A WCF & WPF based Chat Application

S. Narayanaswami and D. Yoon
CIS Department, University of Michigan, Dearborn, MI, USA

Abstract - We present a chat system based on Windows Communication Foundation (WCF) and Windows Presentation Foundation (WPF) with emphasis on security aspects of WCF. As the name indicates, WCF lays down the foundation for the communication between server and client, while WPF is a graphical user interface system in .NET. Through the chat system, we explore the security measures implemented in X.509, which include the contracts, endpoints and the usage of the service. This is then presented by developing an application which implements the WCF security aspects in a client-server architecture and effectively handles the security. The client and server will communicate by using xml messages. WCF provides two mechanisms for transfer security which is transport security and message security. We will make use of these security mechanisms provided by the WCF in our application.

One common use of the WCF is to provide a way to distribute and collect data and information to and from multiple clients at the same time, with no hassle in the user experience on the client side.

The four core security features that WCF addresses are Authentication, Authorization, Confidentiality, Integrity and auditing. WCF supports a variety of authentication options including username, Windows, and certificate authentication which will be studied in detail.

Keywords: Security, X509, Client, Server, cloud system

1 Introduction

A chat system is an internet communication system in which a client asks questions and the server answers them. It is a relatively simple system, yet contains all aspects of the client-server model.

A client login into the system with the valid id. The server checks the clients certificate and allows a chat session if the client has the legitimate certificate. The activities of the system is briefly summarized below.

• Allows the registered users to Login using their Credentials.
• Allow users to join a peer-peer chat
• Display Online users/contact list
• Allow sending and receiving message from and to Contact
• Server records user login and logoff time.
• User list updated when user goes offline
• Secure message using Certificate

2 System Architecture and Design

Service
• Interface IChattingService
  Login, Logoff, Send Message
• Interface IClient
  GetMessage, GetUpdate

Host
• Define Endpoints
• Define Bindings
• Define the Security-Message Level-Certificate (Reliable Sessions)

Client
• Implement IClient in Client Callback

Others
• Allows user to login with proper Credentials

3 Implementation of the System

This system was built on WCF and WPF using .NET framework 4.5.2 in Visual Studio 2015. The WPF acts as a wrapper around the WCF making the Application look nicer that ordinary windows form or console application. The WPF is the library that provides us with all GUI elements and user interactions. We have secured the Application by making use of Certificates, one for the Client and Server using makecert.exe. (Makecert.exe is a free tool provided by Microsoft which helps to create X.509 certificates that are signed by a system test root key or by another specified key) [6].

WCF is used to build service oriented applications using which we can transfer messages and data asynchronously from one service end point to another.[3]. In WCF when we think about service, we are going to create a class of service
and Interface called Service Contract. Some of the key terminologies to know on WCF include:

**Service**: The service is the implementation for things that are exposed to the outside client. Every client starting a session does not initiate a new instance of the service.

**Service Contract**: Service contract is what is exposed to the outside world which specifies the methods that need to be implemented. (Login/Send Message/List Users)

The Client sees what the server has to offer through the Service Contract. Service contract is marked with the [ServiceContract] attribute. The Operation Contract attribute defines an operation and is part of the service contract and is decorated with [OperationContract] attribute. The service and the service contract need to be connected or hosted to make use of the features provided by them.

**Endpoint**: An endpoint of WCF service acts as a gateway for communicating with the outside world. An endpoint is composed of

**Address**: Every service has unique address which is of the format

\[Transport\] :// [Domain Name]: [Port]// [Service Name]

**Binding**: Binding indicates the protocol that are supported by the service (TCP/HTTP/NETCP)

**Contract**: This defines the Operations/or what the service can do.

### 3.1 Client

The Client is implemented using the WPF (Windows Presentation Foundation). The client pages consist of the MainWindow.xaml. This is the screen where users will exchange messages. The right side list holds the list of users online. We have use the dictionary data structure to hold the users and we remove the user from the dictionary when the user goes offline.

