

08 | CSRF/SSRF：为什么避免了XSS，还是“被发送”了一条微博？

2019-12-25 何为舟

安全攻防技能30讲

[进入课程 >](#)



讲述：何为舟

时长 18:00 大小 16.50M



你好，我是何为舟。

前面我们讲了 2 种常见的 Web 攻击：XSS 和 SQL 注入。它们分别篡改了原始的 HTML 和 SQL 逻辑，从而使得黑客能够执行自定义的功能。那么除了对代码逻辑进行篡改，黑客还能通过什么方式发起 Web 攻击呢？

我们还是先来看一个例子。在平常使用浏览器访问各种网页的时候，是否遇到过，自己的银行应用突然发起了一笔转账，又或者，你的微博突然发送了一条内容？

在我们学习 XSS 之后，你可能会联想到，这是银行或者微博中出现了某个 XSS 漏洞。但问题是，你今天并没有访问过银行或者微博的页面，所以并没有“被 XSS”的机会。这时，你想到，会不会是你今天访问的其他网页里存在一些恶意的攻击，实现了你不知道的转账和发博行为呢？好了，你肯定很想知道黑客究竟是怎么做到的，那你不妨先自己思考一下，写出几个可能的答案，然后跟着我开始学习今天的内容！

CSRF 攻击是如何产生的？

我们几乎每天都要用到浏览器，我们的信息也会被浏览器“保存”。那我们首先来看一下，浏览器是如何保存你的身份信息的。

当我们在访问一个 Web 页面的时候，并不是我们自己去获取页面信息，而是浏览器去获取了这些信息，并将它们进行了展示。这就说明，你允许浏览器代表你去和 Web 的服务端进行交互。为了能够准确地代表你的身份，浏览器通常会在 Cookie 中存储一些必要的身份信息。所以，在我们使用一个网页的时候，只需要在首次访问的时候登录就可以了。

从用户体验上来说，这当然是非常方便的。但是，黑客正是利用这一点，来编写带有恶意 JavaScript 脚本的网页，通过“钓鱼”的方式诱导你访问。然后，黑客会通过这些 JavaScript 脚本窃取你保存在网页中的身份信息，通过仿冒你，让你的浏览器发起伪造的请求，最终执行黑客定义的操作。而这一切对于你自己而言都是无感知的。这就是 **CSRF**（Cross-Site Request Forgery，跨站请求伪造）攻击。

接下来，我们就以银行转账为例子，来详细讲解一下这个攻击过程。

当你在银行页面发起一笔转账时，这个过程其实是通过一个转账接口来完成的。这个接口的内容可能包括下面这些内容：

接口地址：`http://bank.com/transfer`；

HTTP 方法：`POST`；


接口参数：`to`（目标账户）、`amount`（金额）。

在转账之前，你肯定进行了一次登录。这样一来，这个转账接口就可以通过你之前存储在 Cookie 中的相关字段来完成认证了。所以，这个接口参数中不需要包含任何身份认证相关的信息。也正是因为如此，这个接口满足了 CSRF 攻击的基本条件：

使用 Cookie 进行认证;

