SQL Server Security Best Practices

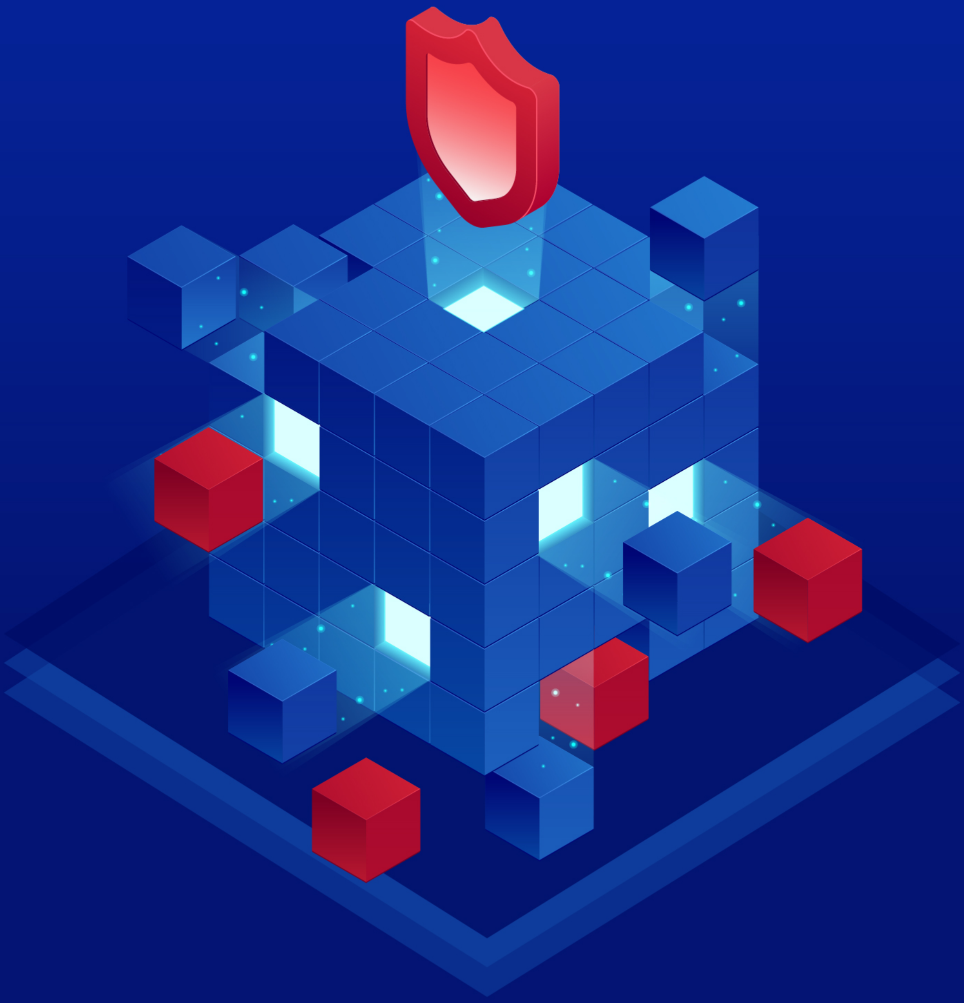

Table of Contents

Introduction

SQL Server is designed to be a secure database platform, but using the default settings leaves security gaps in the system. Moreover, SQL Server has many security features you should configure individually to improve security. Here are the top SQL Server security best practices you should follow.

Harden the Windows Server where SQL Server Operate

It is important to harden the Windows Server operating system before installing SQL Server on it. Otherwise, attackers who cannot defeat your SQL Server security measures can simply go around them: They will gain access to the OS and copy the data files to their own server, where they can breaking passwords and encryption at their leisure. To learn what to do, review these Windows Server hardening best practices.

Install Only the Required SQL Database Components

You should limit the installation to just the components needed for your database to perform its tasks. This approach reduces your attack surface area by eliminating components that could have security vulnerabilities. It also minimizes resource utilization by the database and simplifies administration by eliminating services and components needed to be managed.

Limit the Permissions of Service Accounts According to the Principle of Least Privilege

Each SQL Server service is configured to run under a specified Windows or Active Directory account. You should plan which accounts should be permitted to run which services based on the principle of least privilege, which states that each account should have the minimum permissions and system rights it needs to function.

It's a best practice to assign each service a separate account that is a member of a relevant security group. That way, even if the account for one service is compromised or damaged, other services will still operate normally. You can configure these permissions in Server Configuration Manager.

Here are the types of accounts you can use for SQL Server services:

- **Active Directory managed service account** — This is usually the best option, for two reasons. First, since you cannot use managed service accounts to log on to a server, they are more secure than domain user accounts. Second, you do not need to manually manage password resets for service accounts, as you must for regular domain user accounts.
- **Domain user account** — This is most common type of account used to run services. This account type is quite secure in a domain environment because it doesn't have administrator privileges.
- **Local user account** — This is a good choice for non-domain environments.
- **Local system account** — These accounts are highly privileged so you should avoid using them to run services.
- **Network service account** — This type of account has fewer privileges than a system account, but it does enable a service to have access to network resources, so you should avoid using it whenever possible.
- **Virtual service account** — A virtual service account is similar to an AD managed service account, but it is a type of local account that you can use to manage services without a domain. Technically, it is simply an instance of the builtin Network Service account with its own unique identifier. Virtual service accounts are great to use for SQL services.

