

# SQL SERVER SECURITY PRACTICES

BY PINAL DAVE

# INTRODUCTION

There are a number of examples where data theft has brought businesses to a halt or resulted in bad press that will leave a tarnished image. For example, in 2017, Equifax disclosed that the personal information for over 145 million people was exposed, including social security numbers and credit card numbers. In 2018, Facebook suffered at least two separate data privacy incidents, affecting over 90 million user accounts. There are many more horror stories from banks, financial institutions, healthcare providers, and retail organizations in the recent past and they get even scarier as we move into this digital only world. As organizations start looking at security as a first-class citizen and work on it proactively, these incidents will still keep happening and we will always be playing the catch-up game.

If you are in the job of consulting or if you are in sales-oriented jobs, the likelihood of engaging with new customers on a daily basis is inevitable – and I am one of them. This is something I do a lot in my day job and sometimes I have to deal with some really rude security procedures at the clients' place of business. I always wonder why we get grilled, frisked, and put under so much scrutiny. In one of my recent trips while hopping via airports, the security guard stopped me and told me to write my phone number and name on my baggage tag. I first thought about it and then out of curiosity asked him politely (in frustration), "Why are you doing this and forcing me? I am a frequent traveler and have never been asked to do this in my 8+ years of travel anywhere. This is just a domestic trip, and please relax, because I have a flight to catch."

The guard smiled and politely said, "All the more reason, sir – you now need to write it every time you plan to travel." Seeing me getting a little impatient, he added, "Sir, since you are in a hurry and you are distributing your laptop, bag, cellphone, etc. in different bins, it is quite possible you might miss picking up your materials because of the rush. So if you had written down the phone number, we can reach you individually rather than making a public announcement in this airport." He added, "Sir, I am just doing my job."

This incident changed the way I see security personnel. They are doing a job and we need to just respect them. A deep introspection got me into thinking about security in a different way. What about security in the software we develop? Why is security not a consideration during the design phase itself? Securing your data is one of the most important aspects when it comes to keeping your trade secrets from prying competition.

In this paper let me share some of the common security practices that I propose to my customers on a daily basis and explain how one needs to start implementing security measures within their deployment of SQL Server. I am sure if you are a DBA, these are like checklists you don't want to miss when working with a database engine like SQL Server. We will cover details around the logins and authentication area.

# AUDIT LOGINS

Keep the setting of “Both failed and successful logins” for login auditing. It is critical we track or at least keep an eye on who is accessing the system or attempting to access the system. By default, these settings are not to be tampered with. But I have seen sometimes inexperienced DBAs do turn this setting off. The default is to Audit the “Failed logins only”, but in mission critical systems it might be a better practice to audit both failed and successful logins. See Figure 1 for the login auditing options.

