Exploiting MS SQL Servers

redfoxsec.com/blog/exploiting-ms-sql-servers

Shashi Kant Prasad

July 18, 2023

- July 18, 2023
- Active Directory
- Shashi Kant Prasad

As companies continue to rely on databases to store sensitive information, securing the data has become a top priority. MS SQL Server is a popular database management system that integrates with Windows and Active Directory domains, creating trust relationships that can be leveraged for attacks. As a Red Teamer, it's crucial to understand the fundamentals of MS SQL Server, how to locate and access it, and how to escalate privileges within it. This blog will explore these topics to provide a comprehensive guide for Red Teamers using MS SQL Server.

Fundamentals

MS SQL Server is a collection of Windows services that run on the operating system in the context of the service account. When an instance of SQL Server is installed, a set of unique Windows services is installed. There are three types of SQL Server accounts: Windows accounts, SQL Server logins, and database users. Unless you are a system administrator, you must map an SQL Server login to a database user to access data. At the SQL Server, you use Windows accounts and SQL Server logins for signing in. However, a database user is created independently within the database level. MS SQL Server has several roles, including the sysadmin role, equivalent to the Windows admin for SQL Server. The public role is like the Everyone group in Windows and has the least privilege.

Locating and Accessing MS SQL Server

As a Red Teamer, knowing how to locate and access MS SQL Server is essential. There are several tools and techniques to accomplish this.

Unauthenticated Perspective

Tools like OSQL, SQLPing3, and sqlcmd Utility can be used to identify MS SQL Servers. For example, the sqlcmd -L command can identify MS SQL Servers. PowerUPSQL can also be used to identify MS SQL Servers by importing the module and using the following command.

```
Get-SQLInstanceScanUDP
```

In Azure environments, databases can be located using a DNS dictionary attack against the format x.databases.windows.net. Configuration files containing connection strings on public repositories can also be used to locate databases.

The Local User Perspective

As a local user, SQL Server instances can be identified by checking system services and registry settings. PowerUpSQL includes a function to quickly identify local instances using the follwoing command.

```
Get-SQLInstanceLocal
```

The Domain User Perspective

For SQL Server installed inside a domain, instances are automatically registered in Active Directory with an associated service account to support Kerberos Authentication. Instances can be identified using PowerUpSQL by using the command given below.

```
Get-SQLInstanceDomain
```

Escalating Privileges

As someone working as a Red Teamer, it is crucial to have the skill of escalating privileges in MS SQL Server. One way to do this is through dictionary attacks using commonly used login credentials, but account lockouts should be avoided. PowerUpSQL can be used to perform such attacks, such as Get-SQLInstanceScanUDP | Invoke-SQLAuditWeakLoginPw to start the attack from an unauthenticated user perspective and Get-SQLInstanceDomain | Invoke-SQLAuditWeakLoginPw to start the attack from a domain user perspective.

It is common for applications that utilize SQL Server Express as the backend to be set up with specific credentials and instance names based on recommendations from vendors. PowerUpSQL can be used to check these credentials using Get-SQLInstanceDomain | Invoke-SQLAuditDefaultLoginPw and Get-SQLInstanceDomain | Get-SQLServerLoginDefaultPw.

It is possible to perform MITM attacks to inject queries if communications with the MS SQL Server are unencrypted. Depending on the spoofed user's privilege, we can inject SQL Logins. As a local or domain user, we can try to log in to MS SQL Servers with the current account since excessive login privileges are a common practice on enterprise networks.

Public Role -> Sysadmin Privileges

One way to escalate privileges within MS SQL Server is through the public role, which is similar to the Everyone group in Windows. We can attempt to list all SQL Server logins by using the SELECT name FROM sys.syslogins and SELECT name FROM sys.server_principals commands. To identify SQL logins, one can fuzz the principal ID value within the suser_name function using the public role. Blind domain account/object enumeration can be performed with the public role, which is useful in case of a remote SQL injection attack.

Attackers can exploit stored procedures and trigger creation/injection issues to escalate privileges. Many developers tend to collect the various functions they need and store them in a procedure that the database owner can execute (using the "EXECUTE AS OWNER" command) to gain access to more resources. However, if these procedures are not implemented securely, attackers could potentially extend them to perform SQL injection or command injection attacks. We can use the Get-SQLStoredProcedureCLR command of PowerUpSQL to identify such databases and then use the Create-SQLFileCLRDll command to export all custom CLR assemblies to DLLs, backdoor any of the DLLs, and finally import the backdoored CLR.

Common Post-Exploitation Activities

As a Red Teamer, our job doesn't end with escalating privileges within MS SQL Server. We need to identify sensitive data, extract SQL Server login password hashes, and establish persistence.

Identifying Sensitive Data

We can use regular expressions to filter data and identify sensitive information. PowerUpSQL can be used to accomplish this with the Get-SQLInstanceDomain | Get-SQLConnectionTest | Get-SQLColumnSampleDataThreaded command. Transparent encryption can also be identified using the Get-SQLInstanceDomain | Get-SQLConnectionTest | Get-SQLDatabaseThreaded command.

Extracting SQL Server Login Password Hashes

PowerUpSQL can be used to extract SQL Server login password hashes with the command given below.

```
Get-SQLServerPasswordHash
```

Establishing Persistence

As a Red Teamer, we need to establish persistence to maintain access to MS SQL Server. PowerUpSQL can be used to accomplish this with functions like Get-SQLPersistentRegDebugger, Get-SQLPersistentRegRun, and Get-SQLStoredProcedureCLR.

Poisoning the MS SQL Server Resolution Protocol

It is possible to poison the MS SQL Server Resolution Protocol, which allows us to authenticate to a server that we control. As a Red Teamer, it's crucial to understand this vulnerability and how to exploit it.

TL;DR

MS SQL Server is a popular database management system that Red Teamers need to understand to conduct successful attacks. By understanding MS SQL Server fundamentals, locating and accessing MS SQL Server, escalating privileges, and performing common post-exploitation activities, Red Teamers can gain access to sensitive information and establish persistence. By using tools like PowerUpSQL and understanding vulnerabilities like the MS SQL Server Resolution Protocol, Red Teamers can become skilled, innovative, and reliable security experts.

Redfox Security is a diverse network of expert security consultants with a global mindset and a collaborative culture. If you are looking to improve your organization's security posture, **contact us** today to discuss your security testing needs. Our team of security professionals can help you **identify vulnerabilities and weaknesses in your systems, and provide recommendations to remediate them**.

"Join us on our journey of growth and development by signing up for our comprehensive **courses**."

PreviousBloodHound Cheat Sheet
NextUnderstanding CRLF Injection Attacks

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker's Swiss Army Knife. Have You Heard About It?