Silverlight 4

پژوهش و نگارش: وحید نصیری
Vahid_Nasiri@Yahoo.Com
بهار ۱۳۸۹

فهرست مطالب

فصل ۱۵ – امنیت در **Silverlight** ۳۱۰
چگونه مدل امنیتی Silverlight ، کاربران را از سایت‌ها و برنامه‌های مخرب محافظت می‌کند؟ ۳۱۰
توصیه‌هایی جهت ایجاد برنامه‌های امن Silverlight ۳۱۲
مقابله با حملات XSS ۳۱۲
جلوگیری از استفاده‌ی غیرمجاز از فایل‌های XAP برنامه‌ی شما ۳۱۳
استفاده از برنامه‌های obfuscator مناسب جهت محافظت از کدهای Silverlight ۳۱۴
محافظت از داده‌های Isolated Storage ۳۱۵
امکانات رمزنگاری مهیا در Silverlight به صورت پیش فرض ۳۱۵
منع دسترسی کاربران ناشناس به فایل‌های XAP موجود در یک برنامه‌ی ASP.NET Web forms ۳۱۶
مخفی سازی محل قرارگیری فایل‌های XAP و تعیین اعتبار کاربران مجاز ۳۱۸
امن سازی دسترسی به WCF Services حین کار با Silverlight ۳۲۰
نکته‌ای در مورد فراهم آوردن صفحه‌ی Login توسط برنامه‌ی Silverlight ۳۲۸
بررسی روش حذف Meta Data مرتبط با یک WCF Service ۳۳۰

فصل ۱۵ – امنیت در Silverlight

برنامه‌های Silverlight در سه حالت قابل اجرا هستند:

- **درون مرورگر** : در این حالت برنامه‌ی Silverlight به عنوان قسمتی از یک سایت درون مرورگر اجرا شده و محدود است به سطوح دسترسی مرورگر.
- **خارج از مرورگر با دسترسی محدود (sandboxed)** : برنامه‌های Silverlight در این حالت امکان اجرا درون مرورگر و همچنین خارج از آن‌را خواهند داشت. در حین نصب در خارج از مرورگر ، ابتدا کاربر باید این عملیات را تائید نموده و مجوز آن‌را صادر نماید. سطوح دسترسی این نوع برنامه‌ها نیز همانند حالتی است که درون مرورگر اجرا می‌شوند (این مورد را در طی فصول آتی با جزئیات کاربردی بیشتری بررسی خواهیم نمود).
- **خارج از مرورگر با دسترسی بالا (trusted applications)** : این نوع برنامه‌ها نیز قابلیت اجرا درون مرورگر را دارند و هنگامیکه خارج از مرورگر نصب می‌شوند، ابتدا سطح دسترسی مورد نیاز را درخواست کرده و پس از تائید کاربر، عملکرد آن‌ها همانند یک فایل exe. نوشته شده با .NET Framework خواهد بود با این تفاوت که تنها به My Documents دسترسی داشته و همچنین توانایی کار و تعامل با اشیاء COM را نیز دارند. این نوع برنامه‌ها محدودیت‌های دسترسی به منابع شبکه یا درخواست مجوز صریح از کاربر برای انجام یک سری از امور را نخواهند داشت.

چگونه مدل امنیتی Silverlight ، کاربران را از سایت‌ها و برنامه‌های مخرب محافظت می‌کند؟

علاوه بر مدل امنیتی بسته‌ی مرورگرها که به برنامه‌های Silverlight اعمال می‌شوند، اگر قابلیت جدیدی به این مجموعه اضافه گردیده، بیشترین حد امنیت برای کار با آن‌ها درنظر گرفته شده است:

- **نصب محدود شده**: تنها مدیران سیستم مجوز نصب افزونه‌ی Silverlight را دارند.
- **نیاز به آغاز عملیات توسط کاربر**: برای مثال جهت خواندن اطلاعات فایلی و یا ذخیره سازی اطلاعاتی بر روی Hard-disk کاربر، حتما باید نمایش File Dialogs توسط کاربر درخواست و سپس عملیات مورد نظر آن‌ها تائید گردد. در غیر اینصورت هیچ نوع دسترسی خودکار دیگری به فضایی خارج از Isolated storage امن معرفی شده وجود نخواهد داشت. همچنین این نوع درخواست‌ها حتما باید توسط اعمال صفحه کلید و یا Mouse آغاز گردند و امکان آغاز خودکار آن‌ها توسط برنامه وجود ندارد.

