHERE d.name <> 'msdb'
```

If trustworthy is not enabled, this is not a finding.
If trustworthy is enabled and the database owner is not a privileged account, this is not a finding.
If trustworthy is enabled and the database owner is a privileged account, review the system documentation to determine if the trustworthy property is required and authorized. If this is not documented, this is a finding.

Remediation:

Disable trustworthy on the database.

```
ALTER DATABASE [<database name>] SET TRUSTWORTHY OFF;
```

Additional Information:

CCI-000166 Provide irrefutable evidence that an individual (or process acting on behalf of an individual) falsely denying having performed organization-defined actions to be covered by non-repudiation.

- NIST SP 800-53::AU-10
- NIST SP 800-53A::AU-10.1
- NIST SP 800-53 Revision 4::AU-10
- NIST SP 800-53 Revision 5::AU-10

1.5 SQLD-22-000700 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must allow only the information system security manager (ISSM) (or individuals or roles appointed by the ISSM) to select which auditable events are to be audited.

```
GROUP ID: V-271124
RULE ID: SV-271124r1109215
```

Rationale:

Without the capability to restrict which roles and individuals can select which events are audited, unauthorized personnel may be able to prevent or interfere with the auditing of critical events.

Suppression of auditing could permit an adversary to evade detection.

Misconfigured audits can degrade the system's performance by overwhelming the audit log. Misconfigured audits may also make it more difficult to establish, correlate, and investigate the events relating to an incident or identify those responsible for one.

Audit:

Obtain the list of approved audit maintainers from the system documentation.
Use the following query to review database roles and their membership, all of which enable the ability to create and maintain audit specifications.

```
SELECT
    R.name AS role_name,
    RM.name AS role_member_name,
    RM.type_desc
FROM sys.database_principals R
JOIN sys.database_role_members DRM ON
    R.principal_id = DRM.role_principal_id
JOIN sys.database_principals RM ON
    DRM.member_principal_id = RM.principal_id
WHERE R.type = 'R'
    AND R.name = 'db_owner'
ORDER BY
    role_member_name
```

If any role memberships are not documented and authorized, this is a finding.
Review the database roles and individual users that have the following permissions, all of which enable the ability to create and maintain audit definitions.
ALTER ANY DATABASE AUDIT
CONTROL
Use the following query to determine the roles and users that have the listed permissions:

```
SELECT
PERM.permission_name,
DP.name AS principal_name,
DP.type_desc AS principal_type,
DBRM.role_member_name
FROM sys.database_permissions PERM
JOIN sys.database_principals DP ON PERM.grantee_principal_id =
DP.principal_id
LEFT OUTER JOIN (
SELECT
R.principal_id AS role_principal_id,
R.name AS role_name,
RM.name AS role_member_name
FROM sys.database_principals R
JOIN sys.database_role_members DRM ON R.principal_id = DRM.role_principal_id
JOIN sys.database_principals RM ON DRM.member_principal_id = RM.principal_id
WHERE R.type = 'R'
) DBRM ON DP.principal_id = DBRM.role_principal_id
WHERE PERM.permission_name IN ('CONTROL','ALTER ANY DATABASE AUDIT')
ORDER BY
permission_name,
principal_name,
role_member_name
```

If any of the roles or users returned have permissions that are not documented, or the documented audit maintainers do not have permissions, this is a finding.

Remediation:

Create a database role specifically for audit maintainers, and give it permission to maintain audits without granting it unnecessary permissions. (The role name used below is an example; other names may be used.)

```
CREATE ROLE DATABASE_AUDIT_MAINTAINERS;
GO

GRANT ALTER ANY DATABASE AUDIT TO DATABASE_AUDIT_MAINTAINERS;
GO

Use REVOKE and/or DENY and/or ALTER ROLE ... DROP MEMBER ... statements to
remove the ALTER ANY DATABASE AUDIT permission from all users. Then, for each
authorized database user, run the statement:

ALTER ROLE DATABASE_AUDIT_MAINTAINERS ADD MEMBER;
GO
```

Use REVOKE and/or DENY and/or ALTER SERVER ROLE ... DROP MEMBER ... statements to remove CONTROL DATABASE permission from logins that do not need it.

Additional Information:

CCI-000171 Allow organization-defined personnel or roles to select the event types that are to be logged by specific components of the system.

- NIST SP 800-53::AU-12 b
- NIST SP 800-53A::AU-12.1 (iii)
- NIST SP 800-53 Revision 4::AU-12 b
- NIST SP 800-53 Revision 5::AU-12 b

1.6 SQLD-22-001200 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must limit privileges to change software modules, to include stored procedures, functions, and triggers, and links to software external to SQL Server.

```
GROUP ID: V-271143
RULE ID: SV-271143r1108045
```

Rationale:

If the system were to allow any user to make changes to software libraries, then those changes might be implemented without undergoing the appropriate testing and approvals that are part of a robust change management process.

Accordingly, only qualified and authorized individuals must be allowed to obtain access to information system components for purposes of initiating changes, including upgrades and modifications.

Unmanaged changes that occur to the database software libraries or configuration can lead to unauthorized or compromised installations.

Audit:

Obtain a listing of schema ownership from the server documentation.
Execute the following query to obtain a current listing of schema ownership.

```
SELECT S.name AS schema_name, P.name AS owning_principal
FROM sys.schemas S
JOIN sys.database_principals P ON S.principal_id = P.principal_id
ORDER BY schema_name
```

If any schema is owned by an unauthorized database principal, this is a finding.

Remediation:

Transfer ownership of database schemas to authorized database principals.

```
ALTER AUTHORIZATION ON SCHEMA::[<Schema Name>] TO [<Principal Name>]
```

Additional Information:

CCI-001499 Limit privileges to change software resident within software libraries.

- NIST SP 800-53::CM-5 (6)
- NIST SP 800-53A::CM-5 (6).1
- NIST SP 800-53 Revision 4::CM-5 (6)
- NIST SP 800-53 Revision 5::CM-5 (6)

1.7 SQLD-22-001300 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Database objects (including but not limited to tables, indexes, storage, stored procedures, functions, triggers, links to software external to SQL Server, etc.) must be owned by database/DBMS principals authorized for ownership.

