IDERA® | SQL Secure

MAINTAIN THE SECURITY OF YOUR SQL SERVERS

WITH SQL SECURE

BY GREG ROBIDOUX

INTRODUCTION

Wow, SQL Server database security is difficult! There are so many options it is difficult to make the right decisions to ensure your database environment is secure. There are several security settings you need to think about during installation. These include settings in Windows, network security settings, firewalls, ports, on-premises, cloud, hybrid, exchanging data with business partners, and storage of backups. Beyond that, they also include encrypting databases, encrypted network traffic, secure storage of import and export files, use of application programming interfaces, application security, and permissions based on job roles. As you can see, security is difficult. Therefore, it is so difficult to get security right. Either not enough is in place causing potential security holes or too much is in place where it often makes simple things quite difficult to get implemented.

So where do you begin? As you can tell, there are many levels of securing SQL Server from both within and outside the database engine. For this document, we will focus on the internal side of SQL Server, how you can best manage what permissions you granted, and which users have these permissions. As with most things, security is an ever-changing entity. This is the reason you need to stay on top of changes that are made to make sure you catch and plug any potential security holes as soon as possible.

SQL SERVER'S LOGINS

SQL Server offers two methods of creating a login: Windows Authentication and SQL Server Authentication. Windows authentication allows you to use your Domain level access credentials to manage what users can access SQL Server. This can be individual logins or through Windows Groups. The advantage of this approach is that you can maintain permissions at the domain level for users. For SQL Server authentication, you can control all access levels from within SQL Server. There are some options for enforcing password policies using this method, but it is not as robust or centralized as using Windows authentication. So we have three potential access methods to SQL Server, an individual Windows account, a Windows group, and a SQL Server login. In addition, some applications will use a fixed SQL Server login to connect to the database and then use data from some database table or tables that determine another level of security. This therefore takes away any control of logins at the SQL Server or Windows level, but also adds another level of security auditing that should be in place. As you can see, just the initial access to SQL Server has many options.

SQL SERVER'S SERVER ROLES

When you create a login in SQL Server, it has very limited permissions. SQL Server offers server level roles that provide permissions to manage various aspects in SQL Server. Each login that you created can be in none, many, or all of these roles. Here is a list of these roles:

- bulkadmin
- dbcreator
- diskadmin
- processadmin
- securityadmin
- serveradmin
- setupadmin
- sysadmin

Each of these roles provides elevated privileges to do something at a SQL Server instance level, with the highest level being sysadmin, which gives the login full permission to do whatever they want in the SQL Server instance. We often use this as the default go-to when we cannot figure out security. Just give someone all permissions and they will not have any security related obstacles. Although this works, it is not the best option. Also, sometimes you can do this as a temporary measure. But these things often go unchecked as the elevated permissions persist.

SQL SERVER'S DATABASE ROLES

For a login to gain access to a database, a database user has to be created that connects to this login. There is also a user in each database named guest that provides some level of access for all logins, but you disable this user except in the system databases. There is also a role named public, which gives limited access to certain system objects.

When you tie a login to a database user, there is the ability to assign database level roles, such as:

- db_accessadmin
- db_backupoperator
- db_datareader
- db_datawriter
- db_ddladmin
- db_denydatareader
- db_denydatawriter
- db_owner
- db_securityadmin
- public

The db_owner role has the highest level of permissions and therefore any user with this role can do anything they want within this database. This is like the sysadmin server role mentioned above. When security problems get in the way, the easy option is to just make the user a db_owner. This is not the best way of managing security, but this occurs way too often.

GRANULAR PERMISSIONS

Outside server roles and database roles, you can assign or deny specific permissions using these SQL commands: GRANT, DENY and REVOKE. You can grant permissions to tables, views, stored procedures, functions, and columns. You can grant permissions to individual database users. Or you can create user defined database roles where you grant the permissions to the database role and then you assign database users to this new role. To further complicate this, you can also deny access to specific objects or revoke access.

As you can see, there are lots and lots of layers of security. Therefore, security can become very complex and why, sometimes, just assigning db_owner or sysadmin roles is the easy way out. But on the flip side, a complex security model can also create security holes that lend itself to exploitation. So there has to be a balance and a way to check and audit security access within SQL Server.

