



□  
(/)  
•••  
(/n  
ew  
se  
nd)

- 不应有增删改的操作
- 不应包含敏感信息

当你的实现不符合别人对你的预期，就可能产生漏洞，如

- 隐私泄露, 被csrf漏洞利用, 账号被盗.....

## 0x04 若你非要用get实现增删改

会被重放，导致服务端资源状态发生改变

- 浏览器的重新打开可能会重放请求，而不会提示用户
- 爬虫或安全扫描会重放你的请求
- 获取到你get请求的各种势力可能会重放此请求，如安全厂商，搜索引擎，神秘力量（除了山岗上那个黑客，因为他已经被大灰狼吃掉了）

get操作的csrf防护很难实施，因为get没有防伪造的需求，它的场景不一定配合你的防护。referrer信任可能被利用，token可能被偷。举个例子，一个塑料盒子，它本就不是被设计用来存钱的，你若非要用它存钱，并还要加上一把锁，效果肯定不会好。见下面例子：

- 网站允许用户发表第三方链接、图片等，那么用户构造的csrf请求的referrer是可信域的url，可以绕过referrer的防护
- 存在js端的跳转漏洞跳到第三方，同理可以绕过referrer
- Get请求中防护的token容易被偷，原理同上，后面的章节会细讲

常见的场景：一些使用了mvc框架的程序，直接用urlrewrite后的url来实现了增删改等操作

## 0x06 若你非用get传输敏感信息

互联网上常见的敏感信息举例：

## 隐私信息

<http://weibo.com/lvwei>

大家可能觉得微博id不算隐私，但一旦你的id和某些操作绑定的时候，那就算是隐私了

### 校验信息

[https://mp.weixin.qq.com/cgi-bin/home?t=home/index&lang=zh\\_CN&token=371767643](https://mp.weixin.qq.com/cgi-bin/home?t=home/index&lang=zh_CN&token=371767643)

这是微博公众平台管理后台的首页，首页url里会包含csrf防护的token

## 认证信息

```
http://XXX.XXXXXX.XXX/index.php?ticket=*****
http://XXX.XXXXXX.XXX/index.php?gsid=*****
```

很多登录认证如单点登录，绑定第三方账号登录等会用get请求来完成登录

如果你的get请求中包含如上一信息，那么你的敏感信息将会被偷被泄露，然后你会被搞！！

## 0x07 敏感信息泄露举例

□  
(/w  
p-  
logi  
n.p  
hp  
?  
acti  
on  
=lo  
go  
ut&  
red  
ire  
ct\_  
to=  
htt  
p%  
3A  
%2  
F%  
2F  
dro  
ps.  
wo  
oy  
un.  
org  
)

## 隐私信息泄露举例



用户登录微博后，首页的url会含有用户ID信息。所以timeline上的链接的主人会通过referrer知道哪些用户访问了它。可能大家都不会在意，它可能会帮你逮微博马甲、捉奸在网.....

比如如下场景：

某天你男友出差，长夜漫漫，你很无聊，写了一篇博客，记录下了盛夏夜中你此刻的心情。喝完咖啡，你正打算上床睡觉了，突然你又好奇想知道自己青春的文采已被多少人阅读。于是你打开电脑，登上服务器，去查看你博客的访问日志，从referrer中你发现，你的男朋友和你的男同事在凌晨一点，都访问了你发的链接，并且IP一样。这个时候，作为一个男子汉，你可能要考虑下，应该哭多大声才不会吵到邻居……当然，你还可以安慰自己，他们是一起在网吧通宵玩游戏

我还曾经用此方法帮人揪出了一些人的微博马甲。只要你够无聊，你还可以玩些别的，比如在用户打开的页面上再放上些兴趣爱好治病寻医类的广告，如果他点了，你就可能会知道他平时爱逛的是不是三里屯的优衣库了。

我不清楚微博的首页地址为何要这样设计，服务端若要读当前userId，完全从当前会话中就可读取，而且从安全的角度考虑，也不应该会从url中读取userId信息。那为什么要显示用户的id呢...