```
GROUP ID: V-271146
RULE ID: SV-271146r1109183
```

Rationale:

Within the database, object ownership implies full privileges to the owned object, including the privilege to assign access to the owned objects to other subjects. Database functions and procedures can be coded using definer's rights. This allows anyone who uses the object to perform the actions if they were the owner. If not properly managed, this can lead to privileged actions being taken by unauthorized individuals.

Conversely, if critical tables or other objects in SQL Server rely on unauthorized owner accounts, these objects may be lost when an account is removed.

Audit:

Review system documentation to identify SQL Server accounts authorized to own database objects.
If the SQL Server database ownership list does not exist or needs to be updated, this is a finding.

```
Use the following query to make this determination:

;with objects_cte as
(SELECT o.name, o.type_desc,
   CASE
    WHEN o.principal_id is null then s.principal_id
     ELSE o.principal_id
    END as principal_id
 FROM sys.objects o
 INNER JOIN sys.schemas s
 ON o.schema_id = s.schema_id
 WHERE o.is_ms_shipped = 0
)
SELECT cte.name, cte.type_desc, dp.name as ObjectOwner
FROM objects_cte cte
INNER JOIN sys.database_principals dp
ON cte.principal_id = dp.principal_id
ORDER BY dp.name, cte.name
```

If any of the listed owners are not authorized, this is a finding.

Remediation:

Document and obtain approval for any account(s) authorized for object ownership. If necessary, use the ALTER AUTHORIZATION command to change object ownership to an authorized account.
Example provided below.

```
ALTER AUTHORIZATION ON OBJECT::test.table TO AuthorizedUser;
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-authorization-transact-sql

Additional Information:

CCI-001499 Limit privileges to change software resident within software libraries.

- NIST SP 800-53::CM-5 (6)
- NIST SP 800-53A::CM-5 (6).1
- NIST SP 800-53 Revision 4::CM-5 (6)
- NIST SP 800-53 Revision 5::CM-5 (6)

1.8 SQLD-22-001400 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The role(s)/group(s) used to modify database structure (including but not limited to tables, indexes, storage, etc.) and logic modules (stored procedures, functions, triggers, links to software external to SQL Server, etc.) must be restricted to authorized users.

```
GROUP ID: V-271147
RULE ID: SV-271147r1111078
```

Rationale:

If SQL Server were to allow any user to make changes to database structure or logic, those changes might be implemented without undergoing the appropriate testing and approvals that are part of a robust change management process.

Accordingly, only qualified and authorized individuals must be allowed to obtain access to information system components for purposes of initiating changes, including upgrades and modifications.

Unmanaged changes that occur to the database software libraries or configuration can lead to unauthorized or compromised installations.

DBMS functionality and the nature and requirements of databases will vary; so, while users are not permitted to install unapproved software, there may be instances where the organization allows the user to install approved software packages such as from an approved software repository. The requirements for production servers will be more restrictive than those used for development and research.

The DBMS must enforce software installation by users based on what types of software installations are permitted (e.g., updates and security patches to existing software) and what types of installations are prohibited (e.g., software whose pedigree regarding being potentially malicious is unknown or suspect) by the organization.

In the case of a database management system, this requirement covers stored procedures, functions, triggers, views, etc.

Satisfies: SRG-APP-000133-DB-000362, SRG-APP-000133-DB-000179, SRG-APP-000378-DB-000365

Audit:

If the SQL Server instance supports only software development, experimentation, and/or developer-level testing (i.e., excluding production systems, integration testing, stress testing, and user acceptance testing), this is not a finding.
Obtain a listing of users and roles who are authorized to create, alter, or replace logic modules from the server documentation.
In each user database, execute the following query:

```
SELECT P.type_desc AS principal_type, P.name AS principal_name,
O.type_desc,
CASE class
WHEN 0 THEN DB_NAME()
WHEN 1 THEN OBJECT_SCHEMA_NAME(major_id) + '.' + OBJECT_NAME(major_id)
WHEN 3 THEN SCHEMA_NAME(major_id)
ELSE class_desc + '(' + CAST(major_id AS nvarchar) + ')'
END AS securable_name, DP.state_desc, DP.permission_name
FROM sys.database_permissions DP
JOIN sys.database_principals P ON DP.grantee_principal_id = P.principal_id
LEFT OUTER JOIN sys.all_objects O ON O.object_id = DP.major_id AND O.type IN
('TR','TA','P','X','RF','PC','IF','FN','TF','U')
WHERE DP.type IN ('AL','ALTG') AND DP.class IN (0, 1, 53)

SELECT R.name AS role_name, M.type_desc AS principal_type, M.name AS
principal_name
FROM sys.database_principals R
JOIN sys.database_role_members DRM ON R.principal_id = DRM.role_principal_id
JOIN sys.database_principals M ON DRM.member_principal_id = M.principal_id
WHERE R.name IN ('db_ddladmin','db_owner')
AND M.name <> 'dbo'
```

If any users or role permissions returned are not authorized to modify the specified object or type, this is a finding.
If any user or role membership is not authorized, this is a finding.

Remediation:

Document and obtain approval for any nonadministrative users who require the ability to create, alter, or replace logic modules.
Revoke the ALTER permission from unauthorized users and roles:

```
REVOKE ALTER ON [<Object Name>] FROM [<Principal Name>]
```

Additional Information:

CCI-001499 Limit privileges to change software resident within software libraries.

- NIST SP 800-53::CM-5 (6)
- NIST SP 800-53A::CM-5 (6).1
- NIST SP 800-53 Revision 4::CM-5 (6)
- NIST SP 800-53 Revision 5::CM-5 (6)

CCI-003980 Allow user installation of software only with explicit privileged status.

- NIST SP 800-53 Revision 5::CM-11 (2)

1.9 SQLD-22-001500 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

In the event of a system failure, hardware loss or disk failure, SQL Server must be able to restore necessary databases with least disruption to mission processes.

```
GROUP ID: V-271168
RULE ID: SV-271168r1109218
```

Rationale:

Failure to a known state can address safety or security in accordance with the mission/business needs of the organization. Failure to a known secure state helps prevent a loss of confidentiality, integrity, or availability in the event of a failure of the information system or a component of the system. In the event of a system failure, SQL Server must be able to bring the database back to a consistent state.