HOW TO FIND WHO HAS WHAT PERMISSION

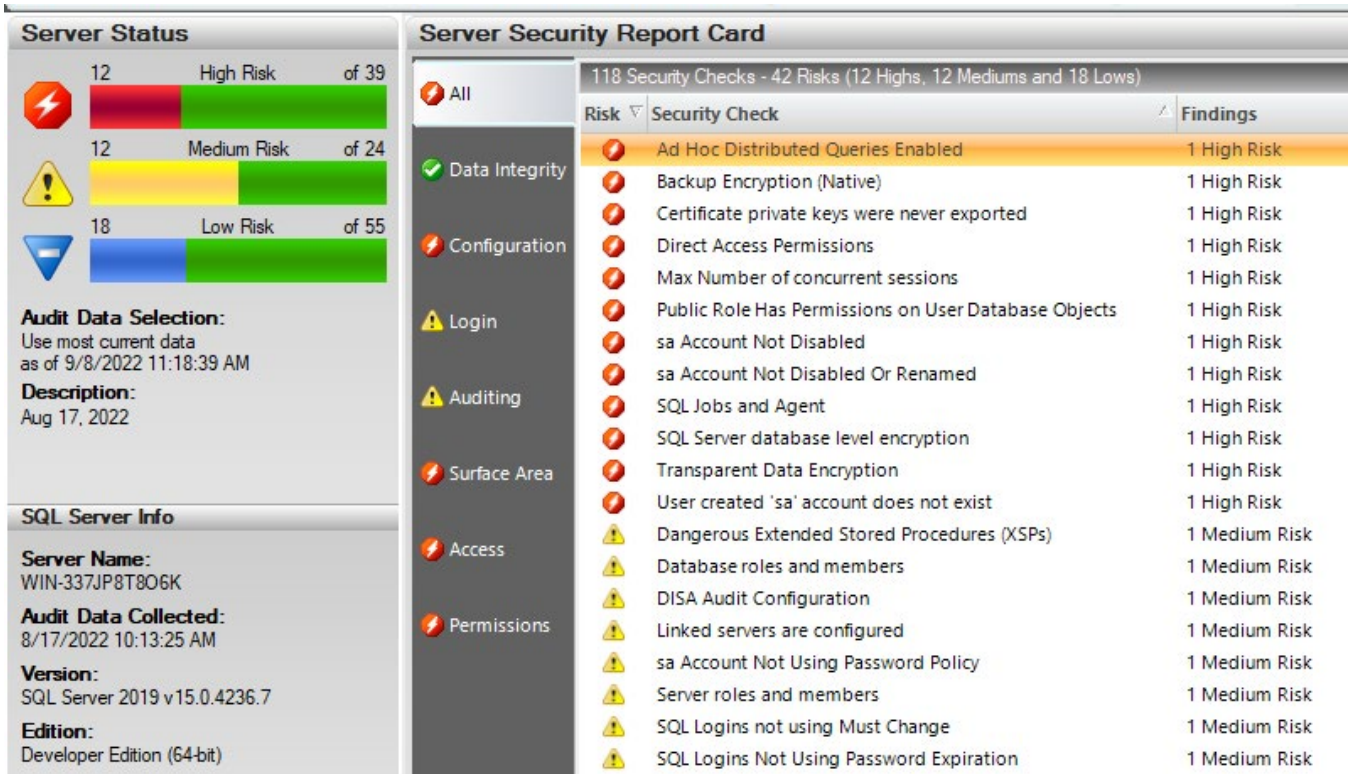
To determine who has permissions, it is easy to navigate in SQL Server Management Studio (SSMS) and review various things such as logins, server roles, database uses, database roles and even drilling into specific permissions that were granted. The problem with this approach is that you have to go to multiple places to get the overall picture of what a login can do within SQL Server or specific databases. Besides using SSMS, you could also use system functions, dynamic management views, and system views to build queries to capture and gather the data. But for each scenario you are trying to gather data, you would need various queries to gather the data and still need a mechanism to combine all the results to make informed decisions. These are great for having a snapshot in time, but how can you tell what new permissions you granted or revoked?

