Running MQ .NET Core SSL Applications on Linux

Ram Subbarao
Published on 02/08/2019 / *Updated on 12/08/2019*

On Linux where should the certificates be installed for .NET applications?
It is always recommended to use a .NET application to install the certificates on Linux where .NET can look up for the certificates.

X509Store class is used to install certificates.

Wrapping the following few lines of code in a .NET Core application and running on Linux will install the certificates.

```
   using System;
   using System.Security.Cryptography.X509Certificates;
   namespace SimplePut
   {
       class SimplePut
       {
           [STAThread]
           static void Main(String[] args)
           {
               try
               {
                   X509Store store = new X509Store(StoreName.My,
StoreLocation.CurrentUser);
                   store.Open(OpenFlags.ReadWrite);
                   X509Certificate2 certificate1 = new
X509Certificate2("client.p12", "12345");
                   //Create a collection and add two of the
certificates.
                   X509Certificate2Collection collection = new
X509Certificate2Collection();
                   collection.Add(certificate1);
                   //Add certificates to the store.
                   store.Add(certificate1);
                   store.AddRange(collection);
                   X509Certificate2Collection collection2 = new
X509Certificate2Collection();
                   collection2.Import("client.p12", "12345",
X509KeyStorageFlags.PersistKeySet);
                   foreach (X509Certificate2 cert in collection2)
                   {
                       store.Add(cert);
```

```
                    Console.WriteLine("Certificate installed
successfully");
                }
            }catch(Exception e)
            {
                throw e;
            }
        }
    }
}
```

Once we have the certificates ready for both the client and server follow the below steps:

1. Copy the above code into a file by name Program.cs.
2. Create a file by name SSL.csproj in the same directory where Program.cs file is placed and copy the below content into the SSL.csproj
3. `<Project Sdk="Microsoft.NET.Sdk">`
4. `  <PropertyGroup>`
5. `    <OutputType>Exe</OutputType>`
6. `    <TargetFramework>netcoreapp2.1</TargetFramework>`
7. `  </PropertyGroup>`
8. `</Project>`
9. Build the SSL.csproj using the following command:
   `dotnet build SSL.csproj`
10. Once the application is built successfully you should see "SSL.dll" under the same directory.
11. Run the SSL.dll application to install the certificates. Use the following command to run the application
    `dotnet run SSL.dll`
12. By running the .NET Core application, the certificates get added into the keystore. The certificates are installed in the following location **".dotnet/corefx/cryptography/x509stores/my"**
    Note: Directly interacting with "**dotnet/corefx/cryptography /x509stores/"** directory or its child directories and files can put .NET in an invalid state and is not supported.
13. Run the MQ .NET Core SSL enabled application to test the setup. Run the MQ .NET Core SimplePut sample located at "/opt/mqm/samp/dotnet/samples/cs/core/base/bin"

```
            [root@abjures1 bin]# dotnet SimplePut.dll -q Q1 -h 127.0.0.1
-p 5454 -l SYSTEM.DEF.SVRCONN -k *USER -s
       TLS_RSA_WITH_AES_128_CBC_SHA256
       Start of SimplePut Application
       MQ Parameters
       1) queueName = Q1
       2) keyRepository = *USER
       3) cipherSpec = TLS_RSA_WITH_AES_128_CBC_SHA256
       4) host = 127.0.0.1
       5) port = 5454
```

```
        6) channel = SYSTEM.DEF.SVRCONN
        7) numberOfMsgs = 1
        8) sslPeerName =
        9) keyResetCount = 0
        10) sslCertRevocationCheck = False
        Connecting to queue manager.. done
        Accessing queue Q1.. done
        Message 1 <test message>.. put
        Closing queue.. done
        Disconnecting queue manager.. done
        End of SimplePut Application
```