Audit:

Review the system documentation to determine whether the database is static, the recovery model to be used, the backup schedule, and the plan for testing database restoration.
If the documentation does not state that the database is static, assume that it is not static. If any of the other information is absent, this is a finding.
If the database is not static, and the documented recovery model is Bulk Logged, but the justification and authorization for this are not documented, this is a finding.
Run the following to determine Recovery Model:

```
USE [master]
GO

SELECT name, recovery_model_desc
FROM sys.databases
ORDER BY name
```

If the recovery model description does not match the documented recovery model, this is a finding.
Review the jobs set up to implement the backup plan. If they are absent, this is a finding.
Check the history of the backups by running the following query. It checks the last 30 days of backups by database.

```
USE [msdb]
GO

SELECT database_name,
   CASE type
    WHEN 'D' THEN 'Full'
    WHEN 'I' THEN 'Differential'
    WHEN 'L' THEN 'Log'
   ELSE type
   END AS backup_type,
 is_copy_only,
 backup_start_date, backup_finish_date
FROM dbo.backupset
WHERE backup_start_date >= dateadd(day, - 30, getdate())
ORDER BY database_name, backup_start_date DESC
```

If the history indicates a pattern of job failures by missing or gaps in backups, this is a finding.
Review evidence that database recovery is tested annually or more often, and that the most recent test was successful. If not, this is a finding.

Remediation:

Modify the system security plan to include whether the database is static, the correct recovery model to be used, the backup schedule, and the plan for testing database restoration.
In SQL Server Management Studio Object Explorer, right-click on the name of the database; select Properties >> Options page. Set the Recovery Model field (near the top of the page) to the correct value.
In Object Explorer, expand >> SQL Server Agent >> Jobs. Create, modify, and delete jobs to implement the backup schedule. (Alternatively, this may be done using T-SQL code or third-party backup software.)
Correct any issues that have been causing backups to fail.
Test the restoration of the database at least once a year; correct any issues that cause it to fail. Maintain a record of these tests.

Additional Information:

CCI-001665 Preserve organization-defined system state information in the event of a system failure.

- NIST SP 800-53::SC-24
- NIST SP 800-53A::SC-24.1 (v)
- NIST SP 800-53 Revision 4::SC-24
- NIST SP 800-53 Revision 5::SC-24

1.10 SQLD-22-001600 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The Database Master Key encryption password must meet DOD password complexity requirements.

```
GROUP ID: V-271169
RULE ID: SV-271169r1109188
```

Rationale:

Weak passwords may be easily guessed. When passwords are used to encrypt keys used for encryption of sensitive data, the confidentiality of all data encrypted using that key is at risk.

Current DOD passwords require the following:

- minimum of 15 characters;
- at least one uppercase character;
- one lowercase character;
- one special character;
- one numeric character, and
- at least eight characters changed from the previous password.

Audit:

From the query prompt:

```
SELECT name
FROM [master].sys.databases
WHERE state = 0
```

Repeat for each database:
From the query prompt:

```
USE [database name]
SELECT COUNT(name)
FROM sys.symmetric_keys s, sys.key_encryptions k
WHERE s.name = '##MS_DatabaseMasterKey##'
AND s.symmetric_key_id = k.key_id
AND k.crypt_type in ('ESKP', 'ESP2', 'ESP3')
```

If the value returned is zero, this is not applicable.
If the value returned is greater than zero, a Database Master Key exists and is encrypted with a password.
Review procedures and evidence of password requirements used to encrypt Database Master Keys.
If the passwords do not meet DOD password standards, this is a finding.

Remediation:

Assign an encryption password to the Database Master Key that is a minimum of 15 characters with at least one uppercase character, one lowercase character, one special character, one numeric character, and at least eight characters changed from the previous password. To change the Database Master Key encryption password:

```
USE [database name];
ALTER MASTER KEY REGENERATE WITH ENCRYPTION BY PASSWORD = 'new password';
```

Note: Do not change the Database Master Key encryption method until the effects are thoroughly reviewed. Changing the master key encryption causes all encryption using the Database Master Key to be decrypted and reencrypted. This action should not be taken during a high-demand time.
Refer to the SQL Server documentation found here prior to reencrypting the Database Master Key:
https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/create-a-database-master-key?

Additional Information:

CCI-001199 Protects the confidentiality and/or integrity of organization-defined information at rest.

- NIST SP 800-53::SC-28
- NIST SP 800-53A::SC-28.1
- NIST SP 800-53 Revision 4::SC-28
- NIST SP 800-53 Revision 5::SC-28

1.11 SQLD-22-001700 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The Database Master Key must be encrypted by the Service Master Key, where a Database Master Key is required and another encryption method has not been specified.

```
GROUP ID: V-271170
RULE ID: SV-271170r1109190
```

Rationale:

When not encrypted by the Service Master Key, system administrators or application administrators may access and use the Database Master Key to view sensitive data that they are not authorized to view. Where alternate encryption means are not feasible, encryption by the Service Master Key may be necessary. To help protect sensitive data from unauthorized access by DBAs, mitigations may be required. Mitigations may include automatic alerts or other audit events when the Database Master Key is accessed outside of the application or by a DBA account.

Audit:

If no databases require encryption, this is not a finding.
From the query prompt:

```
SELECT name
FROM [master].sys.databases
WHERE is_master_key_encrypted_by_server = 1
AND state = 0;
```

If no databases are returned by the query, this is not a finding.
For any databases returned, verify in the System Security Plan that encryption of the Database Master Key using the Service Master Key is acceptable and approved by the Information Owner, and the encrypted data does not require additional protections to deter or detect DBA access. If not approved, this is a finding.
If approved and additional protections are required, verify the additional requirements are in place in accordance with the System Security Plan. These may include additional auditing on access of the Database Master Key with alerts or other automated monitoring.
If the additional requirements are not in place, this is a finding.

Remediation:

Where possible, encrypt the Database Master Key with a password known only to the application administrator.
Where not possible, configure additional audit events or alerts to detect unauthorized access to the Database Master Key by users not authorized to view sensitive data.

Additional Information:

CCI-001199 Protects the confidentiality and/or integrity of organization-defined information at rest.

