CSRF 漏洞概述

【学习目标、重难点知识】

【学习目标】

1. CSRF简介
2. CSRF基本原理和方法
3. GET型CSRF
4. POST型CSRF

【重难点知识】

1. CSRF基本原理和方法

【演示环境】

Pikachu

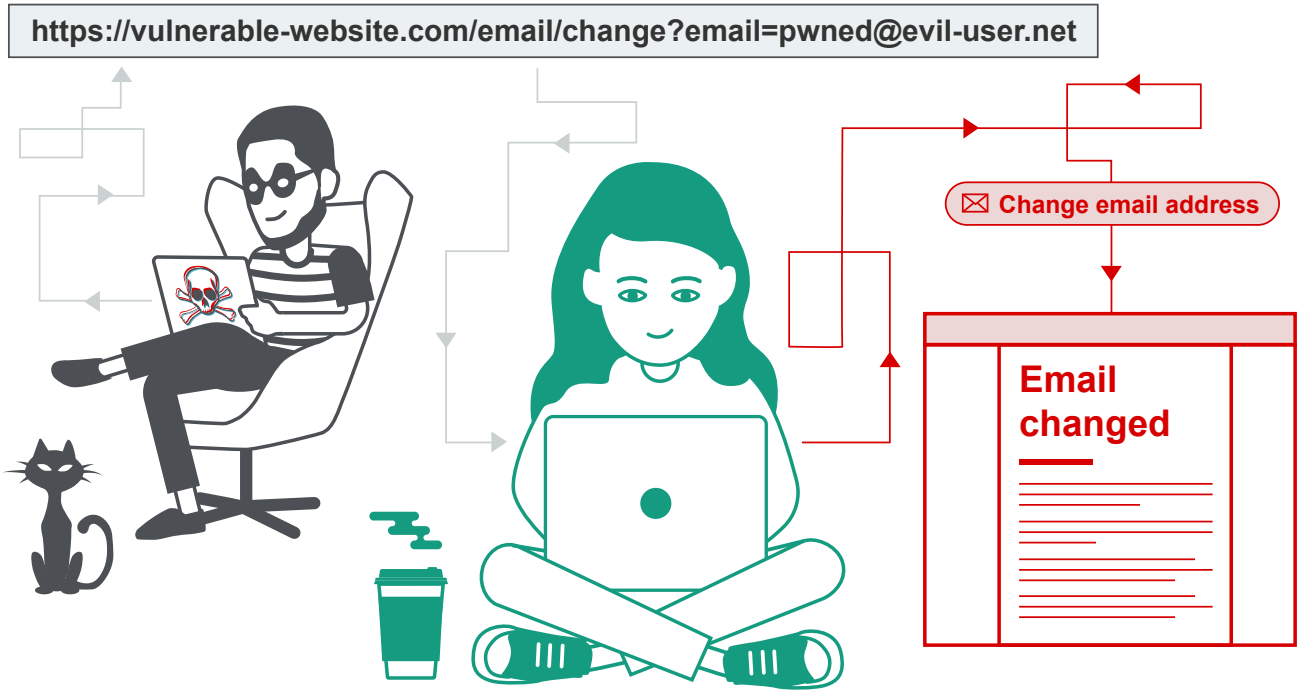
百度网盘为您提供文件的网络备份、同步和分享服务。空间大、速度快、安全稳固，支持教育网加速，…

https://pan.baidu.com/s/1wKC_FfZKhLX7X7zF79lKNw?pwd=GOKT

提取码：GOKT

简介

- `CSRF`（`Cross-site request forgery`）跨站请求伪造：攻击者诱导受害者进入第三方网站，在第三方网站中，向被攻击网站发送跨站请求。利用受害者在被攻击网站已经获取的注册凭证，绕过后台的用户验证，达到冒充用户对被攻击的网站执行某项操作的目的。
- `CSRF` 是一种“挟制用户”在当前已登录的Web应用程序上执行“非本意的操作”的攻击方法。
- `CSRF` 攻击也被称为 `one click` 攻击。



利用条件：

- 目标用户必须处于登录状态
- 用户必须访问攻击者构造的url

csrf和xss区别

XSS漏洞

- 目标是拿到用户的cookie
- 除了偷cookie还可以做其他的事情，弹框，重定向

CSRF漏洞

- 借cookie

与XSS攻击相比，CSRF攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比XSS更具危险性。

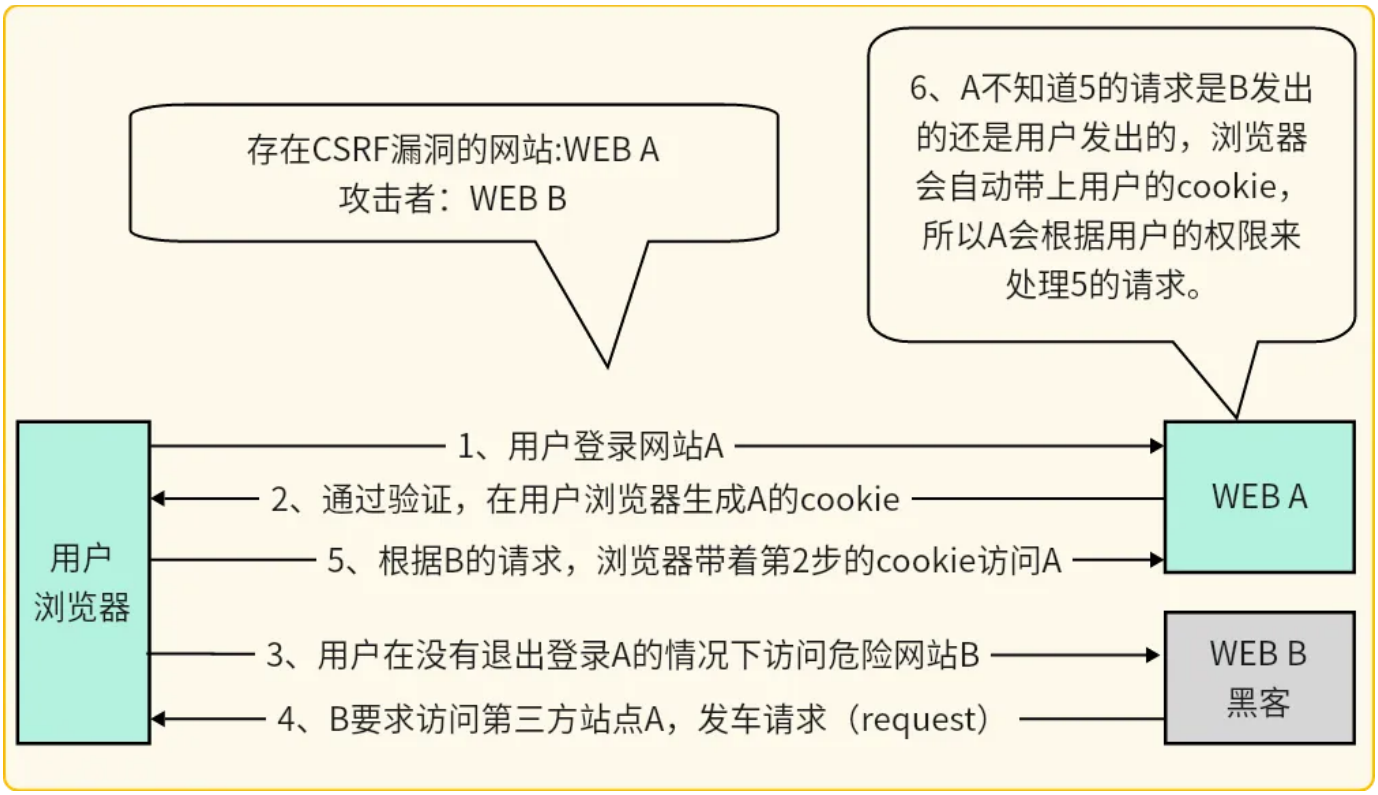
csrf漏洞的成因就是网站的cookie在浏览器中不会过期，只要不关闭浏览器或者退出登录，那以后只要是访问这个都网站，会默认你已经登录的状态。而在这个期间，攻击者发送了构造好的csrf脚本或包含csrf脚本的链接，可能会执行一些用户不想做的功能（比如是添加账号等）

