

Web Service (WS) Security Tutorial with SOAP

What is WS Security?

WS Security is a standard that addresses security when data is exchanged as part of a Web service. This is a key feature in SOAP that makes it very popular for creating web services.

Security is an important feature in any web application. Since almost all web applications are exposed to the internet, there is always a chance of a security threat to web applications. Hence, when developing web-based applications, it is always recommended to ensure that application is designed and developed with security in mind.

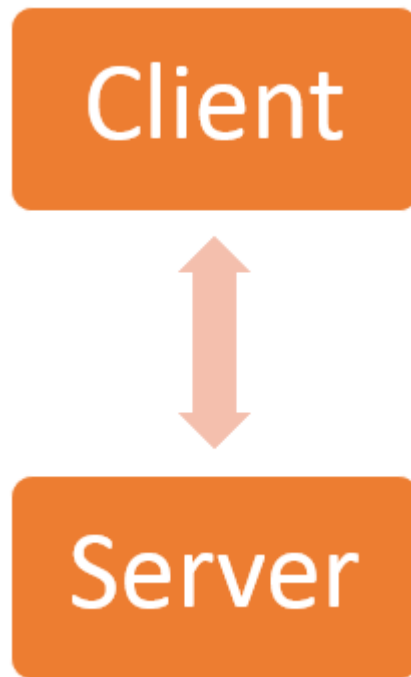
In this tutorial, you will learn-

- [Security Threats and Countermeasure](#)
- [Web Service Security Standards](#)
- [How to build secure web services](#)
- [Web Service Security Best Practices](#)

Security Threats and Countermeasure

To understand security threats which can be hostile to a web application, let's look at a simple scenario of a web application and see how it works in terms of Security.

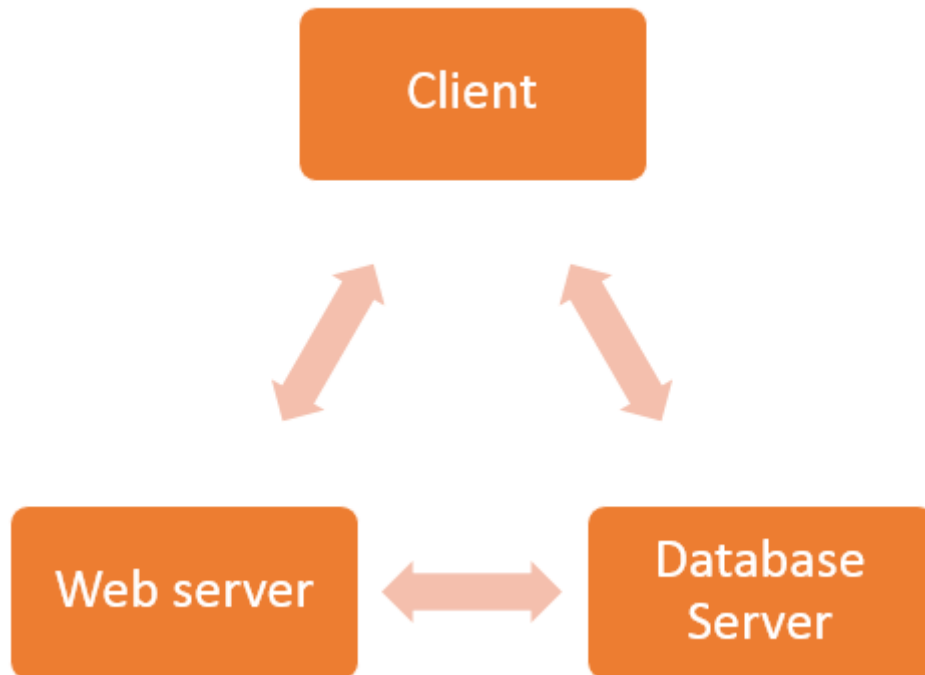
One of the security measures available for the HTTP is the HTTPS protocol. HTTPS is the secure way of communication between the client and the server over the web. HTTPS makes use of the Secure Sockets layer or SSL for secure communication. Both the client and the server will have a digital certificate to identify themselves as genuine when any communication happens between the client and the server.



In a standard HTTPS communication between the client and the server, the following steps take place

1. The client sends a request to the server via the client certificate. When the server sees the client certificate, it makes a note in its cache system so that it knows the response should only go back to this client.
2. The server then authenticates itself to the client by sending its certificate. This ensures that the client is communicating with the right server.
3. All communication thereafter between the client and server is encrypted. This ensures that if any other users try to break the security and get the required data, they would not be able to read it because it would be encrypted.