- NIST SP 800-53::SC-28
- NIST SP 800-53A::SC-28.1
- NIST SP 800-53 Revision 4::SC-28
- NIST SP 800-53 Revision 5::SC-28

1.12 SQLD-22-001800 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The certificate used for encryption must be backed up and stored in a secure location that is not on the SQL Server.

```
GROUP ID: V-271171
RULE ID: SV-271171r1109192
```

Rationale:

Backup and recovery of the certificate used for encryption is critical to the complete recovery of the database. Not having this key can lead to loss of data during recovery.

Audit:

If the application owner and authorizing official have determined that encryption of data at rest is not required, this is not a finding.
Review procedures for and evidence of backup of the certificate used for encryption in the System Security Plan.
If the procedures or evidence does not exist, this is a finding.
If the procedures do not indicate that a backup of the certificate used for encryption is stored in a secure location that is not on the SQL Server, this is a finding.
If procedures do not indicate access restrictions to the certificate backup, this is a finding.

Remediation:

Document and implement procedures to safely back up and store the Certificate used for encryption. Include in the procedures methods to establish evidence of backup and storage, and careful, restricted access and restoration of the Certificate. Also, include provisions to store the backup off-site.

```
BACKUP CERTIFICATE 'CertificateName' TO FILE = 'path_to_file' WITH PRIVATE
KEY (FILE = 'path_to_pvk', ENCRYPTION BY PASSWORD = 'password');
```

As this requires a password, ensure it is not exposed to unauthorized persons or stored as plain text.

Additional Information:

CCI-001199 Protects the confidentiality and/or integrity of organization-defined information at rest.

- NIST SP 800-53::SC-28
- NIST SP 800-53A::SC-28.1
- NIST SP 800-53 Revision 4::SC-28
- NIST SP 800-53 Revision 5::SC-28

1.13 SQLD-22-001900 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must isolate security functions from nonsecurity functions.

```
GROUP ID: V-271172
RULE ID: SV-271172r1109195
```

Rationale:

An isolation boundary provides access control and protects the integrity of the hardware, software, and firmware that perform security functions.

Security functions are the hardware, software, and/or firmware of the information system responsible for enforcing the system security policy and supporting the isolation of code and data on which the protection is based.

Developers and implementers can increase the assurance in security functions by employing well-defined security policy models; structured, disciplined, and rigorous hardware and software development techniques; and sound system/security engineering principles.

Database Management Systems typically separate security functionality from nonsecurity functionality via separate databases or schemas. Database objects or code implementing security functionality should not be commingled with objects or code implementing application logic. When security and nonsecurity functionality are commingled, users who have access to nonsecurity functionality may be able to access security functionality.

Audit:

Determine elements of security functionality (lists of permissions, additional authentication information, stored procedures, application specific auditing, etc.) being housed inside SQL Server.
For any elements found, check SQL Server to determine if these objects or code implementing security functionality are located in a separate security domain, such as a separate database, schema, or table created specifically for security functionality.
Review the system documentation to determine if the necessary database changes cannot be made and that the blockers are also documented. If the necessary changes are documented as not possible, this is not a finding.
Review the database structure to determine where security-related functionality is stored. If security-related database objects or code is not kept separate, this is a finding.

Remediation:

Check documentation and locate security-related database objects and code in a separate database, schema, table, or other separate security domain from database objects and code implementing application logic.
Schemas are analogous to separate namespaces or containers used to store database objects. Security permissions apply to schemas, making them an important tool for separating and protecting database objects based on access rights. Schemas reduce the work required, and improve the flexibility, for security-related administration of a database.
User-schema separation allows for more flexibility in managing database object permissions. A schema is a named container for database objects, which allows the user group objects into separate namespaces.
Where possible, locate security-related database objects and code in a separate database, schema, or other separate security domain from database objects and code implementing application logic. In all cases, use GRANT, REVOKE, DENY, ALTER ROLE … ADD MEMBER … and/or ALTER ROLE …. DROP MEMBER statements to add and remove permissions on server-level and database-level security-related objects to provide effective isolation.
Consider submitting a request to the vendor for changes to a COTS database when database structure does not isolate security functions and cannot be altered directly by the database administrators without loss of official support.

Additional Information:

CCI-001084 Isolate security functions from nonsecurity functions.

- NIST SP 800-53::SC-3
- NIST SP 800-53A::SC-3.1 (ii)
- NIST SP 800-53 Revision 4::SC-3
- NIST SP 800-53 Revision 5::SC-3

1.14 SQLD-22-002000 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Database contents must be protected from unauthorized and unintended information transfer by enforcement of a data transfer policy.

```
GROUP ID: V-271173
RULE ID: SV-271173r1109197
```

Rationale:

Applications, including DBMSs, must prevent unauthorized and unintended information transfer via shared system resources.

Data used for the development and testing of applications often involves copying data from production. It is important that specific procedures exist for this process, including the conditions under which such transfer may take place, where the copies may reside, and the rules for ensuring sensitive data are not exposed.

Copies of sensitive data must not be misplaced or left in a temporary location without the proper controls.

Audit:

Review the procedures for the refreshing of development/test data from production.
Review any scripts or code that exists for the movement of production data to development/test systems, or to any other location or for any other purpose.
Verify that copies of production data are not left in unprotected locations.
If the code that exists for data movement does not comply with the organization-defined data transfer policy and/or fails to remove any copies of production data from unprotected locations, this is a finding.

Remediation:

Modify any code used for moving data from production to development/test systems to comply with the organization-defined data transfer policy, and to ensure copies of production data are not left in unsecured locations.

Additional Information:

CCI-001090 Prevent unauthorized and unintended information transfer via shared system resources.

- NIST SP 800-53::SC-4
- NIST SP 800-53A::SC-4.1
- NIST SP 800-53 Revision 4::SC-4
- NIST SP 800-53 Revision 5::SC-4

1.15 SQLD-22-002100 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must check the validity of all data inputs except those specifically identified by the organization.

```
GROUP ID: V-271176
RULE ID: SV-271176r1109200
```

Rationale:

Invalid user input occurs when a user inserts data or characters into an application's data entry fields and the application is unprepared to process that data. This results in unanticipated application behavior, potentially leading to an application or information system compromise. Invalid user input is one of the primary methods employed when attempting to compromise an application.

