SQL Server Security Best Practices (Part 2)

SQL Server Security is perhaps one of the most overlooked facets of database server maintenance. In Part 1 of this SQL Security Best Practices article, we further discussed physical security, operating system/application maintenance, disabling of unnecessary features, encryption, and data masking to further enhance security.

In Part 2 of this article series, we will continue to discuss additional security measures that can be taken to enhance SQL Server security.

SQL Server Authentication

Protection of SQL Server data depends upon the ability to authenticate access to specific data. To accomplish this in a Windows or Linux environment, SQL Server provides two options for database authentication: Windows/Linux Authentication and SQL Server and Windows/Linux Authentication (also known as Mixed-mode).

The type of SQL Server authentication is selected during SQL Server setup. During setup, you will be initially prompted to select one of these two authentication modes.

Windows or Linux Authentication Mode:

In this mode, when an installer logs in using his Windows or Linux account, SQL Server validates the account name and password using the Windows or Linux operating system. The Windows or Linux authentication uses Active Directory accounts for authentications. Hence, you can have a centralized policy control for strong password complexity, password expiration, account lockout and active directory groups in the active directory.

SQL Server does not prompt for the password, and does not perform the validation. As it turns out, Windows or Linux-based authentication is the default authentication mode, and is much more robustly secure than SQL Server Authentication (discussed below). Windows or Linux Authentication uses Kerberos security protocol to support the above-mentioned security features.

A connection made using Windows or Linux Authentication is sometimes called a trusted connection, because SQL Server trusts the credentials provided by the underlying Windows or Linux operating system.

SQL Server and Windows/Linux Authentication Mode (Mixed-mode):

When using SQL Server Authentication, logins are created in SQL Server proper and are not based on Windows or Linux user accounts. Both the user name and the password are created by using SQL Server and stored within SQL Server. Users connecting using SQL Server Authentication must provide their credentials (login and password) every time that they connect to SQL Server.

Since this mode does not avail itself of Windows or Linux Kerberos security protocol, it is considered to be inferior to Windows or Linux Authentication mode.

NOTE: It should be noted that the authentication mode can be changed subsequent to the initial installation decision to choose one of these two modes.

Passwords

If you are using SQL Server (Mixed-mode) authentication, it automatically creates a user login of SA (system administrator) with the sysadmin privileges and permissions. To increase the security of your SQL Server, the following should be performed:

1) Rename the SA login account to a different, more obscure, name;
2) Disable the account entirely, if you do not plan on using it;
3) For the SA (or renamed) account, select a complex password, consisting of lower/upper case letters, number, and punctuation symbols;
4) Do not allow applications to use the SA (or equivalently renamed) account in any of the application connection strings;

Any other user-based (lower-privileged) SQL Server accounts, should also use complex passwords, as noted above.

High-Privileged Operating System Accounts

SQL Services uses a Windows or Linux account to run its services. Typically one should not assign high-privileged, built-in accounts (or equivalents) such as Network Service or Local System to the various SQL Services, since this can increase the risk of nefarious database/server activity, should someone be able to log into these types of accounts.

Only assign the appropriate level of security-required accounts to SQL Services. Moreover, if not needed, any high-privileged operating system accounts on the server housing SQL server should be disabled as appropriate.

Restrict SQL Traffic

Database servers typically only have another server (or perhaps several) connecting to it. If this is the case, access to the server on the required database ports should be blocked everywhere else.

By only allowing SQL traffic to and from designated IP addresses, one can potentially prevent a nefarious user inside the firewall to not have access to the server. In certain cases, clients (and applications of clients) of SQL Server may need to connect directly to the database server itself. Restricting those SQL connections to the specific IP addresses (or at least IP class block or segment) that require it, should be implemented.

Because these are endpoints, they need to be secured properly, since infiltrated malware can scan and attack SQL servers. These IP restrictions can be managed via IPTables on Linux operating systems, and via the Windows firewall on Microsoft platforms. Alternatively (or additionally), a dedicated hardware firewall can achieve similar results.

SQL Server Patches (Service Packs)

Microsoft regularly releases SQL Server service packs and/or cumulative packs for fixing known issues, bugs, and security issues. It is extremely advisable to apply SQL Server patching on production instances of SQL Server. However, before being applied to production systems, it is highly advisable to apply and test these patches first in a test or development environment, to validate the ramifications of the changes in the patch.

Backups

When dealing with production instances of SQL Server, it is important to regularly backup the databases, to guard against database failure (due to corruption, disk array failure, power outages, disasters, etc).

Full database backups (on a regularly scheduled basis) and incremental backups (on a daily or running time basis) are all very advisable, and should be scheduled accordingly. In addition to guarding against failure, the retaining of backups can be very crucial in the event that a rollback of a database back to a particular date is necessary due to a particular business requirement.

Further, regarding SQL Server security, securing your backups is critical. Typically, database professionals do not consider all the requirements for securing database backups. Performing database backups is the process of creating a copy of the operational state, architecture and stored data of a database. Therefore, it is essential to protect it. This requires restriction of access to backup files and encrypting them properly to minimize the ability to read this offline data. Additionally, when it comes to securing backups, do not provide all people in an organization with the rights on the backup folder to create, view, modify and delete backup files.

Finally, to further secure database backups, it is imperative to store these backups in an off-site facility, as a further safeguard of preserving snapshots of the data. Depending on the organization and critical nature of the database data, backups from months to perhaps years, should be preserved and archived.

Auditing

Auditing is another key component/tool when it comes to SQL Server security. A designated database administrator or database security team should regularly review SQL Server auditing logs for failed logins. SQL Server provides a default login audit mechanism for reviewing all of the login accounts. These audit facilities record incoming requests by username and client IP

address, and any login failures can assist in discovering and eliminating any suspicious database activity. The following types of activity can show up in the SQL Server audit logs:

Extended events - Extended events is a lightweight performance monitoring system that enables users to collect data needed to monitor and troubleshoot problems in SQL Server.

SQL Trace - SQL Trace is SQL Server's built-in utility that monitors and records SQL Server database activity**.** This utility can display server activity, create filters that focus on the actions of users, applications, or workstations, and can filter at the SQL command level.

Change Data Capture - Change Data Capture (CDC) uses a SQL Server agent to record insert, update, and delete activity that applies to a specific table.

Triggers - Application-based SQL Server Triggers can be written specifically to populate a used-defined audit tables to store changes to existing records in specific tables.

SQL Server-Level Audit Specifications - A Server Audit Specification defines which Audit Action Groups will be audited for the entire server (or "instance"). Some audit action groups consist of server-level actions such as the creation of a table or modification of a server role and hence are only applicable to the server itself.

Additionally, hardware and/or software firewall logs (that is, external to SQL Server) should be regularly examined to monitor and detect any nefarious attempts at server penetration.

Conclusion

In Part 2 of this article series, we have reviewed additional methods of enhancing the security of SQL Server databases.

Judicious choice of authentication modes, SQL account naming, password choices, restricting SQL traffic, application of patch updates, assignment of security-friendly accounts to SQL Services, disabling of superfluous operating system accounts, backup strategies, and use of auditing, will further increase the security associated with SQL Server instances.