- **بررسی Domain صفحات و فایل‌های XAP** : اگر صفحه‌ای که برنامه‌ی Silverlight را نمایش می‌دهد از یک Domain و فایل XAP معرفی شده در آن از Domain دیگری آدرس دهی شوند، تعامل بین این دو محدود خواهد گشت. در تگ Object ارائه دهنده‌ی افزونه‌ی Silverlight می‌توان به کمک ویژگی EnableHtmlAccess ، امکان دسترسی به المان‌های صفحه‌ی HTML و فراخوانی توابع جاوا اسکریپت آن‌را در Silverlight مهیا نمود. اگر منشاء صفحه‌ی در برگیرنده‌ی تگ Object و فایل XAP موجود یک Domain و Port باشند، این ویژگی به صورت پیش فرض به true تنظیم شده است؛ در غیر اینصورت مقدار آن false خواهد بود. همچنین لازم به ذکر است تا زمانیکه یک نوع و یا متد تعریف شده در یک برنامه‌ی Silverlight مزین به ویژگی ScriptableTypeAttribute و یا ScriptableMemberAttribute نگردند، از طریق کدهای جاوا اسکریپتی صفحه قابل فراخوانی نخواهند بود.
- **ملاحظات امنیتی حالت تمام صفحه (Full screen)** : برای ورود به حالت تمام صفحه نیز حتما درخواست کاربر باید از طریق صفحه کلید و یا Mouse دریافت گردد و برنامه به صورت خودکار مجوز ورود به این حالت را نخواهد داشت. همچنین تحت هر شرایطی پیغامی مشکی رنگ مبتنی بر ورود به این حالت به کاربر نمایش داده خواهد شد. به علاوه جهت جلوگیری از عدم جعل صفحاتی مانند ورود به سیستم و امثال آن، استفاده از صفحه کلید در این حالت بسیار محدود است.
- **دسترسی به Webcam/microphone** : این مورد در طی فصل‌های بعدی بیشتر توضیح داده خواهد شد. دسترسی به این موارد نیز حتما نیاز به درخواست و تائید کاربر را داشته و همچنین توسط مدیر سیستم هم قابل غیرفعال شدن است.
- **دسترسی به چاپگر**: این مورد نیز تنها پس از نمایش صفحه‌ی مربوطه و تائید دستی کاربر میسر خواهد بود و امکان ارسال صفحه‌ای به صورت خودکار به چاپگر مهیا نیست.
- **دسترس به Clipboard** : برنامه‌های Silverlight تنها پس از نمایش یک اخطار امنیتی و تائید کاربر، مجوز نوشتن و یا خواندن اطلاعات را در Clipboard سیستم خواهند داشت.
- **کشیدن و رها کردن (Drag and drop)**: در این حالت دسترسی به فایل کشیده شده و رها شده بر روی یک صفحه‌ی Silverlight همانند امکانات OpenFileDialog بوده و تنها به Stream فقط خواندنی آن فایل(ها) دسترسی خواهیم داشت. همچنین امکان کشیدن و رها کردن از یک برنامه Silverlight به برنامه‌ای که Silverlight نیست، پشتیبانی نمی‌شود.
- **کلیک راست Mouse** : در Silverlight 4 امکان مدیریت کلیک راست Mouse و نمایش منویی دلخواه وجود دارد. در این حالت نیز کاربران از طریق منوی Start ویندوز دسترسی به تنظیمات افزونه‌ی Silverlight خواهند داشت.
- **دسترسی به منابع شبکه**: در این مورد در طی فصل کار با Web Services توضیحات لازم ارائه شد و اگر سایت مقصد فایل‌های clientaccesspolicy.xml و یا crossdomain.xml را در ریشه‌ی سایت خود به همراه دسترسی‌های لازم ارائه ندهد، از طریق یک برنامه‌ی Silverlight هیچگونه دسترسی به

آن منابع وجود نخواهد داشت. لازم به ذکر است که همانند یک صفحه‌ی HTML ، امکان تعریف تگ‌های **<Image>** و **<Media>** نیز در یک برنامه‌ی Silverlight جهت دسترسی به تصاویر و فایل‌های چند رسانه‌ای سایر Domain ها بدون نیاز به فایل‌های امنیتی ذکر شده وجود دارد اما با این محدودیت که دسترسی به محتوای این منابع دریافت شده از طریق برنامه نویسی وجود نخواهد داشت. در حین کار با Sockets در Silverlight فایل clientaccesspolicy.xml باید از طریق Port شماره ۹۴۳ در دسترس باشد و همچنین این نوع برنامه‌ها به Port های ۴۵۰۲ تا ۴۵۳۴ محدود هستند.

- **تغییر اندازه‌ی برنامه‌های Silverlight** : جهت جلوگیری از تغییر اندازه‌ی غیرمجاز صفحات، تغییر اندازه‌ی آن‌ها نیز تنها با درخواست دستی کاربر میسر خواهد بود. در مورد برنامه‌هایی که خارج از مرورگر اجرا می‌شوند نیز وضعیت به همین صورت است.
- **حملات سرریز بافر**: از آنجائیکه برنامه‌های Silverlight نیز از کدهای مدیریت شده (.NET Managed codes) تشکیل می‌شوند، امکان حملات سرریز بافر در این نوع برنامه‌ها در حد صفر است.
- **امکان غیرفعال کردن یک سری از ویژگی‌های Silverlight توسط مدیران سیستم**: اگر در شبکه‌ی خود علاقمند به بستن یک سری از ویژگی‌های Silverlight مانند دسترسی به Webcam و امثال آن هستید، تنها کافی است با دسترسی مدیریتی کلیدهای رجیستری واقع در مسیر زیر را تغییر دهید (برای مثال AllowWebcam=0):

HKey_Local_Machine\Software\Microsoft\Silverlight

- **به روز رسانی خودکار افزونه‌ی Silverlight** : به روز رسانی‌های جدید افزونه‌ی Silverlight به صورت خودکار توسط سیستم به روز رسانی آن هر هفت روز یکبار (اگر Windows update فعال باشد) و یا هر ۳۰ روز یکبار اگر Windows update غیرفعال گردد، بررسی و دریافت خواهند شد.
- **عدم اجرا با سطح دسترسی مدیریتی**: برنامه‌هایی که خارج از مرورگر اجرا می‌شوند همواره بدون دسترسی مدیریتی به فعالیت خواهند پرداخت (شبیه به UAC در ویندوزهای ویستا به بعد). حتی اگر این نوع برنامه‌ها را عمدا به صورت دستی با دسترسی مدیریتی اجرا نمائیم، پروسه‌ی دومی از برنامه با سطح دسترسی عادی و محدود تشکیل شده و پروسه‌ی قبلی خاتمه خواهد یافت.

توصیه‌هایی جهت ایجاد برنامه‌های امن Silverlight

مقابله با حملات XSS

همانند برنامه‌های متداول مبتنی بر HTML ، برنامه نویسان Silverlight نیز باید اطلاعات کافی را در مورد نحوه‌ی مقابله با حملات یا cross site scripting (XSS) کسب کنند. در طی یک حمله‌ی XSS، مهاجم ممکن است

به اطلاعات سمت کاربر مانند اطلاعات Cookies ، Isolated storage و امثال آن دسترسی پیدا کند. البته باید در نظر داشت که امکان حملات XSS به سایت‌های مبتنی بر Silverlight بسیار کم بوده و محدود به موارد بعد می‌شود که باید به آن‌ها دقت داشت:

یک TextBox ساده Silverlight را در نظر بگیرید که خاصیت متن آن به صورت زیر مقدار دهی شده است:

```
textblock.Text = attackerString;
```