CSRF原理

CSRF攻击过程有以下两个重点：

1.目标用户已经登录了网站，能够执行网站的功能

2.目标用户访问了攻击者构造的URL



- 用户访问网站A，输入账号密码登录网站A
- 登录成功之后，网站A产生的cookie会保存到浏览器中，此时用户可以向网站A正常发送请求
- 用户没有退出网站A之前，访问了攻击者构造的网站B
- 网站B，要求用户去请求网站A，此时请求里面包括攻击者构造好的数据
- 浏览器就会将网站A的cookie+所请求的内容，一起发送给网站A(用户的cookie+黑客的请求)
- 这时候网站A就认为这是一个合法请求，进一步导致构造好的数据被执行

案例：虹膜望远镜、指纹采集。

1. CSRF 攻击流程

- 其实我们不能挟持用户，但是我们可以挟持用户的浏览器发送任意的请求。
- 一个典型的 `CSRF` 攻击有着如下的流程：

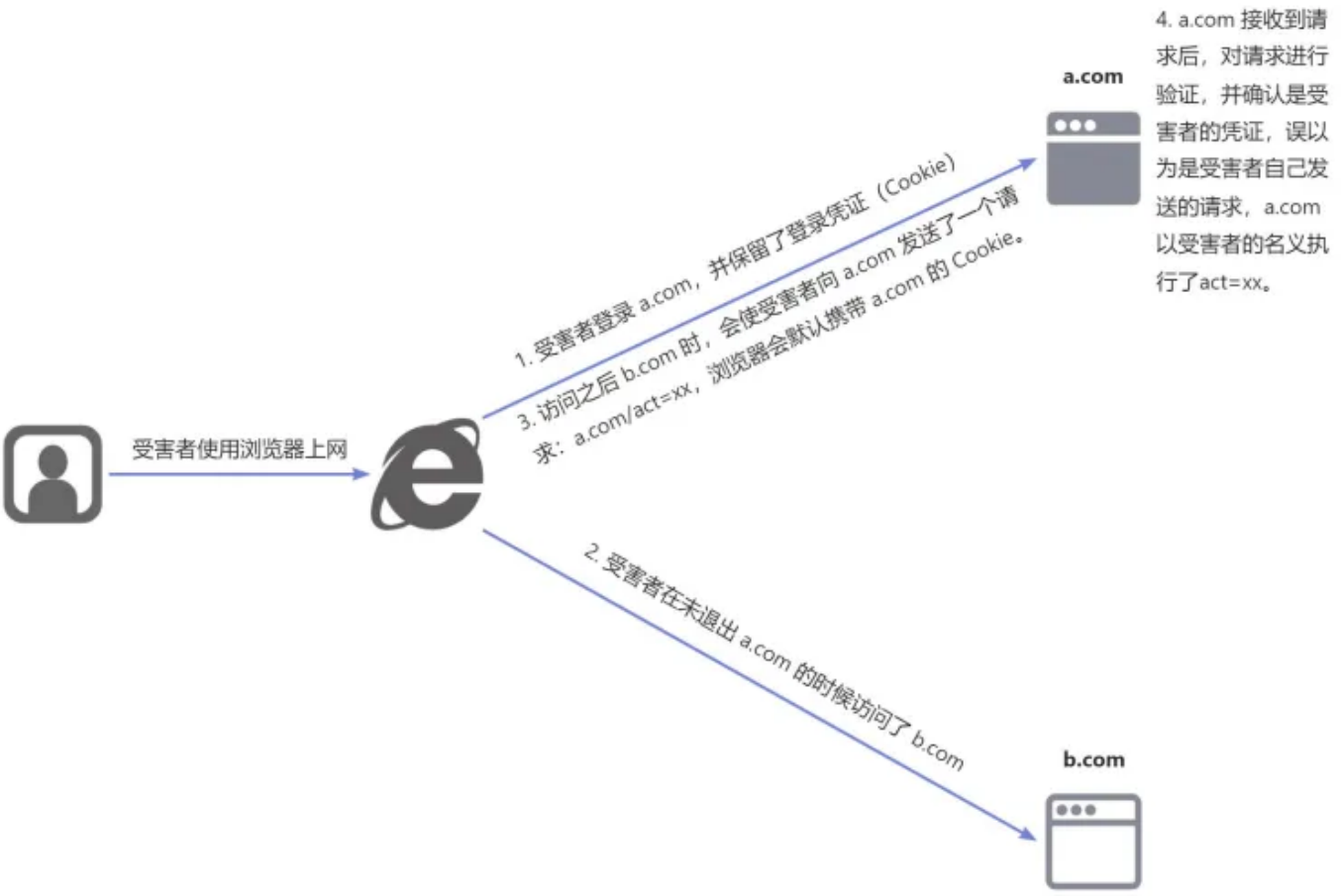
```
受害者登录 a.com，并保留了登录凭证（Cookie）。

攻击者引诱受害者访问了 b.com。

访问之后 b.com 让用户向 a.com 发送了一个请求：a.com/act=xx，浏览器会默认携带 a.com 的 Cookie。

a.com 接收到请求后，对请求进行验证，并确认是受害者的凭证，误以为是受害者自己发送的请求，a.com 以受害者的名义执行了act=xx。
```