## token信息泄露举例

```
GET https://www.baidu.com/?from=timeline&isappinstalled=0 HTTP/1.1
Host: www.baidu.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2414.0 Safari/537.36
Referer: https://mp.weixin.qq.com/cgi-bin/message?test=message&list&count=20&day=7&token=262076938d1e1c44xvntor
```

如上图，这是微信公众平台后台管理员点击了网友发来的信息后的打开第三方页面的请求，referrer字段中的url中包含了管理后台的csrf防护的token

微信公众平台的操作大多是post，csrf的防护有token和referrer，但在每个页面的get请求中，也会含有这个token。这样的话token很容易被referrer偷，见上图，token已经发给了第三方域了。这样csrf的防护体系就被削弱了

顺便说一句，这个后台是https的，所以我们的链接也要是https的才会发送referrer。在网上申请个免费的https站点即可

好在问题目前没什么危害，管理后台的csrf防护是referrer和token都做了的，而且用户想给公众号发一个链接需要一些小技巧，直接发链接在后台是不可以形成链接的。

## 认证信息泄露举例——被referrer发到第三方



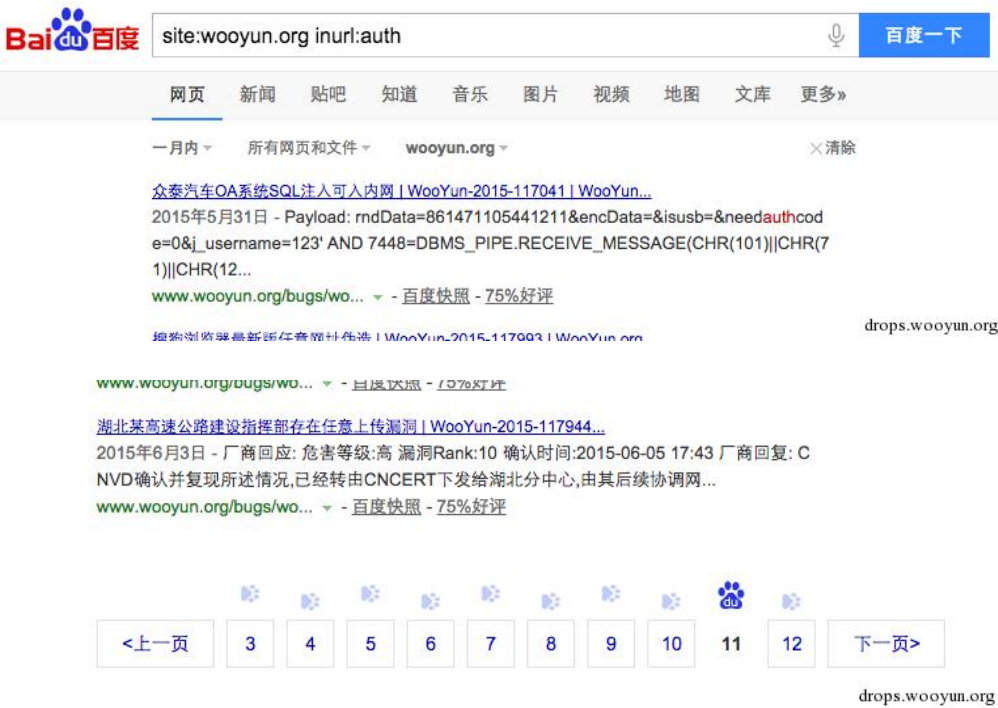
上图是现在的乌云的厂商用户的查看漏洞详情的临时页面，原来的页面是没有查看密码的，是可以通过地址栏里那个含有auth信息的get请求直接查看漏洞详情的

但某一漏洞详情页包含了一个优酷的视频，这个查看详情的链接会在优酷的视频页显示。因为优酷为了告诉用户展示来源，显示了referrer信息。这样这个漏洞详情的临时查看页面就可以被网友在网上无意撞见了，厂商的漏洞详情可能会被提前泄露。见下图