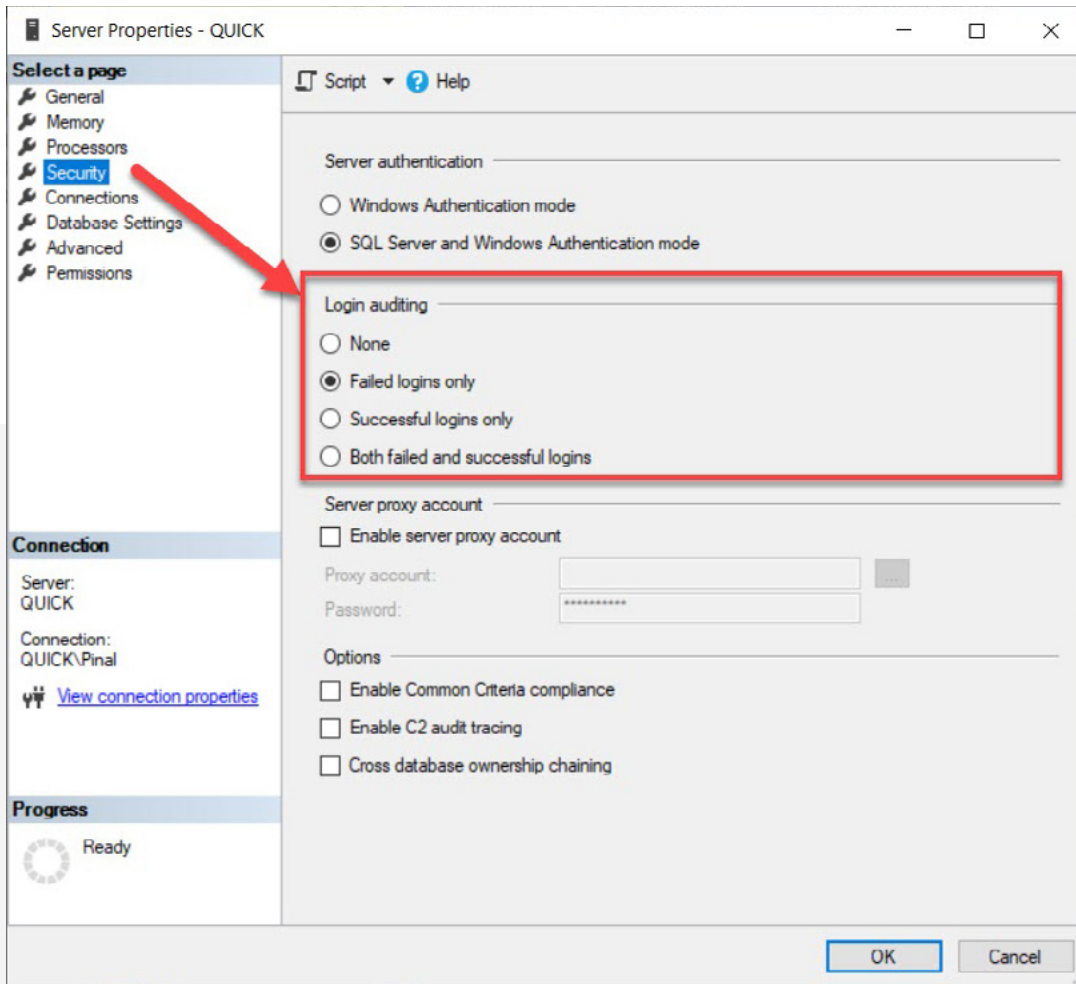


Figure 1 Login auditing options in server properties

The idea is to make sure there are no Denial-of-Service (DOS) attacks made on the system and we are tracking the connections coming into the server. From SQL Server 2008, it is also possible to use the new Audit feature to log connection attempts. If you plan to use this, as a best practice you should use a secured location for audit data files.

# AUTHENTICATION AND DISABLE SA

Try to use Windows Authentication as much as possible. Usage of mixed mode authentication is recommended for legacy applications and non-Windows users. I recommend this religiously because when multiple users use the same login, it becomes difficult to know which “NT User” actually did an operation on the server. I recommend this for SQL Server Administration too, instead of using the default SYSADMIN SA account. Try not to share the SA account across multiple users. We can change the authentication type on the Server Properties > Security > Server Authentication; this setting can be found as shown on the previous page in Figure 1.

If possible, make sure to disable the SA login inside SQL Server. This is because this user account can be easily guessed by anyone working with SQL Server. Ideally, rename or disable this account and use some other account to administer your SQL Server instance. The following command can help you DISABLE / ENABLE the SA user account.

```
/* Disable SA Login */  
ALTER LOGIN [sa] DISABLE  
GO  
/* Enable SA Login */  
ALTER LOGIN [sa] ENABLE  
GO
```

As we move to the next recommendation, if you have to use mixed mode authentication then I highly recommend enabling the password expiration and password complexity values for all the logins created on the server for all the authentication types.

If there are multiple users administering a given SQL Server instance, then it is recommended to track password changes for SQL Server logins. Enabling auditing on password changes for SQL Server logins allows you to investigate when and by whom the passwords of specific logins have changed. Consider creating a server audit for the action 'LOGIN\_CHANGE\_PASSWORD\_GROUP'. See Figure 2 for setting up the server audit specification.