参数中不包含任何隐私信息。

于是，黑客可以构造一个如下的空白网页。我们假设这个网页的地址为 hacker.com。

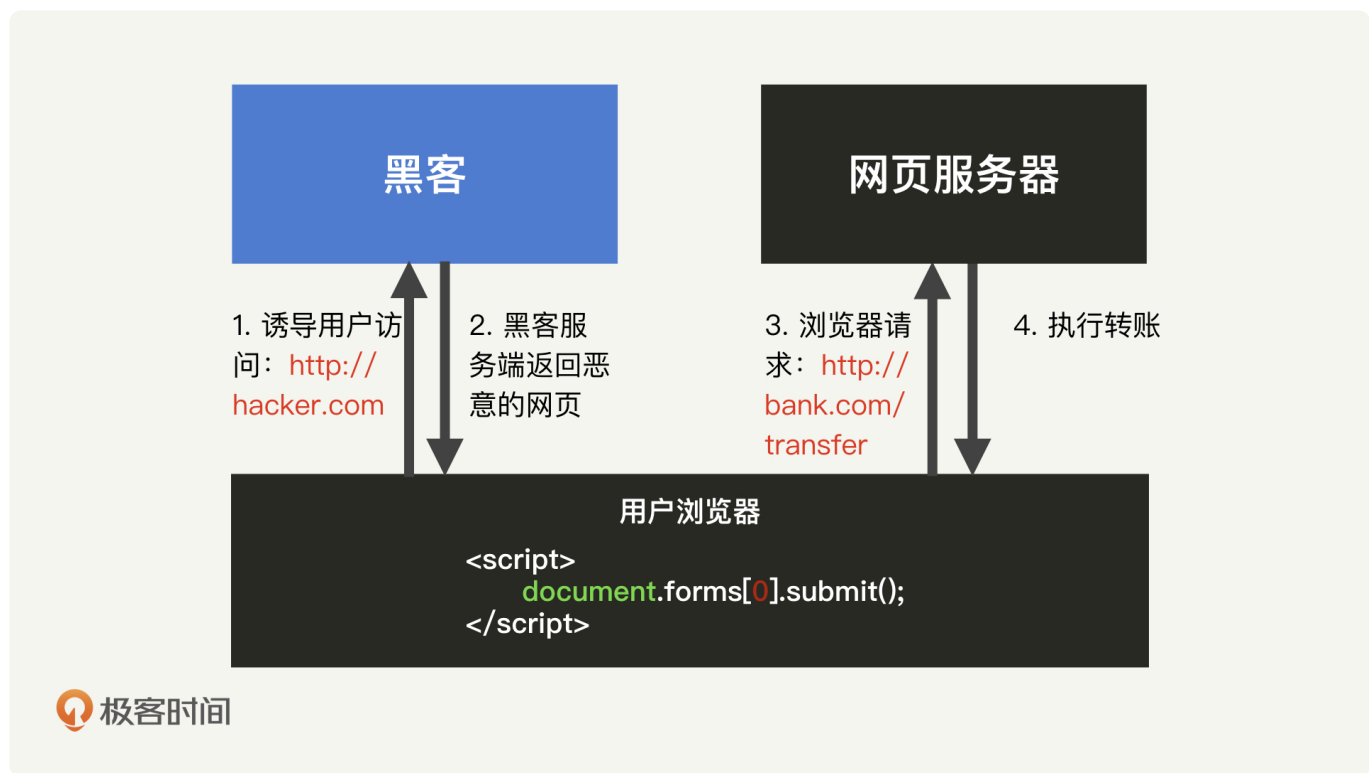
 复制代码

```
1 <html>
2   <body>
3     <form action="http://bank.com/transfer" method="POST">
4       <input type="hidden" name="to" value="hacker" />
5       <input type="hidden" name="amount" value="10000.00" />
6     </form>
7     <script>
8       document.forms[0].submit();
9     </script>
10  </body>
11 </html>
```

在 HTML 中，<script>标签内的 JavaScript 脚本会在打开网页的时候自动执行。因此，一旦用户访问了这个 hacker.com 的页面，它就会自动提交 form 表单，向 http://bank.com/transfer 这个接口（假设为转账接口）发起一个 POST 请求。

其中，to 和 amount 这两个参数，代表着用户向黑客的账号转账 10000 元。只要这个用户之前登录过 bank.com，并且账户余额大于 10000 元，那么黑客就能够成功地收到这 10000 元的转账了。在这个网页中，<input>的标签带有“hidden”属性，所以这个过程对于用户来说都是不可见的。

