Eugene Borts

Applied Database I

Dr. Ron Eaglin

Assignment 13

SQL Security Report

Introduction

The purpose of this report is to establish a step-by-step walkthrough of two scenario's pertaining to the security of a SQL Server database. In the first scenario, a hacker is attempting to penetrate the SQL Server. In the second scenario, an administrator attempts to defend the SQL Server against intrusion. This report is divided into two parts. The first part is a demonstration of various SQL Injection methods from the perspective of a hacker. The second part is a step-by-step guide to establishing a multitude of security measures as an administrator.

Part I: Hacker

One of the most common attacks used by hackers is known as an SQL Injection Attack. SQL Injection occurs when a server requests information from a user such as a username and password, but instead of providing the requested information, the user runs a malicious SQL statement which is passed to the SQL Server for parsing and execution, unbeknownst to the database administrator.

In this example, the user will be playing the role of a hacker by attempting to execute an SQL Injection on an unsuspecting SQL Server. The goal of the hacker is to penetrate the database and gain access to all the usernames and passwords. There are multiple methods that can be used to successfully execute such an attack.

The "1=1 is Always True" method is one of the simpler ways of gaining access to a table and the information it holds. While the original purpose of the code was to select a user with a given ID, the code could be exploited to view the contents of a table if the database does not have the proper restrictions in place. An example of this code can be seen below.

```
SELECT ID, Name, Pass FROM Users WHERE UserId = 10 or 1=1;
```

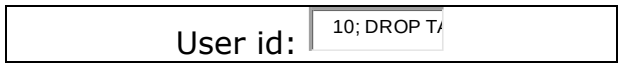
Another simple method of gaining access to user names and passwords in a database is the “”=” is Always True” method. The user, who is a hacker in this case, can insert “ OR “”=” into the username and password fields, which can cause the server to create a valid SQL statement that will display all of the rows in the table. An example of the query generated by the server is shown below.

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

If the methods above fail to gain access to the usernames and passwords, there is a third injection technique based on Batched SQL Statements. A batched SQL statement is simply a group of SQL statements separated by semicolons, as shown below.

```
SELECT * FROM Users; DROP TABLE Clients
```

To gain access to a table using Batched SQL Statements, the hacker may enter his code via an input field such as a text box, as shown below.



If all these methods fail, the hacker can still attempt to breach the SA, or Super Administrator, account. This is a default account with administrative privileges that is created when the database is installed. The reason this is such a common access point for intruders is because this account is created with a very weak default password, which is often left unchanged. By simply guessing the password to the SA account, the hacker may then gain access not only to the usernames and passwords, but to the rest of the database as well.

Part II: Defender

In this scenario, we will take on the role of a server administrator attempting to prevent and defend against attacks from hackers. A primary source of protection against various forms of attacks and malicious code is known as hardening, which is the process of configuring a database to increase security. This includes, but is not limited to, the configuration of accounts, settings, and tools.

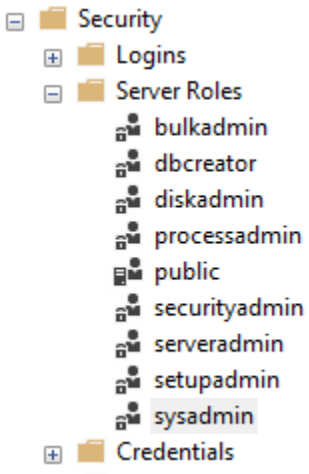
The first step of hardening a SQL server is to ensure that an attacker cannot take over the default administrator account, known as the SA Account or Super Administrator Account.