- 攻击完成，攻击者在受害者不知情的情况下，冒充受害者，让 `a.com` 执行了自己定义的操作。
- 示例如下：



- `CSRF` 实现必要条件：
  - 受害者登录信任站且能够执行网站的功能，并在本地保存了对应 `Cookie` 信息。
  - 受害者在使用同一浏览器且 `Cookie` 存活的情况下，访问了攻击者构造的危险网站。

2. CSRF 漏洞产生原因

- `CSRF` 攻击的主要原因是应用程序未能正确验证请求的来源，使得攻击者可以伪装请求，以便在用户不知情的情况下执行操作。
- `CSRF` 漏洞产生的位置通常是在 `Web` 应用程序的不安全设计或实现中，主要存在于以下场景：
    - **缺乏足够的请求验证**： 如果应用程序在执行重要操作（例如更改密码、提交订单等）时没有对请求的来源进行验证，攻击者可以通过构造恶意网站，诱使用户在另一个标签页或浏览器中访问，并触发恶意请求。
    - **未使用防御性措施**： 应用程序可以使用一些防御性措施来防范 CSRF 攻击，例如生成随机的令牌（CSRF Token）并将其嵌入到表单中。攻击者无法获得受保护的令牌，从而无法成功伪造请求。
    - **会话管理不当**： 如果应用程序的会话管理机制存在缺陷，攻击者可能能够获取受害者的会话凭证并在其名义下执行操作。
    - **敏感操作缺乏确认**： 如果应用程序没有在执行敏感操作之前要求用户确认，攻击者可以构建一个诱骗用户执行操作的链接或按钮，从而在用户不知情的情况下执行该操作。

3. CSRF 漏洞利用

CSRF 漏洞挖掘

- 跨站请求伪造，主要是伪造用户一些重要操作，但是因为攻击者看不到伪造请求的响应结果，因此在漏洞挖掘时，着重点要放在用户的“增“、“删”、“改”这些操作上。
    - **增**：自动增加用户、收藏指定的店铺等。
    - **删**：删除一条留言、好友等。
    - **改**：修改密码、转账等。

CSRF可以做什么?

你这可以这么理解CSRF攻击：

攻击者盗用了你的身份，以你的名义发送恶意请求。

CSRF能够做的事情包括：

以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账.....

造成的问题包括：个人隐私泄露以及财产安全。

相关知识回顾

想要深入理解CSRF的攻击特性我们有必要了解一下网站Session的工作原理。

Session大家都不陌生，无论是用.net还是PHP开发过网站的程序员都肯定用过Session对象，然而Session它是如何工作的呢？如果我们把浏览器的Cookie禁用了，大家认为Session还能正常工作吗？ 答案是否定的。

```
在这里举个简单的例子帮助大家理解Session。

比如我买了一张高尔夫俱乐部的会员卡，俱乐部给了我一张带有卡号的会员卡。我能享受哪些权利呢?
如果我是高级会员卡可以打19洞和后付费喝饮料，而初级会员卡只能在练习场挥杆。
我的个人资料都是保存在高尔夫俱乐部的数据库里，我每次去高尔夫俱乐部只需要出示这张高级会员卡，俱乐部就知道我是谁了，并且为我服务了。 因此我们的高级会员卡卡号相当于保存在Cookie的Sessionid；而我的高级会员卡权利和个人信息就相当于服务端的Session对象。
```

我们知道Http是无状态的协议，它不要求浏览器在每次请求中标明客户端自己的身份，并且浏览器以及服务器之间并没有保持一个持久性的连接用于多个页面之间的访问。为了维持web应用程序状态的问题，每次Http请求都会将本域下的所有Cookie作为Http请求头的一部分发送给服务端，服务器端就可以根据请求中的Cookie所存放的Sessionid去Session对象中找到该会员资料了。

我们理解了Session的工作机制后，CSRF也就很容易理解了。CSRF攻击就相当于攻击用户复制了我的高级会员卡，然后攻击用户就可以拿着这张假冒的高级会员卡去高尔夫俱乐部打19洞，享受美味的饮料，而我这个受害者在月底就会收到高尔夫俱乐部的账单。

CSRF 漏洞检测

检测CSRF漏洞是一项比较繁琐的工作，最简单的方法就是抓取一个正常请求的数据包，去掉Referer字段后再重新提交，如果该提交还有效，那么基本上可以确定存在CSRF漏洞。 随着对CSRF漏洞研究的不断深入，不断涌现出一些专门针对CSRF漏洞进行检测的工具，如CSRFTester，CSRF Request Builder等。以CSRFTester工具为例，CSRF漏洞检测工具的测试原理如下：使用CSRFTester进行测试时，首先需要抓取我们在浏览器中访问过的所有链接以及所有的表单等信息，然后通过在CSRFTester中修改相应的表单等信息，重新提交，这相当于一次伪造客户端请求。如果修改后的测试请求成功被网站服务器接受，则说明存在CSRF漏洞，当然此款工具也可以被用来进行CSRF攻击。

- 在漏洞的检测方面，不管是 `GET` 型还是 `POST` 型，都可以分为下述两步进行：
  a. 先判断重要操作（增删改）的数据包中是否有 `token` 等随机值。如果存在随机值，先去除该值尝试能否正常发送请求。如果可以发送则可能存在 `CSRF` 漏洞。
  b. 再判断是否存在 `Referer` 字段，去掉 `Referer` 字段后再重新提交，如果该提交还

有效，那么基本上可以确定存在 `CSRF` 漏洞。

CSRF 漏洞攻击

- `CSRF` 最初的一个错误观点，认为 `CSRF` 只能由 `GET` 请求发起，因此一些开发者认为只要把重要的操作改为只允许 `POST` 请求就能防 `CSRF` 。如果一些请求是 `POST` 类型的，最简单的方法就是在攻击页面构造好一个 `form` 表单，然后用 `JavaScript` 自动提交这个表单。

pikachu靶场演示

GET 方式构造

- 当发现存在 `CSRF` 的漏洞点使用的是 `GET` 型传参时，可以直接通过构造 `URL` 的方式进行攻击。

构造 URL

- 只需要一个 HTTP 请求，让对方点击都就会直接提交数据，一般会这样利用：

```
http://pikachu/vul/csrf/csrfget/csrf_get_edit.php?sex=boy&phonenum=12345678901&add=japan&email=123456&submit=submit
```