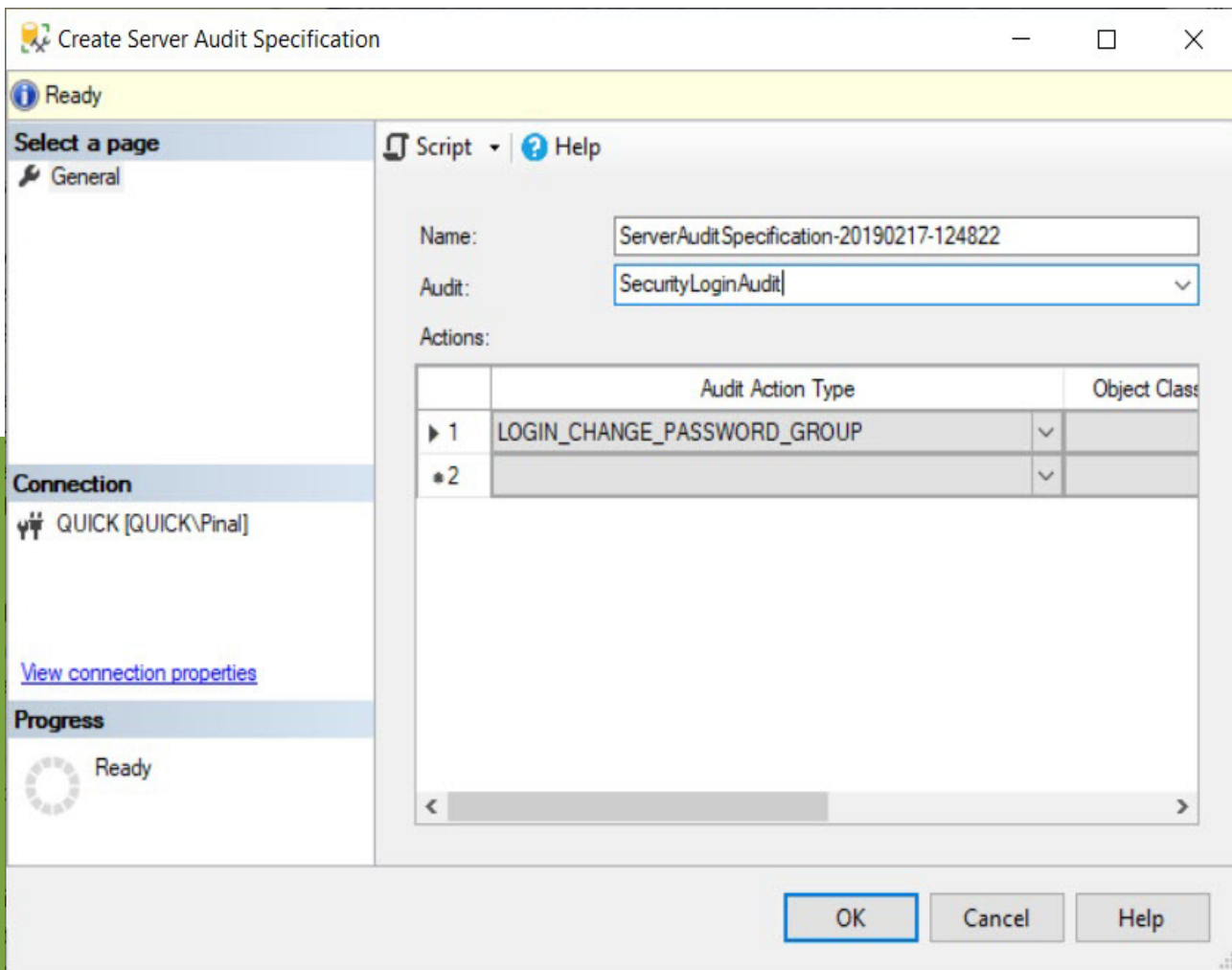


Figure 2 Create server audit specifications

## VIEW DATABASE PERMISSIONS

This is a server level permission which is often less appreciated. If this permission is given then the login can see all the metadata for all databases irrespective of the login owning the database or not. By controlling this permission, we effectively restrict the amount of data a login can see in the SYS.DATABASES, SYS.SYSDATABASES views and SP\_HELPDB calls. To limit visibility to database metadata, deny a login the VIEW ANY DATABASE permission.

```
/* REVOKE VIEW DEFINITION PERMISSION */  
USE master  
GO  
REVOKE VIEW ANY DEFINITION TO Pinal
```

After this permission is denied, a login can see only metadata for master, tempdb and databases that the login owns. By default the public role is granted this permission; don't use DENY on the "public" role, otherwise no one will be able to list database objects.

## SYSADMINS HAVE SUPER POWERS

From SQL Server 2008 onwards, the Windows built-in administrators group is not part of the SQL Server SYSADMIN fixed server role. This is a great "secure by default" strategy and best practice in my opinion, and I am glad this was done. Now during the installation, we are requested to explicitly mention / grant the logins who will have SYSADMIN privileges. Be very cautious in giving SYSADMIN privileges to logins. The below script lets you find out who are the Sysadmins in your server today.

```
SELECT  
prin2.name AS [Name],  
prin2.is_disabled,  
prin2.type_desc AS [Login Type],  
prin2.create_date  
FROM sys.server_principals prin  
INNER JOIN sys.server_role_members mem  
ON prin.principal_id = mem.role_principal_id  
INNER JOIN sys.server_principals prin2  
ON prin2.principal_id = mem.member_principal_id  
WHERE prin.type = 'R' and prin.name = N'sysadmin'
```

Sysadmins as the name suggests have all the permissions to do anything inside SQL Server. If you have to give permission, then provision the logins in the SYSADMIN fixed server role directly; don't use a common SQL Authentication with the SYSADMIN rights on the server – this helps while auditing. As shown in Figure 3, on my local server you can see I have hardly given rights to anyone apart from my own Login.

