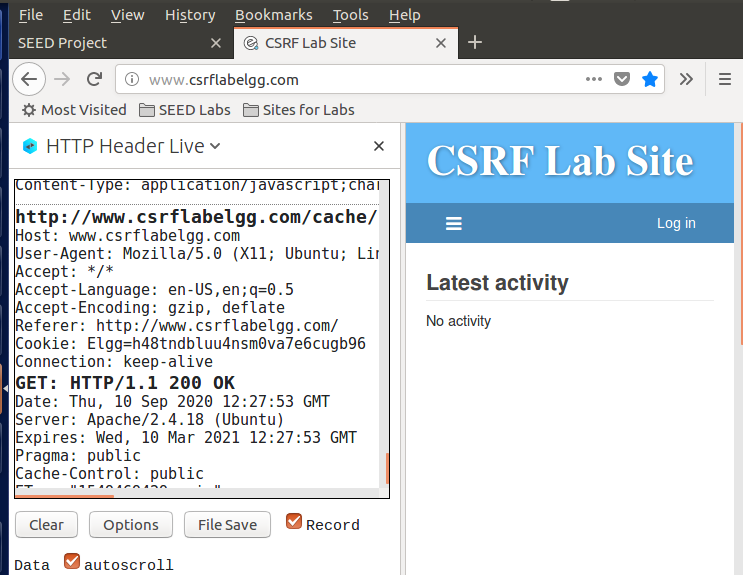
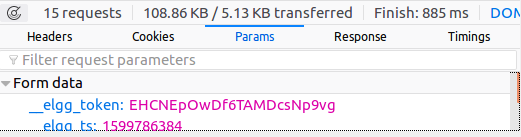
网安实训实验报告

Lab 4

57118139 顾宸玮

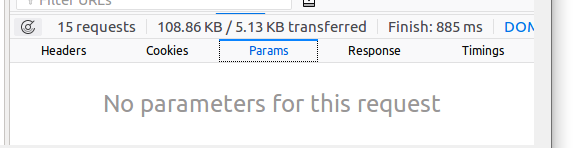
**Task 1: Observing HTTP Request.**







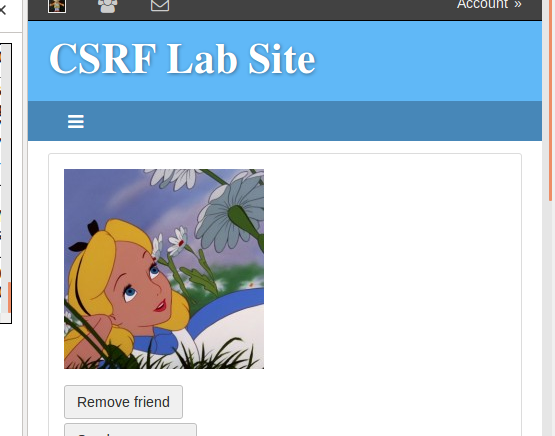
通过查看params选项查看到POST的参数，有elgg的访问令牌\_elgg\_token，\_elgg\_ts,登录密码，用户名等参数。



而对于GET来说，此请求并没有具体参数。

**Task 2: CSRF Attack using GET Request**

首先登录Boby账号，然后添加Alice为好友：



在使用HTTP Header Live后，抓包显示的网址如下：



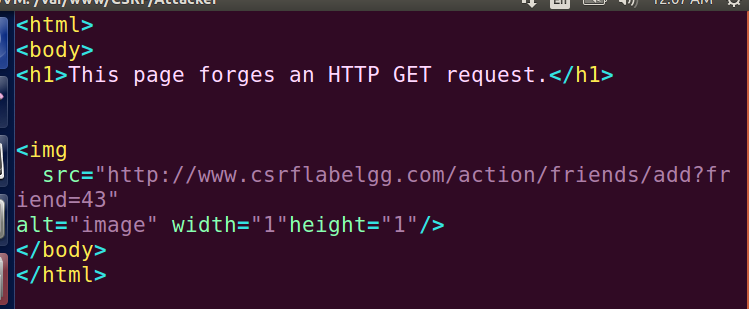
开头为Elgg添加好友请求的URL，在伪造请求中，需要设置目标URL，除此之外，还需要添加指定用户ID，即为friend参数。