详情可参考 WooYun: 乌云某临时授权查看链接在某些极小可能性下有泄露的可能 (<http://www.wooyun.org/bugs/wooyun-2015-0102609>)

然后我想看看这个查看漏洞的授权页在网上泄露了多少，去百度搜了下



一个月内泄露的乌云厂商用户的临时查看链接竟然有十二页之多，我觉得应该不可能全是厂商管理员分享出去的。所以我有了一个猜测，不一定是错的，那就是：乌云的所有get请求都已被百度云加速收集，用来帮助用户进行搜索的seo优化。

## 0x08 偷最敏感的信息——认证信息

使用get请求认证的一些场景

- 单点登陆从sso拿ticket信息，参数名如ticket、auth
- 网站绑定第三方账号登陆，由第三方给的登陆凭证
- App给内嵌页面请求加上认证信息，参数名如sid、gsid

□  
(/)  
-  
(/n  
ew  
se  
nd)  
  
□  
(/w  
p-  
logi  
n.p  
hp  
?  
acti  
on=  
lo  
go  
ut&  
red  
ire  
ct\_  
to=  
htt  
p%  
3A  
%2  
F%  
2F  
dro  
ps.  
wo  
oy  
un.  
org  
)

Xss偷不了httponly的cookie了？

- 你可以试试偷上面的这些认证信息
- Xss能做的比你想象的要多，它毕竟是个代码执行漏洞
- 如果Xss不好找？你还可以试试referrer，它不产生漏洞，但它是漏洞的搬运工

首先我们了解些背景知识，我简单介绍下单点登陆

- 需求：如果用户已经登陆B站，则自动登陆A站
- 实现：用户访问A站，A站把用户跳转到B站，B站验证用户已登陆，给用户一张票，用户拿着票去找A站，A拿着票去B那，验证成功后放用户进去

下文中将大量出现如下示例站点

- A:http://www.t99y.com
- B:http://passport.wangzhan.com

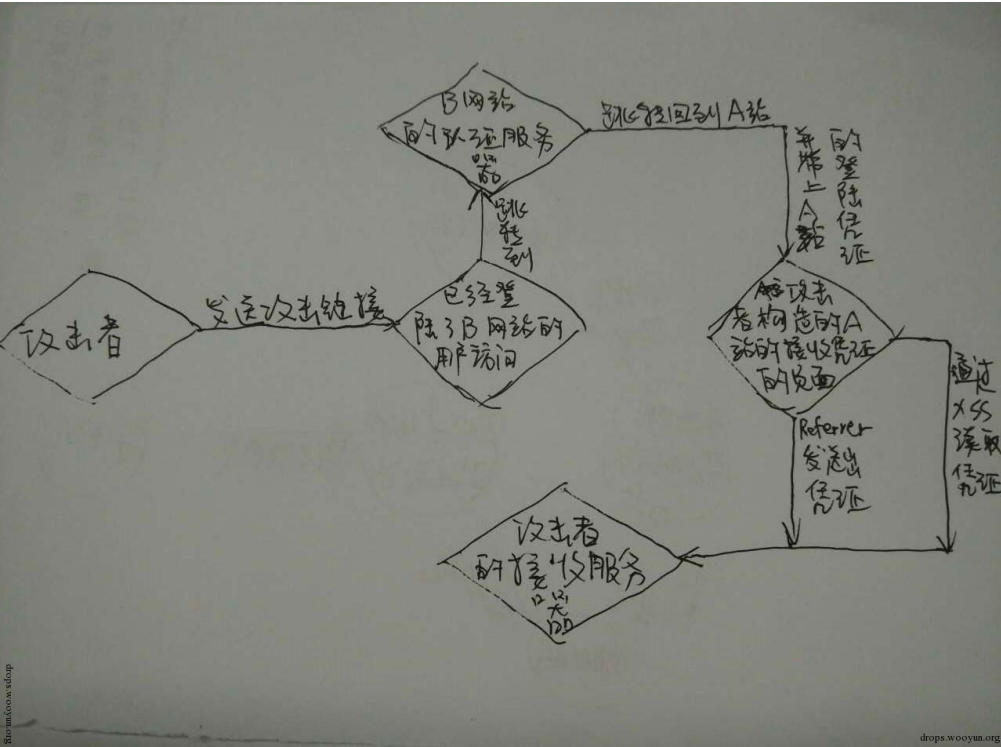
举例：用户访问 http://passport.wangzhan.com/login.php?url=http://www.t99y.com/a.php