This account is a common attack point for security breaches and is created automatically when a database is installed. To prevent the breaching and malicious use of this account, a new Super Administrator account with a very strong password should be created to replace the default SA account, and the default SA account must be disabled or deleted. An example of this is shown below.

```
USE MASTER
ALTER LOGIN sa DISABLE
ALTER LOGIN sa WITH NAME = newadmin
```

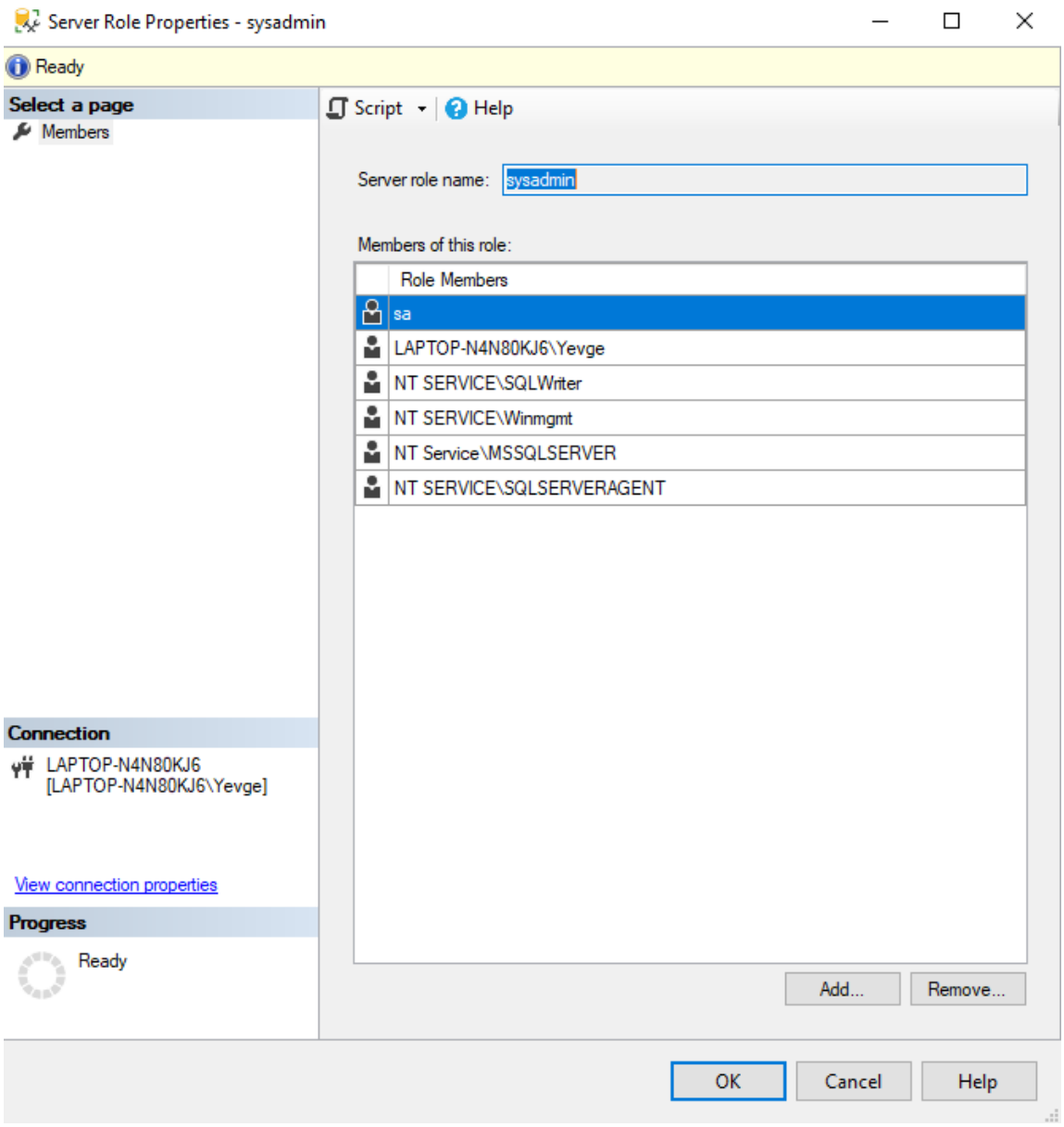
The next step in hardening the server is to go through each user account to make sure there are no other vulnerable accounts, such as members of SYSADMIN or users with CONTROL SERVER permission. A step-by-step example of this process is shown below.

Step I: Display sysadmin role members.