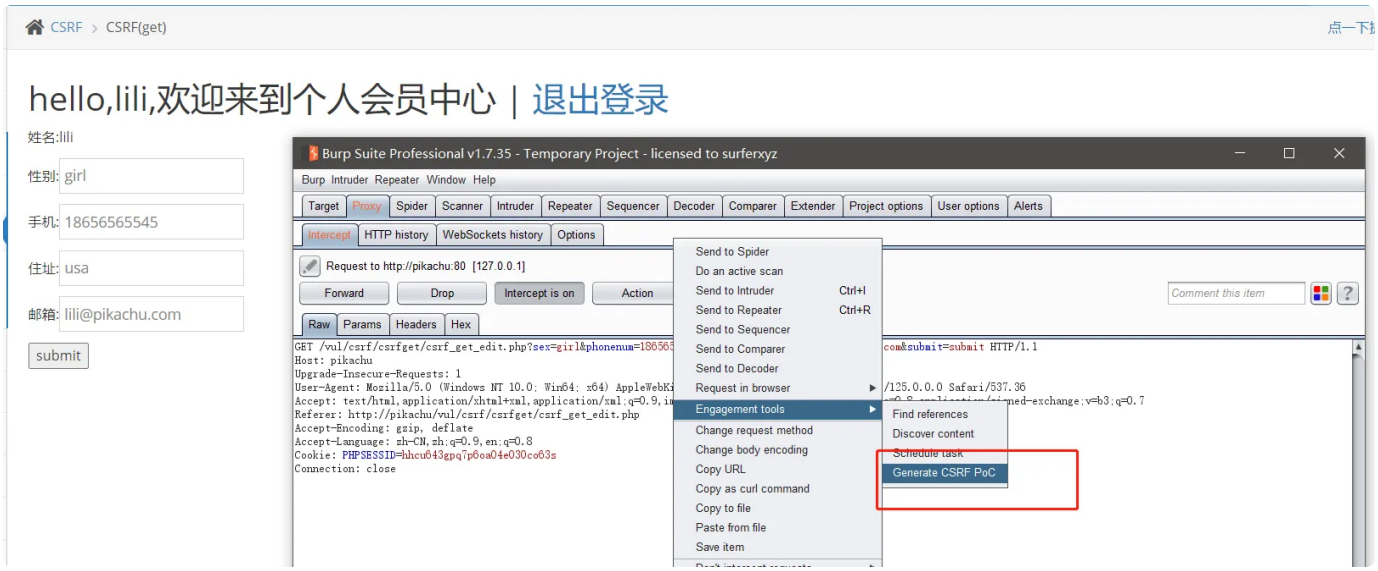
构造短链接

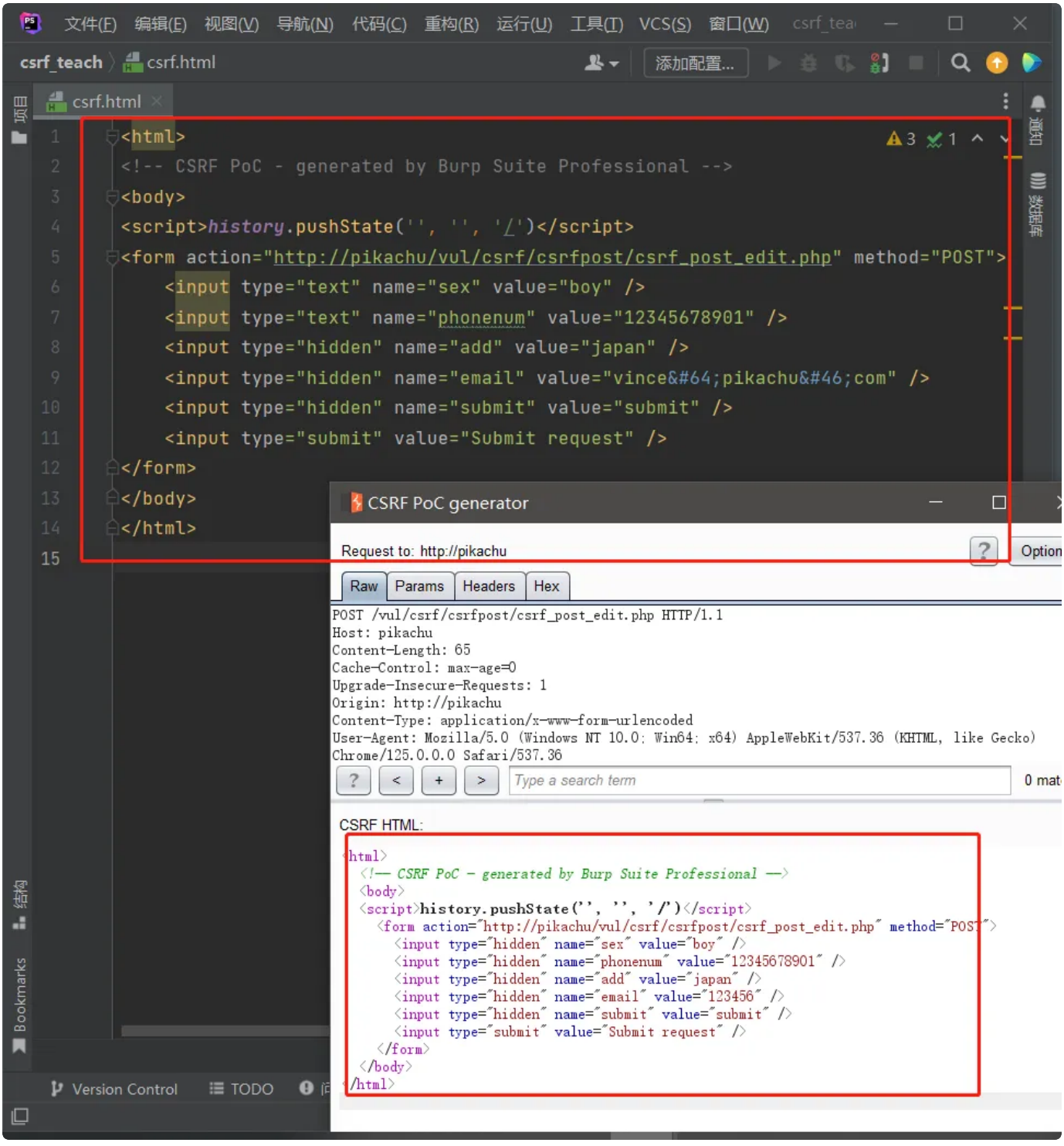
- 这样直接构造出的 `URL` 隐蔽性太低，可以通过短链接的方式对 `URL` 进行隐蔽，这里推荐三个短链接生成网站：
  - https://www.985.so/
  - https://www.dwz.lc/
  - https://www.mynb8.com/