为了方便你理解，我把这个流程，我画成了一张图，如下所示：



通过 CSRF 攻击，黑客能做什么？

和 XSS 一样，CSRF 也可以仿冒用户去进行一些功能操作的请求，比如修改密码、转账等等，相当于绕过身份认证，进行未授权的操作。

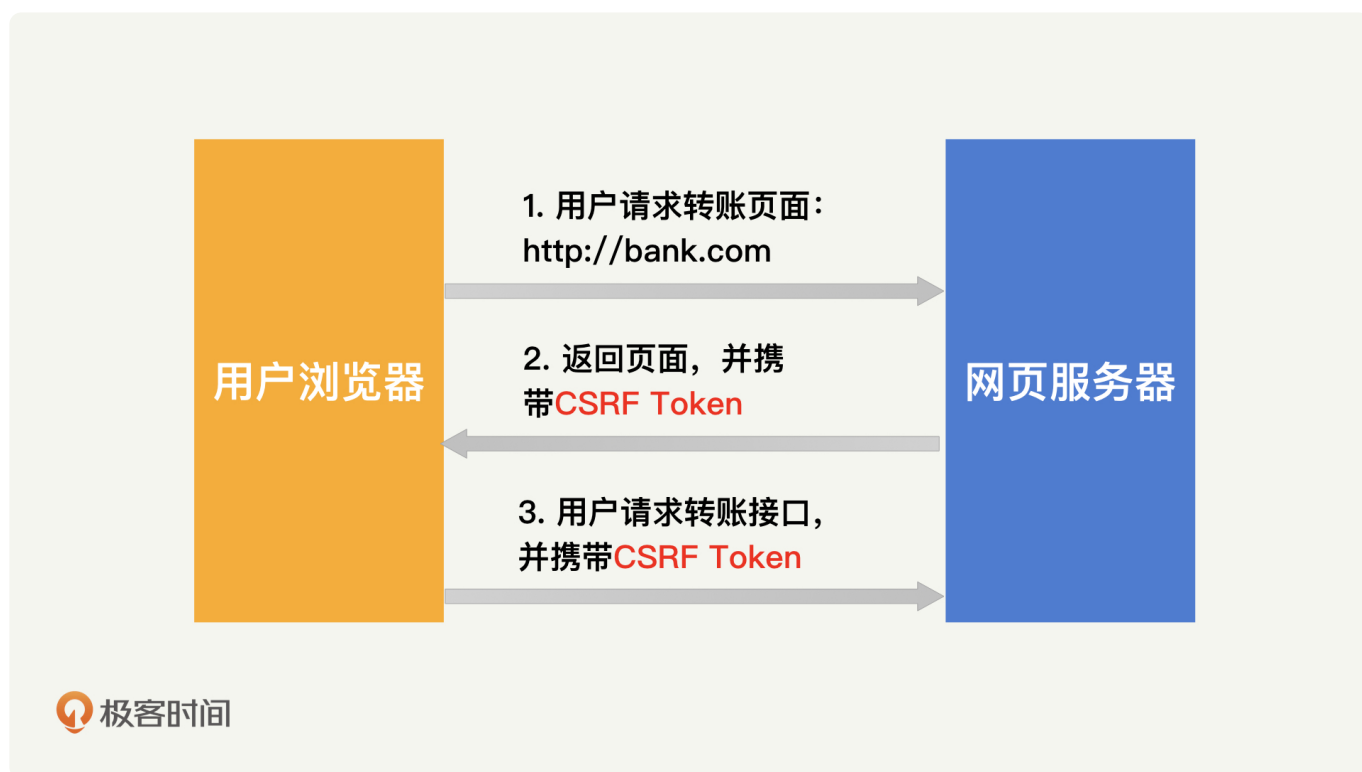
值得一提的是，尽管黑客通过 CSRF 能进行的操作没有 XSS 丰富，但 CSRF 在传播和攻击成本上都低于 XSS。这也就是说，即使你的网页中没有任何注入漏洞，但只要接口配置不当，就能够被 CSRF 利用。而黑客也只需要在自己的域名中，搭建一个诱导性的网页，就可以让任何访问网页的用户都遭受到 CSRF 攻击。而且，用户每天需要访问大量的网页，根本没有办法确认每一个网页的合法性。而从严格意义上来说，用户根本没有办法防止 CSRF 攻击。因此，我们只能从应用本身入手去加强防护。

如何进行 CSRF 防护？

那究竟该怎么进行 CSRF 防护呢？我们有两种方法。**行业内标准的 CSRF 防护方法是 CSRFToken。**我们先来看这个方法。

通过前面的学习，我们知道，CSRF 是通过自动提交表单的形式来发起攻击的。所以，在前面转账的例子中，黑客可以通过 [抓包](#) 分析出 <http://bank.com/transfer> 这个接口所需要的参数，从而构造对应的 form 表单。因此，我们只需要在这个接口中，加入一个黑客无法猜到的参数，就可以有效防止 CSRF 了。这就是 **CSRF Token** 的工作原理。

它的工作流程，我也总结了一下，如下图所示：



因为 CSRF Token 是每次用户正常访问页面时，服务端随机生成返回给浏览器的。所以，每一次正常的转账接口调用，都会携带不同的 CSRF Token。黑客没有办法进行提前猜测，也就没有办法构造出正确的表单了。

除了 CSRF Token 之外，我们也可以通过二次验证来加强防护。

回想一下，当你进行各类支付操作的时候，银行网页通常会要求你输入支付密码。你可能会觉得奇怪，明明自己已经登录了，为什么还需要输入一个独立的支付密码呢？这其实和 CSRF Token 的原理一样：这个独立的支付密码是需要用户输入的，只存在于用户的记忆中，因此，也是黑客无法获取到的参数。

怎么理解呢？假如说，黑客通过 CSRF 攻击，替你发起了一笔转账。在支付的时候，银行会发起一个全新的页面，让你验证支付密码。这个时候你发现，这个支付请求不是你本人发起的，那你肯定不会输入支付密码来完成验证。所以，在用户进行支付这样的敏感操作时，应用通常会要求用户提供一些私密的信息，就是为了对 CSRF 攻击进行防护。

讲到这里，你现在对 CSRF 的攻击和防护，应该有了一个大概的了解。简单来说，CSRF 其实就是黑客利用浏览器存储用户 Cookie 这一特性，来模拟用户发起一次带有认证信息的请求，比如转账、修改密码等。防护 CSRF 的原理也很简单，在这些请求中，加入一些黑客