در این حالت خطری متوجه کاربران نخواهد بود زیرا ورودی دریافتی در اینجا تنها به صورت یک رشته‌ی ساده تفسیر خواهد شد. اما در مثال زیر که یک تگ XAML پویا را ارائه خواهیم داد، حتما باید ورودی کاربر تعیین اعتبار گردد (البته این نوع برنامه نویسی در Silverlight بسیار نادر و محدود است) :

```
XamlReader.Load("<TextBlock Text=’” + attackerString + "’/>”);
```

تنها در موارد زیر است که امکان حملات XSS در Silverlight وجود داشته و باید ورودی‌های کاربران را در این موارد کاملا بررسی نمود:

۱. استفاده مستقیم از متد XamlReader.Load به همراه رشته ارائه شده توسط کاربر
۲. استفاده مستقیم از متد AssemblyPart.Load به همراه DLL ارائه شده توسط مهاجم
۳. تشکیل XML با ترکیب رشته‌ها جهت ارسال به REST services. در این حالت‌ها بهتر است از System.Xml.XmlWriter استفاده گردد.
۴. تولید HTML به کمک Silverlight با استفاده از امکانات فضای نام System.Windows.Browser .
۵. اجازه دادن به کاربر جهت Upload فایل‌های XAP و استفاده و نمایش آن‌ها. در Silverlight چون MIME Type دریافتی باید از طرف Web Server به صورت application/x-silverlight-app ارائه گردد، نمی‌توان یک فایل XAP را به شکل DOCX تغییر پسوند داده و پس از Upload، همانند یک برنامه‌ی Silverlight از آن استفاده نمود.

جلوگیری از استفاده‌ی غیرمجاز از فایل‌های XAP برنامه‌ی شما

برای اینکه بررسی نمود آیا میزبانی که فایل‌های XAP شما را مورد استفاده قرار می‌دهد، همان میزبان مجاز مورد نظر شما است، کدهای زیر را به سازنده‌ی کلاس App برنامه خود اضافه نمائید (مزیت استفاده از سازنده‌ی کلاس نسبت به روال رخدادگردان آغازین برنامه، امکان عدم بارگذاری برنامه پیش از هر رخداد ممکنی است):

App.xaml.cs

```
public App()
{
   if (App.Current.Host.Settings.EnableHTMLAccess == false)
               throw new Exception();
```

```
    string htmlurl =
              System.Windows.Browser.HtmlPage.
              Document.DocumentUri.ToString().ToLower();
    if (htmlurl != "http://foo.com/mypage.html") //آدرس سایت شما
                throw new Exception();

    string xapServer = this.Host.Source.ToString();
    //بررسی دقیق تر نام و محل قرارگیری فایل‌های برنامه
    if (xapServer != "http://localhost:60338/TestApp.xap")
    {
      throw new Exception("Application came from an unexpected server");
    }

    this.Startup += this.Application_Startup;
    this.Exit += this.Application_Exit;
    this.UnhandledException += this.Application_UnhandledException;

    InitializeComponent();
}
```

در اینجا ابتدا بررسی می‌شود که آیا دسترسی به HTML وجود دارد یا خیر؟ اگر خیر به این معنا است که سایت دیگری از مسیر فایل XAP شما سوء استفاده نموده و مشغول به بکارگیری غیر مجاز از آن شده است؛ زیرا همانطور که پیشتر نیز ذکر شد، تنها اگر Domain صفحه و Domain فایل XAP یکسان نباشند، ویژگی EnableHTMLAccess به false تنظیم می‌گردد.

همچنین ممکن است کلا صفحه‌ی دربرگیرنده و فایل XAP را توسط یک میزبان دیگر ارائه دهند. در این حالت هم می‌توان مطابق بررسی دوم، مسیر قرارگیری اصلی را بررسی نمود و در صورت عدم تحقق شرایط ، برنامه را از کار انداخت.

بررسی سوم صورت گرفته در مورد نام و آدرس دقیق سرور ارائه دهنده‌ی XAP فایل ما است که در صورت استفاده‌ی غیرمجاز، سبب از کار افتادن برنامه می‌شود.

استفاده از برنامه‌های obfuscator مناسب جهت محافظت از کدهای Silverlight

فایل‌های XAP برنامه‌های شما، توسط مرورگرها قابل دریافت و بررسی هستند. بنابراین امکان مهندسی معکوس برنامه‌های Silverlight نیز به همین ترتیب میسر است (برای مثال با استفاده از برنامه‌ی .NET Reflector). بنابراین به هیچ عنوان اطلاعات حساسی را در کدهای Silverlight خود قرار ندهید. همچنین

می‌توان ابزارهای Code obfuscation را با پروسه‌ی Build برنامه یکپارچه نمود. برای نمونه نحوه‌ی یکپارچه سازی برنامه‌ی رایگان Babel Obfuscator با پروسه‌ی Build در VS.NET به شرح ذیل است:

۱. ابتدا نگارش رایگان این برنامه را از آدرس زیر دریافت نمائید:
http://www.babelfor.net/Downloads.aspx
۲. برنامه‌ی رایگان 7-Zip را از آدرس زیر جهت اعمال فشرده سازی مورد نیاز دریافت کنید:
http://www.7-zip.org/
۳. به خواص پروژه‌ی Silverlight خود مراجعه نموده و در برگه‌ی Build Events آن، دستورات ذیل را در قسمت Post-build event command line وارد نمائید:

```
"C:\Program Files\Babel\babel.exe" $(TargetPath) --noildasm --nomsil --noinvalidopcodes
"C:\Program Files\7-Zip\7z.exe" a $(TargetDir)$(ProjectName).xap
$(TargetDir)BabelOut\$(TargetFileName) -y -tZIP
```

به این صورت فایل اسمبلی تولیدی برنامه‌ی ما پس از پایان رخداد Build به صورت خودکار وارد یک پروسه‌ی Code obfuscation و سپس فشرده سازی می‌شود تا فایل XAP نهایی امن‌تری تشکیل گردد.

نکته‌ی مهمی را که باید اینجا در نظر داشت این است که اکثر اینگونه ابزارها از Obfuscation اسمبلی‌های حاوی کدهای XAML برنامه عاجز هستند. بنابراین بهتر است تمامی کدهای C# یا VB.NET برنامه‌ی خود را توسط یک پروژه Class library تهیه کنید تا Obfuscation آن به سادگی و مجزای از اسمبلی اصلی برنامه صورت گیرد. بدیهی است استفاده از الگوی MVVM ، جدا سازی کدهای برنامه را از رابط کاربری آن تسهیل بخشیده و به این صورت یکی از انتخاب‌های مناسب برای توسعه‌ی برنامه‌های امن نیز به شمار خواهد رفت.