POST 方式构造

- 当发现存在 `CSRF` 的漏洞点使用的是 `POST` 型传参时，可以直接通过构造 `form` 表单的方式进行攻击。

构造表单



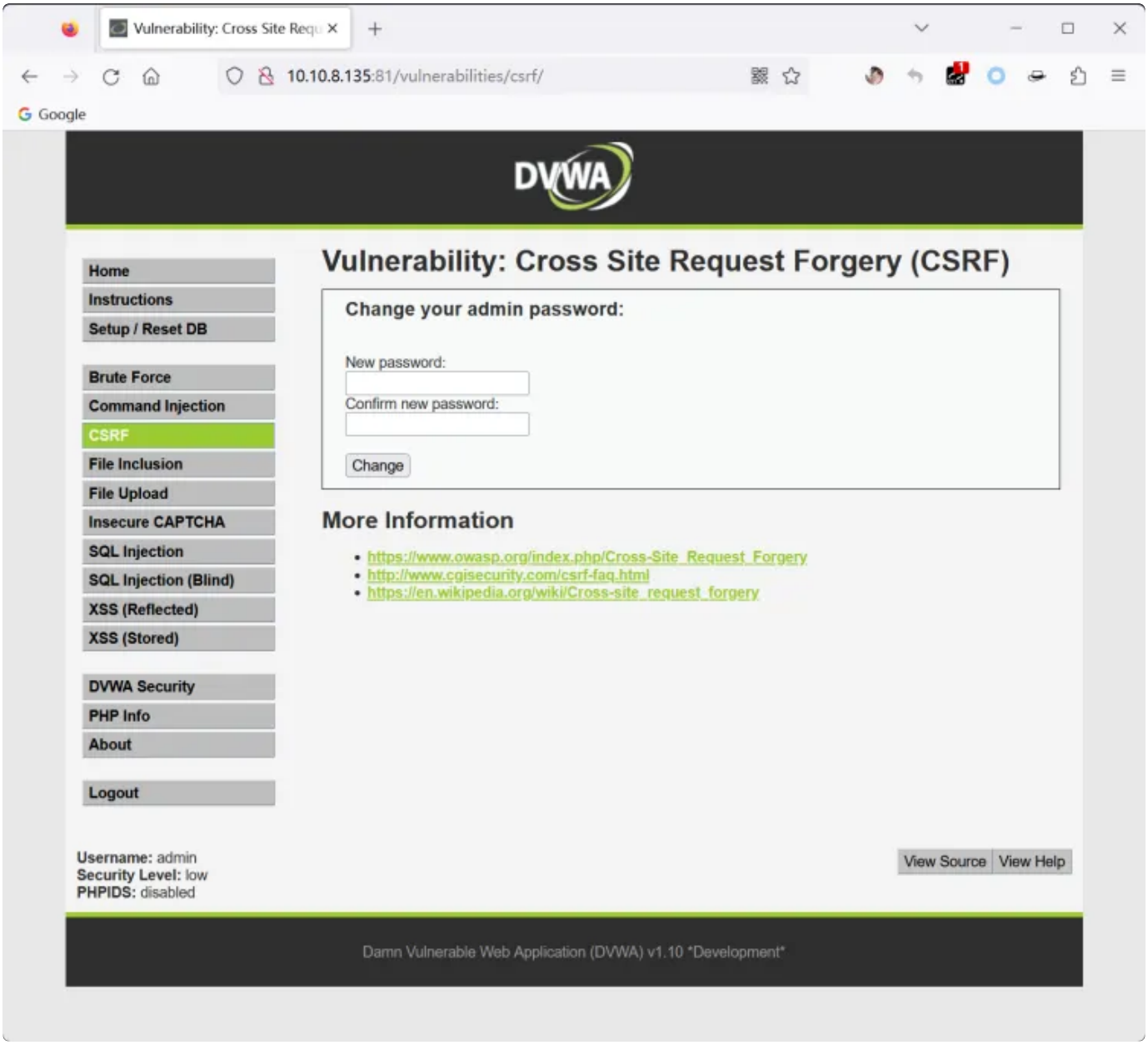


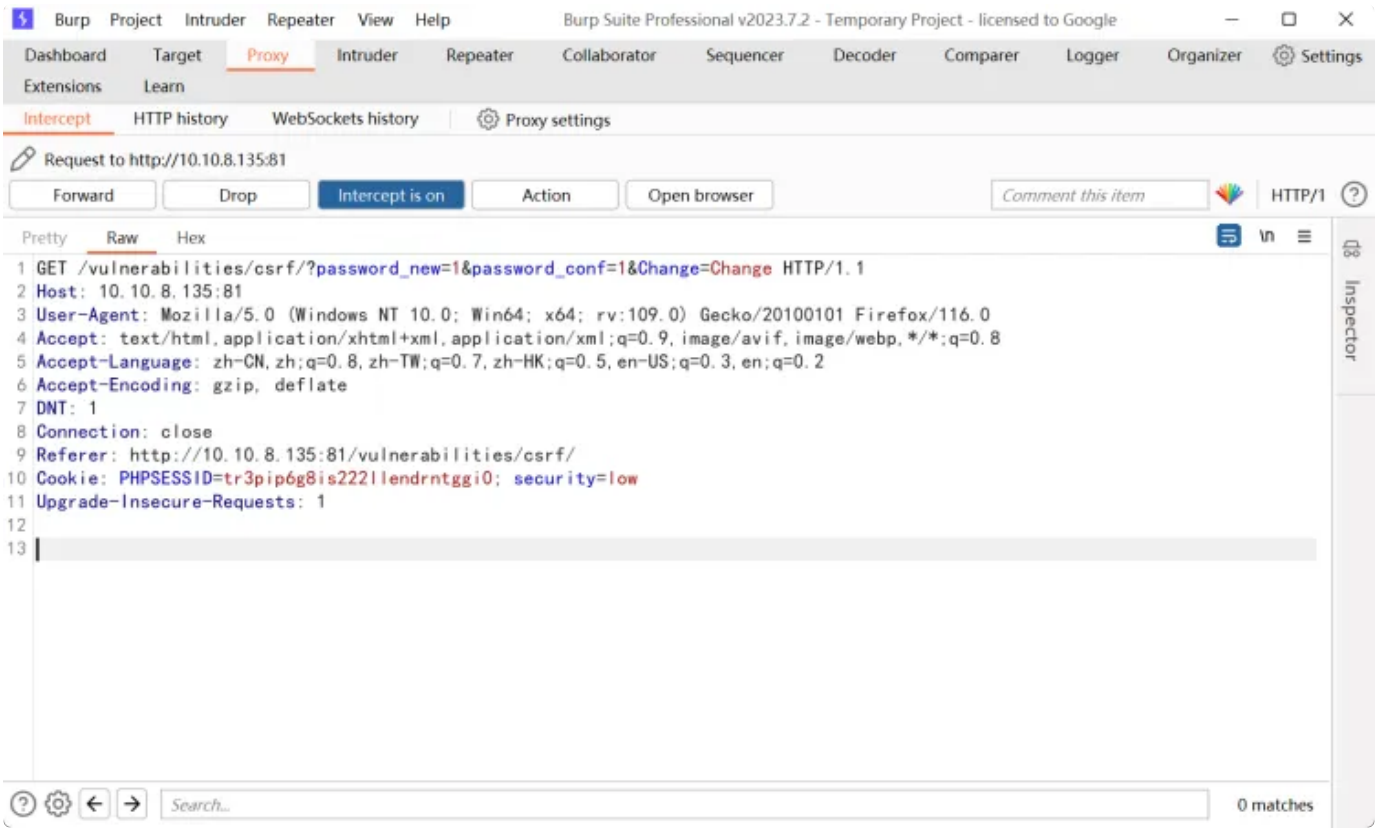
- 只需要简单编写一个 `form` 表单即可，一般会这样利用：

```html
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://pikachu/vul/csrf/csrfpost/csrf_post_edit.php" method="POST">
      <input type="hidden" name="sex" value="boy" />
      <input type="hidden" name="phonenum" value="15988767673" />
      <input type="hidden" name="add" value="nba&#32;lakes" />
      <input type="hidden" name="email" value="kobe&#64;pikachu&#46;com" />
      <input type="hidden" name="submit" value="submit" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

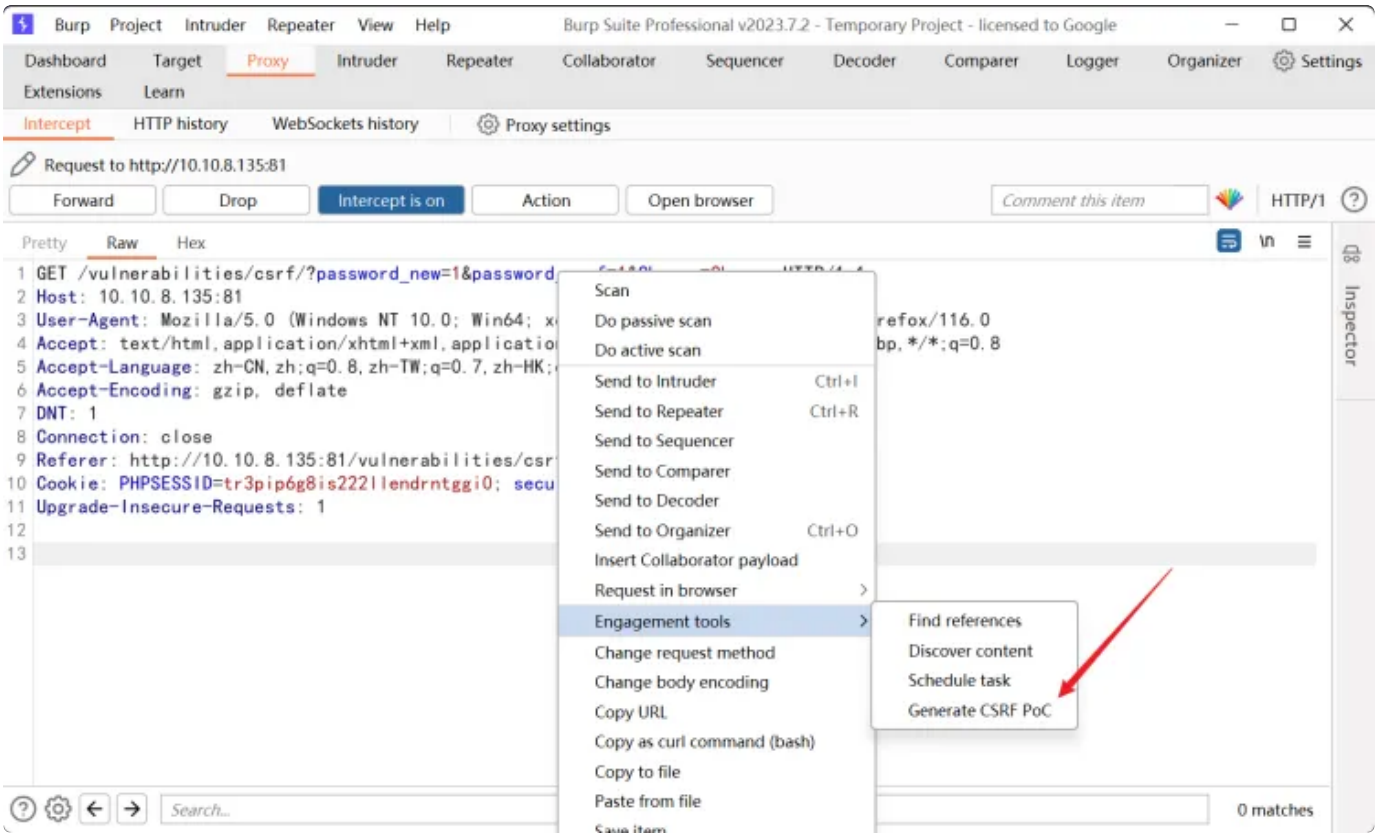
DVWA

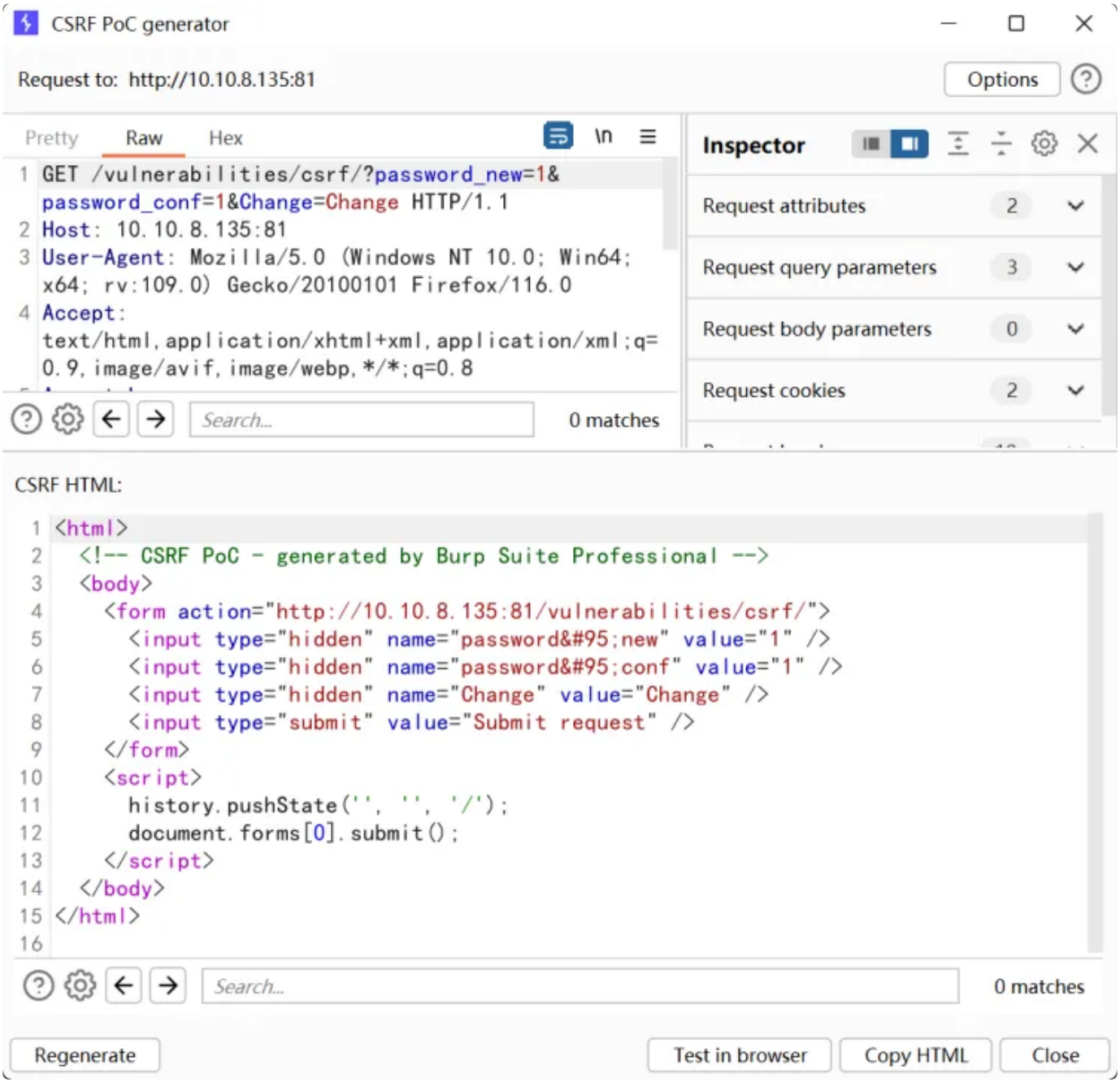
- 使用 `BurpSuite` 代理抓取对应报文：


Vulnerability: Cross Site Requ
10.10.8.135:81/vulnerabilities/csrf/
Google
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*



- 右键选择生成 `CSRF POC`：





注：这样类型的工具还有 – OWASP CSRF Tester

了解CSRF的机制之后，危害性我相信大家已经不言而喻了，我可以伪造某一个用户的身份给其好友发送垃圾信息，这些垃圾信息的超链接可能带有木马程序或者一些欺骗信息（比如借钱之类的），如果CSRF发送的垃圾信息还带有蠕虫链接的话，那些接收到这些有害信息的好友万一打开私信中的链接就也成为了有害信息的散播着，这样数以万计的用户被窃取了资料种植了木马。整个网站的应用就可能在瞬间崩溃，用户投诉，用户流失，公司声誉一落千丈甚至面临倒闭。曾经在MSN上，一个美国的19岁的小伙子Samy利用CSS的background漏洞几小时内让100多万用户成功的感染了他的蠕虫，虽然这个蠕虫并没有破坏整个应用，只是在每一个用户的签名后面都增加了一句“Samy 是我的偶像”，但是一旦这些漏洞被恶意用户利用，后果将不堪设想，同样的事情也曾经发生在新浪微博。