无法得到的参数信息即可，比如 CSRF Token 或者独立的支付密码等。掌握了这些内容，其实 CSRF 的知识基本上就差不多了。

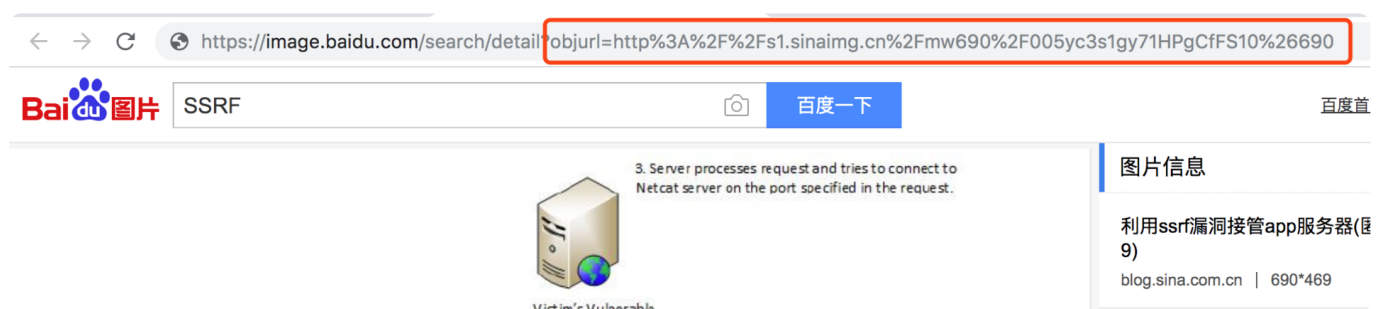
SSRF：同样的原理，发生在服务端又会发生什么？

在 CSRF 中，黑客通过诱导用户访问某个网站，让用户的浏览器发起一个伪造的请求。那么，如果服务端发起了这个伪造的请求，又会发生什么呢？

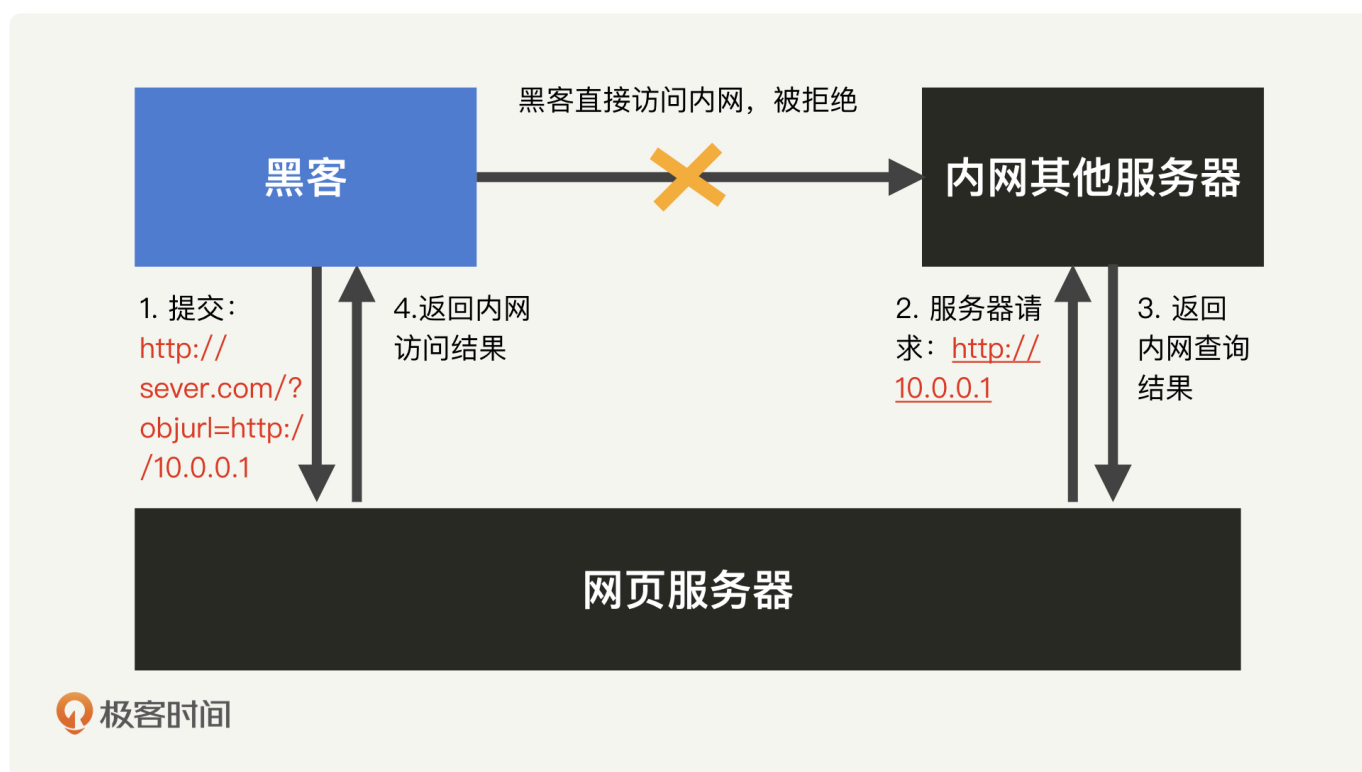
我们知道，服务端也有代理请求的功能：用户在浏览器中输入一个 URL（比如某个图片资源），然后服务端会向这个 URL 发起请求，通过访问其他的服务端资源来完成正常的页面展示。