محافظت از داده‌های Isolated Storage

حتما کلیه اطلاعات حساس قابل ذخیره سازی در Isolated storage را پیش از ذخیره سازی، رمزنگاری نمائید؛ زیر این اطلاعات توسط مدیر سیستم و یا سایر برنامه‌هایی در همان Domain و Port و یا در حالاتی که حملات DNS رخ داده است، قابل دسترسی خواهند بود.

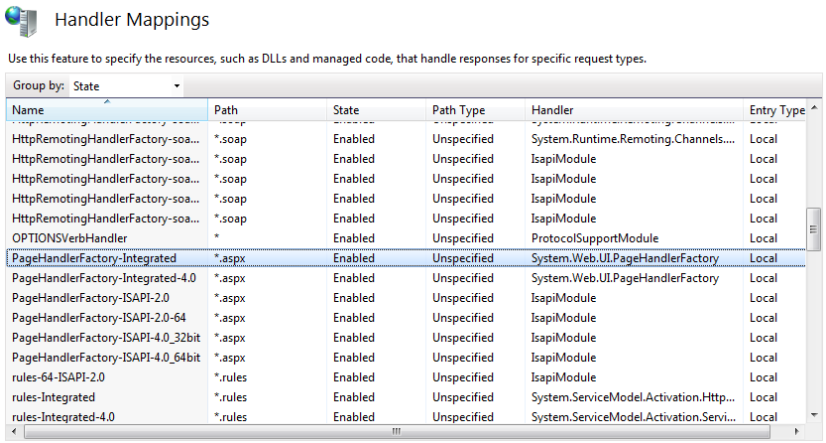
امکانات رمزنگاری مهیا در Silverlight به صورت پیش فرض

- پروتکل HTTPS به صورت کامل توسط Silverlight پشتیبانی می‌شود و اگر برنامه‌ای تحت این شرایط ارائه گردد کلیه تراکنش‌های آن نیز باید از طریق همین پروتکل ارائه گردد.
- فضای نام System.Security.Cryptography در Silverlight از الگوریتم‌های AES برای رمزنگاری اطلاعات و SHA1 ، SHA256 و HMAC برای عملیات Hashing و بررسی امضای دیجیتال پشتیبانی به عمل می‌آورد.

- Silverlight از DRM جهت ارائه‌ی محتوای چندرسانه‌ای پشتیبانی می‌کند.

منع دسترسی کاربران ناشناس به فایل‌های XAP موجود در یک برنامه‌ی ASP.NET Web forms

در یک برنامه‌ی ASP.NET با استفاده از سیستم‌های اعتبار سنجی استاندارد آن مانند Forms Authentication و امثال آن، می‌توان سطوح دسترسی متفاوتی را برای صفحات سایت تعریف نمود. اما باید در نظر داشت که این اعتبار سنجی و عدم ارائه‌ی محتوا صرفا برای یک سری از پسوندهای شناخته شده‌ی ASP.NET عمل کرده و فایل‌های XAP از این امر مستثنا هستند. برای رفع این نقیصه می‌توان یک نگاشت جدید را جهت فایل‌هایی با پسوند XAP در قسمت handler mappings برنامه IIS تعریف کرد (شکل ۱) و یا می‌توان یک HttpModule جدید را نیز جهت بسط سیستم اعتبار سنجی پیش فرض، تعریف نمود.



شکل ۱– نگاشت‌های پیش فرض راه اندازهای تعریف شده در IIS 7.x .

برای این منظور یک پروژه‌ی Silverlight جدید را به همراه ASP.NET Web site آن آغاز نمائید. سپس کلاس محافظت کننده‌ی ذیل را به آن اضافه کنید:

ProtectXap.cs

```
using System;
using System.Web;

namespace SilverlightApplication65.Web
{
    public class ProtectXap : IHttpModule
    {
        public void Dispose()
        { }

        public void Init(HttpApplication context)
        {
```

```
            context.EndRequest += context_EndRequest;
        }

        static void context_EndRequest(object sender, EventArgs e)
        {
            var app = (HttpApplication)sender;
            HttpContext context = app.Context;

            if (context.Request.Url.AbsolutePath
                                .ToUpper().Contains(".XAP"))
            {
                if (context.User == null ||
                    context.User.Identity.IsAuthenticated == false)
                    context.Response.Redirect(
                      "login.aspx?ReturnUrl=" + context.Request.Url);
            }
        }
    }
}
```

دراین کلاس تمامی درخواست‌هایی که حاوی یک فایل XAP باشند بررسی شده و در صورتیکه کاربر جاری اعتبار سنجی نشده باشد، به صفحه‌ی login هدایت خواهد شد. برای ثبت این httpModule و همچنین httpHandler مرتبط با فایل‌های XAP جهت محافظت از آن‌ها، فایل Web.Config برنامه را باید مطابق تغییرات ذیل ویرایش نمود :

Web.config

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />

    <authentication mode="Forms">
      <!--one month ticket-->
      <forms
        name=".403AuthXap"
        cookieless="UseCookies"
        loginUrl="login.aspx"
        defaultUrl="default.aspx"
        slidingExpiration="true"
        protection="All"
        path="/"
        timeout="43200"
        />
    </authentication>

    <!--برای آی آی اس ٦ لازم است-->
    <httpModules>
     <add type="SilverlightApplication65.Web.ProtectXap, SilverlightApplication65.Web"
```

```
          name="ProtectXap" />
    </httpModules>
    <httpHandlers>
      <add path="*.xap" verb="*"
           type="System.Web.StaticFileHandler"
           validate="false"  />
    </httpHandlers>
  </system.web>