But the above type of security will not work in all situations. There can come a time when the client can talk to multiple servers. An example given below shows a client talking to both a database and a web server at a time. In such cases, not all information can pass through the https protocol.



This is where SOAP comes in action to overcome such obstacles by having the WS Security specification in place. With this specification, all security related data is defined in the SOAP header element.

The header element can contain the below-mentioned information

1. If the message within the SOAP body has been signed with any security key, that key can be defined in the header element.
2. If any element within the SOAP Body is encrypted, the header would contain the necessary encryptions keys so that the message can be decrypted when it reaches the destination.

In a multiple server environments, the above technique of SOAP authentication helps in the following way.

- Since the SOAP body is encrypted, it will only be able to be decrypted by the web server that hosts the web service. This is because of how the SOAP protocol is designed.
- Suppose if the message is passed to the database server in an HTTP request, it cannot be decrypted because the database does not have right mechanisms to do so.
- Only when the request actually reaches the Web server as a SOAP protocol, it will be able to decipher the message and send the appropriate response back to the client.

We will see in the subsequent topics on how the WS Security standard can be used for SOAP.

Web Service Security Standards

As discussed in the earlier section, the WS-Security standard revolves around having the security definition included in the SOAP Header.

The credentials in the SOAP header is managed in 2 ways.

First, it defines a special element called UsernameToken. This is used to pass the username and password to the web service.

The other way is to use a Binary Token via the BinarySecurityToken. This is used in situations in which encryption techniques such as Kerberos or X.509 is used.

The below diagram shows the flow of how the security model works in WS Security



Below are the steps which take place in the above workflow

1. A request can be sent from the Web service client to Security Token Service. This service can be an intermediate web service which is specifically built to supply usernames/passwords or certificates to the actual SOAP web service.
2. The security token is then passed to the Web service client.
3. The Web service client then called the web service, but, this time, ensuring that the security token is embedded in the SOAP message.
4. The Web service then understands the SOAP message with the authentication token and can then contact the Security Token service to see if the security token is authentic or not.

The below snippet shows the format of the authentication part which is part of the WSDL document. Now based on the below snippet, the SOAP message will contain 2 additional elements, one being the Username and the other being the Password.

```
<xs:element name="UsernameToken">
```

```
<xs:complexType>
  <xs:sequence>
    <xs:element ref="Username"/>
    <xs:element ref="Password" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID"/>
</xs:complexType></xs:element>
```

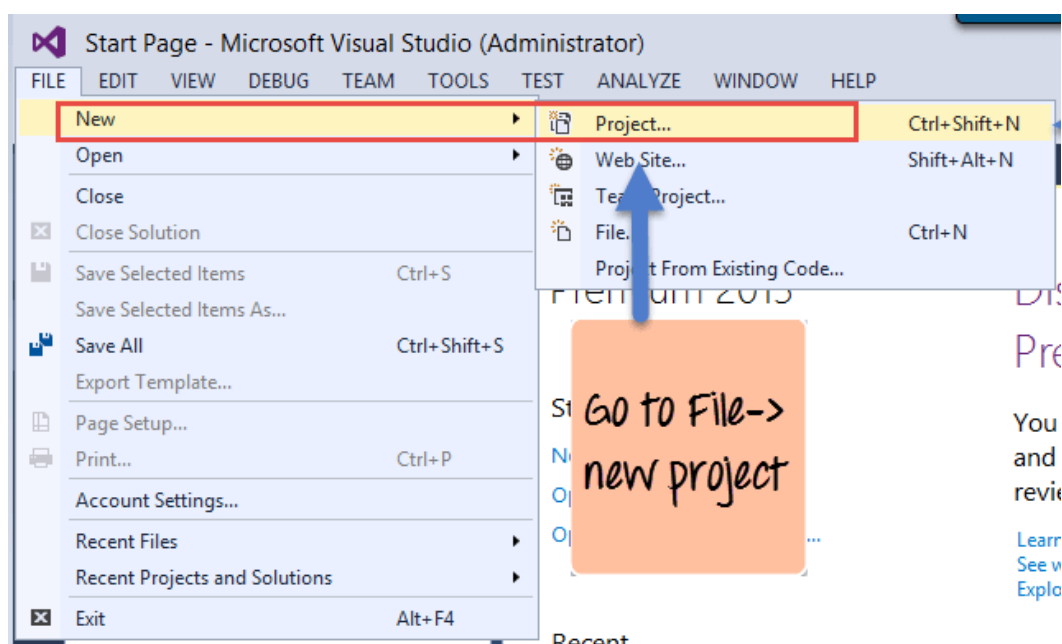
When the SOAP Message is actually passed between the clients and the server, the part of the message which contains the user credentials could look like the one shown above. The wsse element name is a special element named defined for SOAP and means that it contains security based information.

