

حالتی را در نظر بگیرید که سرویس های یک برنامه در آدرس مشخص هاست شده اند. اگر اعتبار سنجی برای این سرویس ها در نظر گرفته نشود به راحتی می توان با در اختیار داشتن آدرس مورد نظر تمام سرویس های برنامه را فراخوانی کرد و اگر رمزگذاری اطلاعات بر روی سرویس ها فعال نشده باشد می توان تمام اطلاعات این سرویس ها را به راحتی به دست آورد. کمترین تلاش در این مرحله برای پیاده سازی امنیت این است که برای فراخوانی هر سرویس حداقل یک شناسه و رمز عبور چک شود و فقط در صورتی که فراخوانی سرویس همراه با شناسه و رمز عبور درست بود اطلاعات در اختیار کلاینت قرار گیرد. قصد داریم طی یک مثال این مورد را بررسی کنیم:

ابتدا یک پروژه با دو Console Application با نام های Service و Client ایجاد کنید. سپس در پروژه Service یک سرویس به نام BookService ایجاد کنید و کدهای زیر را در آن کپی نمایید:

Contract مربوطه به صورت زیر است:

```
[ServiceContract]
public interface IBookService
{
    [OperationContract]
    int GetCountOfBook();
}
```

کدهای مربوط به سرویس:

```
[ServiceBehavior(IncludeExceptionDetailInFaults = true)]
public class BookService : IBookService
{
    public int GetCountOfBook()
    {
        return 10;
    }
}
```

فایل Program در پروژه Service را باز نمایید و کدهای زیر را که مربوط به hosting سرویس مورد نظر است در آن کپی کنید:

```
class Program
{
    static void Main(string[] args)
    {
        ServiceHost host = new ServiceHost(typeof(BookService));

        var binding = new BasicHttpBinding();

        host.AddServiceEndpoint(typeof(IBookService), binding, "http://localhost/BookService");
        host.Open();

        Console.WriteLine("BookService host");

        Console.ReadKey();
    }
}
```

بر اساس کدهای بالا، سرویس BookService در آدرس <http://localhost/BookService> هاست می شود. نوع Binding نیز BasicHttpBinding انتخاب شده است.

حال نوبت به پیاده سازی سمت کلاینت می رسد. فایل Program سمت کلاینت را باز کرده و کدهای زیر را نیز در آن کپی نمایید:

```
static void Main(string[] args)
{
    Thread.Sleep(2000);
```

```

        BasicHttpBinding binding = new BasicHttpBinding();
        ChannelFactory<IBookService> channel = new ChannelFactory<IBookService>(binding, new
EndpointAddress("http://localhost/BookService"));
        Console.WriteLine("Count of book: {0}", channel.CreateChannel().GetCountOfBook());
        Console.ReadKey();
    }

```

در کدهای عملیات ساخت ChannelFactory برای برقراری اطلاعات با سرویس مورد نظر انجام شده است. پروژه را Build نمایید و سپس آن را اجرا کنید. خروجی زیر مشاهده می‌شود:



تا اینجا هیچ گونه اعتبارسنجی انجام نشد. برای پیاده سازی اعتبارسنجی باید یک سری تنظیمات بر روی Binding و Hosting سمت سرور و البته کلاینت برقرار شود. فایل Program پروژه Service را باز نمایید و محتویات آن را به صورت زیر تغییر دهید:

```

static void Main(string[] args)
{
    ServiceHost host = new ServiceHost(typeof(BookService));

    var binding = new BasicHttpBinding();
    binding.Security = new BasicHttpSecurity();
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;

    host.Credentials.UserNameAuthentication.UserNamePasswordValidationMode =
System.ServiceModel.Security.UserNamePasswordValidationMode.Custom;

    host.Credentials.UserNameAuthentication.CustomUserNamePasswordValidator = new
CustomUserNamePasswordValidator();

    host.AddServiceEndpoint(typeof(IBookService), binding, "http://localhost/BookService");
    host.Open();

    Console.WriteLine("BookService host");
    Console.ReadKey();
}

```

تغییرات اعمال شده:

ابتدا نوع Security در Binding را به حالت TransportCredentialOnly تنظیم کردیم. در یک جمله هیچ گونه تضمینی برای صحت اطلاعات انتقالی در این حالت وجود ندارد و فقط یک اعتبارسنجی اولیه انجام خواهد شد. در نتیجه هنگام استفاده از این حالت باید با دقت عمل نمود و نباید فقط به پیاده سازی این حالت اکتفا کرد. (Encryption اطلاعات سرویس‌ها مورد بحث این پست نیست)