  <!--برای آی آی اس ۷ لازم است-->
  <system.webServer>
    <modules>
      <add name="ProtectXap"
       type="SilverlightApplication65.Web.ProtectXap, SilverlightApplication65.Web"/>
    </modules>
    <handlers>
      <add name="ProtectXap" verb="POST,GET,HEAD,DEBUG" path="*.xap"
           type="System.Web.StaticFileHandler" />
    </handlers>
  </system.webServer>
</configuration>
```

در این فایل تنظیمات لازم جهت IIS 6.0 و همچنین IIS 7.x نیز ذکر شده‌اند.

اکنون پس از این تغییرات اگر برنامه‌ی Silverlight فوق را اجرا کنید، کاربران اعتبار سنجی نشده موفق به مشاهده‌ی محتوای فایل XAP برنامه نخواهند شد. ابتدا باید از طریق سیستم Forms Authentication اعتبار سنجی شوند و سپس مجاز خواهند بود تا به صفحه‌ی حاوی برنامه‌ی Silverlight در ASP.NET Web site جاری مراجعه نمائید.

مخفی سازی محل قرارگیری فایل‌های XAP و تعیین اعتبار کاربران مجاز

علاوه بر روش قبل، می‌توان محتوای XAP برنامه را توسط یک HttpHandler نیز ارائه داد. برای این منظور از منوی پروژه، گزینه‌ی افزودن یک آیتم جدید، یک Generic handler را به نام GetXap.ashx به برنامه‌ی ASP.NET پروژه جاری اضافه کنید. سپس کدهای آن را به شرح بعد تغییر دهید:

GetXap.ashx.cs

```
using System.IO;
using System.Web;

namespace SilverlightApplication67.Web
{
```

```
    /// <summary>
/// Summary description for GetXap
/// </summary>
public class GetXap : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        if (!context.User.Identity.IsAuthenticated)
            return;

        // Render XAP Stream Back To Client
        var buffer = File.ReadAllBytes(
            context.Server.MapPath(
            "ClientBin/SilverlightApplication67.xap"));

        context.Response.ContentType =
                    "application/x-silverlight-app";
        context.Response.BinaryWrite(buffer);
    }

    public bool IsReusable
    {
        get
        {
            return false;
        }
    }
}
}
```

یکی از روش‌های محافظت از فایل‌های غیرپویا مانند تصاویر و امثال آن در برنامه‌های ASP.NET ، استفاده از یک HttpHandler می‌باشد که نحوه‌ی پیاده سازی آن‌را ملاحظه می‌نمائید. در اینجا پس از بررسی اینکه آیا کاربر جاری اعتبار سنجی شده است یا خیر، محتوای فایل ClientBin/SilverlightApplication67.xap خوانده شده و به مرورگر کاربر ارائه می‌شود.

سپس نحوه‌ی استفاده از آن در فایلی که تگ Object متناظر با افزونه‌ی Silverlight در آن قرار می‌گیرد به شکل بعد خواهد بود:

ASPX

```
<object data="data:application/x-silverlight-2,"
     type="application/x-silverlight-2" width="100%" height="100%">
        <param name="source" value="GetXap.ashx"/>
...
```

به این ترتیب دیگر کاربران با مراجعه به Source فایل فوق، نام و مسیر فایل‌های XAP را نخواهد دید و همچنین امکان ارائه‌ی آن‌ها به کاربران اعتبار سنجی نشده نیز به سادگی وجود نخواهد داشت.

امن سازی دسترسی به WCF Services حین کار با Silverlight

هنگام کار با WCF Services توسط برنامه‌های Silverlight ، دو مشکل امنیتی مهم باید در نظر گرفته شوند:

۱. تبادل اطلاعات از طریق basicHttpBinding ، یعنی ارسال داده‌ها به صورت Text ساده و با استفاده از packet sniffers قابل دریافت و آنالیز کامل هستند.
۲. تنها برنامه‌های Silverlight هستند که به فایل‌های clientaccesspolicy.xml جهت بررسی دسترسی به یک WCF Service احترام می‌گذارند. سایر برنامه‌ها به سادگی می‌توانند متدهای یک WCF Service را صدا زده و اطلاعات لازم را از آن دریافت کنند.

مشکل اول را می‌توان با استفاده از پروتکل HTTPS برطرف نمود. مورد دوم را با توجه به اینکه عموما و در اکثر موارد WCF Service و همچنین برنامه‌ی Silverlight توسط یک ASP.NET Web Site ارائه خواهند شد، می‌توان با بهره‌گیری از مدل‌های امنیتی ASP.NET نسبت به امن سازی دسترسی به WCF Service اقدام نمود. برای این منظور مثال کاملی را در ادامه با هم مرور خواهیم نمود:

ابتدا یک پروژه‌ی جدید Silverlight را به همراه ASP.NET Web site آن، ایجاد نمائید. سپس صفحات جدید login.aspx و default.aspx و یک Silverlight-enabled WCF Service را همانطور که در فصل به کارگیری Web Services در Silverlight ذکر گردید، به نام TestService.svc به برنامه‌ی ASP.NET پروژه‌ی جاری اضافه کنید.

در ادامه تنظیمات زیر را به Web.Config برنامه‌ی ASP.NET اعمال نمائید. به این صورت تمام Site محافظت خواهد شد (منجمله WCF Service برنامه) و تنها پس از اعتبار سنجی در حالت Forms یا Windows، امکان دسترسی به امکانات WCF Service وجود خواهد داشت.

Web.config

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />
    <authentication mode="Forms">
      <!--one month ticket-->
      <forms
        name=".403AuthXap"  cookieless="UseCookies"
        loginUrl="login.aspx"  defaultUrl="default.aspx"
        slidingExpiration="true"  protection="All"
        path="/"  timeout="43200" />
    </authentication>

    <authorization>
```

```
        <deny users="?"/>
    </authorization>
  </system.web>
...
```

بدیهی است این تنظیمات صرفا به فایل‌های برنامه‌های ASP.NET اعمال شده و فایل‌های XAP جزو این مجموعه نیستند که روش محافظت از آن‌ها با یک HttpModule سفارشی در قسمت قبل عنوان گردید.

