使用 Wave 文件绕过 CSP 策略

转载

hhhsan 于 2018-05-22 17:00:38 发布 617 收藏

分类专栏： XSS 文章标签： xss


XSS 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

转载自：https://mp.weixin.qq.com/s/ljBB5jStB7fcJq4cgdWnnw

> 本文作者：梅子酒（来自信安之路学生渗透小组）
> 赠送书籍：《Android应用安全防护和逆向分析》
> 活动地址：信安之路五月送书活动

CSP Introduction

CSP 全称 Content Security Policy，即内容安全策略。CSP 是一个额外的安全层，用于检测并削弱某些特定类型的攻击，包括 XSS 和注入。

CSP 被设计为完全向后兼容，在不同的浏览器上，不会因是否支持 CSP 而产生冲突问题。在不进行特定设置之前，默认为网页使用标准的同源策略。

通常可以使用 meta 标签来设置 CSP，或者使用 php 的 header 函数来进行相关属性的设置。

CSP 的配置会直接影响到页面加载资源的方式，在适当配置的情况下，可以有效的防范 XSS 攻击。

CSP Configuration

我们通过 Content-Security-Policy 头信息来进行 CSP 的设置，通常格式如下:

```
Content-Security-Policy: policy
```

其中，policy 部分对应的为具体的内容安全策略。

一个策略由一系列的策略指令组成，每个策略都描述了一个针对某个特定类型资源以及生效范围的策略。

网上对于相关指令和资源表的说明已经很多了，我就不再赘述。

Bypass CSP With Different Policy

Common Policy Bypass

目前在比赛中常见的绕过 CSP 一般是:

```
script-src ‘self’ ‘unsafe-inline’
script-src ‘self’ ‘unsafe-eval’
script-src ‘nonce-*’
```

通常情况下，除了 CSP 之外，还都会搭配一定的过滤措施来让选手进行绕过。

这里有几个例子，我就不再多说：

1、0CTF 2018 h4x0rs.club2 writeup

> http://sec2hack.com/ctf/0ctf2018-h4x0rs-club2.html

2、Google CTF 2016 Wallowing Wallabies - Part Three

> http://countersite.org/articles/web-vulnerability/83-web-writeup-googlectf-wallowing-wallabies.html

Bypass “script-src ‘self’ “

在前几天的 PlaidCTF 中，出现了如下的 CSP:

> Content-Security-Policy: style-src ‘self’ https://fonts.googleapis.com; font-src ‘self’ https://fonts.gstatic.com; media-src ‘self’ blob:; script-src ‘self’; object-src ‘self’; frame-src ‘self’

`script-src 'self'` 代表着只能加载符合同源策略的文件，直接插入至 html 页面中的静态 script 标签将无法执行。结合其他 CSP 来看，常用的 iframe，object 等标签也是无法被绕过的。

我尝试着使用 link 的预加载机制去带出 cookie，然而受限于 `script-src 'self'` 的限制，虽然能够通过 dns 带出信息，但是无法将 cookie 带出来，因此预加载也是无法使用的。

于是只能另外寻找突破口，在查阅大量资料后发现，可以通过引入正常的非 js 文件来达到引入 js 脚本的效果。考虑到题目中具有上传点，可以将 js 代码插入到尾部来进行绕过。

本地搭建环境进行测试：

`xss.html` 内容为：

`cc.wave` 内容为：

执行效果为:

可以看到注释部分并未对js的执行起到干扰，因此这种攻击手法是可行的。

在进行上传时，后端会进行文件格式校验，因此需要在保证文件格式验证正确的情况进行绕过，在录音选项中，上传的文件为 webm 格式，文件头是不可见字符，在引入 js 文件时，会产生错误，因此需要引入文件的文件头是可见字符。

这里对比下两者的文件格式便很明显了:

wav 文件文件头（第三十五行）：

webm 文件文件头（红线处开始）

wav 格式的文件是以 RIFF 明文开头的，可以使用我上面所用到的攻击方法去构造 xss 代码，而 webm 开头为乱码，在执行时，会因为产生报错而中止执行。

在绕过文件格式检查之后，js 会根据文件格式给定一个 MIME-TYPE，在带入 src 属性的时候，audio 的 Type 会和可执行脚本产生冲突，因此 wav 文件无法代入，而 wave 在 MIME 转换的名单之外，因此在上传成功 wave 文件时，其 MIME-TYPE 并不会与 src 冲突。

因此直接在 description 中写入:

```
<script src="https://idiot.chal.pwning.xxx/uploads/upload_5aee08ef0275e0.94993532.wave"></script>
```

即可拿到 cookie 作为 idolt1 登陆，接下来的就是审计 js，上传 wave 文件动态添加 xss 代码即可。

Bypass 文件格式的重点在于 javascript 在遇到"变量+运算符+变量"格式的表达式时，可以将注释插入其中，并且不会产生干扰。

在这个题目中，难点有两个：

1、绕过 self 限制

2、构造出符合文件内容检查的文件

同样的，我对于 php 是否会有此类特性也感到好奇，在经过测试后发现，在插入注释后，php 的执行也不会被干扰。而 python 因为不具有行内注释的操作而无法做到上述操作。

之前看某篇 paper，其中曾提到使用 GIF 绕过 CSP，其中的部分思路和我上面用到的有部分相似。Paper 链接如下：

```
https://www.slideshare.net/x00mario/jsmvcomfg-to-sternly-look-at-javascript-mvc-and-templating-frameworks/3-TodayJavaScript_MVC_Templating_FrameworksWhy_Because
```

首先，同样是上传 GIF，使得 GIF 与目标网站处于同源下，然后使用 Angular 的 class 去调用这个 GIF 文件，然后会生成 script 标签，并且其 src 属性是 GIF 文件，此时 GIF 中的内容被作为 js 代码解析了，并且因为处于同源情况下，因此可以顺利触发 xss。

Comments On CSP

CSP 作为内容安全策略，在合理配置的情况，可以极大的提高 xss 的攻击成本，以达到较好的防御效果，然而部署成本同样较高，一是熟练掌握相关策略带来的难度，一是配置 CSP 所带来的工作量。

CSP 的不当配置不仅会引发安全问题，还有可能导致页面资源加载失败，但总的来说，CSP 仍然是防范 XSS 攻击较为优秀的措施。


图片违规！


图片违规！