How to build secure web services

Now let's look at SOAP web service security example. We will build a web service security upon the example demonstrated earlier in the SOAP chapter and will add a security layer to it.

In our example, we are going to create a simple web service, which will be used to return a string to the application which calls the web service. But this time, around, when the web service is invoked, the credentials need to be supplied to the calling service. Let's follow the below steps to create our SOAP web service and add the security definition to it.

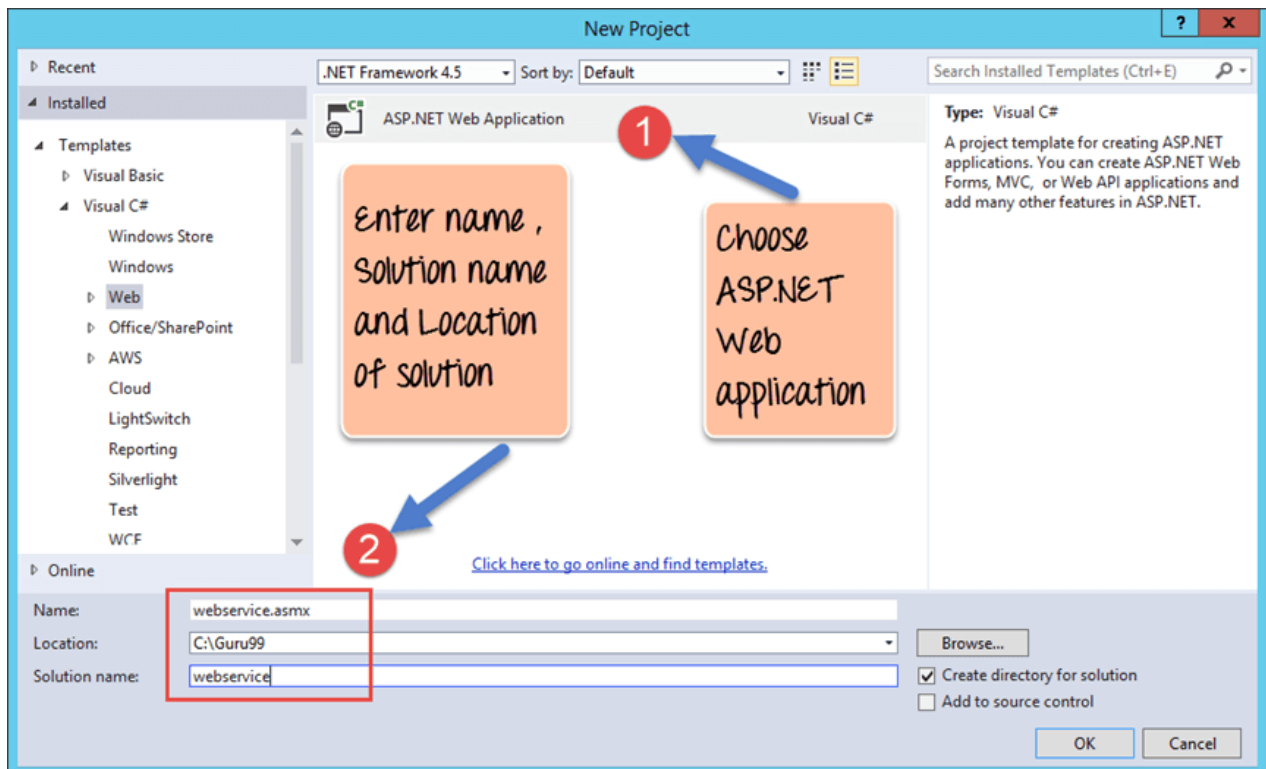
Step 1) The first step is to create an empty [Asp.Net](#) Web application. From Visual Studio 2013, click on the menu option File->New project.