这个时候，只要黑客在输入中提交一个内网 URL，就能让服务端发起一个黑客定义的内网请求，从而获取到内网数据。这就是 **SSRF**（Server Side Request Forgery，服务端请求伪造）的原理。而服务端作为内网设备，通常具备很高的权限，所以，这个伪造的请求往往因为能绕过大部分的认证和授权机制，而产生很严重的后果。

比方说，当我们在百度中搜索图片时，会涉及图片的跨域加载保护，百度不会直接在页面中加载图片的源地址，而是将地址通过 GET 参数提交到百度服务器，然后百度服务器请求到对应的图片，再返回到页面展示出来。



这个过程中，百度服务器实际上会向另外一个 URL 地址发起请求（比如，上图中的 `http://s1.sinaimg.cn`）。利用这个代理发起请求的功能，黑客可以通过提交一个内网的地址，实现对内网任意服务的访问。这就是 SSRF 攻击的实现过程，也就是我们常说的“内网穿透”。



通过 SSRF 攻击，黑客能做什么？

了解了 SSRF 攻击的过程之后，我们知道，在服务端不做任何保护措施的情况下，黑客可以利用 SSRF 向内网发起任意的 HTTP 请求。那么，这些请求会产生什么样的后果呢？我总结了一下，主要会有这样两种动作：内网探测和文件读取。

1. 内网探测

我们先来看内网探测。内外网一般是隔离的。所以，黑客在外网环境中，是无法知道内网有哪些服务器，这些服务器又分别提供了哪些服务。但是，通过一个加载图片的 SSRF 漏洞，黑客就能够对内网进行探测。这是怎么做到的呢？别着急，我们慢慢来看。

在前面百度搜图的例子中，我们请求的地址是：

🔗 <https://image.baidu.com/search/detail?>

`objurl=http://s1.sinaimg.cn/picture🔗.jpg`。因为 🔗 `http://s1.sinaimg.cn/picture🔗.jpg` 会正常返回一个图片，所以网页会展示出来对应的图片。

我们假定这样一个服务端逻辑：在这个请求过程中，服务端会判断 `objurl` 返回数据的 Content Type 是否为 `image/jpeg`。那么，可能的返回结果就有三种：

“是”，则展示图片；

“不是”，则返回“格式错误”；

无响应，则返回“找不到图片”。

基于这三种返回逻辑，黑客可以构造一个恶意的请求地址：

🔗 <https://image.baidu.com/search/detail?objurl=127.0.0.1:3306>。如果服务器返回“格式错误”，则代表服务端本地的 3306 端口可用；如果返回“找不到图片”，则代表不可用。我们知道，3306 是 MySQL 对应的端口号，因此，根据这个返回的信息，黑客就能够知道服务端本地是否开启了一个 MySQL 服务。接下来，黑客只需要不断重复这个过程，尝试不同的 IP 和端口号，就能够一点一点探测出整个内网的结构。

2. 文件读取

接下来，我们说一下文件读取。服务器除了对图片的代理不做合法性判断之外，对很多其他的代理也不做判断，而是直接将代理的结果返回到前端。我们称这种情况为“有回显的 SSRF”。在这种情况下，黑客不仅能够知道请求是否成功了，还能够知道具体返回的内容。这时候你肯定会好奇，黑客究竟是怎么做到呢？

在 URI 中，开头的 `http://` 和 `https://` 代表需要使用什么协议去进行请求。除了 HTTP 之外，URI 还有很多种协议可以选择，比如 `file://` 就是直接读取本地的文件。通过输入 `file://etc/passwd`，黑客就能够通过一个请求获取到本地的 `passwd` 文件，从而知道本地有哪些用户。经过不断地尝试，黑客就能够把整个服务器中的文件内容都给拉取出来，这其中包括密钥、源码等极度敏感的信息。

我曾经就遇到过一个黑客。他通过 SSRF 攻击拿到了服务端的源码，然后通过对源码的分析，找到了一个 SQL 注入的漏洞，再利用 SSRF 发起对内网的 SQL 注入攻击，从而拿到了内网的命令执行权限。

如何进行 SSRF 防护？

因为 SSRF 漏洞起源于业务的正常功能需求（比如百度图片的图片请求等等）。因此，我们很难真正消除它。尽管如此，我还是会为你介绍几种常见的防护手段，来尽可能地提高应用的安全性。这些常见的手段主要包括：白名单限制、协议限制和请求端限制。接下来，我们——来看。

白名单的限制永远是最简单、最高效的防护措施。 SSRF 中的白名单，就是对用户提交上来的目标 URL 进行限制。比如，只允许是同一个域名下的 URL。你可以理解为，让百度图片的代理服务只允许代理 baidu.com 的 URL。但是，很多时候，因为业务功能的设计，白名单的限制并不可行。比如，上述百度图片的例子，这个功能的设计思路就是，baidu.com 这个域名下能够请求各类域名下的图片资源（比如上述例子中的 sinaimg.cn）。