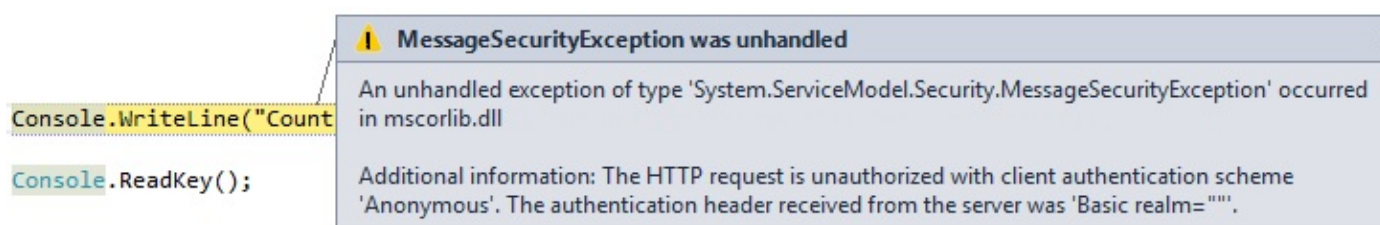
ClientCredentialType نیز باید به حالت Basic تنظیم شود. در WCF اعتبارسنجی به صورت پیش فرض در حالت Windows است (یعنی UserNamePasswordValidationMode برابر مقدار Windows است و اعتبارسنجی بر اساس کاربر انجام می‌شود). این مورد باید به مقدار Custom تغییر یابد. در انتها نیز باید مدل اعتبارسنجی دلخواه خود را به صورت زیر پیاده سازی کنیم: در پروژه سرویس یک کلاس به نام CustomUserNamePasswordValidator بسازید و کدهای زیر را در آن کپی کنید:

```
public class CustomUserNamePasswordValidator : UserNamePasswordValidator
{
    public override void Validate(string userName, string password)
    {
        if (userName != "Masoud" || password != "Pakdel")
            throw new SecurityException("Incorrect userName or password");
    }
}
```

Validator مورد نظر از کلاسی abstract به نام UserNamePasswordValidator ارث می‌برد، در نتیجه باید متد abstract به نام Validate را override نماید. در بدنه این متد شناسه و رمز عبور با یک مقدار پیش فرض چک می‌شوند و در صورت عدم درستی این پارامترها یک استثنا پرتاب خواهد شد.

تغییرات مورد نیاز سمت کلاینت:

اگر در این حالت پروژه را اجرا نمایید از آن جا که از این به بعد، درخواست‌ها سمت سرور اعتبار سنجی می‌شوند در نتیجه با خطای زیر روبرو خواهید شد:



این خطا از آن جا ناشی می‌شود که تنظیمات کلاینت و سرور از نظر امنیتی با هم تناسب ندارد. در نتیجه باید تنظیمات Binding کلاینت و سرور یکی شود. برای این کار کد زیر را به فایل Program سمت کلاینت اضافه می‌کنیم:

```
static void Main(string[] args)
{
    Thread.Sleep(2000);
    BasicHttpBinding binding = new BasicHttpBinding();

    binding.Security = new BasicHttpSecurity();
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;

    ChannelFactory<IBookService> channel = new ChannelFactory<IBookService>(binding, new
    EndpointAddress("http://localhost/BookService"));

    channel.Credentials.UserName.UserName = "WrongUserName";
    channel.Credentials.UserName.Password = "WrongPassword";

    Console.WriteLine("Count of book: {0}", channel.CreateChannel().GetCountOfBook());

    Console.ReadKey();
}
```

توسط دستور زیر، مقدار شناسه و رمز عبور به درخواست اضافه می‌شود.

```
channel.Credentials.UserName.UserName = "WrongUserName";
channel.Credentials.UserName.Password = "WrongPassword";
```

در اینجا UserName و Password اشتباه مقدار دهی شده اند تا روش کار Validator مورد بررسی قرار گیرد. حال اگر پروژه را اجرا نمایید خواهید دید که در Validator مورد نظر، عملیات اعتبار سنجی به درستی انجام می شود:

```
1 reference
public class CustomUserNamePasswordValidator : UserNamePasswordValidator
{
    0 references
    public override void Validate(string userName, string password)
    {
        if (userName != "Masoud" || password != "Pakdel")
            throw new SecurityException("WrongUserName or password");
    }
}
```

 **SecurityException was unhandled by user code**
An exception of type 'System.Security.SecurityException' occurred in Service.exe but was not handled in user code

[دریافت سورس مثال بالا](#)

نظرات خوانندگان

نویسنده:

بهمن

تاریخ:

۱۱:۵۲ ۱۳۹۲/۱۰/۲۱

سلام. ممنون به خاطر زحماتتون.