![Figure 1 A login window and the code behind](image)

### 3.1.1 Server

The server is split into two parts. The first is the Interface which has the Contract, methods and properties and the second part being the implementation of the Interface. The server contains a contract that determines the Service behavior.

The ChattingInterfaces is where we have defined the OperationContract, which includes:

1. **Login** - this accepts the parameter username, which is used to login with the desired name to join the chat window.
2. **SendMessageToAll** - this operation accepts the message and username as parameter and is used to distribute the message across the active clients.
3. **Logout** - this operation Logs the client out of the chat window/application
4. **GetCurrentUsers** – This returns a list of String which holds the Current active users in the application.

In the Chatting Server, we define the Operation Contract methods. In the chatting server, we create a Dictionary to hold the list of logged in users in the Application. On user exit, the user is removed from the dictionary.

The ChattingServer code is pretty straight forward. It establishes the connection and prints the client’s status.

### 4. Securing It Using Certificates

Certificates are a type of Identification to ensure that the communication is known and reliable. Technically a Certificate binds together a name and a public key [6]. (Makecert.exe is a free tool provided by Microsoft which helps to create X.509 certificates that are signed by a system test root key or by another specified key)

To have a secured communication between the Client and Server, I have used the X509 certificate authentication. This consist of the WCF Services-clients should be authenticated by X509 certificate. Client should validate the service using this certificate.

Steps to be followed to create the certificate [7]:

**Step 1:**
Open Visual Studio Command Prompt and go to the location where you want to save the certificate.

**Step 2:**
Execute the below command

```
makecert -n "CN=RootCATest" -r -sv RootCATest.pvk RootCATest.cer
```

You will be prompted to enter a password and hit OK.

**Step 3:**
The next step is to install the Root Certificate Authority in Client and Server machine.
For the Certificate to be made a trusted one, we need to import this to the Trusted Root Certification Authority Store. The Microsoft Management Control (MMC), can be accessed by typing MMC in the command prompt. In the MMC dialog box, we have to click the Add or Remove Snap in from the File menu and select Certificate from the available Snap-in and Add the certificate.

![Figure 2 Microsoft Management Control Dialog box](image)

This will prompt the Certificate Snap in dialog, Select the Computer account and use the default option in the following steps and hit Finish.

**Step 4:**
We now need to add the Certificate to the Store by importing it. To Import the certificate, go-to the Console Root and expand the Certificates menu item. Expand the 'Trusted Root Certificate Authority, right click and select ‘All Task’ and ‘Import’ (Fig 6.5).

The welcome wizard appears the one below (Fig: 6.6)

**Step 5:**
The next step is to install the Temporary Service Certificate to the Server machine. We follow the same steps as we followed to the Client machine.

```bash
makecert -sk <<UniqueKeyName>> -iv RootCATest.pvk -n "CN=<<MachineName>>" -ic RootCATest.cer -sr localmachine -ss my -sky exchange -pe
```

Configuring the Temporary Service Certificate in IIS

**Step 6:**
Once the certificates are installed in the Client and Server machines, we need to update our configuration files of the Client and Server. In the config, we add the client and server credentials to the respective project config and specify the certificate name in the find value and the store location will be of that of the current user.

5  **Putting It Altogether**

After creating the Certificates and updating the config file, this section has the screenshot of the application. We will be using two instance of the client and see how the message transfer is achieved.

The below screen is the Login screen of the application. The user has to provide the login information and this is validated with the stored values in the backend. If the authentication is successful, the User is taken to chat window

6  **Conclusions**

WCF Chat application has been implemented successfully. The following service functions have been verified:

- Allows only authenticated users to login to the application
- Allow multiple instances of the Client with single server
- Peer to peer message transfer
- Update Active user list
- Secure the message transfer with Certificate at both the client and server
- Server records all the information about the client instances login and logoff

It can be concluded that the implementation of WCF is simple but versatile in its effectiveness. It is a powerful way of establishing a secured communication either using transport or message level security. WCF inherently provides authentication, authorization, integrity and confidentiality for any application built on the WCF framework. The implementation of the service was smooth and the created service worked as intended.

The application was developed using the best practice of coding standard.

7  **References**