A product like SQL Secure makes security management so much easier. You can identify what permissions a login or database user has with a couple of clicks of the mouse. This is a major benefit of its graphical environment and the ability to drill in and navigate the security levels.

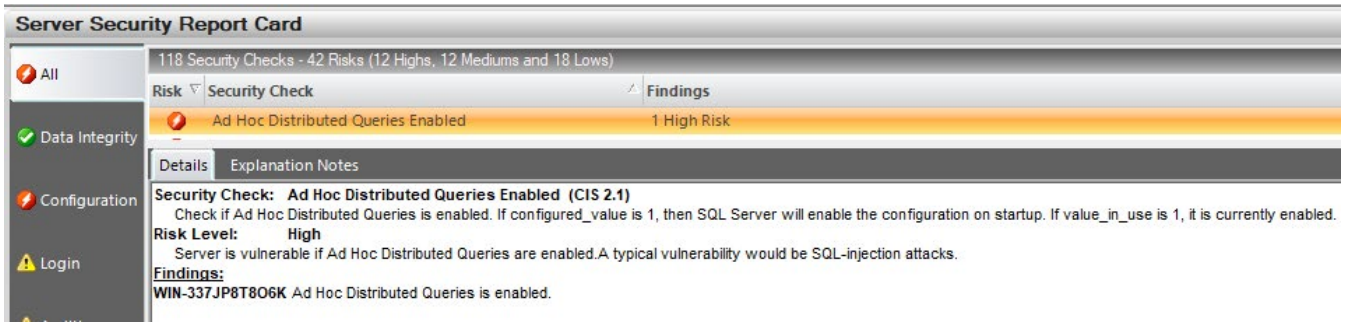
SQL SECURE

It only takes a few minutes to install to get started with SQL Secure. After the install of SQL Secure, you register the servers you want to manage and begin the initial data collection.

After the initial data collection, here is a view of the “Server Security Report Card” identifying security issues that you should review. You can segment this data with several views, such as Data Integrity, Configuration, Login, Auditing, Surface Area, Access, and Permissions.

