Neden Microsoft Anti-XSS Kullanmalıyım?

Bedirhan Urgun, bedirhanurgun {at} gmail.com, webguvenligi.org

1. Yeni bir C# Web Uygulaması

Default.aspx sayfasına TextBox, Button ve bir Label koyup, bunları birbirine yapıştırın; kullanıcı tarafından TextBox'a ne girilirse Buton'a tıklandığında Label'a yazılsın.

2. Ekseses

Uygulamayı çalıştırdıktan sonra TextBox'a basit bir XSS saldırı dizgisi koyun (mesela **<plaintext>**) ve Buton'a tıklayın. ASP.NET'in Request Validation özelliği isteği yakalayacaktır. Sonra TextBox'a basit ama anlamlı olabilecek bir girdi koyun (mesela **bu doğrudur a<b**) ancak Request Validation özelliği bu dizgiyi de saldırı zannedip yakalayacaktır. İstenmeyen bir durum…

3. Request Validation Özelliğini Kapatın

Request Validation özelliğini deaktive etmek için, Web.config dosyasına aşağıdakini girin;

```
<pages validateRequest="false">
```

Sonra uygulamayı build edin ve TextBox'a basit ama anlamlı bir girdiyi tekrar koyun. Bu kez istek doğru çalışacaktır ama XSS saldırı dizgisi de yaklanmayacak ve saldırı gerçekleştirilecektir. İstenmeyen bir durum…

4. ASP.NET Web Control Encoding

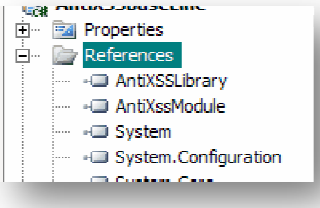
Saldırı gerçeklendiğinde oluşan HTML kaynak koduna bakıldığında, **Label.Text** değerinin (HTML dilinde **span**) otomatik HTML kodlanmadığı görülecektir. Öyleyse Label yerine TextBox kullanabiliriz. Uygulama içerisinde Label yerine bir TextBox koyun ve tekrar anlamlı bir girdiyi (hey bu benim <bedirhan>) ve XSS dizgisini deneyin (<plaintext>). Herşey beklendiği gibi çalışacaktır ama şimdi de sağlanan bazı Web Kontrollerini güvenlik nedeniyle kullanamaz durumdayız. Örnek olarak;

LinkButton.Text
HyperLink.Text
HtmlImage.Src
Literal.Text

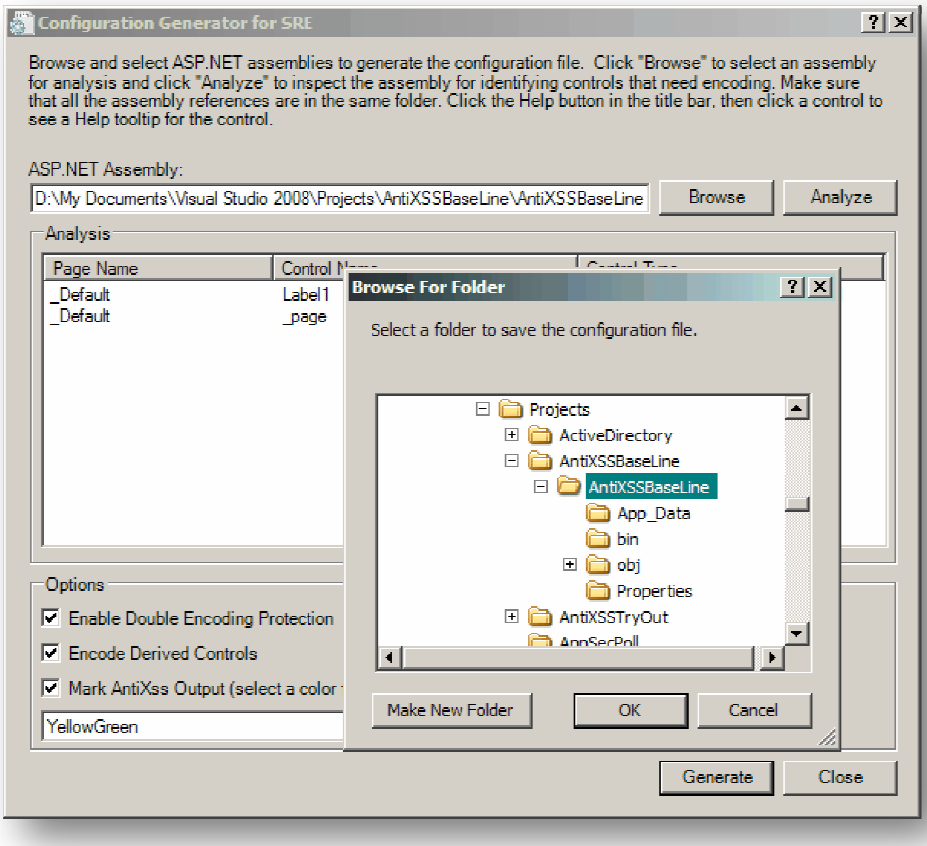
Bunun gibi daha bir çok kontrolün değerleri ASP.NET tarafından kodlanmaz. Daha kapsamlı bir listeyi [1, 2] kaynaklarında bulabilirsiniz. Bu geliştirici kesinlikle kabul edilebilir değildir ve bu nedenle istenmeyen bir durumdur…

5. Ve Anti-XSS Modülü (aka SRE)…

Anti-XSS kütüphanesini [3] kurun ve kurulum ile beraber gelen iki dll'i uygulamaya ekleyin (modül ve kütüphane).