要使Alice加boby为好友，需要知道boby的ID

经过测试，发现boby的ID为43，所以最终伪造的URL为

<http://www.csrflabelgg.com/action/friends/add?friend=43>

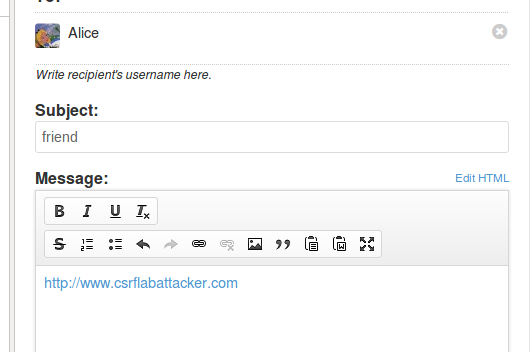
我们现在做一个恶意网站：



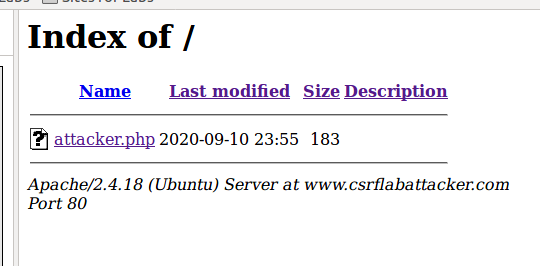
将以上php文件放到/var/www/CRSF/Attacker文件夹中。

登录boby账号，发送一个邮件给Alice。将以下链接发给Alice

<http://www.csrflabattacker.com>，出现以下界面：



Alice点开链接：

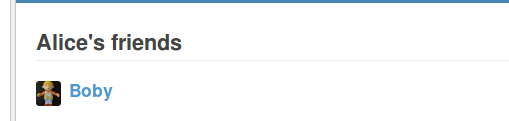




Alice点完链接，发现成功加了boby好友。



成功加为好友



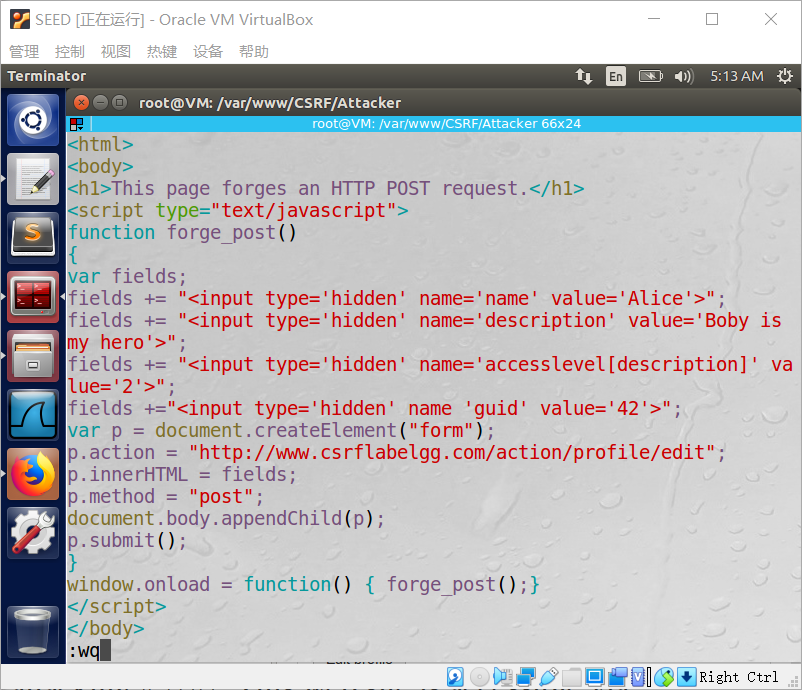
**Task 3: CSRF Attack using POST Request**