在这种时候，**我们可以对协议和资源类型等进行限制。**比如：对于使用协议，我们只允许 HTTP 或者 HTTPS 协议；对于返回的内容，我们只允许图片格式的内容。通过这些限制，虽然不能完全阻止黑客发起 SSRF 攻击，但也大大降低了黑客能够造成的危害。

除此之外，因为 SSRF 最终的结果，是接受代理请求的服务端发生数据泄漏。所以，SSRF 防护不仅仅涉及接收 URL 的服务端检测，也需要接受代理请求的服务端进行配合。在这种情况下，我们就需要用到**请求端限制**，它的防护措施主要包括两个方面。

第一，为其他业务提供的服务接口尽量使用 POST，避免 GET 的使用。因为，在 SSRF 中（以及大部分的 Web 攻击中），发起一个 POST 请求的难度是远远大于 GET 请求的。因为默认的请求方式是 GET，而发起 POST 请求，需要在发起 HTTP 请求的时候进行配置。很多安全漏洞中不包含能够配置协议的地方。在上述百度图片的例子中，黑客显然就只能发起 GET 请求。如果某个敏感服务是 POST 的，黑客就无法请求到相关资源了。

第二，为其他业务提供的服务接口，最好每次都进行验证。通过 SSRF，黑客只能发起请求，并不能获取到服务端存储的验证信息（如认证的 key 和 secret 等）。因此，只要接受代理请求的端对每次请求都进行完整的验证，黑客无法成功通过验证，也就无法完成请求了。