در ادامه دو متد را جهت بازگردان نوع اعتبار سنجی برنامه ASP.NET جاری و نام کاربر اعتبار سنجی شده، توسط این WCF Service ارائه خواهیم داد:

TestService.svc.cs

```
using System.ServiceModel;
using System.ServiceModel.Activation;
using System.Web;

namespace SilverlightApplication66.Web
{
    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode
        = AspNetCompatibilityRequirementsMode.Allowed)]
    public class TestService
    {
        [OperationContract]
        public string GetAuthenticationType()
        {
          return HttpContext.Current.User.Identity.AuthenticationType;
        }

        [OperationContract]
        public string GetLoggedUser()
        {
          return HttpContext.Current.User.Identity.Name;
        }
    }
}
```

همانطور که پیشتر نیز ذکر شد از آنجائیکه یک Silverlight-enabled WCF Service را به برنامه اضافه کرده‌ایم، تنظیمات پیش فرض آن جهت سازگاری با موتور ASP.NET (aspNetCompatibilityEnabled=true) و همچنین Silverlight مناسب می‌باشند.

سپس قصد داریم کدهای ساده‌ای را جهت صفحات login و پیش فرض برنامه ایجاد نمائیم. در کلاس SecurityManager می‌توانید منطق اعتبار سنجی سفارشی خود را مانند دریافت اطلاعات از یک بانک اطلاعاتی، در سمت سرور پیاده سازی نمائید :

SecurityManager.cs

```
namespace SilverlightApplication66.Web
```

```
{
    public class SecurityManager
    {
        public static bool Authenticate(
                string username, string password)
        {
            //منطق اعتبار سنجی سفارشی خود را در اینجا پیاده سازی نمائید
            return username == password;
        }
    }
}
```

سپس کدهای صفحه login برنامه به شرح زیر خواهند بود:

login.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeBehind="login.aspx.cs" Inherits="SilverlightApplication66.Web.login" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div align="center">
        <asp:Login ID="ctrlLogin" runat="server"
             OnAuthenticate="ctrlLogin_Authenticate" />
    </div>
    </form>
</body>
</html>
```

و مثالی ساده در مورد نحوه‌ی استفاده از کنترل Login و سیستم اعتبار سنجی مبتنی بر Forms در ASP.NET در ادامه ذکر گردیده است:

login.aspx.cs

```
using System;
using System.Web.Security;
using System.Web.UI.WebControls;

namespace SilverlightApplication66.Web
{
    public partial class login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {}
        protected void ctrlLogin_Authenticate(object sender,
```

```
                AuthenticateEventArgs e)
        {
            if (SecurityManager.Authenticate(
                ctrlLogin.UserName, ctrlLogin.Password))
            {
                //authenticated
                FormsAuthentication.RedirectFromLoginPage(
                    ctrlLogin.UserName, ctrlLogin.RememberMeSet);
            }
        }
    }
}
```

با توجه به تنظیمات Forms authentication ذکر شده در فایل Web.Config ، کاربر پس از اعتبار سنجی به صفحه‌ی default.aspx رهنمون خواهد شد. بنابراین در اینجا یک Response.Redirect ساده برای هدایت کاربر به صفحه‌ی آزمایشی برنامه‌ی Silverlight قرار داده شده است:

default.aspx.cs

```
using System;

namespace SilverlightApplication66.Web
{
    public partial class _default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
          Response.Redirect("SilverlightApplication66TestPage.aspx");
        }
    }
}
```

تا اینجا کار امن سازی برنامه و سرویس WCF ایجاد شده به پایان می‌رسد و تنها از طریق امکانات برنامه‌ی ASP.NET و مشخصات کاربر اعتبار سنجی شده، امکان دسترسی به این WCF Service وجود خواهد داشت.

اکنون قصد داریم کدهای کلاینت Silverlight پروژه جاری را (شکل ۲) برای اتصال به این WCF Service و نمایش اطلاعات آن، توسعه دهیم. این مثال را به کمک الگوی MVVM پیاده سازی کرده و از کلاس معروف DelegateCommand توسعه داده شده در فصل‌های قبلی کتاب جاری، استفاده خواهیم نمود.

ابتدا نیاز است تا ارجاعی را به WCF Service ایجاد شد در پروژه ASP.NET به پروژه‌ی Silverlight خود اضافه نمائیم (توسط منوی پروژه، گزینه‌ی Add Service reference). اگر به این طریق عمل نمائید موفق نخواهید شد تا ارجاعی را اضافه نمائید زیرا کلاینت افزودن ارجاع به WCF Service ، اعتبار سنجی نشده است و دسترسی لازم را برای انجام اینکار ندارد. برای این منظور ابتدا قسمت authorization ذکر شده در Web.Config برنامه‌ی ASP.NET را به طور موقت حذف نموده، سپس به برنامه‌ی Silverlight مراجعه کرده

و ارجاع لازم را به WCF Service اضافه نمائید. اکنون مجددا کدهای قسمت authorization را به فایل Web.Config اضافه کنید.

مدل برنامه‌ی Silverlight ما از دو خاصیت دریافتی از WCF Service به همراه پیاده سازی اینترفیس INotifyPropertyChanged تشکیل می‌گردد:

ServiceInfo.cs

```
using System.ComponentModel;

namespace SilverlightApplication66.Model
{
    public class ServiceInfo : INotifyPropertyChanged
    {
        string _authenticationType;
        public string AuthenticationType
        {
            get { return _authenticationType; }
            set
            {
                if (_authenticationType == value) return;
                _authenticationType = value;
                raisePropertyChanged("AuthenticationType");
            }
        }

        string _identityName;
        public string IdentityName
        {
            get { return _identityName; }
            set
            {
                if (_identityName == value) return;
                _identityName = value;
                raisePropertyChanged("IdentityName");
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        void raisePropertyChanged(string propertyName)
        {
            var handler = PropertyChanged;
            if (handler == null) return;
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

سپس ViewModel برنامه، کار ارائه‌ی خاصیت ServiceInfo و دو Comamnd به نام‌های GetAuthType و GetLoggedInUserName را به View برنامه یا همان صفحه‌ی اصلی پروژه‌ی Silverlight ، انجام می‌دهد :