首先编写java文件：

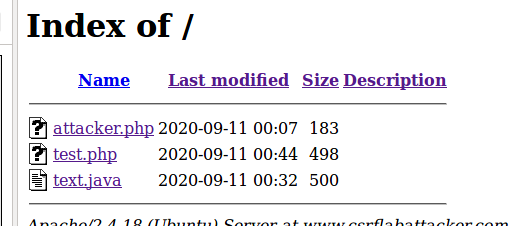
通过boby访问Alice的主页，查询HTTP Header Live的GET请求发现Alice的guid为42：



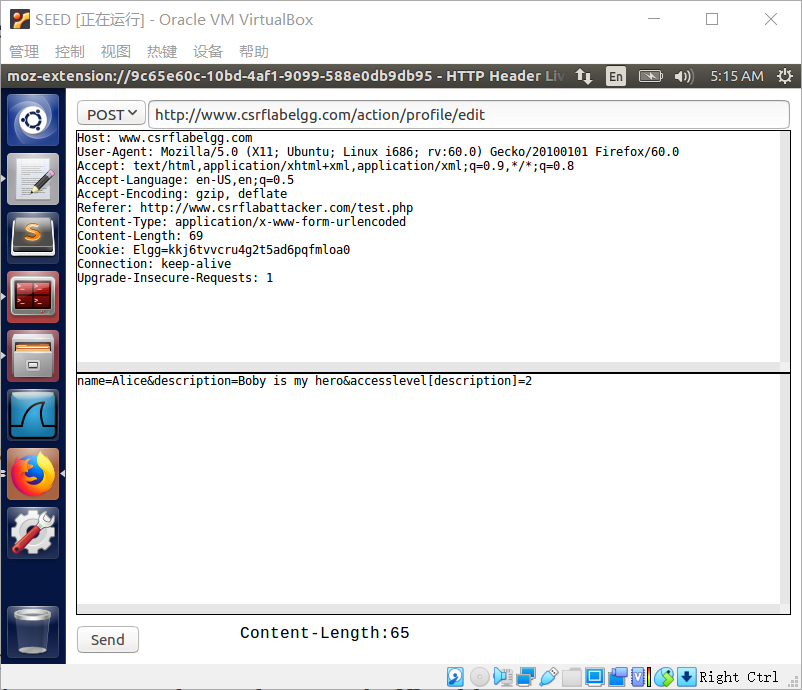
编写java文件，修改Alice的description为“Boby is my hero”：