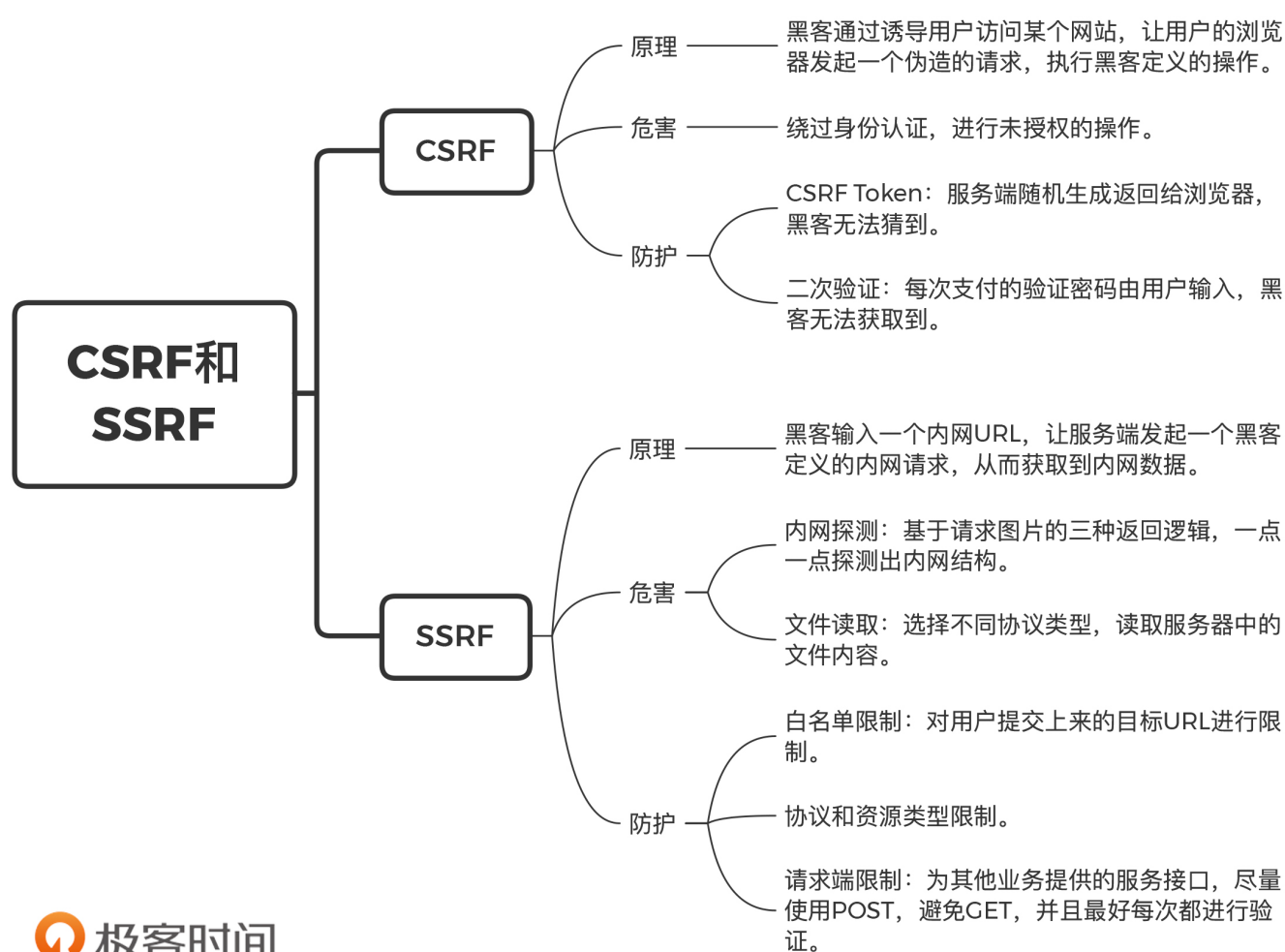
总结

好了，今天的内容差不多了，让我们来回顾一下，你要掌握的重点内容。

今天我们介绍了 CSRF 和 SSRF 这两种攻击方式。其中，CSRF 是黑客控制用户的浏览器发起伪造的请求，SSRF 则是黑客控制服务端发起伪造的请求。通过伪造的请求，黑客可以伪造用户或者服务器的身份，越权获取数据或者发起请求。应用中的请求接口越敏感，黑客能够造成的伤害就越大。

除此之外，CSRF 和 SSRF 产生于正常的业务功能逻辑中，因此，我们没有办法从根本上组织黑客发起伪造的请求。但是，你可以通过加强接口的安全验证，来避免伪造请求造成影响。在 CSRF 中，我们可以通过 CSRF Token 或者二次验证等操作来加强防护。这样，黑客无法获取到隐私信息，也就无法发起连续的请求了。在 SSRF 中，我们则需要限制可请求的域名，来限制黑客能够访问到的资源。另外，目标服务端，也需要加强接口的验证，来避免伪造请求成功通过授权。

今天的内容比较多，为了方便你记忆，我总结了一个知识脑图，你可以通过它来对今天的重点内容进行复习巩固。



思考题

接下来，让我们来看一道思考题。

通过今天的讲解，你可以回忆一下，你的企业是否遇到过 CSRF/SSRF 攻击呢？如果遇到过，当时是如何处理的呢？如果没有遇到过，那你负责的 Web 或者应用中，是否实现了

CSRF/SSRF 的保护逻辑呢？具体又是怎么实现的呢？

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友。我们下一讲再见！


点击查看 

来参加打卡，攻克
工作中 80% 的安全问题



PC端用户扫码参与



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | SQL注入：明明设置了强密码，为什么还会被别人登录？

下一篇 09 | 反序列化漏洞：使用了编译型语言，为什么还是会被注入？

精选留言 (14)

 写留言



tt

2019-12-25

首先向老师请教一个问题：从hacker.com向bank.com发起HTTP请求不会遇到同源策略的限制么？

其次谈谈自己对CSRF的理解。

...

展开 

作者回复：会，所以是通过form.submit的POST形式，跳转过去的。黑客并拿不到返回的结果。

1

3



瑞泉

2019-12-29

老师，csrf xss sql注入这些Web安全有没有比较好的测试工具推荐？后续课程中会有工具介绍吗？

作者回复: 你好，感谢你的留言。xss可以用beef。sql注入可以用sqlmap。csrf好像没听说过，从原理上来说，也不容易做自动化的检测。

这类工具主要是如何发起攻击，不是本专栏的重点，因此不会做介绍，感兴趣可以自行了解。

1

1



LEON

2019-12-25

请教老师一个问题，通过CSRF token 来进行防护的话，有没有可能黑客通过自己转账确认CSRF token的位置或者标识，然后进行CSRF模拟表单进行提交的时候，通过JS脚本把CSRF token取出来，加在黑客模拟的表单中发送给server。从而造成CSRF token 防护失效？
谢谢老师

展开

作者回复: CSRF token每个人每次请求都不一样，提前拿没有意义。如果黑客能够拿到用户的token，说明已经通过XSS等控制了用户的浏览器，则没有CSRF的意义了。

1

1



小晏子

2019-12-25

目前还没遇到过CSRF和SSRF的攻击，首先对于CSRF攻击，主要是使用CSRF-token进行防护的，这个很多web框架都提供了现成的模块可供使用。对于SSRF主要是用了白名单和接口认证的方式，文中提到的一个方案：全部使用POST，也考虑过，可是很多人认为这种方式可奇怪，不好理解，所以就没用。

这里也请教下老师，接口全部使用POST请求的这种方式在业界是否用的普遍？

展开

作者回复: 敏感的上行操作全部使用POST，既符合GET和POST本身的设计初衷，也能够提升安全性，所以还是挺普遍的。



1

**Middleware**

2019-12-25

尽量使用 POST 请求方式，似乎不太好吧？有违 RESTFul

作者回复: 嗯，有些绝对了。应该是上行操作都用POST



1

**Zerolce**

2019-12-25

老师我有个问题不太懂：现在很多接口安全机制，不可能仅仅是直接访问一个接口而不带header验证（例如：cookie）就可以成功。不需要验证，还不如直接通过postman直接请求呢。这样子的话黑客怎样实施csrf？用户身份（cookie或token）黑客怎样添加到表单里面？