بر طبق آموزشهای گوناگون برای اعمال امنیت روی سرویس میتوان از Certificate هایی استفاده کرد که خودمان آنها را تولید کرده ایم. البته سفارش شده که در زمان برنامه نویسی و پیاده سازی پروژه از آن استفاده شود نه برای زمان واقعی استفاده از سرویس.

آیا این امکان وجود دارد که از Certificate هایی که خودمان ایجاد کرده ایم در پروژه های واقعی استفاده کنیم؟ اگر این امکان وجود دارد آیا این Certificate ها کار رمز گزاری و رمز گشایی را برای ما انجام میدهند؟ و چه محدودیتهایی دارند؟ با تشکر؟

نویسنده:

مسعود پاکدل

تاریخ:

۱۲:۴۵ ۱۳۹۲/۱۰/۲۱

اگر به مثال بالا دقت کرده باشید حتما متوجه شدید که از BasicHttpBinding استفاده کردم. دلیل این موضوع این است که BasicHttpBinding به صورت پیش فرض هیچ گونه تمهیدات امنیتی را بر روی سرویس ها در نظر نمی گیرد. اگر قصد پیاده سازی مثال بالا را به وسیله WsHttpBinding (این binding به صورت توکار مباحث رمزگذاری و امضای دیجیتال را در خود دارد) داشته باشیم حتما باید از Certificate ها بهره ببریم. در نتیجه برای پیاده سازی مثال بالا به روش WsHttpBinding از [makecert.exe](#) برای تولید certificate ها استفاده می شد (عموما در مثال ها و نمونه ها از همین روش استفاده میشود) که در اجرای واقعی سرویس ها مناسب نیست. در [Soap](#) این Certificate ها شامل اطلاعات رمزگذاری و مجوزها و کلیدهای عمومی و خصوصی خواهند بود در نتیجه از اهمیت به سزایی برخوردارند. برای حفظ امنیت سرویس ها توصیه می شود certificate ها را از یک CA (برای مثال [VeriSign](#)) خریداری شود یا حداقل می توانید از Microsoft Certificate Services که در ویندوزهای سرور نصب می شود استفاده نمایید. در واقع اگر یک Certificate Authority وجود نداشته باشد بهتر است از این روش استفاده نشود.

نویسنده:

مهرسا

تاریخ:

۱۳:۱۵ ۱۳۹۲/۱۲/۰۵

سلام؛ من کدهای شمارو امتحان کردم ولی در کلاینت من نمیتونم اینو پیدا کنم channel.Credentials برای من اینو داره channel.ClientCredentials هر چی هم گشتم نتونستم پیدااش کنم میگه کلاس Credentials وجود نداره

نویسنده:

مسعود پاکدل

تاریخ:

۱۴:۲ ۱۳۹۲/۱۲/۰۵

Credential خود یک property از نوع ClientCredential در نمونه های وهله سازی شده از ChannelFactory است. شما از روش Add Service Reference و proxy استفاده کرده اید در نتیجه ChannelFactory به صورت یک خاصیت در نمونه وهله سازی شده از client proxy در دسترس است. به صورت زیر عمل نمایید:

```
proxy.ChannelFactory.Credentials.UserName.UserName = "WrongUserName";
proxy.ChannelFactory.Credentials.UserName.Password = "WrongPassword";
```

در همین رابطه : [مقایسه بین روش Proxy و ChannelFactory](#)

نویسنده: مهرسا
تاریخ: ۱۱:۱۹ ۱۳۹۲/۱۲/۰۶

مرسی از جوابتون

امکان ست کردن تنظیمات سرور در وب کانفیگ هم هست؟ چون من سرویسو در یک وب سایت گذاشتم.

نویسنده: مسعود پاکدل
تاریخ: ۱۴:۱۵ ۱۳۹۲/۱۲/۰۶

بله. می‌توانید تمام تنظیمات را در فایل config قرار دهید. برای نمونه:

```
<behaviors>
  <serviceBehaviors>
    <behavior name="yourServiceNameBehavior">
      <serviceDebug includeExceptionDetailInFaults ="true"/>
      <serviceCredentials>
        <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="MyCustomUserNameValidator, service" />
      </serviceCredentials>
    </behavior>
  </serviceBehaviors>
</behaviors>

</system.serviceModel>
```

در صورتی که از certificate ها استفاده کرده اید آن را هم باید به صورت زیر در این بخش قرار دهید:

```
<serviceCertificate findValue="localhost" storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectName" />
```