相关举例：

受害者Bob 在银行有一笔存款，通过对银行的网站发送请求

“http://bank.example/withdraw?account=bob&amount=1000000&for=bob2”可以使 Bob把1000000块的存款转到Bob2的账号下。通常情况下，该请求发送到网站后，服务器会先验证该请求是否来自一个合法的Session，并且该Session的用户Bob已经成功登陆。黑客Hacker自己在该银行也有账户，他知道上文中的URL可以把钱进行转帐操作。

Hacker可以自己发送一个请求给银行：

http://bank.example/withdraw?account=bob&amount=1000000&for=Hacker。

但是这个请求来自Hacker而非 Bob，他不能通过安全认证，因此该请求不会起作用。这时，Hacker想到使用CSRF的攻击方式，他先自己做一个网站，在网站中放入如下代码：

src=“http://bank.example/withdraw?account=bob&amount=1000000&for=Hacker”，并且通过广告等诱使 Bob 来访问他的网站。当Bob访问该网站时，上述URL就会从Bob的浏览器发向银行，而这个请求会附带Bob浏览器中的 Cookie 一起发向银行服务器。大多数情况下，该请求会失败，因为他要求Bob的认证信息。但是，如果Bob当时恰巧刚访问他的银行后不久，他的浏览器与银行网站之间的Session尚未过期，浏览器的Cookie之中含有Bob的认证信息。这时，悲剧发生了，这个URL请求就会得到响应，钱将从Bob的账号转移到Hacker的账号，而Bob当时毫不知情。等以后Bob发现账户钱少了，即使他去银行查询日志，他也只能发现确实有一个来自于他本人的合法请求转移了资金，没有任何被攻击的痕迹。而Hacker则可以拿到钱后逍遥法外。

4. CSRF防御

目前防御 CSRF 攻击主要有三种策略：

- 验证 HTTP Referer 字段；
- 在请求地址中添加token并验证；
- 在HTTP头中自定义属性并验证。

验证Referer

根据 HTTP 协议，在HTTP头中有一个字段叫Referer，它记录了该 HTTP 请求的来源地址。在通常情况下，访问一个安全受限页面的请求来自于同一个网站，比如需要访问

`http://bank.example/withdraw?account=bob&amount=1000000&for= Hacker`，