展开 ∨

作者回复: 用浏览器发起请求，会自动带上cookie的。



1

**leslie**

2019-12-30

CSRF的防御方式算是提前学习到了：正准备做相关的事情，之前不理解为何现在许多现金支付为何越来越多的使用手机验证；其实目的就是避免该环节用户的密码和支付密码被保存从而被利用。

作者回复: 是的。支付密码的意义，除了防CSRF这类攻击，其实也是一定程度上强化用户对密码的保护意识。支付密码其实强调了说这个密码和普通密码意义不一样，因此用户一般只在脑子里面记忆，不会轻易写下来。

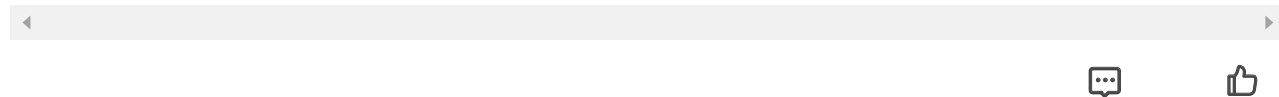
**早起不吃虫**

2019-12-26

老师总结的很好，不过由于篇幅所限，讲的稍显不够详细，所以看到评论区有很多相关的疑问，建议增加篇幅，哈哈😊

展开 ▾

作者回复: 篇幅确实有限, 对很多内容进行了删减。有不明白的欢迎留言讨论~



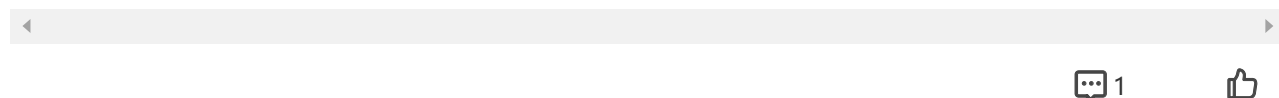
仰望星空

2019-12-25

如果csrf token也存储在cookie里是不是就不安全了, 但奇怪的是spring security 框架就是把card token存储在cookie里返回给浏览器的, 似乎也没人说不安全。

展开 ▾

作者回复: 存cookie里面是不安全了。你说的card token是干啥用的? 不太了解。

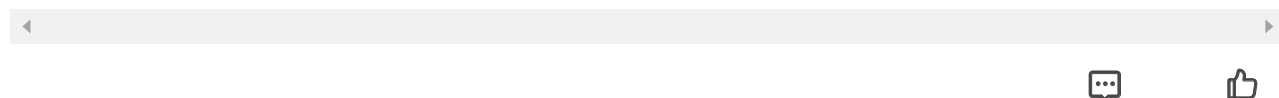


LEON

2019-12-25

感觉XSS攻击和CSRF攻击很像, 这两种攻击比较起来具体有什么关系和区别吗?

作者回复: XSS攻击发生在当前域名, CSRF攻击发生在其他域名。总体来说, XSS攻击能够覆盖CSRF的危害, 但XSS难度更好, 传播能力更弱。



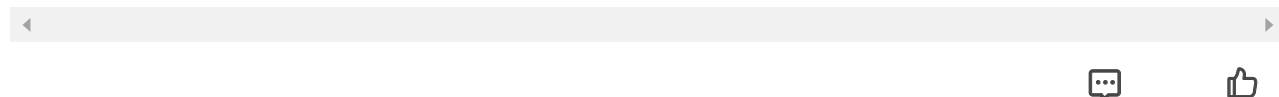
稳

2019-12-25

想请教老师个问题, 前后端分离项目中, 怎样做csrf? 如果通过接口返回, 是不是黑客也可以额外做一次接口请求呢?

展开 ▾

作者回复: 做好同源保护, 黑客没办法在其他域名下, 拿到用户在当前域名下的接口返回结果。然后, 把token和session等会话标识绑定即可。



alan

2019-12-25

讲得太好了！

展开 ▾



Cy23

2019-12-25

CSRF了解了，SSRF攻击原理理解了，SSRF发起攻击的细节还有很多不了解的，需要扩展学习下

作者回复: SSRF其实就是看，通过HTTP请求，都能从内网获取些什么信息。



Zerolce

2019-12-25

上一条留言：

老师我有个问题不太懂：现在很多接口安全机制，不可能仅仅是直接访问一个接口而不带header验证（例如：cookie）就可以成功。不需要验证，还不如直接通过postman直接请求呢。这样的话黑客怎样实施csrf？用户身份（cookie或token）黑客怎样添加到表单里面...

展开 ▾

作者回复: 浏览器发起请求，会自动带上这个请求域名的cookie，所以黑客不需要主动添加。。。。看浏览器怎么处理认证了，我理解Basic Auth浏览器应该也是会自动带上的，吧？

