International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Secure Integration: .NET Core Authentication with SQL Server in Linux Docker Containers

Vishnupriya S Devarajulu

Email: *vishnupriyasupriya[at]gmail.com*

Abstract: ***Integrating a. NET Core application using Integrated Security like Kerberos Authentication from a Linux Docker container with SQL Server is a challenging process due to protocols security differences in platform. This article provides a step - by - step solution and sample code to configure and authenticate it in a simple way.***

Keywords: .NET Core, SQL Server, Security, Docker, Linux, Kerberos Authentication, Docker Container

1. Introduction

Generally, to connect the application to SQL Server, we use the integrated security method of Windows Authentication where we enter the credentials of the user that needs to be logged in with. But this method is usually preferred in enterprise environments such as Active directory where it cross checks the entered login information with the user authentication group or log in information registered in the Active directory. But if we need to run the. NET Core applications in Docker containers on Linux using Integrated Security to connect to SQL Server, it needs additional configuration setup, especially when that Integrated Security is Kerberos Authentication. This Article provides detailed solution steps to achieve this.

Prerequisites

- . NET Core application
- Docker installed on a Linux machine
- Kerberos setup for the Linux environment
- SQL Server instance configured for Windows Authentication

Steps to Authenticate. NET Core Client with SQL Server Using Kerberos Authentication

Configure Kerberos on the Linux Host: First thing to configure is Kerberos on the Linux Host. Kerberos must be set up on the Linux host to allow Integrated Security Authentication. This includes installing the necessary packages and configuring the Kerberos client. To communicate with the underlying KDC, a proper *krb5. conf* file should be created and stored. The *krb5. conf* file must be in the containers at */etc/kerb5. conf*

Install Required Packages: These packages enable us to run kinit command within the container for fetching Kerberos tickets from the KDC.

```
bash
sudo apt - get update
sudo apt - get install - y krb5 - user libpam - krb5
```

or you can also do

```
RUN apt install - y krb5 - config
RUN apt - get install - y krb5 - user
```

Configure Kerberos Client: Edit /**etc/krb5. conf** to match environment.

```
conf
 [libdefaults]
 default_realm = TESTDOMAIN. COM
 dns_lookup_realm = false
 dns_lookup_kdc = true

 [realms]
 TESTDOMAIN. COM = {
 kdc = testdomain. com
 admin_server = testdomain. com
 }

 [domain_realm]
 . testdomain. com = TESTDOMAIN. COM
 testdomain. com = TESTDOMAIN. COM
```

Create a Keytab File: A keytab file is required to authenticate the service. We will need to create this on a Windows machine and copy it to the Linux host.

```
powershell
ktpass - princ HTTP/yourhostname[at]TESTDOMAIN. COM - mapuser youruser - crypto AES256 - SHA1 - ptype KRB5_NT_PRINCIPAL - pass yourpassword - out yourkeytabfile. keytab
```

Copy yourkeytabfile. keytab to the Linux host, typically to /etc/krb5. keytab.

The AES256 - SHA1 in the ktpass command refers to the encryption and hashing algorithms used to create the Kerberos keytab file. Here's a breakdown of each component:

- **AES256**: This stands for Advanced Encryption Standard (AES) with a 256 - bit key. AES is a symmetric encryption algorithm widely used for securing data.
- **SHA1**: This stands for Secure Hash Algorithm 1. It is a cryptographic hash function that produces a 160 - bit hash value, typically rendered as a 40 - digit hexadecimal number.

Dockerfile for. NET Core Application: The next step is to create a Dockerfile for. NET Core application. The Dockerfile needs to set up the Kerberos client and copy the keytab file.

The job of the Dockerfile can be described as:

- Building the. NET Core application in a container based on microsoft/dotnet: sdk AS build - env image
- Copying the build artefacts to a runtime image (from microsoft/dotnet: aspnetcore - runtime)

- Installing kerberos packages for debian: stretch - slim image
- Copying the krb5. conf file and keytab
- Copying a script that executes kinit and launches the. NET application
- Specifying the launch script as entry point

dockerfile
FROM mcr. microsoft. com/dotnet/aspnet: 5.0
WORKDIR /app

Install Kerberos client
RUN apt - get update && apt - get install - y krb5 - user

Copy keytab file
COPY krb5. keytab /etc/krb5. keytab

Copy the application files
COPY. .

ENTRYPOINT ["dotnet", "YourApp. dll"]

Configure. NET Core Application: Update. NET Core application to use Integrated Security for SQL Server connection. Need to ensure we have the necessary NuGet packages installed.
xml
<ItemGroup>
<PackageReference Include="Microsoft. Data. SqlClient" Version="5.0.0" />
<PackageReference Include="Microsoft. Data. SqlClient. SNI" Version="5.0.0" />
</ItemGroup>

In application's appsettings. json, configure the connection string to use Integrated Security Authentication.
json
{
"ConnectionStrings": {
"DefaultConnection": "Server=your_sql_server; Database=your_database; Integrated Security=true; TrustServerCertificate=true; "
}
}
In application code, the connection string must be used to connect to the SQL Server.
using Microsoft. Data. SqlClient;

public class DatabaseService
{
private readonly string _connectionString;

public DatabaseService (IConfiguration configuration)
{
_connectionString = configuration. GetConnectionString ("DefaultConnection");
}

public void ConnectToDatabase ()
{
using (SqlConnection connection = new SqlConnection (_connectionString))
{
connection. Open ();
// Perform database operations
}
}
}

Build and Run the Docker Container: Build Docker image and run the container. Ensure that the container runs with the necessary Kerberos configurations.

Build the Docker Image:
bash
docker build - t yourapp: latest.

Run the Docker Container:
bash
docker run - it - - rm - - name yourapp - v /etc/krb5. conf: /etc/krb5. conf - v /etc/krb5. keytab: /etc/krb5. keytab yourapp: latest

2. Conclusion

The Process of Authenticating a. NET Core application with SQL Server using Integrated Security such as Kerberos Authentication from a Linux container needs at most care. As, even if there is a small mistake, the developer would run into multiple issues and must spend lot of time identifying and fixing and redoing the whole process. By following the detailed steps outlined in this article, developers can successfully set up and run application securely. This setup combines the advantages of Docker's containerization with the security and simplicity of Authentication, ensuring secure and seamless access to SQL Server Database.

References

[1] Code Project - https: //www.codeproject. com/Articles/1272546/Authenticate - NET - Core - Client - of - SQL - Server - with - In
[2] Learn. Microsoft. com - https: //learn. microsoft. com/en - us/windows - server/security/kerberos/kerberos - authentication - overview
[3] Learn. Microsoft. com - https: //learn. microsoft. com/en - us/windows - server/administration/windows - commands/ktpass
[4] Ubuntu - https: //help. ubuntu. com/community/Kerberos
[5] GitHub - https: //github. com/dotnet/SqlClient/issues/143