ServiceInfoViewModel.cs

```
using System.Windows.Input;
using SilverlightApplication66.Helper;
using SilverlightApplication66.Model;
using SilverlightApplication66.TestServiceReference;

namespace SilverlightApplication66.ModelViews
{
    public class ServiceInfoViewModel
    {
        public ServiceInfoViewModel()
        {
            ServiceInfo = new ServiceInfo();
            GetLoggedInUserName = new DelegateCommand<ServiceInfo>(
                getLoggedInUserName, canGetLoggedInUserName);
            GetAuthType = new DelegateCommand<ServiceInfo>(
                getAuthType, canGetAuthType);
        }

        public ICommand GetAuthType { set; get; }

        public ICommand GetLoggedInUserName { set; get; }

        public ServiceInfo ServiceInfo { set; get; }

        private bool canGetAuthType(ServiceInfo arg)
        {
            return true;
        }

        private bool canGetLoggedInUserName(ServiceInfo arg)
        {
            return true;
        }

        private void getAuthType(ServiceInfo obj)
        {
            var srv = new TestServiceClient();
            srv.GetAuthenticationTypeCompleted +=
                      srv_GetAuthenticationTypeCompleted;
            srv.GetAuthenticationTypeAsync();
        }

        private void getLoggedInUserName(ServiceInfo obj)
        {
```

```
            var srv = new TestServiceClient();
            srv.GetLoggedUserCompleted += srv_GetLoggedUserCompleted;
            srv.GetLoggedUserAsync();
        }

        void srv_GetAuthenticationTypeCompleted(object sender,
            GetAuthenticationTypeCompletedEventArgs e)
        {
            if (e.Error == null)
                ServiceInfo.AuthenticationType = e.Result;
        }

        void srv_GetLoggedUserCompleted(object sender,
            GetLoggedUserCompletedEventArgs e)
        {
            if (e.Error == null)
                ServiceInfo.IdentityName = e.Result;
        }
    }
}
```

با جزئیات اعمال غیرهمزمان کار با WCF Services در طی فصل مرتبط با آن پیشتر آشنا شده‌ایم. در اینجا نیز به همین ترتیب اطلاعات لازم، دریافت و به دلیل اینکه شیء ServiceInfo اینترفیس INotifyPropertyChanged را پیاده سازی نموده است، هر تغییری در مقدار خواص آن، بلافاصله از طریق سیستم Binding ، در اختیار View برنامه نیز قرار می‌گیرد.

در ادامه کدهای XAML برنامه را مشاهده می‌نمائید:

MainPage.xaml

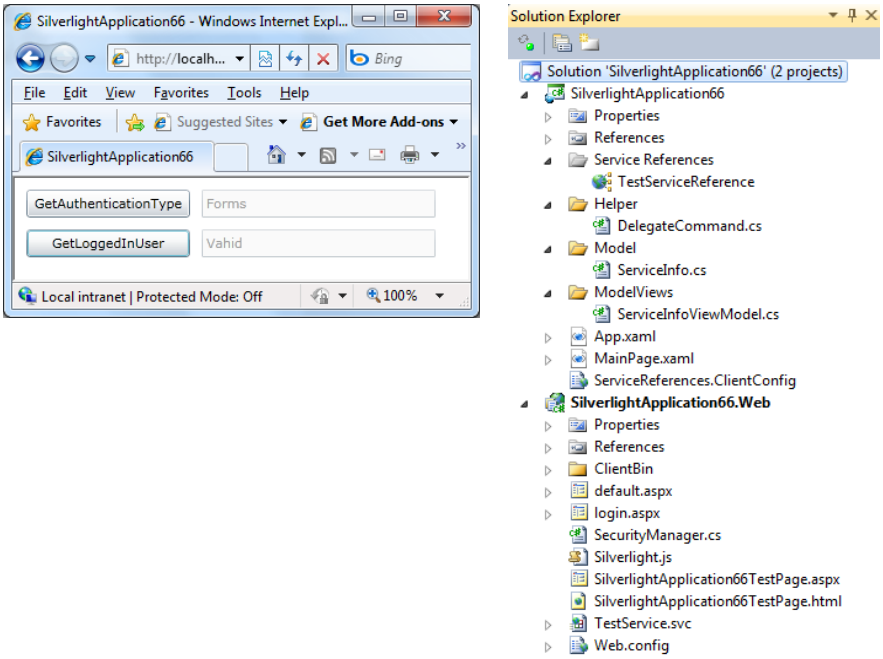
```
<UserControl x:Class="SilverlightApplication66.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:
     mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:vm="clr-namespace:SilverlightApplication66.ModelViews"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <vm:ServiceInfoViewModel x:Key="vmServiceInfoViewModel" />
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White"
          Margin="5"
          DataContext="{StaticResource vmServiceInfoViewModel}"
          >
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
```

```
            <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Button Grid.Row="0" Grid.Column="0"
                Command="{Binding GetAuthType}"
                Margin="5"
                CommandParameter="{Binding ServiceInfo}"
            Content="GetAuthenticationType"  />
        <TextBox Grid.Row="0" Grid.Column="1"
                 Margin="5" HorizontalAlignment="Left"
                 Text="{Binding ServiceInfo.AuthenticationType}"
            Width="200" IsEnabled="False" />
        <Button Grid.Row="1" Grid.Column="0"
                Command="{Binding GetLoggedInUserName}"
                Margin="5"
                CommandParameter="{Binding ServiceInfo}"
            Content="GetLoggedInUser"  />
        <TextBox Grid.Row="1" Grid.Column="1"
                 Margin="5" HorizontalAlignment="Left"
                 Text="{Binding ServiceInfo.IdentityName}"
            Width="200" IsEnabled="False" />
    </Grid>
</UserControl>
```

در این View ابتدا ViewModel برنامه در قسمت منابع User control جاری اضافه شده است و سپس به DataContext گرید جاری انتساب می‌یابد. کلیه اعمال کلیک بر روی دکمه‌ها و همچنین دریافت اطلاعات نهایی از طریق سیستم Binding مدیریت خواهند شد.

اکنون اگر پروژه را در VS.NET آغاز نمائید، ابتدا به صفحه‌ی login هدایت خواهید شد؛ زیرا تمام صفحات ASP.NET Site برنامه، محافظت شده‌اند و صفحه‌ی login در Web.Config برنامه به عنوان نقطه‌ی آغازین اعتبار سنجی، معرفی شده است. پس از ورود نام کاربری و کلمه‌ی عبوری یکسان (مطابق منطق ساده‌ی پیاده سازی شده در کلاس SecurityManager)، مجوز دسترسی به برنامه‌ی Silverlight را خواهید یافت و اکنون این برنامه دسترسی لازم را جهت کار با WCF Service، دارا می‌باشد.