用户必须先登陆bank.example，然后通过点击页面上的按钮来触发转账事件。这时，该转帐请求的Referer值就会是转账按钮所在的页面的URL，通常是以bank.example域名开头的地址。而如果黑客要对银行网站实施 CSRF 攻击，他只能在他自己的网站构造请求，当用户通过黑客的网站发送请求到银行时，该请求的 Referer 是指向黑客自己的网站。因此，要防御CSRF攻击，银行网站只需要对于每一个转账请求验证其Referer值，如果是以bank.example开头的域名，则说明该请求是来自银行网站自己的请求，是合法的。如果Referer是其他网站的话，则有可能是黑客的CSRF攻击，拒绝该请求。

这种方法的显而易见的好处就是简单易行，网站的普通开发人员不需要操心CSRF的漏洞，只需要在最后给所有安全敏感的请求统一增加一个拦截器来检查Referer的值就可以。特别是对于当前现有的系统，不需要改变当前系统的任何已有代码和逻辑，没有风险，非常便捷。

然而，这种方法并非万无一失。Referer的值是由浏览器提供的，虽然 HTTP协议上有明确的要求，但是每个浏览器对于Referer的具体实现可能有差别，并不能保证浏览器自身没有安全漏洞。使用验证 Referer 值的方法，就是把安全性都依赖于第三方（即浏览器）来保障，从理论上来讲，这样并不安全。事实上，对于某些浏览器，比如 IE6 或 FF2，目前已经有一些方法可以篡改Referer值。如果bank.example网站支持 IE6 浏览器，黑客完全可以把用户浏览器的Referer值设为以bank.example域名开头的地址，这样就可以通过验证，从而进行 CSRF 攻击。

添加token

CSRF 攻击之所以能够成功，是因为黑客可以完全伪造用户的请求，该请求中所有的用户验证信息都是存在于Cookie中，因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的Cookie来通过安全验证。要抵御CSRF，关键在于在请求中放入黑客所不能伪造的信息，并且该信息不存在于Cookie之中。可以在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。　　这种方法要比检查Referer要安全一些，token可以在用户登陆后产生并放于session之中，然后在每次请求时把token从session中拿出，与请求中的token进行比对，但这种方法的难点在于如何把token以参数的形式加入请求。对于 GET 请求，token将附在请求地址之后，这样 URL 就变成 http://url?csrftoken=tokenvalue。 而对于 POST 请求来说，要在 form 的最后加上

`<input type="hidden" name="csrftoken" value="tokenvalue"/>`，

这样就把 token 以参数的形式加入请求了。

token存在于哪儿?