	Name	is_disabled	Login Type	create_date
1	sa	0	SQL_LOGIN	2003-04-08 09:10:35.460
2	QUICK\Pinal	0	WINDOWS_LOGIN	2018-06-14 20:39:46.430
3	NT SERVICE\SQLWriter	0	WINDOWS_LOGIN	2018-06-14 20:39:46.440
4	NT SERVICE\Winmgmt	0	WINDOWS_LOGIN	2018-06-14 20:39:46.443
5	NT Service\MSSQLSERVER	0	WINDOWS_LOGIN	2018-06-14 20:39:46.447
6	NT SERVICE\SQLSERVERAGENT	0	WINDOWS_LOGIN	2018-06-14 20:39:46.930

Figure 3 Sysadmin and server permissions

It's highly recommended to give SYSADMIN privileges to the lowest number of accounts to maximize security. And do not assign any elevated rights to a user if that user does not need it.



# HOW TO PROTECT FROM SYSADMINS

As we can see, the Sysadmins are really powerful, and when using SYSADMIN privileges, note that no permission check is done. So use this with caution and give access to only limited and, if possible, the least number of users.

Organizations have a need to protect their data and I have seen Management asking if there is a way to protect data even from DBAs. The first question I ask is, "What rights have you given them?" The standard answer I get is, "They are DBAs and hence we have given them the administrative rights of Sysadmins." If you look at the examples and analogy I gave before, this is surely dangerous, right? So the immediate question is, what should be the rights for a DBA?

These are tough questions but with every release of SQL Server there have been enhancements that we don't want to miss. Let me build the set of permissions for you leading to a feature introduced in SQL Server 2014 and later versions.

Let us first start creating our super user SQLAuthority.

```
USE [master]
GO
CREATE LOGIN SQLAuthority WITH PASSWORD=N'pass@word1',
DEFAULT_DATABASE=[master],
CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
```

The idea now is NOT to give SYSADMIN rights but to give rights that still enable the DBA to do their day-to-day activities without any problem. We are going to give this user CONTROL SERVER rights.

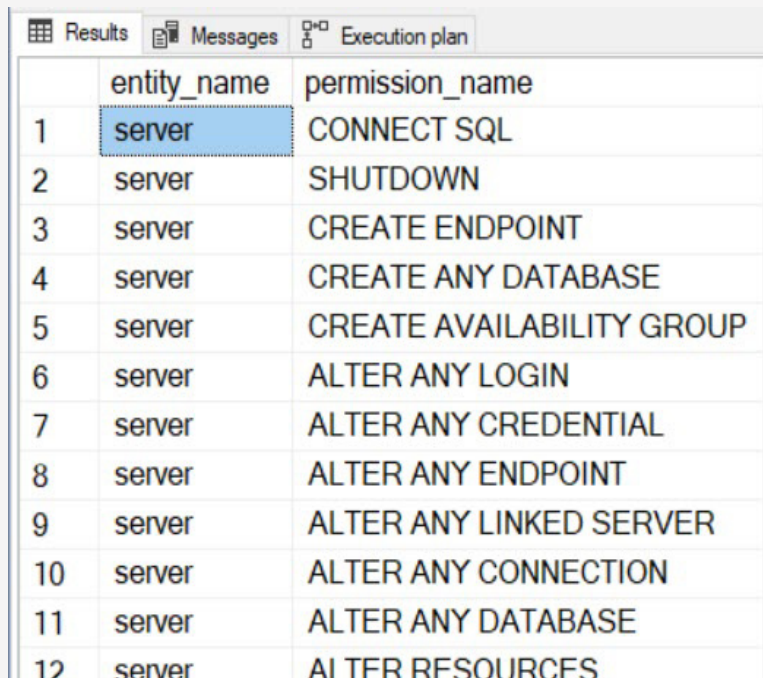
```
GRANT CONTROL SERVER TO SQLAuthority;
GO
```



Let us query the permissions DMV to find out the effective permission SQLAuthority has after this GRANT. We will be using the following command:

```
SELECT entity_name, permission_name
FROM sys.fn_my_permissions(NULL, NULL);
```

Which will generate the permissions list shown in Figure 4.