With respect to database management systems, one class of threat is known as SQL Injection, or more generally, code injection. It takes advantage of the dynamic execution capabilities of various programming languages, including dialects of SQL. Potentially, the attacker can gain unauthorized access to data, including security settings, and severely corrupt or destroy the database.

Even when no such hijacking takes place, invalid input that gets recorded in the database, whether accidental or malicious, reduces the reliability and usability of the system. Available protections include data types, referential constraints, uniqueness constraints, range checking, and application-specific logic. Application-specific logic can be implemented within the database in stored procedures and triggers, where appropriate.

This calls for inspection of application source code, which will require collaboration with the application developers. It is recognized that in many cases, the database administrator (DBA) is organizationally separate from the application developers, and may have limited, if any, access to source code. Nevertheless, protections of this type are so important to the secure operation of databases that they must not be ignored. At a minimum, the DBA must attempt to obtain assurances from the development organization that this issue has been addressed and must document what has been discovered.

Audit:

Review SQL Server code (stored procedures, functions, triggers), application code, settings, column and field definitions, and constraints to determine whether the database is protected against invalid input. If code exists that allows invalid data to be acted upon or input into the database, this is a finding.
If column/field definitions do not reflect the data, this is a finding.
If columns/fields do not contain constraints and validity checking where required, this is a finding.
Where a column/field is noted in the system documentation as necessarily free-form, even though its name and context suggest that it should be strongly typed and constrained, the absence of these protections is not a finding.
Where a column/field is clearly identified by name, caption, or context as Notes, Comments, Description, Text, etc., the absence of these protections is not a finding.

Remediation:

Use parameterized queries, constraints, foreign keys, etc., to validate data input. Modify SQL Server to properly use the correct column data types as required in the database. Consider submitting a request to the vendor for changes to a COTS database when code is discovered that could create invalid inputs and cannot be altered directly by the database administrators without loss of official support.

Additional Information:

CCI-001310 Checks the validity of organization-defined information inputs to the system.

- NIST SP 800-53::SI-10
- NIST SP 800-53A::SI-10.1
- NIST SP 800-53 Revision 4::SI-10
- NIST SP 800-53 Revision 5::SI-10

1.16 SQLD-22-002400 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must provide nonprivileged users with error messages that provide information necessary for corrective actions without revealing information that could be exploited by adversaries.

```
GROUP ID: V-271179
RULE ID: SV-271179r1108921
```

Rationale:

Any DBMS or associated application providing too much information in error messages on the screen or printout risks compromising the data and security of the system. The structure and content of error messages need to be carefully considered by the organization and development team.

Databases can inadvertently provide a wealth of information to an attacker through improperly handled error messages. In addition to sensitive business or personal information, database errors can provide host names, IP addresses, usernames, and other system information not required for troubleshooting but very useful to someone targeting the system.

Carefully consider the structure/content of error messages. The extent to which information systems are able to identify and handle error conditions is guided by organizational policy and operational requirements. Information that could be exploited by adversaries includes, for example, logon attempts with passwords entered by mistake as the username, mission/business information that can be derived from (if not stated explicitly by) information recorded, and personal information, such as account numbers, social security numbers, and credit card numbers.

This calls for inspection of application source code, which will require collaboration with the application developers. It is recognized that in many cases, the database administrator (DBA) is organizationally separate from the application developers, and may have limited, if any, access to source code. Nevertheless, protections of this type are so important to the secure operation of databases that they must not be ignored. At a minimum, the DBA must attempt to obtain assurances from the development organization that this issue has been addressed and must document what has been discovered.

Audit:

Review application behavior and custom database code (stored procedures, triggers), to determine whether error messages contain information beyond what is needed for explaining the issue to general users.
If database error messages contain PII data, sensitive business data, or information useful for identifying the host system or database structure, this is a finding.

Remediation:

Adjust database code to remove any information not required for explaining the error to an end user.
Consider enabling trace flag 3625 to mask certain system-level error information returned to nonadministrative users.

1. Launch SQL Server Configuration Manager >> SQL Services.
2. Open the instance properties.
3. Select the "Service Parameters" tab.
4. Enter "-T3625".
5. Click "Add" and then click "OK".
6. Restart SQL instance.

Additional Information:

CCI-001312 Generate error messages that provide information necessary for corrective actions without revealing information that could be exploited.

- NIST SP 800-53::SI-11 b
- NIST SP 800-53A::SI-11.1 (iii)
- NIST SP 800-53 Revision 4::SI-11 a
- NIST SP 800-53 Revision 5::SI-11 a

1.17 SQLD-22-002600 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must associate organization-defined types of security labels having organization-defined security label values with information in process, transit, or storage.

```
GROUP ID: V-271184
RULE ID: SV-271184r1109203
```

Rationale:

Without the association of security labels to information, there is no basis for SQL Server to make security-related access-control decisions.

Security labels are abstractions representing the basic properties or characteristics of an entity (e.g., subjects and objects) with respect to safeguarding information.

These labels are typically associated with internal data structures (e.g., tables, rows) within the database and are used to enable the implementation of access control and flow control policies; reflect special dissemination, handling, or distribution instructions; or support other aspects of the information security policy.

One example includes marking data as classified or CUI. These security labels may be assigned manually or during data processing, but, either way, it is imperative these assignments are maintained while the data is in storage. If the security labels are lost when the data is stored, there is the risk of a data compromise.

The mechanism used to support security labeling may be a feature of SQL Server, a third-party product, or custom application code.

Satisfies: SRG-APP-000311-DB-000308, SRG-APP-000314-DB-000310

Audit:

If security labeling is not required, this is not a finding.
If security labeling requirements have been specified, but neither a third-party solution nor a SQL Server Row-Level security solution is implemented that reliably maintains labels on information, this is a finding.

Remediation:

Deploy SQL Server Row-Level Security (refer to link below) or a third-party software, or add custom data structures, data elements, and application code, to provide reliable security labeling of information.
https://msdn.microsoft.com/en-us/library/dn765131.aspx

Additional Information:

CCI-002262 Provide the means to associate organization-defined types of security attributes having organization-defined security attribute values with information in storage.