B站检验A站是白名单域后，然后302跳转到

```
http://www.t99y.com/a.php?ticket=*****
```

然后a.php用ticket参数去B站验证用户合法后，再给用户种认证cookie

偷认证信息的大概流程如下，后面会细讲。总之攻击的目的就是，拿到用户的ticket信息



0x09 How

互联网上常见的几个单点登陆场景，通行证或第三方站给的登陆凭的证使用的方式各有不同，分别该怎么偷

场景一，直接使用票据来做验证，get型csrf的token和此类似

```
http://t99y.com/a.php?ticket=XXXXXXXXXXXXXXXXXX
```

服务端使用此ticket去sso验证此用户身份，然后在本域种认证cookie

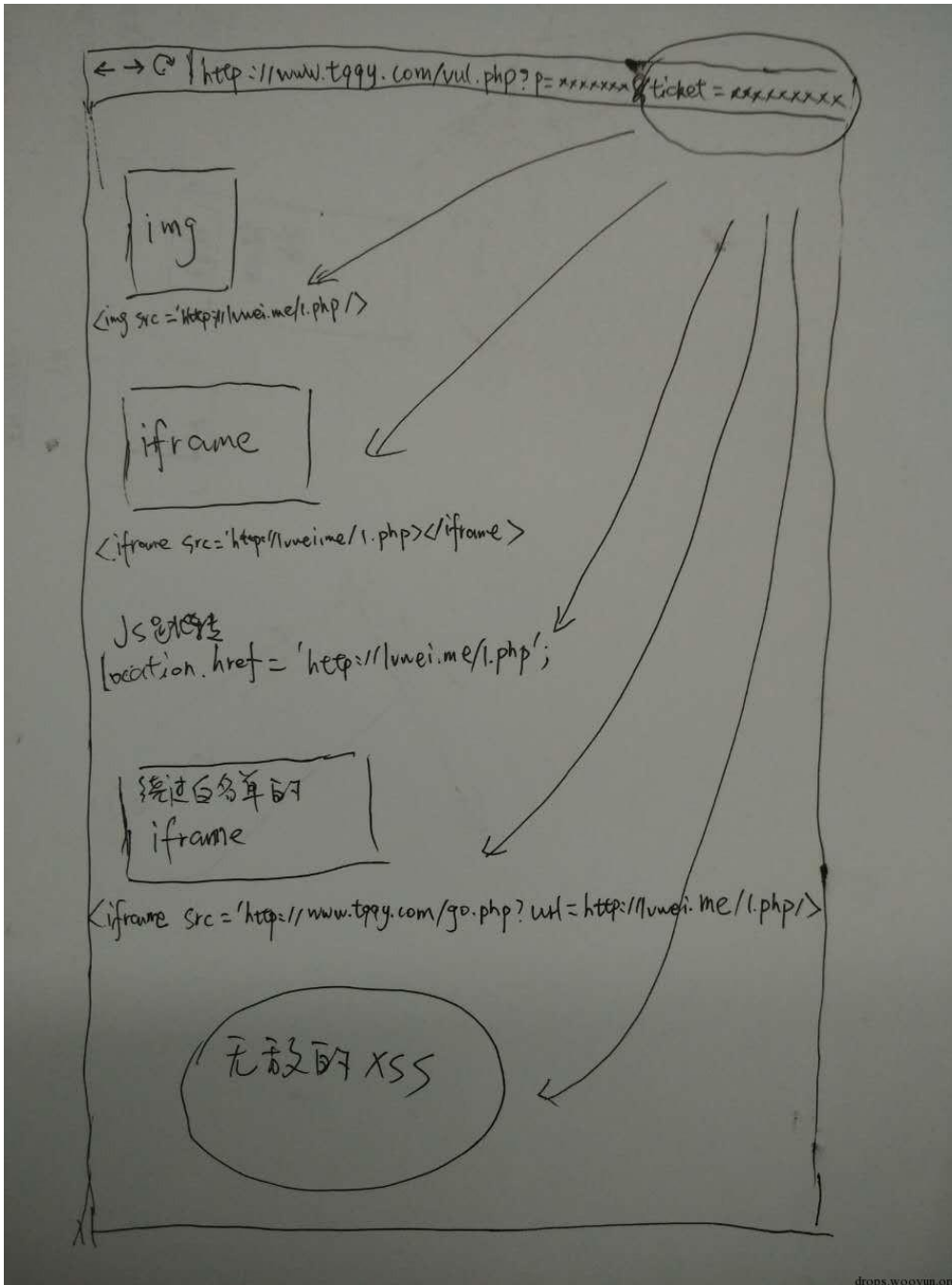
偷的思路：

让我们构造的页面获取到凭证后请求我们控制的服务器上的资源，这样referrer里就有ticket信息了

偷的几种方式

- 找能发自定义src的图片的页面去sso取票，带着ticket信息的页面会发起图片请求，图片服务是我们自己的，我们可以读到请求中的referrer，referrer中会包含ticket信息
- 找能发自定义src的iframe的页面，iframe请求中的referre有ticket
- 找一个有js跳转漏洞的页面去取票，跳转目的地址是我们的服务，js的跳转是带上referrer的，读取此请求的referrer，里面包含ticket
- 如果img和iframe的src值只允许白名单域的url，那就再找一个白名单域的302跳转漏洞来绕过白名单，302跳转可以传递上个请求的referrer
- Xss获取地址栏信息

示意图如下，如下是我画的一个chrome浏览器，地址栏里ticket参数会被包含到下面的一些元素的请求的referrer中







偷的思路:

在xss中，用iframe包含单点登录的请求1，登录跳转后，我们通过src属性可以读到请求1的url，使用

iframe.contentWindow.location.href可以读到跳转最后的请求3的url。但我们需要的是中间的请求2。所以我们想，可不可以让它跳到请求2后，不再继续跳转了，这样我们通过iframe.contentWindow.location.href就可以读到请求2的url了。这时候我想起，cookie超长的拒绝服务终于可以派上用场了