```
token 在客户端一般存放于localStorage、cookie(页面没有刷新、页面没有关闭，cookie也会过期)、或sessionStorage中

localStorage    生命周期是永久，这个特性主要是用来作为本地存储来使用的，解决了cookie存储空间不足的问题                (cookie中每条cookie的存储空间为4k)，localStorage中一般浏览器支持的是5M大小，这个在                不同的浏览器中localStorage会有所不同。它只能存储字符串格式的数据，所以最好在每次存储时                把数据转换成json格式，取出的时候再转换回来。

sessionStorage  生命周期为当前窗口或标签页，存在于会话中。
cookie
```

5. CSRF常见面试问题

1. CSRF和SSRF的区别

CSRF为跨站请求伪造，SSRF为服务器请求伪造,CSRF的核心是让客户端的浏览器去访问，SSRF核心是让服务器去访问。

2. CSRF与XSS的区别

csrf伪造请求,xss为跨站脚本攻击,一个为伪造cookie等验证信息,一个则是使用一个脚本攻击。

XSS 核心是操作目标网站的HTML代码 窃取Cookie 。

CSRF 核心是在非目标网站的HTML代码做手脚 让受害者浏览器偷偷的去访问目标网站。

3. 为什么CSRF会利用的cookie信息可以不过过期?

因为Cookie具有时效性，但时效性不一定，常见的是一次会话，什么是一次会话？就是浏览器关闭开启一次。

4. Cookie是什么?

Cookie就是个验证信息，用于对登陆的用户进行一个验证,看看是什么身份。

5. Cookie在CSRF中的作用?

CSRF中，就是利用这个cookie，来盗用其他用户的身份来进行操作，就相当于有个人拿了你的身份证去上网，cookie就相当于这个身份证

6. CSRF怎么和XSS打组合拳

首先使用XSS直接执行CSRF的JS恶意语句

`<script>csrf构造好的恶意代码</script>`

7. csrf可以用于钓鱼吗

可以，可以和xss一起用，用一个反射型的xss,盗取cookie后，构造csrf脚本或脚本去攻击

8. CSRF的成因

Cookie不过期，没有进行进一步的验证用户信息，没有安全意识访问了恶意站点
<img src=>
<img >

9. csrf的操作流程

* 攻击者构造网站A：http://192.168.13.1/csrf/test.html
* 受害者使用网站B：http://csrf-csb.gxalabs.com/adminscoreedit.php?user_id=2&course_id=1
* 条件：网站B，必须是登录状态
* 这里访问网站A，其实就是请求一个伪造的链接
* 网站A实际上是构造了一个向网站B发送请求的链接
* 实际上目标访问的还是网站B
* 浏览器会讲网站B的cookie，一起发送给服务，服务器就认为，此次请求是合法的

10. csrf能有什么操作

转账，修改密码，越权等N种操作

11. 一定要登陆A时同时访问C吗

不一定,cookie都有时效性，一次性的cookie必须这样，不是一次性的则会存在硬盘中

12. CSRF可以直接盗取在本地的cookie吗

不可以，但这是xss的操作,csrf只是利用cookie

13. CSRF的载体

一个包含恶意链接或脚本

14. csrf poc是什么

一个带验证的数据表单的脚本或链接，内有要操作的数据

15. csrf操作过程

生成一个脚本或链接，当你操作时触发，就获得你在目标网站上的cookie把数据提交到目标地址，当你访问或者操作时，浏览器偷偷执行了攻击者的操作

16. csrf是什么

csrf为跨站请求伪造，伪造利用其它用户来完成请求操作，简单来说就是让别人来进行我想要的操作

17. token防御csrf原理

Token就是一段字母数字的随机值，访问时服务端会生成一个随机的token值并传回到前端的表单里，当我们提交表单时，token会作为一个参数提交到服务端进行验证 [终极方案]

18. referer防御csrf原理

Refer记录了该http请求的来源地址，访问一个安全受限页面请求必须来自同一个网站，如果不是则请求不合法无法生效 [可绕过]

19. 如何绕过token 随机生成字符串+生成的时间

Token的生成一定要随机

有些Token根本就不验证或者时间戳做Token

如果存在xss漏洞，token防御将无效

20. csrf如何绕过referer

当浏览器的检测不严，比如referer值为空时，可以绕过

```
136   Referer可以伪装绕过
```