SQL Server Security Best Practices (Part 1)

SQL Server Security is perhaps one of the most overlooked facets of database server maintenance. Without taking the necessary precautions, an instance of SQL Server can be ripe for abuse and failure.

In the “A Primer on SQL Security” article, we discussed the logical implementation of Users, Groups, Roles, and Permissions, to enhance or limit database user security. In Part 1 of this SQL Security Best Practices article, we discuss a variety of important additional maintenance security topics.

SQL Server Physical Security

A very big part of SQL Server security is the physical security associated with the location of the SQL Server database. When we talk about physical security, we consider things such as the safety and access of the data center, and other physical aspects associated with the server that the database resides on. For example, data center access can be controlled by things like human guards, keys, smart card access, face recognition software, and fingerprint readers.

Data centers not only need to protect the servers where SQL Server resides, but other pieces of infrastructure which may include things like modems, hubs, routers, storage arrays, and physical firewall devices. Physical security requires dealing with hardware devices, software (firewalls, operating systems, layered products), and network infrastructure, and keeping them at arms-length from humans, hackers, and any potential natural disasters (floods, hurricanes, power outages, etc).

Some of the precautionary areas that those in charge of physical security must deal with include things such as 24x7 security guards, climate control monitoring (extreme hot or extreme cold can affect equipment adversely), fire detection and suppression systems, water leakage detection mechanisms, ensuring that necessary equipment is plugged into UPS’s (Uninterruptible Power Supplies), and the scheduling of both hardware and software preventative maintenance.

Operating System (and Application) Security Concerns

Next on the list of security issues is the operating system that SQL Server resides on. SQL Server supports both Microsoft Windows and several flavors of Linux. To protect your operating system from hackers, viruses, and bugs, which could affect the functioning, access to, and integrity of SQL Server, there are many precautions that can be taken.

First and foremost, operating system upgrades and (security) patches should always be applied whenever they become available. Before applying them to production-level machines, it may be prudent to apply them first to test or development environments, and allow them to run for a

period of time, to ensure that the upgrades/patches are stable and are not problematic. Moreover, when an operating system goes end-of-life, it should always be replaced with a supported operating system version.

Next, it is good practice to disable public internet access on your servers, to mitigate outside hacking interference. This can be followed by implementing robust firewalls on your operating system. By definition of firewall rules, one should restrict access to/from database servers that run on the operating system, and limit database access to permissioned applications only.

Further, it is extremely good practice to remove unnecessary (unused) applications from your operating systems, including unneeded operating system features (for example, Email or FTP) that could potentially lend itself to a security threat.

Finally, one can make use of SQL Server's Extended Protection for Authentication option to prevent an authentication relay attack using the service binding and channel binding. To enable extended protection, go to the SQL Server Configuration Manager, expand the screen, right-click on Protocols and then go to Advanced, Extended protection. Note, by default, this is turned off.

Enabling/Disabling/Changing Ports

Another important security measure one can take is to close all unnecessary ports in your operating system via your firewall, and open up select ports, as necessary. For example, by default, SQL Server runs on port 1433. Therefore, you can allow TCP port 1433 (and 3389 for remote server access) if no other application runs on the server. Similarly, the analysis service uses default port 2383 as a standard port. Review of all other SQL Server layered product ports should also be reviewed and opened, on an as-needed basis only.

Further to the port discussion, it may be prudent to change SQL Server's default listening port (1433) to another port number. By not changing it, this well-documented port number can be an invitation to hackers to infiltrate a SQL Server instance. Therefore, you should use a non-default port to solidify your SQL Server security. You can modify this very easily using the SQL Server Configuration Manager tool.

Eliminate Superfluous SQL Server Add-on Features.

The SQL Server consists of database engine features that provide additional functionality that may not be utilized by every installation. Some of these components may be a potential target to gain access to SQL Server by hackers. Therefore, it is good common practice to disable the add-on components and features in SQL Server that are not going to be used, as this will limit the chances of any potential hacker attack. Some of these more obvious components include the following:

OLE Automation Procedures - they enable SQL Server to leverage OLE (Object Linking and Embedding) to interact with other COM objects. Data security-wise, this increases the attack surface.

Database Mail XP's - enables the Database Mail extended stored procedures in the msdb database.

Scan for startup procs - an option to scan for automatic execution of stored procedures at Microsoft SQL Server startup time.

Common language runtime (CLR) integration feature - The CLR provides various functions and services required for program execution, including just-in-time (JIT) compilation, allocating and managing memory, enforcing type safety, exception handling, thread management, and security.

Windows (not Linux) process spawned by xp_cmdshell - Has the same security rights as the SQL Server service account, and spawns a Windows command shell and passes in a string for execution. Any output is returned as rows of text.

Cross-database Ownership Chaining (also known as cross-database chaining) - A security feature of SQL Server that allows users of databases access to other databases besides the one they are currently using. Again, if you do not need a particular SQL Server feature, disabled it if you can.

Encryption

A huge area of security for SQL Server is encryption. SQL Server supports several different encryption mechanisms to protect sensitive data in a database. The different encryption options available are as follows:

Always Encrypted Option - The Always Encrypted option helps to encrypt sensitive data inside client applications. The always encrypted-enabled driver automatically encrypts and decrypts sensitive data in the client applications. The encryption keys are never revealed to the SQL Server database engine. It does an excellent job of protecting confidential data.

Transparent Data Encryption (TDE) - TDE offers encryption at the file level. TDE solves the problem of protecting data at rest, and encrypting databases both on the hard drive and consequently on backup media. It does not protect data in transit or data in use. It helps to secure the data files, log files and backup files.

Column-Level Encryption - Column-level encryption helps to encrypt specific column data; for example, credit card numbers, bank account numbers, and social security numbers.

Data Masking

Data masking is a technique used to create a version of data that looks structurally similar to the original but hides (masks) sensitive information. The version with the masked information can then be used for a variety of purposes, such as offline reporting, user training or software testing.

Specifically, there are 2 types of data masking supported by SQL Server:

Static Data Masking - Static Data Masking is designed to help organizations create a sanitized copy of their databases where all sensitive information has been altered in a way that makes the copy shareable with non-production users. With Static Data Masking, the user configures how masking operates for each column selected inside the database. Static Data Masking will then replace data in the database copy with new, masked data generated according to that configuration. Original data cannot be unmasked from the masked copy. Static Data Masking performs an irreversible operation.

Dynamic Data Masking - Dynamic data masking helps to limit sensitive data exposure to non-privileged users. It can be used to greatly simplify the design and coding of security in your application. Dynamic data masking helps prevent unauthorized access to sensitive data by enabling customers to specify how much sensitive data to reveal with minimal impact on the application layer.

Conclusion

Database security is an extremely important part of database design, operations, and maintenance. It includes things such as physical security, operating system/application maintenance, disabling of superfluous features, port maintenance, encryption, and data masking. Collectively, and if addressed properly, a SQL Server database will remain free from attack, will remain operationally sound, and will ensure database integrity.