- NIST SP 800-53 Revision 4::AC-16 a
- NIST SP 800-53 Revision 5::AC-16 a

CCI-002263 Provide the means to associate organization-defined types of security attributes having organization-defined security attribute values with information in process.

- NIST SP 800-53 Revision 4::AC-16 a
- NIST SP 800-53 Revision 5::AC-16 a

CCI-002264 Provide the means to associate organization-defined types of security attributes having organization-defined security attribute values with information in transmission.

- NIST SP 800-53 Revision 4::AC-16 a
- NIST SP 800-53 Revision 5::AC-16 a

1.18 SQLD-22-002800 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must enforce discretionary access control (DAC) policies, as defined by the data owner, over defined subjects and objects.

```
GROUP ID: V-271186
RULE ID: SV-271186r1109206
```

Rationale:

DAC is based on the notion that individual users are "owners" of objects and therefore have discretion over who should be authorized to access the object and in which mode (e.g., read or write). Ownership is usually acquired because of creating the object or via specified ownership assignment. DAC allows the owner to determine who will have access to objects they control. An example of DAC includes user-controlled table permissions.

When discretionary access control policies are implemented, subjects are not constrained regarding what actions they can take with information for which they have already been granted access. Thus, subjects that have been granted access to information are not prevented from passing (i.e., the subjects have the discretion to pass) the information to other subjects or objects.

A subject that is constrained in its operation by Mandatory Access Control policies is still able to operate under the less rigorous constraints of this requirement. Thus, while Mandatory Access Control imposes constraints preventing a subject from passing information to another subject operating at a different sensitivity level, this requirement permits the subject to pass the information to any subject at the same sensitivity level.

The policy is limited by the information system boundary. Once the information is passed outside of the control of the information system, additional means may be required to ensure the constraints remain in effect. While the older, more traditional definitions of DAC require identity-based access control, that limitation is not required for this use of DAC.

Audit:

Review system documentation to determine requirements for object ownership and authorization delegation.
Use the following query to discover database object ownership:
Schemas not owned by the schema or dbo:

```
SELECT name AS schema_name, USER_NAME(principal_id) AS schema_owner
FROM sys.schemas
WHERE schema_id <> principal_id
 AND principal_id <> 1
```

Objects owned by an individual principal:

```
SELECT object_id, name AS securable,
  USER_NAME(principal_id) AS object_owner,
  type_desc
FROM sys.objects
WHERE is_ms_shipped = 0 AND principal_id IS NOT NULL
ORDER BY type_desc, securable, object_owner
```

Use the following query to discover database users who have been delegated the right to assign additional permissions:

```
SELECT U.type_desc, U.name AS grantee,
 DP.class_desc AS securable_type,
   CASE DP.class
    WHEN 0 THEN DB_NAME()
    WHEN 1 THEN OBJECT_NAME(DP.major_id)
    WHEN 3 THEN SCHEMA_NAME(DP.major_id)
   ELSE CAST(DP.major_id AS nvarchar)
   END AS securable,
       permission_name, state_desc
FROM sys.database_permissions DP
JOIN sys.database_principals U ON DP.grantee_principal_id = U.principal_id
WHERE DP.state = 'W'
ORDER BY grantee, securable_type, securable
```

If any of these rights are not documented and authorized, this is a finding.

Remediation:

To correct object ownership:

```
ALTER AUTHORIZATION ON <Securable> TO <Principal>
```

To revoke any unauthorized permissions:

```
REVOKE [Permission] ON <Securable> FROM <Principal>
```

Additional Information:

CCI-002165 Enforce organization-defined discretionary access control policies over defined subjects and objects.

- NIST SP 800-53 Revision 4::AC-3 (4)
- NIST SP 800-53 Revision 5::AC-3 (4)

1.19 SQLD-22-002900 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Execution of stored procedures and functions that use execute as must be restricted to necessary cases only.

```
GROUP ID: V-271188
RULE ID: SV-271188r1108912
```

Rationale:

In certain situations, to provide required functionality, a DBMS needs to execute internal logic (stored procedures, functions, triggers, etc.) and/or external code modules with elevated privileges. However, if the privileges required for execution are at a higher level than the privileges assigned to organizational users invoking the functionality applications/programs, those users are indirectly provided with greater privileges than assigned by organizations.

Privilege elevation must be used only where necessary and protected from misuse.

This calls for inspection of application source code, which will require collaboration with the application developers. It is recognized that in many cases, the database administrator (DBA) is organizationally separate from the application developers, and may have limited, if any, access to source code. Nevertheless, protections of this type are so important to the secure operation of databases that they must not be ignored. At a minimum, the DBA must attempt to obtain assurances from the development organization that this issue has been addressed and must document what has been discovered.

Audit:

Review the system documentation to obtain a listing of stored procedures and functions that use impersonation. Execute the following query:

```
SELECT S.name AS schema_name, O.name AS module_name,
USER_NAME(
CASE M.execute_as_principal_id
WHEN -2 THEN COALESCE(O.principal_id, S.principal_id)
ELSE M.execute_as_principal_id
END
) AS execute_as
FROM sys.sql_modules M
JOIN sys.objects O ON M.object_id = O.object_id
JOIN sys.schemas S ON O.schema_id = S.schema_id
WHERE execute_as_principal_id IS NOT NULL
and       o.name not in
(
'fn_sysdac_get_username',
                              'fn_sysutility_ucp_get_instance_is_mi',
                              'sp_send_dbmail',
                              'sp_SendMailMessage',
                              'sp_syscollector_create_collection_set',
                              'sp_syscollector_delete_collection_set',
                              'sp_syscollector_disable_collector',
                              'sp_syscollector_enable_collector',

'sp_syscollector_get_collection_set_execution_status',
                              'sp_syscollector_run_collection_set',
                              'sp_syscollector_start_collection_set',
                              'sp_syscollector_update_collection_set',
                              'sp_syscollector_upload_collection_set',
                              'sp_syscollector_verify_collector_state',
                              'sp_syspolicy_add_policy',
                              'sp_syspolicy_add_policy_category_subscription',
                              'sp_syspolicy_delete_policy',

'sp_syspolicy_delete_policy_category_subscription',
                              'sp_syspolicy_update_policy',
                              'sp_sysutility_mi_add_ucp_registration',
                              'sp_sysutility_mi_disable_collection',
                              'sp_sysutility_mi_enroll',
                              'sp_sysutility_mi_initialize_collection',
                              'sp_sysutility_mi_remove',
                              'sp_sysutility_mi_remove_ucp_registration',
                              'sp_sysutility_mi_upload',

'sp_sysutility_mi_validate_enrollment_preconditions',
                              'sp_sysutility_ucp_add_mi',
                              'sp_sysutility_ucp_add_policy',
```

```
'sp_sysutility_ucp_calculate_aggregated_dac_health',

'sp_sysutility_ucp_calculate_aggregated_mi_health',
                              'sp_sysutility_ucp_calculate_computer_health',

'sp_sysutility_ucp_calculate_dac_file_space_health',
                              'sp_sysutility_ucp_calculate_dac_health',

'sp_sysutility_ucp_calculate_filegroups_with_policy_violations',
                              'sp_sysutility_ucp_calculate_health',

'sp_sysutility_ucp_calculate_mi_file_space_health',
                              'sp_sysutility_ucp_calculate_mi_health',
                              'sp_sysutility_ucp_configure_policies',
                              'sp_sysutility_ucp_create',
                              'sp_sysutility_ucp_delete_policy',
                              'sp_sysutility_ucp_delete_policy_history',
                              'sp_sysutility_ucp_get_policy_violations',
                              'sp_sysutility_ucp_initialize',
                              'sp_sysutility_ucp_initialize_mdw',
                              'sp_sysutility_ucp_remove_mi',
                              'sp_sysutility_ucp_update_policy',

'sp_sysutility_ucp_update_utility_configuration',
                              'sp_sysutility_ucp_validate_prerequisites',
                              'sp_validate_user',

'syscollector_collection_set_is_running_update_trigger',
                              'sysmail_help_status_sp'
)

ORDER BY schema_name, module_name
```

If any procedures or functions are returned that are not documented, this is a finding.

Remediation:

Alter stored procedures and functions to remove the "EXECUTE AS" statement.

Additional Information:

CCI-002233 Prevent the organization-defined software from executing at higher privilege levels than users executing the software.

- NIST SP 800-53 Revision 4::AC-6 (8)
- NIST SP 800-53 Revision 5::AC-6 (8)

1.20 SQLD-22-003100 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

SQL Server must enforce access restrictions associated with changes to the configuration of the database(s).

```
GROUP ID: V-271195
RULE ID: SV-271195r1109208
```

Rationale:

Failure to provide logical access restrictions associated with changes to configuration may have significant effects on the overall security of the system.

When dealing with access restrictions pertaining to change control, it should be noted that any changes to the hardware, software, and/or firmware components of the information system can potentially have significant effects on the overall security of the system.

Accordingly, only qualified and authorized individuals should be allowed to obtain access to system components for the purposes of initiating changes, including upgrades and modifications.

Audit:

Execute the following query to obtain a listing of user databases whose owner is a member of a fixed server role:

```
 SELECT
              D.name AS database_name, SUSER_SNAME(D.owner_sid) AS
owner_name,
              FRM.is_fixed_role_member
FROM sys.databases D
OUTER APPLY (
              SELECT MAX(fixed_role_member) AS is_fixed_role_member
              FROM (
                            SELECT IS_SRVROLEMEMBER(R.name,
SUSER_SNAME(D.owner_sid)) AS fixed_role_member
                            FROM sys.server_principals R
                            WHERE is_fixed_role = 1
              ) A
) FRM
WHERE D.database_id > 4
              AND (FRM.is_fixed_role_member = 1
                            OR FRM.is_fixed_role_member IS NULL)
ORDER BY database_name
```

If no databases are returned, this is not a finding.
For each database/login returned, review the Server Role memberships:

1. In SQL Server Management Studio, expand "Logins".
2. Double-click the name of the login.
3. Click the "Server Roles" tab.

If any server roles are selected, but not documented and authorized, this is a finding.

Remediation:

Remove unauthorized users from roles:

```
ALTER ROLE DROP MEMBER user;
```

https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-role-transact-sql?
Set the owner of the database to an authorized login:

```
ALTER AUTHORIZATION ON database::DatabaseName TO login;
```

https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-authorization-transact-sql?

Additional Information:

CCI-001813 Enforce access restrictions using organization-defined mechanisms.

- NIST SP 800-53 Revision 4::CM-5 (1)
- NIST SP 800-53 Revision 5::CM-5 (1) (a)

1.21 SQLD-22-003200 (Manual)

Profile Applicability:

- SEVERITY: CAT I

Description:

SQL Server must use NSA-approved cryptography to protect classified information in accordance with the data owner's requirements.

```
GROUP ID: V-271199
RULE ID: SV-271199r1108919
```

Rationale:

Use of weak or untested encryption algorithms undermines the purposes of using encryption to protect data. The application must implement cryptographic modules adhering to the higher standards approved by the federal government since this provides assurance they have been tested and validated.

It is the responsibility of the data owner to assess the cryptography requirements in light of applicable federal laws, Executive Orders, directives, policies, regulations, and standards.

NSA-approved cryptography for classified networks is hardware based. This requirement addresses the compatibility of a DBMS with the encryption devices.

Audit:

Detailed information on the NIST Cryptographic Module Validation Program (CMVP) is available at the following website: http://csrc.nist.gov/groups/STM/cmvp/index.html.
Review system documentation to determine whether cryptography for classified or sensitive information is required by the information owner.
If the system documentation does not specify the type of information hosted on SQL Server as classified, sensitive, and/or unclassified, this is a finding.
If neither classified nor sensitive information exists within SQL Server databases or configuration, this is not a finding.
Verify that Windows is configured to require the use of FIPS-compliant algorithms.

1. Click "Start".
2. Type "Local Security Policy".
3. Press "Enter".
4. Expand "Local Policies".
5. Select "Security Options".
6. Locate "System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing".

If the Security Setting for this option is "Disabled", this is a finding.
Note: The list of acceptable algorithms is "AES 256" and "Triple DES".
If cryptography is being used by SQL Server, verify that the cryptography is NIST FIPS 140-2 or 140-3 certified by running the following SQL query:

```
SELECT DISTINCT name, algorithm_desc
FROM sys.symmetric_keys
WHERE key_algorithm NOT IN ('D3','A3')
ORDER BY name
```

If any items listed show an uncertified NIST FIPS 140-2 algorithm type, this is a finding.