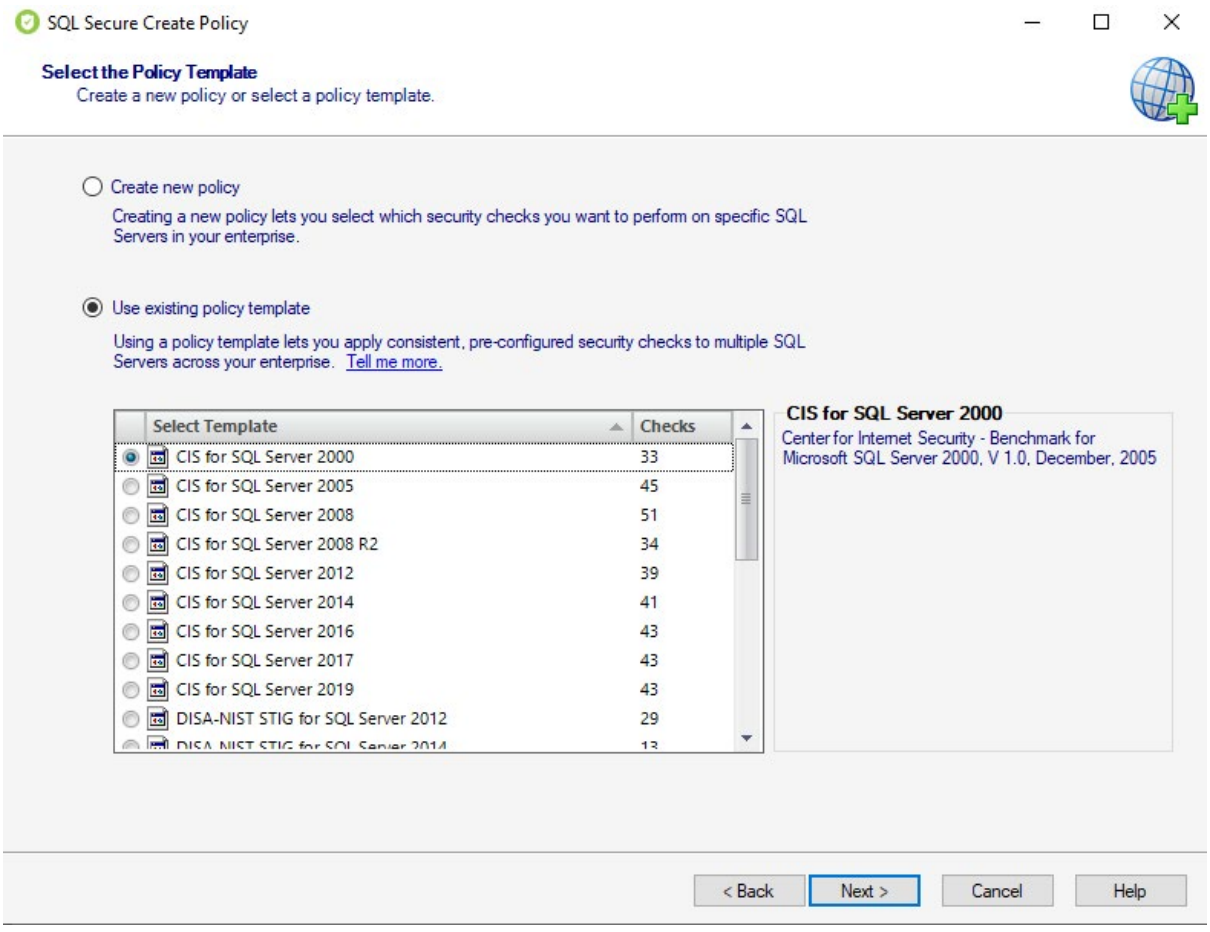


In addition, if you select one of the security checks such as “Ad Hoc Distributed Queries Enabled”, then you will get more details about this issue.