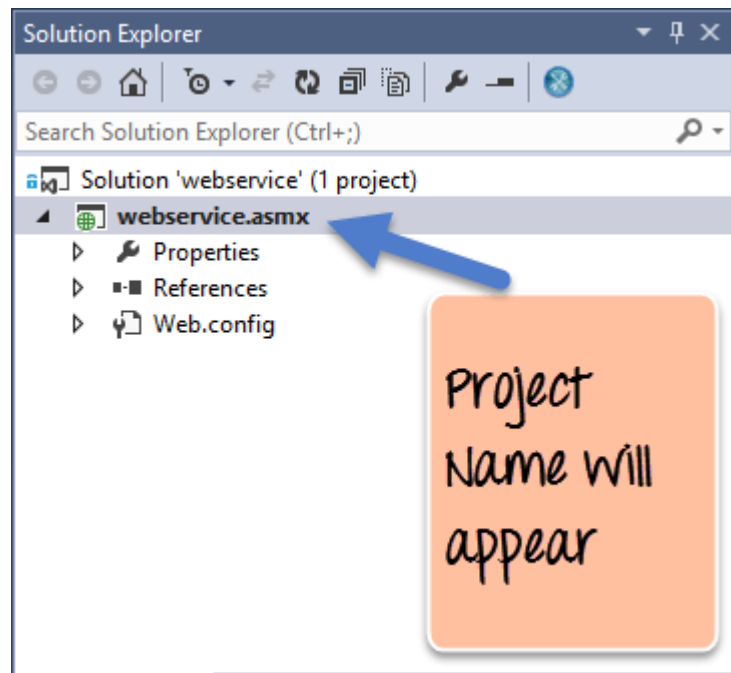
Once you click on the New Project option, Visual Studio will then give you another dialog box for choosing the type of project and to give the necessary details of the project. This is explained in the next step

Step 2) In this step,

1. Ensure that you first choose the [C#](#) web template for ASP.NET Web application. The project has to be of this type in order to create web services project. By choosing this option, Visual Studio will then carry out the necessary steps to add required files which are required by any web-based application.
2. Give a name for your project which in our case has been given as "**webservice.asmx**." Then make sure to give a location, where the project files will be stored.



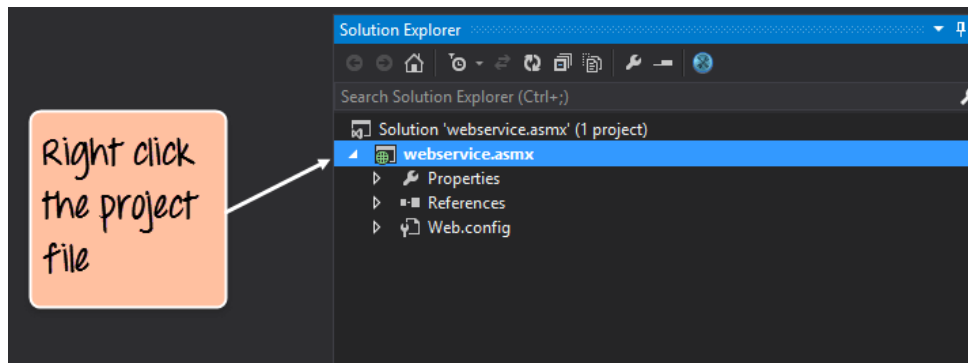
Once done you will see the project file created in your solution explorer in Visual Studio 2013.