Remediation:

Configure cryptographic functions to use NSA-approved cryptography compliant algorithms.
Use DOD code-signing certificates to create asymmetric keys stored in the database used to encrypt sensitive data stored in the database.
Run the following SQL script to create a certificate:

```
USE
CREATE CERTIFICATE
 ENCRYPTION BY PASSWORD = <'password'>
 FROM FILE = <'path/file_name'>
 WITH SUBJECT = 'name of person creating key',
 EXPIRY_DATE = '<'expiration date: yyyymmdd'>'
```

Run the following SQL script to create a symmetric key and assign an existing certificate:

```
USE
CREATE SYMMETRIC KEY <'key name'>
 WITH ALGORITHM = AES_256
 ENCRYPTION BY CERTIFICATE
```

For Transparent Data Encryption (TDE):

```
USE master;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '';
CREATE CERTIFICATE  . . .;
USE ;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE ;
ALTER DATABASE
SET ENCRYPTION ON;
```

Additional Information:

CCI-002450 Implement organization-defined types of cryptography for each specified cryptography use.

- NIST SP 800-53 Revision 4::SC-13
- NIST SP 800-53 Revision 5::SC-13 b

1.22 SQLD-22-003300 (Manual)

Profile Applicability:

- SEVERITY: CAT I

Description:

SQL Server must implement cryptographic mechanisms to prevent unauthorized modification or disclosure of organization-defined information at rest (to include, at a minimum, PII and classified information) on organization-defined information system components.

```
GROUP ID: V-271201
RULE ID: SV-271201r1109210
```

Rationale:

DBMSs handling data requiring "data at rest" protections must employ cryptographic mechanisms to prevent unauthorized disclosure and modification of the information at rest. These cryptographic mechanisms may be native to the DBMS or implemented via additional software or operating system/file system settings, as appropriate to the situation.

Selection of a cryptographic mechanism is based on the need to protect the integrity of organizational information. The strength of the mechanism is commensurate with the security category and/or classification of the information. Organizations have the flexibility to either encrypt all information on storage devices (i.e., full disk encryption) or encrypt specific data structures (e.g., files, records, or fields).

The decision whether and what to encrypt rests with the data owner and is also influenced by the physical measures taken to secure the equipment and media on which the information resides.

Satisfies: SRG-APP-000429-DB-000387

Audit:

Review the system documentation to determine whether the organization has defined the information at rest that is to be protected from disclosure or modification, which must include, at a minimum, PII and classified information.
If no information is identified as requiring such protection, this is not a finding.
Review the configuration of SQL Server, Windows, and additional software as relevant.
If full-disk encryption is required, and Windows or the storage system is not configured for this, this is a finding.
If database transparent data encryption (TDE) is called for, check whether it is enabled:

```
SELECT
DB_NAME(database_id) AS [Database Name], CASE encryption_state WHEN 0 THEN
'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
WHEN 6 THEN 'Protection change in progress'
END AS [Encryption State]
FROM sys.dm_database_encryption_keys
```

For each user database for which encryption is called for and it is marked Unencrypted, this is a finding.
If table/column encryption and/or a separation between those who own the data (and can view it) and those who manage the data (but should have no access) is required for PII or similar types of data, use Always Encrypted. The details for configuring Always Encrypted are located here: https://msdn.microsoft.com/en-us/library/mt163865.aspx.
Review the definitions and contents of the relevant tables/columns for the Always Encryption settings, if any of the information defined as requiring cryptographic protection is not encrypted this is a finding.

Remediation:

Where full-disk encryption is required, configure Windows and/or the storage system to provide this.
Where TDE is required, create a master key, obtain a certificate protected by the master key, create a database encryption key and protect it by the certificate, and then set the database to use encryption. For guidance from MSDN on how to do this, refer to: https://msdn.microsoft.com/en-us/library/bb934049.aspx.
Where table/column encryption is required, enable encryption on the tables/columns in question. For guidance from the Microsoft Developer Network on how to do this with Always Encrypted, refer to: https://msdn.microsoft.com/en-us/library/mt163865.aspx.

Additional Information:

CCI-002475 Implement cryptographic mechanisms to prevent unauthorized modification of organization-defined information at rest on organization-defined system components.

- NIST SP 800-53 Revision 4::SC-28 (1)
- NIST SP 800-53 Revision 5::SC-28 (1)

CCI-002476 Implement cryptographic mechanisms to prevent unauthorized disclosure of organization-defined information at rest on organization-defined system components.

- NIST SP 800-53 Revision 4::SC-28 (1)
- NIST SP 800-53 Revision 5::SC-28 (1)

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**STIG RULES**		
1.1	SQLD-22-000100 (Manual)	☐	☐
1.2	SQLD-22-000300 (Manual)	☐	☐
1.3	SQLD-22-000500 (Manual)	☐	☐
1.4	SQLD-22-000600 (Manual)	☐	☐
1.5	SQLD-22-000700 (Manual)	☐	☐
1.6	SQLD-22-001200 (Manual)	☐	☐
1.7	SQLD-22-001300 (Manual)	☐	☐
1.8	SQLD-22-001400 (Manual)	☐	☐
1.9	SQLD-22-001500 (Manual)	☐	☐
1.10	SQLD-22-001600 (Manual)	☐	☐
1.11	SQLD-22-001700 (Manual)	☐	☐
1.12	SQLD-22-001800 (Manual)	☐	☐
1.13	SQLD-22-001900 (Manual)	☐	☐
1.14	SQLD-22-002000 (Manual)	☐	☐
1.15	SQLD-22-002100 (Manual)	☐	☐
1.16	SQLD-22-002400 (Manual)	☐	☐
1.17	SQLD-22-002600 (Manual)	☐	☐
1.18	SQLD-22-002800 (Manual)	☐	☐
1.19	SQLD-22-002900 (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1.20	SQLD-22-003100 (Manual)	☐	☐
1.21	SQLD-22-003200 (Manual)	☐	☐
1.22	SQLD-22-003300 (Manual)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
Sep 2, 2025	1.0.0	Initial CIS Release