Web 2.0 安全杂谈

长短短

大纲

1. opener 反向劫持详解以及修补
2. OAuth 漏洞的生命力以及正确的修补方式
3. 本地存储缓存 JS 带来的安全隐患
4. 浏览器最新的第三方资源防篡改技术 SRI 详解
5. 如何低成本的提升网站的网站前端安全 -- CSP

opener 反向劫持详解以及修补

- 是什么
- 如何利用
- 如何修补

opener 反向劫持详解

- 没有准确定义，更像是一个特性，而不是 bug。
- 适应的浏览器：Firefox、Safari，Chrome、IE 需要同源的 URL 跳转漏洞做跳板。
- 原理：一个网页与打开它的网页之间有一个 JS 的浏览器对象可以进行交互，它是 opener，意思是「打开我的那个窗口」，这个对象可以控制「打开他的那个窗口」的 location，也就是地址，通过重写 opener.location 或是调用 opener.location.replace 函数，我们可以重定向它的地址。

opener 反向劫持如何利用？

- 利用条件：能在目标站上插入「超链接」，诱导他人点击这个超链接。
- 利用代码：就一句 opener.location = ‘http://x-fishing/xxx.html’
- X-Fishing 做了什么？
  - 自动伪造来源站的登陆页
  - 植入表单劫持代码，劫持登陆账户及密码等敏感数据
  - 自动伪造来源站的 favicon

opener 劫持如何修补呢

- 一般情况

  这样就够了
- 如果是富文本，最后是通过一个跳转页来屏蔽 opener 对象
- 因为 noreferer 可以被这些方式绕过
  - <svg><a xlink:href=“//evil.com” rel=“noreferrer” >CLICK</svg>
  - <form action=“//evil.com” target=“_blank” rel=“noreferrer” ><input type=“submit” ></form>
  - <form id=“test” rel=“noreferrer” ></form><button form=“test” formtarget=“_blank” formaction=“//evil.com” >CLICKME</button>
  - $CLICKME$

opener 劫持 Discuz! 演示

OAuth 漏洞最开始大家关注的点

- http://wooyun.org/bugs/wooyun-2010-059403
- http://wooyun.org/bugs/wooyun-2014-059427
- http://wooyun.org/bugs/wooyun-2010-059639
- http://wooyun.org/bugs/wooyun-2014-059676

OAuth 身份劫持漏洞的生命力

- OAuth 目前的应用有QQ、微博等第三方登陆授权，授权细节我不细。
- 问题核心：redirect_url 的参数非固定，可指定到站点任意地址，或是子域名的任意地址，导致 code 可以通过 referer 泄漏。如：植入一个外部的 图片(<img>) 或是 超链接(<a>)

如何利用 OAuth 劫持身份？

1. 首先需要找到一个支持第三方登陆的网站
2. 在这个网站里找到一个可以插入外部图片或链接的功能
3. 写一个PHP用于插入图片的时候用，功能是把 referer 记录下来
4. 把PHP的地址通过图片植入到目标网站
5. 把最终带有这个图片外链的URL放入 redirect_url
6. 再把这个 OAuth 的授权地址放入 iframe 生成一个页面
7. 将生成的页面通过 URL 形式发送给目标用户
8. 查看 referer 记录重放 oauth 授权 code

OAuth 演示

- 。 。 。

本地存储缓存 JS 带来的安全隐患

- 现在广泛被大家使用的「将 JS 代码缓存在本地 localStorage」方案有很大的安全隐患。网站出现任何 XSS，都有可能被用来篡改缓存在 localStorage 中的代码。之后即使 XSS 被修复，localStorage 中的代码依然是被篡改过的，持续发挥作用。

百度地图本地存储污染案例

详细说明：

由于 BMap API 使用了 HTML5 localStorage 缓存 JS，缓存被修改后导致 Rootkit XSS.

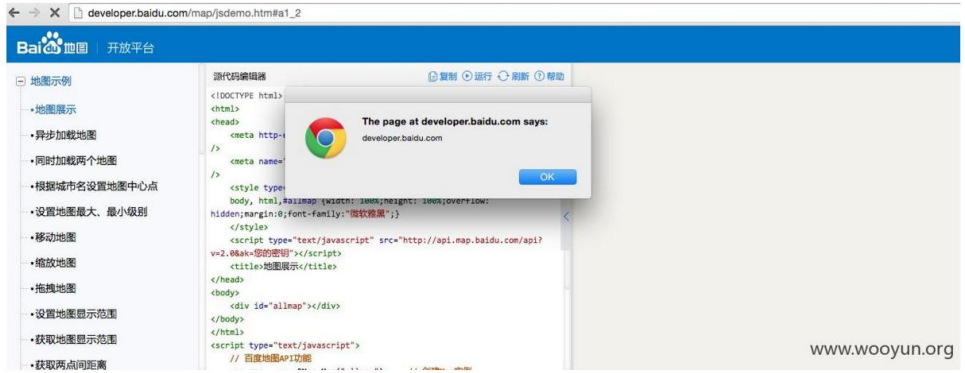
任意一个使用了BMap API 的页面，打开控制台输入

code 区域

```
for(var x in localStorage)localStorage[x]='alert(1);//';
```

如：http://developer.baidu.com/map/jsdemo.htm#a1_2

漏洞证明：



本地存储污染方式

- for(var x in localStorage)localStorage[x]='alert(1);//';

什么是 SRI？

SRI 是 Subresource Integrity 的缩写，主要用于解决网页中的第三方资源以及CDN资源文件，可能被篡改的问题。

目前支持 SRI 的浏览器有 Chrome 45+ 和 FireFox 43+。

SRI 如何使用？

- 原有标签中加入 integrity 属性。
- 支持 sha256、sha384 和 sha512，签名算法和摘要签名内容用 - 分隔。
- sha256 算法生成摘要签名，并进行 Base64 编码：

  echo -n "alert('Hello, world.');" | openssl dgst -sha384 -binary | openssl base64 -A

SRI 如何使用？

- <script crossorigin=“anonymous” integrity=“sha256-Ln/D0mSiCOE4PehbgVN5vsz/VsH5d3FFFdTKx4IO7z4=” src=“https://assets-cdn.github.com/assets/frameworks-2e7fc3d264a208e1383de85b815379beccff56c1f977714515d4cac7820eef3e.js” ></script>
- <link crossorigin="anonymous" href="https://assets-cdn.github.com/assets/github-6670887f84dea33391b25bf5af0455816ab82a9bec8f4f5e4d8160d53b08c0f3.css" integrity="sha256-ZnCIf4TeozORslv1rwRVgWq4Kpvsj09eTYFg1TsIwPM=" media="all" rel="stylesheet" />

THANKS