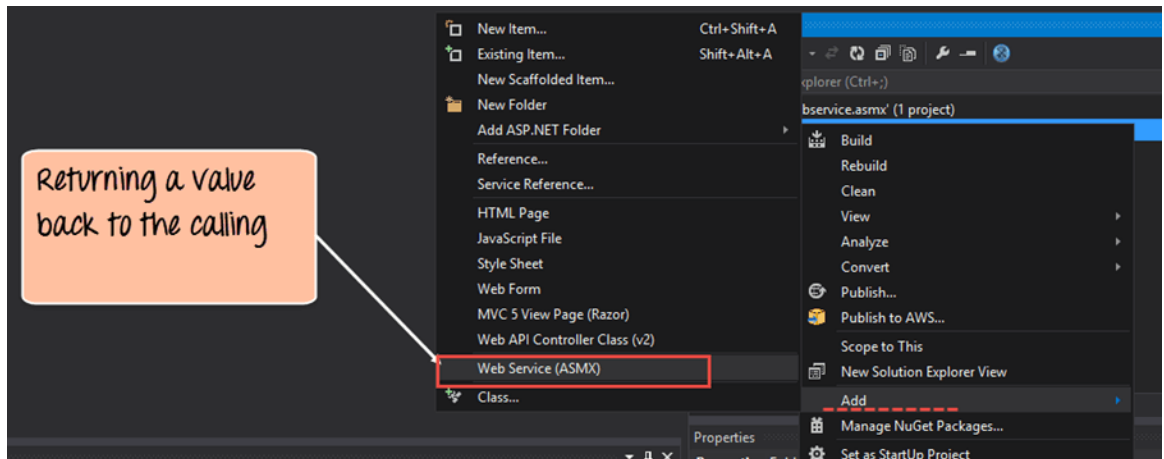
Step 3) In this step,

We are going to add a Web service file to our project

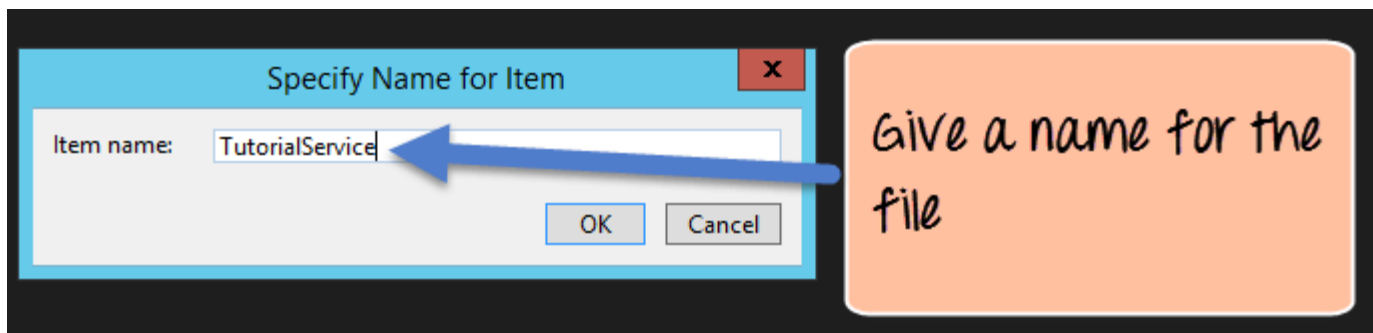
1. First Right-click on the project file as shown below



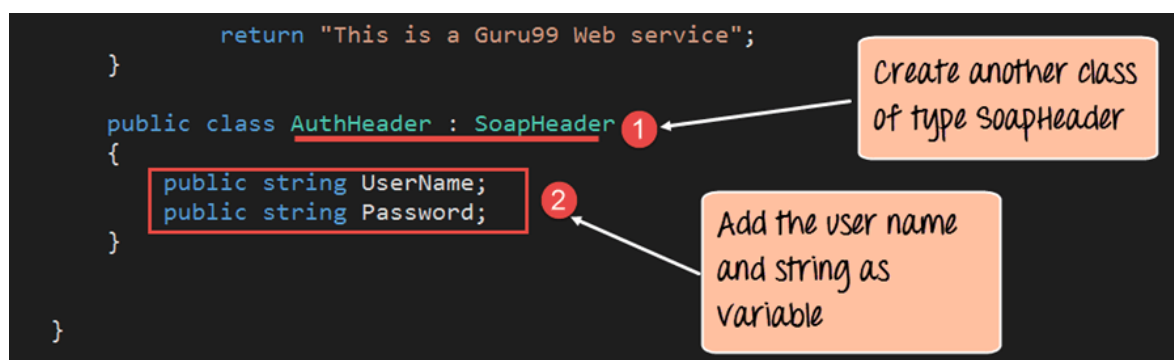
2. Once you right-click on the project file, you have the chance to choose the option "Add->Web Service (ASMX)" to add a web service file. Just provide a name of Tutorial Service for the web service name file.



The above step will prompt a dialog box, wherein one can enter the name of the web service file. So in the below dialog box, enter the name of TutorialService as the file name.