In SQL Server Management Studio, go to Security → Server Roles → sysadmin→ Properties → Role Members

Step II: Add or remove sysadmin role members.



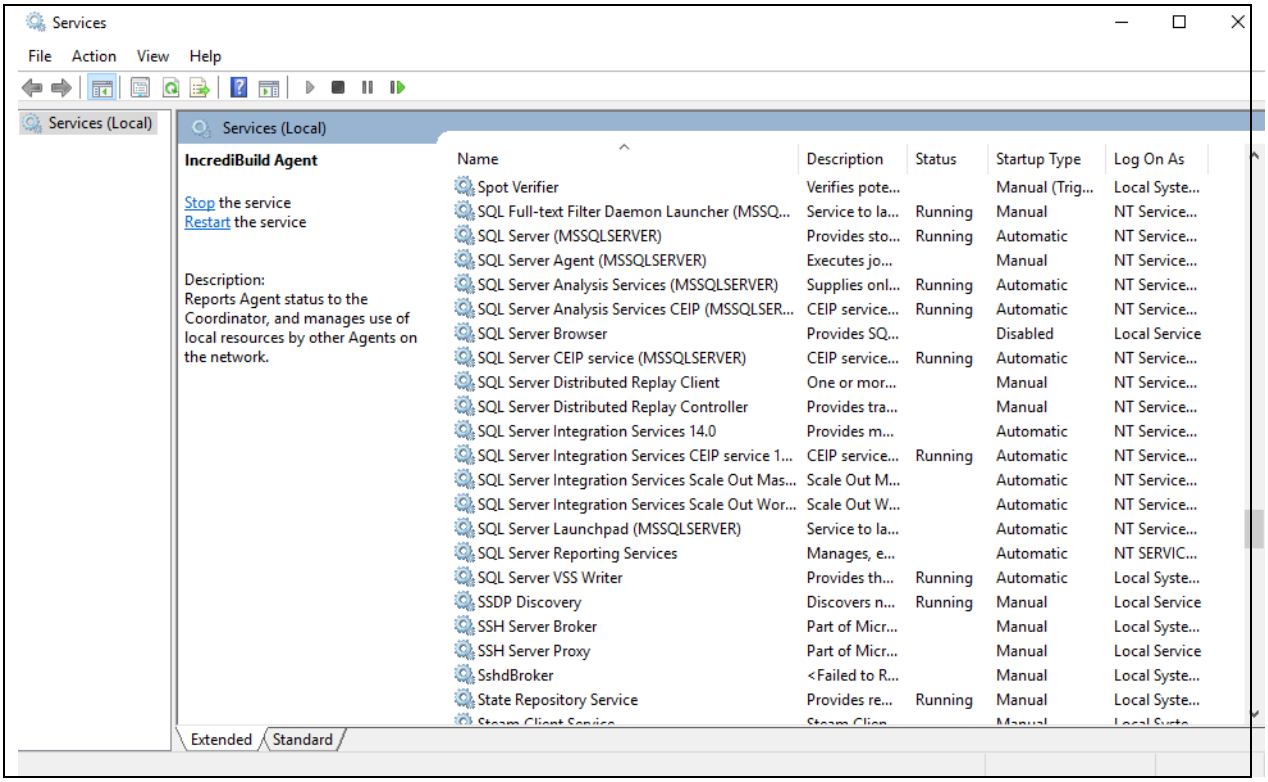
Select any unwanted accounts for the sysadmin role and click remove to remove administrative privileges from a user.

Another step that can be taken to find vulnerable administrative accounts is to view the built-in administrators. An example of this is shown below.