## 偷的方式

- Xss创建iframe, 种超长cookie, 让含ticket的302拒绝服务, 然后使用iframe.contentWindow.location.href读取最后的iframe的当前地址

拒绝服务还有个好处，可以绕过某些ticket的防重放。因为有些票据在受害者端只要被使用后，可能我们盗取后也无法利用了。使用这种方式偷，可以让票据在受害者端不会被使用。

还有，根据path设置cookie可以精准的让我们的iframe停在我们想让它停的url上。

示例代码如下：

```
var iframe=document.createElement('iframe');
iframe.src="http://passport.wangzhan.com/login.php?url=http://www.t99y.com/a.php";
document.body.appendChild(iframe);
for (i = 0; i < 20; i++) {
    document.cookie = i + '=' + 'X'.repeat(2000);//可以根据需求设置path
}
iframe.addEventListener('load', function(){
    document.write(iframe.contentWindow.location.href);
    for (i = 0; i < 20; i++) {
        document.cookie = i + '=' + 'X';
    }
}, false);
```

### 场景五，跨域从通行证获取登陆ticket

形式为类似 `http://www.wangzhan.com/sso/getst.php?callback=jsonp` , 然后通行证会返回个jsonp格式的数据, 里面包含认证信息

参考案例 WooYun: 微博上你点我发的链接我就可以登上你的微博 (web版和app端均可两个漏洞一并提交)  
(<http://www.wooyun.org/bugs/wooyun-2015-0124352>)