Step 4) Add the following code to your Tutorial Service asmx file. The below snippet of code is used to add a custom class which will be used to change the SOAP Header when the SOAP message is generated. Since we now want to add security credentials to the SOAP header, this step is required.



```

return "This is a Guru99 Web Service";
}

public class AuthHeader : SoapHeader
{

```



```

        public string UserName;
        public string Password;
    }
}

```

Code Explanation:-

1. We are now creating a separate class called **AuthHeader** which is of type **SoapHeader class**. Whenever you want to change what gets passed in the SOAP header, one needs to create a class which uses the in-built SoapHeader class of .Net. By customizing the SOAPheader, we now have the ability to pass a 'Username' and 'Password' when the web service is called.
2. We then define variables of 'UserName' and 'Password' which are of type string. They will be used to hold the values of the username and password which are passed to the web service.

Step 5) As the next step, the following code needs to be added to the same **TutorialService.asmx file**. This code actually defines the function of our web service. This function returns a string "This is a Guru99 Web service" to the client. But this time, the string will only be returned if the client application passes the credentials to the web service.

```

public class TutorialService : System.Web.Services.WebService
{
    public AuthHeader Credentials;
    [SoapHeader("Credentials")]
    [WebMethod]
    public string Guru99WebService()
    {
        if (Credentials.UserName.ToLower() != "Guru99" ||
            Credentials.Password.ToLower() != "Guru99Password")
        {
            throw new SoapException("Unauthorized",
                SoapException.ClientFaultCode);
        }
        else
            return "This is a Guru99 Web service";
    }
}

```

Annotations:

- 1. Create an object of the Credentials class
- 2. Ensure the Soapheader now has a security object
- 3. Code to validate the user name and password

```

public class TutorialService : System.Web.Services.WebService
{
    public AuthHeader Credentials;

    [SoapHeader("Credentials")]

```

```

[WebMethod]
public string Guru99WebService()
{
    if (Credentials.UserName.ToLower() != "Guru99" ||
        Credentials.Password.ToLower() != "Guru99Password")
    {
        throw new SoapException("Unauthorized",
            SoapException.ClientFaultCode);
    }
    else
        return "This is a Guru99 Web service";
}

```

Code Explanation:-

1. Here, we are creating an object of the AuthHeader class which was created in the earlier step. This object will be passed to our **Guru99Webservice** in which the username and password can be closely examined.
2. The [SoapHeader] attribute is used now to specify that when the Web service is called, it needs to have the user name and password passed.
3. In this block of code, we are actually examining the user name and password passed when the web service is called. If the Username is equal to "Guru99" and the password is equal to "Guru99Password" then the message of "This is a Guru99 Web service" is passed to the client. Else an error will be sent to the client if the wrong user id and password are passed.

If the code is executed successfully, the following Output will be shown when you run your code in the browser.

Output:



The above output is shown when the program is run, which means that the Web service is now available. Let's click on the Service Description link.



From the service description, you will now be able to see that the username and password are elements of the WSDL file. These parameters need to be sent when the web service is invoked.

Web Service Security Best Practices

Following are the security considerations which should be noted when working with Web services

1. **Auditing and Log management** – Use application logging to log all requests, which comes to the web services. This gives a detailed report on who has invoked the web service and can help in Impact analysis if any security breach occurs.
2. **Flow of calls to the web service** – Try to note the flow of the calls in web services. By default, an application could call multiple web services request with Authentication tokens passed between these web services. All calls between web services need to be monitored and logged.
3. **Sensitive Information** - Do not include sensitive information in your log entries such as passwords or credit card numbers or any sort of other confidential information. If there is an event which has any of this information, it needs to be discarded before logging.
4. **Track Business Operations** - Track significant business operations. For example, instrument your application to record access to particularly sensitive methods and business logic. Let's take an example of an online shopping application. There are multiple steps in a typical application such as the choosing the items to be purchased, the items loaded in the cart and then the final purchase. This entire business workflow needs to be tracked by the web service.
5. **Proper Authentication** - Authentication is the mechanism by which the clients can establish their identity with the web service using a certain set of credentials that can prove that identity. One should never store the user credentials, and hence, if WS Security is used to call the web service, it has to be noted that the web service should not store the credentials which are sent in the SOAP header. These should be discarded by the web service.

Summary

- SOAP provides an additional layer called WS Security for providing additional security when calls are made to Web services.
- The WS Security can be called with a simple username or password or can be used with Binary certificates for authentication
- We have seen that in .Net we can customize the Web service to have a user name and password passed as part of the SOAP header element.