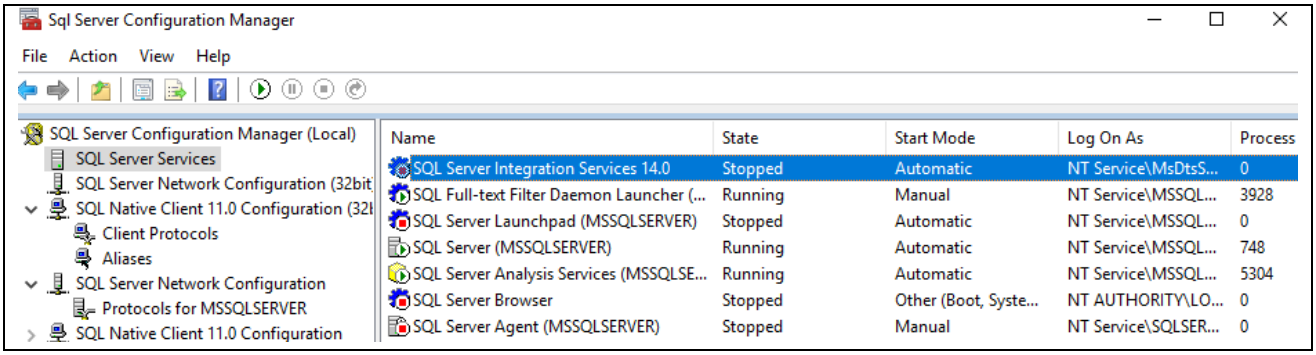
```
USE MASTER
GO

SELECT * FROM sys.server_principals
WHERE name = N'BUILTIN\Administrators'
```

Once all vulnerable administrator accounts have been secured, the next step in database hardening is to view which SQL services are running and disable any unnecessary processes and services. The two tools that can be used to accomplish this are the SQL Server Configuration Manager and the Windows Services Manager. The Services Manager lists all services running on the Windows operating system, so it provides a wider span of information, detail, and control, but the SQL Server Configuration Manager can also be used to disable or enable certain SQL services as needed. Services that may be disabled to increase security include reporting services, integration services, analysis services, and notification services. A comparison between the two applications can be seen below.