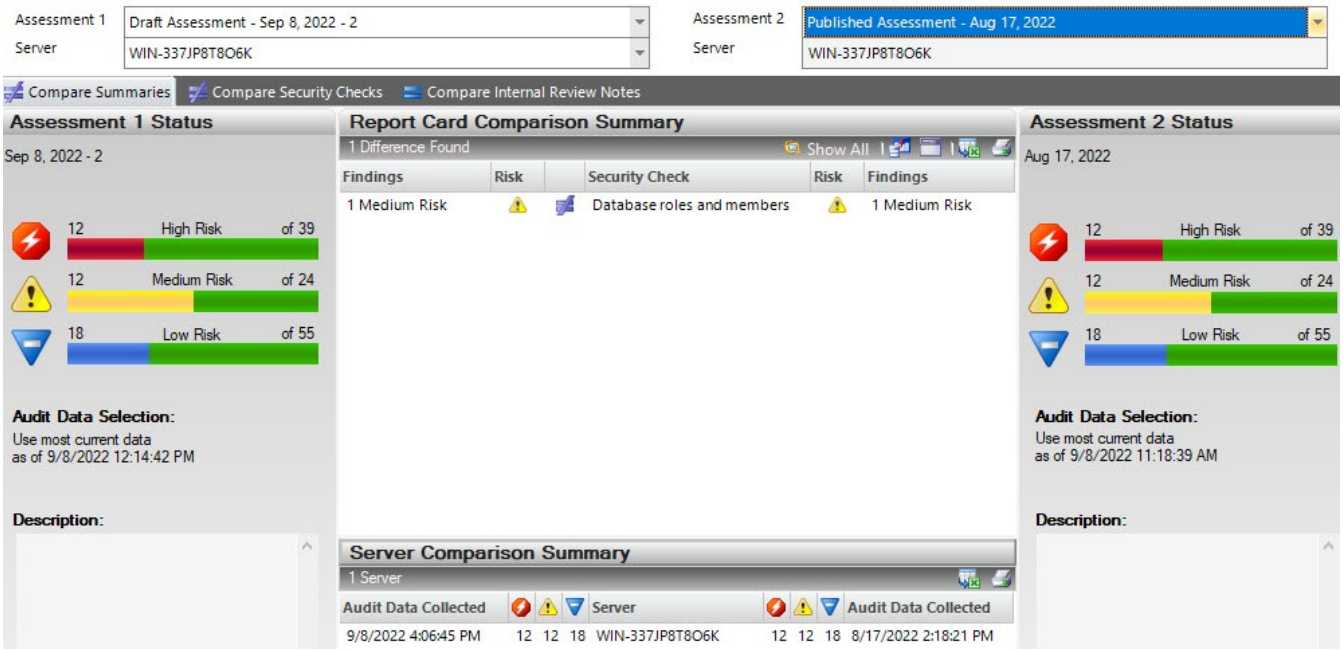
BUILT-IN SECURITY POLICIES

SQL Secure also has several built-in policies for regulations, such as CIS, DISA, HIPAA, PCI, and SOX. This allows you to use these predefined policies to ensure that you address any security issue related to the policy when auditing security. Each policy has several checks associated with the policy, as shown below.



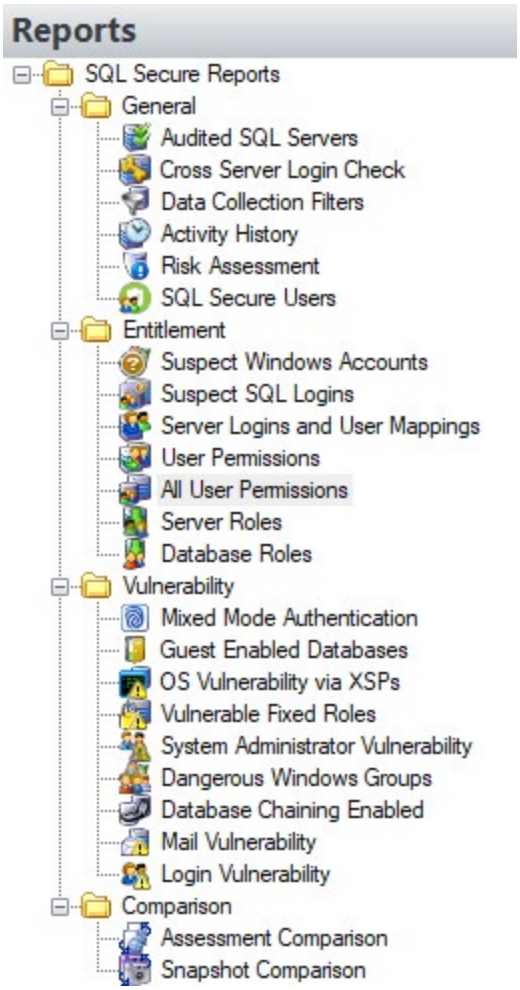
COMPARE POINTS IN TIME

One of the prominent features of SQL Secure is the ability to compare two points in time. This allows you to determine what security changes have changed between snapshots.



REPORTING

SQL Secure provides over 20 built-in reports to get security information at the server level, database level, and user level. These reports allow you to compare different points in time to find security changes that have occurred, and potential vulnerability issues that need to be addressed. You can also create high-level reports to get an overall view of all audited SQL Server instances in your environment.



SQL SECURE IS CONFIGURABLE

Although SQL Secure offers plenty out of the box settings, these settings are configurable. So if there are certain checks you want to ignore, you can disable that check. Also, you can set your own risk level for each of these items to ensure you are focusing on the items that you deem most important are always at the forefront of your audited security data.

GETTING STARTED WITH SQL SECURE

If you are looking for a better way to audit SQL Server security, then consider SQL Secure. The installation only takes a few minutes and you can assess the state of your security setup for your SQL Servers. The ability to identify changes and issues allows you to spend time on addressing the top security priorities for your environment. This is better than taking your time trying to determine what the issues are before you can even address them.

Start for Free

The ability to identify changes and issues allows you to spend time on addressing the top security priorities for your environment. This is better than taking your time trying to determine what the issues are before you can even address them.

ABOUT THE AUTHOR

Greg Robidoux is the President and founder of Edgewood Solutions, a technology services company delivering services and solutions for Microsoft SQL Server. Greg is also a co-founder of MSSQLTips.com. He has been working with Microsoft SQL Server since 1999 and has authored many articles and delivered several presentations online and at local and national Microsoft SQL Server events.