	entity_name	permission_name
1	server	CONNECT SQL
2	server	SHUTDOWN
3	server	CREATE ENDPOINT
4	server	CREATE ANY DATABASE
5	server	CREATE AVAILABILITY GROUP
6	server	ALTER ANY LOGIN
7	server	ALTER ANY CREDENTIAL
8	server	ALTER ANY ENDPOINT
9	server	ALTER ANY LINKED SERVER
10	server	ALTER ANY CONNECTION
11	server	ALTER ANY DATABASE
12	server	ALTER RESOURCES

Figure 4 View enabled server permissions

This gives us explicitly 34 specific server level permissions. This still isn't that restrictive. If SQLAuthority tries to elevate himself to SYSADMIN rights using the following command:

```
ALTER SERVER ROLE sysadmin
ADD MEMBER SQLAuthority
```

He will be presented with an error stating no permissions.

```
Msg 15151, Level 16, State 1, Line 11
Cannot alter the server role 'sysadmin', because it does not exist
or you do not have permission.
```

But from the list of permissions granted, we can find that IMPERSONATE ANY LOGIN is available for this user. If that is the case, then SQLAuthority can impersonate ANY user in the system and gain access to the server. Typically, he would do it this way:

```
EXECUTE AS LOGIN = 'sa';

ALTER SERVER ROLE sysadmin
ADD MEMBER SQLAuthority

REVERT;
```

That is it, now SQLAuthority has gained SYSADMIN rights into the system, and this seems to be a problem prior to SQL Server 2012. With SQL Server 2012, there was an interesting addition to DENY impersonation rights to a user, and even if they have CONTROL SERVER rights, SQL Server will honor the DENY permission. In our example, we can explicitly deny Impersonation to SQLAuthority:

```
DENY IMPERSONATE ON LOGIN::sa TO [SQLAuthority]
```

This is great, and now SQLAuthority cannot impersonate the SA account. But do you see the problem here? We have not solved the problem yet. We have explicitly denied impersonation to this account, but there can be other users inside the system who have SYSADMIN rights and our CONTROL SERVER rights will give SQLAuthority the opportunity to impersonate one of them. It is humanly impossible to DENY such rights to each and every user with SYSADMIN rights for SQLAuthority.

SQL Server comes with a permission called IMPERSONATE ANY LOGIN rights. This is powerful for the scenario we questioned. After giving the CONTROL SERVER rights, we can issue the following command in SQL Server:

```
DENY IMPERSONATE ANY LOGIN TO SQLAuthority
GO
```

After this command the effective permission for SQLAuthority is the same as CONTROL SERVER but he has been explicitly denied IMPERSONATION rights. This is so powerful because as the management wants, we can give close to SYSADMIN rights to a user to perform their daily duties yet not give them the ability to elevate themselves easily.

If we run the command for effective permissions, we can see the list now has one fewer row at 33. The DMF for this is:

```
SELECT entity_name, permission_name
FROM sys.fn_my_permissions(NULL, NULL);
```

Also worth a mention, the other permissions added in SQL Server are CONNECT ANY DATABASE and SELECT ALL USER SECURABLES.

```
use [master]
GO
GRANT CONNECT ANY DATABASE TO [SQLAuthority]
GO
GRANT SELECT ALL USER SECURABLES TO [SQLAuthority]
GO
```

Think of these rights when you want to give permission to an Auditor who visits your organization for compliance and wants access to the Server while they can only select and not update anything. These rights are powerful for those scenarios.

## CONCLUSION

Security is a core area and non-negotiable when it comes to mission critical applications. We know how companies have lost business and lost respect in the industry because of lack of security measures. With every release of SQL Server, there are tons of additions that get added as part of the platform. It is important to start using them in our application deployment design so that loopholes can be avoided. In this paper we discussed some of the general best practices as they evolved with SQL Server versions and we reviewed specific SYSADMIN privileges and how we can secure our server using some of the permissions added in SQL Server.

## ABOUT THE AUTHOR

Pinal Dave is a SQL Server Performance Tuning Expert and an independent consultant. He has authored 11 SQL Server database books, 21 Pluralsight courses and has written over 4500 articles on the database technology on his blog at [sqlauthority.com](http://sqlauthority.com). Along with 16+ years of hands-on experience he holds a Masters of Science degree and a number of database certifications.



# Improve Any SQL Server Audit

## WITH SQL COMPLIANCE MANAGER

- Audit sensitive data to see who did what, when, where, and how
- Monitor and alert on suspicious activity to detect and track problems
- Satisfy audits for multiple industry regulatory requirements
- Select from over 25 pre-defined compliance reports and create custom views
- Lightweight data collection agent minimizes server impact

Start for FREE