Windows Services

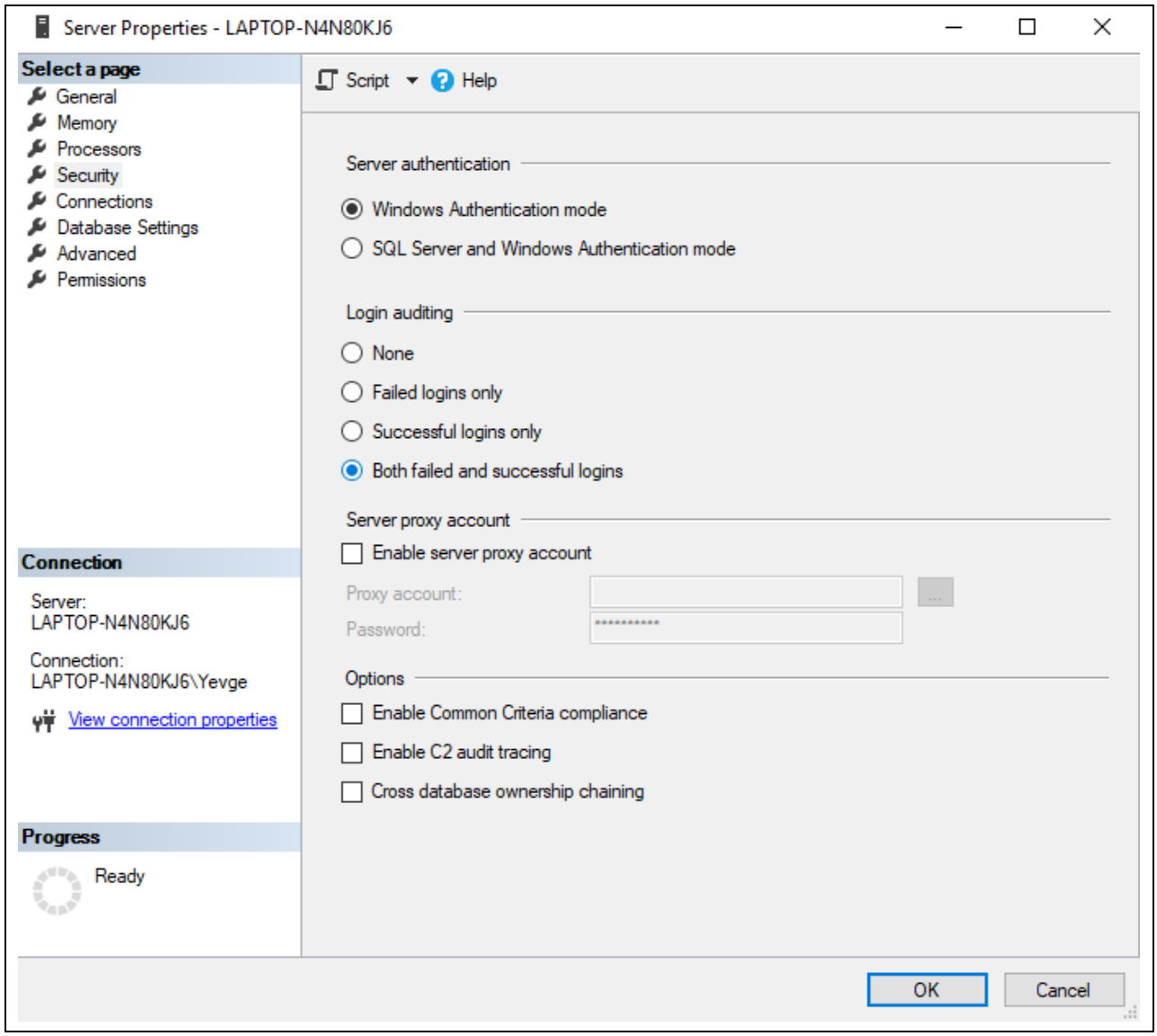


SQL Service Configuration Manager

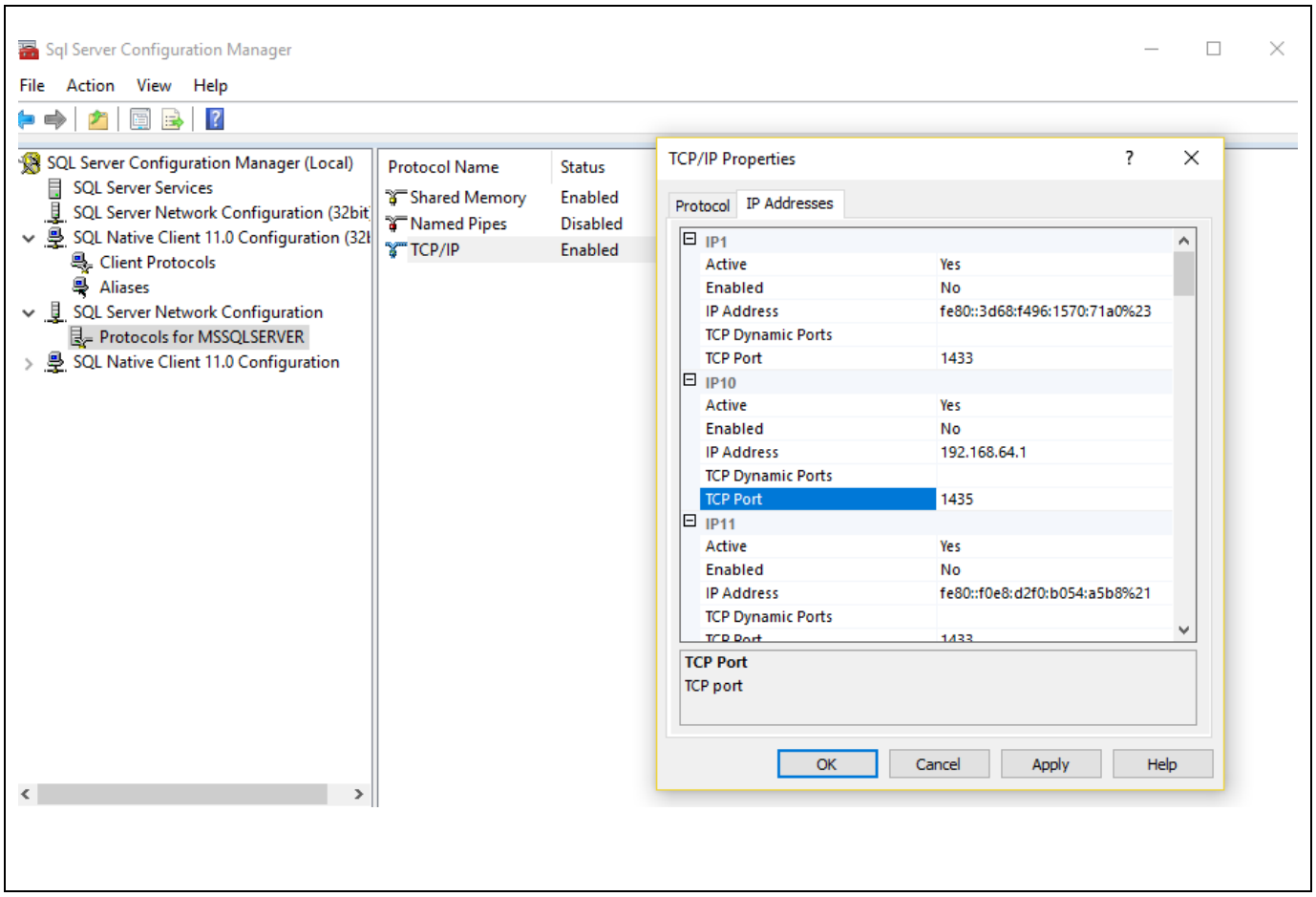


Authentication is another important factor in hardening a SQL database. Poor authentication measures can result in a compromised account, and therefore a compromised database and breached server. A simple security measure to take when selecting and configuring account authentication in SQL Server is to avoid using SQL Authentication and to use Windows Authentication instead. Windows Authentication is more secure than SQL Authentication and will make it more difficult for hackers to breach the database.