Turn Off the SQL Server Browser Service

The SQL Server Browser service eliminates the need to assign port numbers to the instances. It enables SQL admins and authorized users to discover database instances over the network. However, this also makes it possible for attackers to gain knowledge of the available SQL Server resources. Therefore, when running a default instance of SQL Server, you should either turn off the SQL Server Browser service or configure another port for it to use to communicate.

Use Groups and Roles to Simplify Management of Effective Permissions

The effective permissions for a given account on a specific resource result from:

- Explicit permissions granted directly to the account on the resource.
- Permissions inherited from membership in a role or group.
- Permissions inherited from a parent resource.

In order to more easily understand and manage effective permissions, the best practice is to create containers, such as groups or roles, and assign those containers permissions to access resources. Then make accounts members of those groups of roles. That way, it's easier to understand the effective permissions for each account. Moreover, simply by putting a user in the right groups or roles, you can assign the correct permissions to new hires, modify permissions as a user's role changes and remove a user's permissions when they leave the organization.

Follow the Principle of Least Privilege when Assigning SQL Server Roles

Server roles provide an easy way to delegate administrative privileges, but you must assign these roles carefully. Here are all the default SQL server roles and the permissions they have:

- **Sysadmin** — Perform any activity on the SQL server
- **Serveradmin** — Configure SQL server settings and shut down the server
- **Securityadmin** — Manage logins, including their properties, passwords and permissions
- **Processadmin** — Terminate processes on the SQL Server instance
- **Setupadmin** — Add or remove linked servers and manage replication
- **Bulkadmin** — Execute the BULK INSERT statement
- **Diskadmin** — Manage disk files
- **Dbcreator** — Create, alter or drop any database
- **Public** — Every user is a member of this role. It does not have any permissions except to objects that are configured as public.

It is very important that you follow the principle of least privilege when assigning roles to users. For example, if a user only needs permissions to shut down the server and end processes, they should be assigned the serveradmin and processadmin roles; assigning them the sysadmin role would a big violation of principle of least privilege.

If no default server role matches your security requirements, you should create a custom role that does. You can do that using either TransactSQL or the Management console.

Use Strong Passwords for Database Administrators

Strong passwords are a must for all database administrator accounts to make them resistant to brute-force attacks. At a minimum, require these passwords to contain at least 10 characters, including uppercase and lowercase letters, numbers and specific symbols; however, passphrases are the best choice. These password best practices offer additional proven techniques for managing DB admin passwords properly.

Install SQL Server Updates Promptly

Both white-hat and malicious hackers are constantly discovering vulnerabilities and exploits in SQL Server. Microsoft releases several types of updates to fix them:

- **Hotfixes** (also known as QFE or Quick Fix Engineering) are released to solve customer problems ASAP. Due to the tight time constraints, hotfixes receive limited testing, so they should be applied only to systems that are known to have the specific issues they address.
- **Cumulative updates** (CUs) are periodic releases of a set of hotfixes that have had proper testing.
- **Service packs** (SPs) are a big collection of patches and fixes that have been properly tested and can easily be installed as a single package.

The simplest way to secure SQL Server is to keep it up to date. The easiest way to achieve this is to enable automatic updates from Microsoft. Larger organizations or those with strong change processes should apply updates only after testing them in test environments.

Use Appropriate Authentication Options

SQL server offers several options for authenticating users:

- **SQL Server Authentication mode** — Only Windows or AD users are permitted to connect to SQL Server.
- **Windows Authentication mode** — Only Windows or AD users are permitted to connect to SQL Server. SQL Server does not actually authenticate Windows; rather, it allows access based on an access token that has been issued to the user who logged in.
- **SQL Server and Windows Authentication mode** — Both Windows and SQL logins, such as the system administrator (sa) account, can access SQL Server. This mode is often called mixed authentication.

Best practices recommend using Windows Authentication to connect to SQL Server because it can leverage the Active Directory account, group and password policies. If you have to use SQL Server Authentication Mode to connect to SQL Server, do not use an sa account; instead, disable that account because it is the first account attackers will try to compromise in a brute-force attack.

Control Password Options for Logins

In a Windowsbased environment, administrators can enable policies for Windows users that control things such as password complexity and expiration. SQL Server can enforce similar options for SQL Server logins.

When you create a SQL Server login, you can specify the following options:

- **MUST_CHANGE** — SQL Server will prompt the user to change their password when they log on for the first time. Use this option when you create a new user or want a user to reset their password the next time they log on.
- **CHECK_POLICY** — The Windows password policies of the computer on which SQL Server is running will be enforced for the user. Always enable this setting.
- **CHECK_EXPIRATION** — The user will be required to reset their password regularly. Always enable this setting.