Yine kurulum ile beraber gelen ConfigGen.exe'yi çalıştırın ve yazdığınız uygulamanın DLL dosyasını göstererek antixssmodule.config dosyasını üretin. Bu dosyayı uygulamanın kök dizinine koyun.



Aşağıdaki satırları Web.config dosyası içinde httpModules direktifi altına ekleyin;

```
<add name="AntiXssModule"
type="Microsoft.Security.Application.SecurityRuntimeEngine.AntiXssModule"/>
```

Aşağıdaki satırları antixssmodule.config dosyasına ekleyin (eğer hali hazırda yoksa);

```
<ControlEncodingContext FullClassName="System.Web.UI.WebControls.Label"
PropertyName="Text" EncodingContext="Html" />
```

Daha sonra eklenen TexBox'u bir Label ile değiştirin ve uygulamayı tekrar build edin, çalıştırın. Tekrar anlamlı bir girdiyi (hey bu benim <bedirhan>) ve XSS dizgisini deneyin (<plaintext>). Herşey beklendiği gibi çalışacaktır. Ta ki…

6. Biraz Daha Karmaşıklık ve Koda Dokunuş Kaçınılmaz!

Anti-XSS modülü kullanmak çok avantajlıdır çünkü koda dokunmak gerekmez. Ancak biraz daha karmaşık bir durumu inceleyelim. Aşağıdaki satırları uygulamanızın Default.aspx dosyasında <head> tag'inin içine koyun;

```
<%
    string myvar = TextBox1.Text;
%>
<script language="javascript" type="text/javascript">
    var name = '<%= myvar %>';
</script>
```

Hiçbir web kontrol kodlaması bu kodun oluşturduğu problemi çözemez (uygulamayı çalıştırdıktan sonra TextBox içine **a'; alert(2); s ='** girebilirsiniz). Burda Anti-XSS kütüphanesini kullanmaktan başka bir çare yoktur (daha doğrusu koda dokunmaktan başka). Yine de Anti-XSS kullanılması ile kodda yapılacak değişiklik basittir;

```
<%
    string myvar =
Microsoft.Security.Application.AntiXss.JavaScriptEncode(TextBox1.Text,
false) ;
%>
<script language="javascript" type="text/javascript">
    var name = '<%= myvar %>';
</script>
```

7. Yap. Yapma.

Yukarıdaki bütün maddeler (Anti-XSS çözümü de dahil olmak üzere), bir kademeli savunma anlayışının parçalarıdır. Bu adımların, sağlam bir girdi denetimi politikası olmadan %100 güvenliği sağlamaları mümkün olmayabilir.

1. Uygulamalarınızın güvenliği için sadece Request Validation özelliğine güvenmeyin çünkü bazı XSS çeşitlerine karşı etkisiz kalacaktır.
2. Request Validation özelliğini kapatmamaya çalışın, eğer kapatmanız gerekiyorsa komple değil bazı sayfalarda kapatma yoluna gidin.
3. Uygulamalarınızın güvenliği için sadece Web Kontrol Çıktı Kodlamasına (Output Encoding) güvenmeyin çünkü bu özellik bütün kontrolleri kapsamamaktadır ve bazı XSS çeşitlerine karşı etkisiz kalacaktır.
4. "Varsayılı ASP.NET Web Kontrol Çıktı Kodlama" özelliğinin korumadığı web kontrolleri için Anti-XSS modülü kullanılmalıdır. Bu koda dokunulmayan bir çözüm olduğundan öncelikli olarak seçilebilir ama modül kullanımı bazı XSS çeşitlerine karşı etkisiz kalacaktır.
5. Kodlama kapsamını arttırmak (javascript, css, xml) için Anti-XSS kütüphanesi kullanılmalı ve koda dokunulmalıdır.

Son olarak bütün bu karaliste ve çıktı kodlama (output encoding) önlemleri üzerine pozitif bir girdi denetimi politikası izlenmelidir.

References

[1] http://blogs.msdn.com/sfaust/attachment/8918996.ashx
[2] {Cat.NET Installation Directory}\Microsoft\CAT.NET\Config**Sinks_WebResponse.xml**
[3] http://www.codeplex.com/AntiXSS