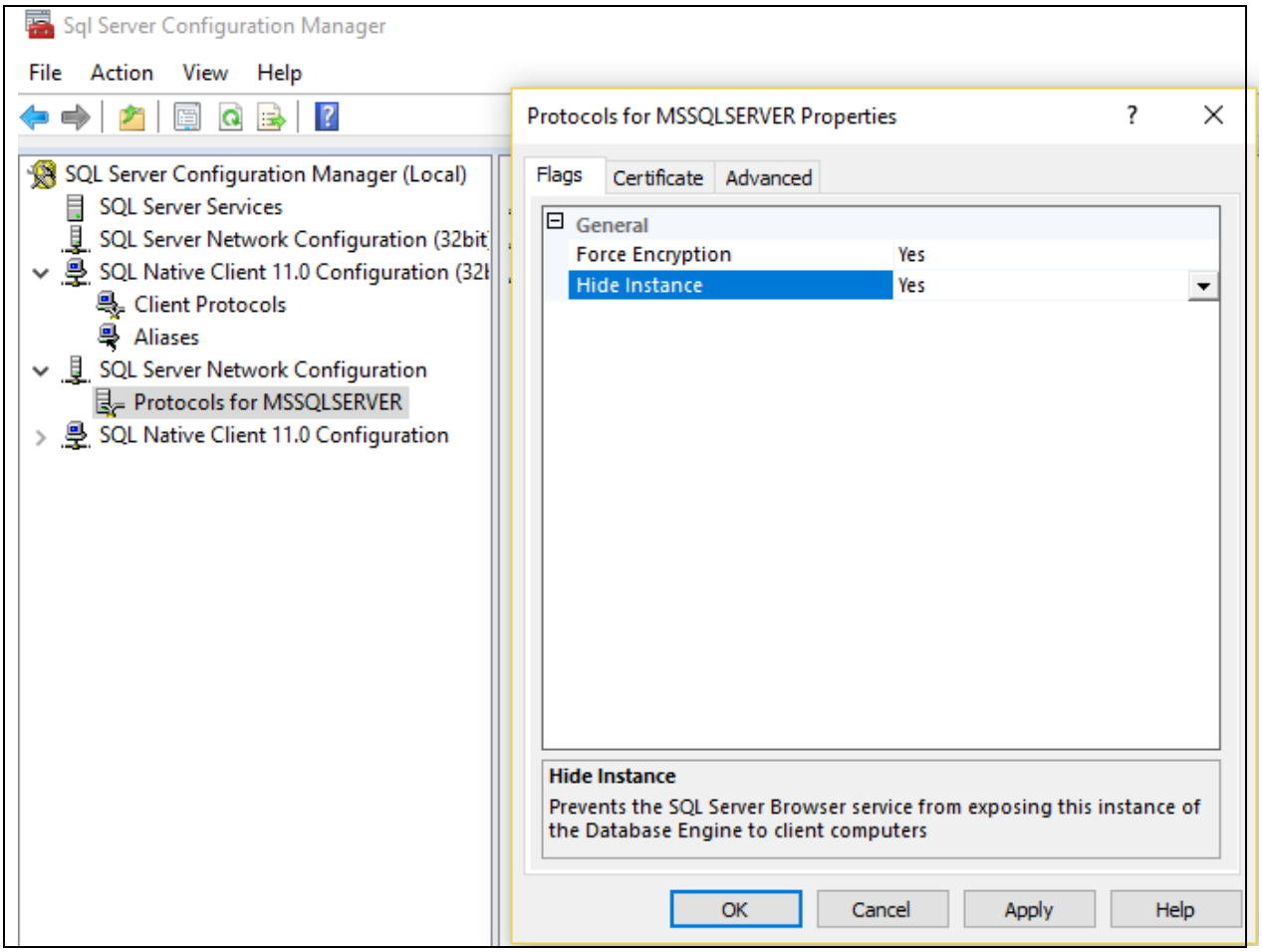
Logging is an extremely important feature in any form of cybersecurity apparatus, so the next logical step in hardening our SQL server is to enable login logging. This step can be completed quickly and easily by right clicking on the server from the object browser in SQL Server Management Studio, clicking on properties to open the properties window, clicking on security to open the security page, and selecting both failed and successful login attempts under login auditing. The server authentication method and the logging options may both be changed on the security page of the server properties window, which can be viewed in the example below.



The next step in hardening our server is to change the default SQL Server port. The default TCP port for SQL Server is 1433, so hackers may already know to sniff this port. If the port is changed to a different open port, for example 1435, then it may create some difficulties for the attacker in finding an opening. This can be accomplished by opening the SQL Server Configuration Manager, navigating to Protocols for MSSQLSERVER under SQL Server Network Configuration, enabling TCP/IP, right clicking on TCP/IP to open its properties, clicking on the IP Addresses tab, and changing the value 1433 in the TCP port to a different value, which in this case will be 1435. An example of this process is shown below.



We can keep the SQL Server Configuration Manager open to take the final step in hardening our SQL Server. This last step we will take is to force encryption and turn off broadcasting for the server. To do this, we right click on Protocols for MSSQLSERVER and click on properties. From there, we will change both the force encryption field and the hide instance field to yes. An example of this process is shown below.



Now that we have completed hardening our SQL Server by securing vulnerable accounts, turning off unnecessary services, selecting the appropriate form of authentication, enabling logging, changing default ports, hiding broadcasting, and forcing encryption, the risk of this SQL server being breached by an attacker has been reduced significantly. The job of protecting against intruders is never completed, and there are always more measures that can be taken to continue hardening our server. These tasks include keeping service packs up to date, enforcing a strong password policy, and disabling additional SQL services such as browser services.

Conclusion

As we have seen, SQL Injection can easily be exploited by hackers to breach SQL databases, allowing them to compromise intricate systems, steal valuable information, cause catastrophic damage, or perform any combination of malicious actions. However, by taking advantage of the various approaches to SQL server hardening, a server administrator can defend against such attacks from hackers and even prevent those attacks from occurring.

Though there will always be ways for hackers to breach and compromise a system, there will always be countermeasures that can be taken to defend against such attacks.