## 偷的几种方式

- 可能存在json劫持漏洞，可以通过json劫持来偷取用户的登陆凭证
- Xss漏洞，去跨域请求此接口得到数据

## 0x0a 总结

综上所述，get请求的滥用大家都没重视过，但综合一些小漏洞，可能会产生一些意想不到的大漏洞

修复方案其实很简单，不要使用get方法进行非读操作，不要使用get方法传输敏感信息，因为你不可能控制你所有页面不向第三方发起带referrer的资源请求，而且get请求很难保护。它只是个天真烂漫的孩子，你不要让它承载太多责任

在前几天的阿里安全峰会上，我讲了这个议题，最后的观众提问环节有人问使用https可不可以解决上面的问题。答案当然是不可以，https解决的只是客户端到服务端的传输加密防篡改。但get请求的数据在两个端，尤其是客户端，https是保护不了的。

至于单点登录问题的修复方案，有很多层面都可以去解决，比如不使用get，不让攻击者发起伪造的单点登录请求等等，这些细节需要具体问题具体对待，这里就不细讲了。有需求的女网友可以私下找我交流，共同为互联网的安全学习进步

©乌云知识库版权所有 未经许可 禁止转载

收藏 分享




碎银子打赏，作者好攒钱娶媳妇：

为您推荐了适合您的技术文章:

1. 我的通行你的证 (<http://drops.wooyun.org/web/12695>)
2. Clickjacking简单介绍 (<http://drops.wooyun.org/papers/104>)
3. 密码找回功能可能存在的问题 (补充) (<http://drops.wooyun.org/web/3295>)
4. CSRF简单介绍及利用方法 (<http://drops.wooyun.org/papers/155>)

昵称

验证码



写下你的评论...

发表



**一只媛** 2016-04-04 23:33:00  
wuwuwu

回复



**yeah** 2016-03-30 13:27:32  
@th000 没错，却又没说更好的解决方式

回复



**thanatos** 2016-01-05 10:10:33  
真不错，学习了！

回复



**socket** 2015-12-13 22:56:19  
多出文章啊，看了后的确涨了不少知识

回复



**风格** 2015-11-11 11:29:37  
赞 ~~涨姿势了

回复



**Jumbo** 2015-07-15 22:25:30  
这个时候，作为一个男子汉，你可能要考虑下，应该哭多大声才不会吵到邻居……当然，你还可以安慰自己，他们是一起在网吧通宵玩游戏

回复

□  
(/)  
.  
(/n  
ew  
se  
nd)

	<b>null_z</b> 2015-07-15 21:33:20 感觉很深入，很棒 ~ ~ ~ 多谢楼主	<a href="#">□回复</a>
	<b>light</b> 2015-07-15 20:33:10 学习了~	<a href="#">□回复</a>
	<b>th000</b> 2015-07-15 16:33:38 故弄玄虚	<a href="#">□回复</a>
	<b>th000</b> 2015-07-15 16:33:03 整个PPT无非表达一个意思：不要再GET请求带上认证相关之类的敏感信息。	<a href="#">□回复</a>
	<b>Hxai11</b> 2015-07-15 15:03:51 写的不错，相比起POST来说，貌似GET涉及的方面更广，还有，男子汉和他什么情况。。。	<a href="#">□回复</a>
	<b>Friday</b> 2015-07-15 13:41:51 卤煮还要娶媳妇？	<a href="#">□回复</a>
	<b>Xser233</b> 2015-07-15 12:24:04 赞，月底的PPT有素材了	<a href="#">□回复</a>
	<b>milan</b> 2015-07-15 11:48:36 牛逼	<a href="#">□回复</a>
	<b>瘦蛟舞</b> 2015-07-15 10:09:02 伟哥牛掰呀	<a href="#">□回复</a>

□  
(/w  
p-  
logi  
n.p  
hp  
?  
acti  
on  
=lo  
go  
ut&  
red  
ire  
ct\_  
to=  
htt  
p%  
3A  
%2F  
dro  
ps.  
wo  
oy  
un.  
org  
)