نمایی از ساختار فایل‌های این پروژه و برنامه‌ی نهایی در حال اجرای آن را در شکل ۲ ملاحظه می‌نمائید.



شکل ۲– نمایی از ساختار پروژه‌ی امن سازی یک WCF Service

نکته‌ای در مورد فراهم آوردن صفحه‌ی Login توسط برنامه‌ی Silverlight

در مثال قبل شاید علاقمند نباشید که از کنترل Login مرتبط با ASP.NET استفاده نمائید و نیاز است تا صفحه‌ی Login را در برنامه‌ی Silverlight خود پیاده سازی کنید. برای این منظور یک متد اعتبار سنجی را به WCF Service به شکل زیر اضافه نمائید:

TestService.svc.cs

```
//using System;
//using System.Web.Security;

[OperationContract]
public bool Authenticate(string Username, string Password)
{
   if (FormsAuthentication.Authenticate(Username, Password))
   {
      FormsAuthentication.SetAuthCookie(Username, false);
      return true;
   }
   return false;
}
```

به این صورت برنامه‌ی Silverlight در ابتدای کار، صفحه‌ی Login خود را نمایش داده و از متد Authenticate فوق استفاده خواهد نمود. علت استفاده از SetAuthCookie ، مقدار دهی شیء HttpContext.Current.User می‌باشد.

اکنون هر متد ارائه شده توسط WCF Service باید پیش از ارائه‌ی خدمات خود، خاصیت IsAuthenticated مرتبط با کاربر جاری را بررسی نماید:

TestService.svc.cs

```
[OperationContract]
public bool GetMyBalance(out decimal Balance)
{
  if (HttpContext.Current.User.Identity.IsAuthenticated)
  {
     Balance = 1000.00M;
     return true;
  }
  else
  {
     Balance = decimal.MinValue;
     return false;
  }
}
```

در این حالت Web.Config برنامه‌ی ASP.NET ارائه دهنده‌ی WCF Service به شکل زیر درخواهد آمد:

TestService.svc.cs

```
<?xml version="1.0"?>
...
       <system.web>
              <authentication mode="Forms">
                     <forms name="secure">
                            <credentials passwordFormat="Clear">
                            <user name="myUser" password="secret"/>
                            </credentials>
                     </forms>
              </authentication>
...
       </system.web>
...
```

به این ترتیب یک کاربر با مشخصات ذکر شده تعریف گردیده است که مورد استفاده‌ی سیستم Forms Authentication قرار خواهد گرفت؛ یا می‌توان از همان کلاس SecurityManager مثال قبل و نحوه‌ی بکارگیری آن در فایل login.aspx.cs ایده گرفت. همچنین در فایل Web.Config فوق، صفحات پیش فرض پس از اعتبار سنجی و login ، دیگر ذکر نشده‌اند؛ زیرا هدف از این مثال پیاده سازی آن‌ها در برنامه‌ی Silverlight بود.

فراخوانی متد Authenticate ذکر شده تنها یکبار باید صورت گیرد و آن هم در صفحه‌ی Login برنامه‌ی Silverlight . در صورت موفقیت آمیز بودن عملیات، شیء HttpContext.Current.User مقدار دهی شده و از خواص آن می‌توان جهت بررسی وضعیت کاربر جاری استفاده نمود.

به علاوه دیگر در اینجا نمی‌توان مانند مثال قبل از تگ authorization استفاده نمود؛ زیرا جهت دسترسی به متد اعتبار سنجی WCF Service ، کلاینت Silverlight ما در بدو امر اعتبار سنجی نشده است و نیاز است تا بتواند این متد را فراخوانی کند. در این حالت دسترسی‌های غیرمجاز توسط بررسی خاصیت زیر دفع خواهند شد:

HttpContext.Current.User.Identity.**IsAuthenticated**

بررسی روش حذف Meta Data مرتبط با یک WCF Service

اگر WCF Service تولیدی شما تنها قرار است توسط برنامه‌ی Silverlight موجود در پروژه‌ی جاری مورد استفاده قرار گیرد، باید Meta Data مرتبط با آن سرویس را جهت بالابردن امنیت سیستم، حذف نمود. توسط این Meta Data می‌توان ServiceContract ، OperationContract و سایر اطلاعات یک WCF Service را استخراج نمود.

زمانیکه در برنامه‌ی Silverlight خود ارجاعی را به یک WCF Service اضافه می‌نمائید، اطلاعات Meta Data آن سرویس به صورت کامل و به شکل کلاس‌هایی به برنامه اضافه خواهند شد. بنابراین پس از افزودن ارجاعی به WCF Service به سادگی می‌توان اطلاعات Meta Data را مخفی نمود و از آنجائیکه این اطلاعات هم اکنون در برنامه‌ی Silverlight ما موجود است، هیچگونه مشکلی رخ نخواهد داد. برای این منظور به Web.Config برنامه مراجعه کرده و سطر زیر را حذف نمائید یا به شکل Comment در آورید:

Web.Config

```
...
<!-- <endpoint address="mex" binding="mexHttpBinding"
               contract="IMetadataExchange" /> -->
...
```

علاوه بر آن تغییرات ذیل را نیز اعمال نمائید:

Web.Config

```
...
    <behavior name="">
       <serviceMetadata httpGetEnabled="false" httpsGetUrl="false" />
       <serviceDebug includeExceptionDetailInFaults="false" />
    </behavior>
...
```

لازم به ذکر است اگر در WCF Service خود تغییری را اعمال نمودید، ابتدا IMetadataExchange فوق را فعال کرده، سپس بر روی ارجاع مرتبط با این سرویس در پروژه‌ی Silverlight کلیک راست نموده و گزینه‌ی

Update service reference را انتخاب کنید. پس از پایان کار به روز رسانی Meta Data سمت کلاینت، مجددا تنظیمات فوق را به Web.Config اعمال نمائید.