Be Diligent about Disabling and Deleting Logins

If a login will not be used for a long period of time, such as a month or longer, you should disable it and then reenable it later if needed. In these situations, it is better to disable the login rather than deleting it from the system. However, you should review your logins periodically and delete any that were disabled more than a year ago.

Use a Strong Database Backup Strategy

You must ensure that your database is backed up properly so the data can be recovered if a failure occurs. There are two types of backups: full backups and incremental backups. A full backup, as the name suggests, backs up the full database. After a full backup has been completed, SQL Server maintains a map of extents that can be backed up, and it backs up only those extents that have changed; this is called a differential backup. SQL Server does not clear the map of modified extents after a differential backup; it clears it only after a full backup.

Differential backups are much faster and occupy less disk space than full backups, so they are very useful for medium and large databases after the initial full backup; however, full backups should still be taken periodically. For small databases, the best practice is to simply use full backups every time.

For very large databases, you can consider using a file and filegroup backup strategy, which backs up only certain files or filegroups. This strategy can reduce the time required to perform backups. It can also speed recovery times, because if a single file is lost, you need to restore only that file or the filegroup that contains that file, instead of the whole database. However, be aware that managing filegroup backups can be complex.

Monitor Activity on Your SQL Server

Effective monitoring is critical to detecting, diagnosing and resolving problems. For example, you might identify longrunning queries that could turn out to be malicious. In particular, be sure to watch for the following:

- **Concurrency issues** — If multiple users attempt access the same data, some requests are blocked until others complete, which can result in deadlock, in which two operations are blocking one another.
- **Deviations from your baseline** — Record regular workloads metrics to establish a baseline so you know what they look like when the SQL database is operating normally. Then watch for deviations from that baseline. If you experience a significant change from your baseline without a clear cause, begin a security investigation as soon as possible. Remember that your environment is constantly evolving, with more data to store and more users working with the data, so you need to reestablish your baselines on a regular basis, such as once a year. Baselining will also help you to plan infrastructure upgrades in a timely way, instead of suddenly discovering you are over capacity.

Audit Access and Changes to SQL Server and Your Databases

You should always audit failed logins to SQL Server. Once you have enabled login auditing in SQL Server, the failed and successful login information will be written to the SQL Server error logs, which should be monitored regularly for suspicious activities.

Also be sure to monitor access, changes and deletions to database objects that contain restricted data. You should also track changes to SQL Server configurations and permissions, so you can block attacks and remediate mistakes before you suffer significant damage. This tracking can be done using SQL traces or third-party software like Netwrix Auditor for SQL Server.

Protect against SQL Injection Attacks

In a SQL injection attack, hackers enter SQL commands into a form field of an application front end. The application then passes that code to the database engine, which executes it; for example, it might create logins, delete data or change permissions.

The best way to protect against these attacks is to parameterize every query sent to the database. You should use properly configured stored procedures; they are much safer than direct dynamic SQL. Never pass string values in the front-end application, and be sure that all queries to the database are sanitized before being executed against the database.

Use Encryption Wisely

Encrypting data helps keep it secure even if unauthorized users gain access to it. There are several encryption features in SQL Server you can use to protect your data:

- **Transparent data encryption (TDE)** — TDE encrypts the physical files, both the data (mdf) and log (ldf) files. The encryption process is completely transparent to the applications accessing the database; the files are encrypted using either Advanced Encryption Standard (AES) or Triple DES and then decrypted as the information goes into memory.
- **Always encrypted** — This feature encrypts the data both at rest and in motion (keeps it encrypted in memory), so it protects the data from rogue administrators, backup thieves and man-in-the-middle attacks. Unlike TDE, it allows you to encrypt only certain columns, rather than the entire database. Enable this feature if you are running Microsoft SQL Server 2016 or later.
- **Column level encryption** — This feature encrypts only certain columns in a database, such as credit card details or Social Security numbers. Enable it if you have Microsoft SQL Server 2014 or earlier; in later versions, the “always encrypted” feature is superior.

About Netwrix

Netwrix is a software company that enables information security and governance professionals to reclaim control over sensitive, regulated and business-critical data, regardless of where it resides. Over 10,000 organizations worldwide rely on Netwrix solutions to secure sensitive data, realize the full business value of enterprise content, pass compliance audits with less effort and expense, and increase the productivity of IT teams and knowledge workers.

Founded in 2006, Netwrix has earned more than 150 industry awards and been named to both the Inc. 5000 and Deloitte Technology Fast 500 lists of the fastest growing companies in the U.S.

For more information, visit www.netwrix.com.

Corporate Headquarters:
300 Spectrum Center Drive, Suite 200, Irvine, CA 92618
Phone: 1-949-407-5125 **Toll-free:** 888-638-9749 **EMEA:** +44 (0) 203-588-3023

netwrix.com/social

Strengthen Server Security

with Netwrix Auditor for SQL Server

- Get notified about changes to critical database objects and permissions
- Spot suspicious access and modification attempts
- Identify sensitive data across SQL Server instances
- Investigate security incidents and troubleshoot issues faster

Download Free 20-Day Trial