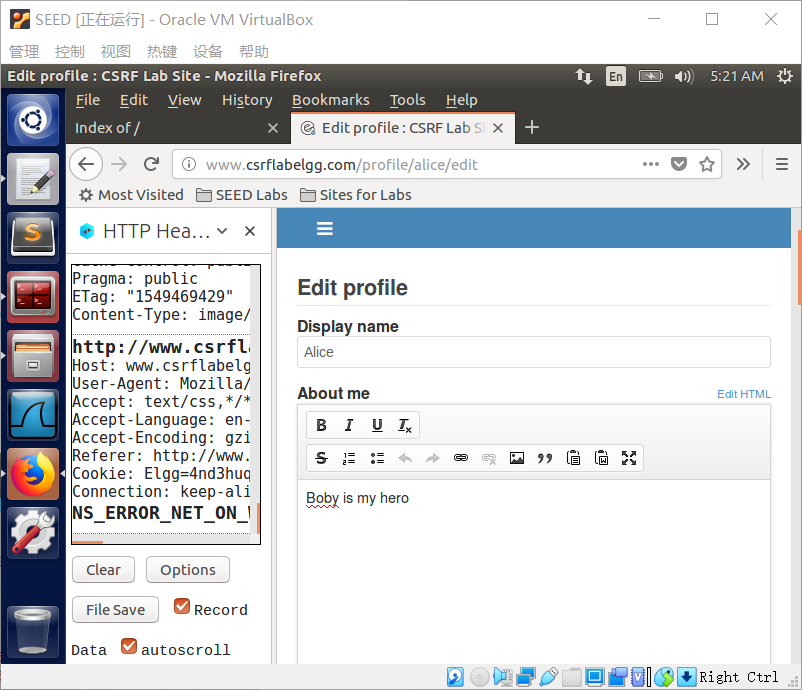
同样boby给Alice发送恶意链接，Alice打开test.php后



通过HTTP Header Live捕获到的POST请求：



最终Alice的profile修改成功！



问题一：The forged HTTP request needs Alice’s user id (guid) to work properly. If Boby targets Alice specififically, before the attack, he can fifind ways to get Alice’s user id. Boby does not know Alice’s Elgg password, so he cannot log into Alice’s account to get the information. Please describe how Boby can solve this problem.

在Alice输入账号密码成功登录Elgg后，网页已经保存了Alice的cookie，在Alice在同一网页浏览boby发送的恶意网站链接，，浏览器就带着Alice登录后认证通过的cookie信息访问这个恶意链接的第三方网站，恶意请求就会在这个时候执行，boby可通过一些抓包软件抓取到Alice的用户信息。

问题二：If Boby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim’s Elgg profifile? Please explain.

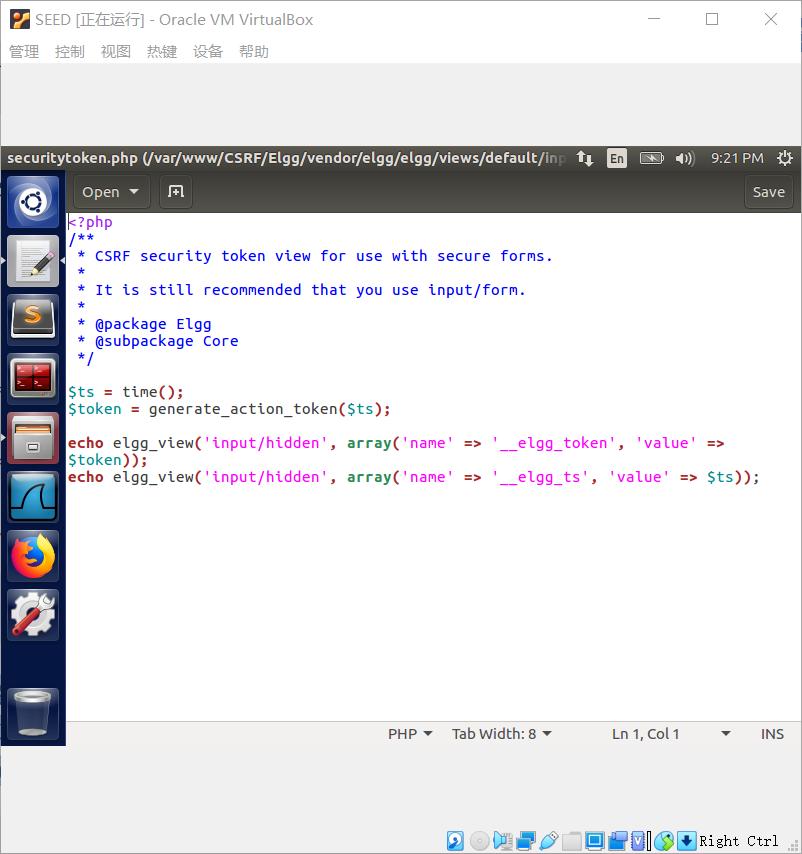
可以。因为用户登录账户的信息被存在了cookie里，boby可以在不知道验证信息的情况下直接利用用户自己的cookie来通过安全验证，从而进入用户账号修改信息。

**Task 4: Implementing a countermeasure for** Elgg

\_\_elgg\_ts和\_\_elgg\_token由

/var/www/CSRF/Elgg/vendor/elgg/views/default/input/securitytoken.php生成。

并添加到web页面。下面的代码片段显示了如何将其动态添加到web页面。

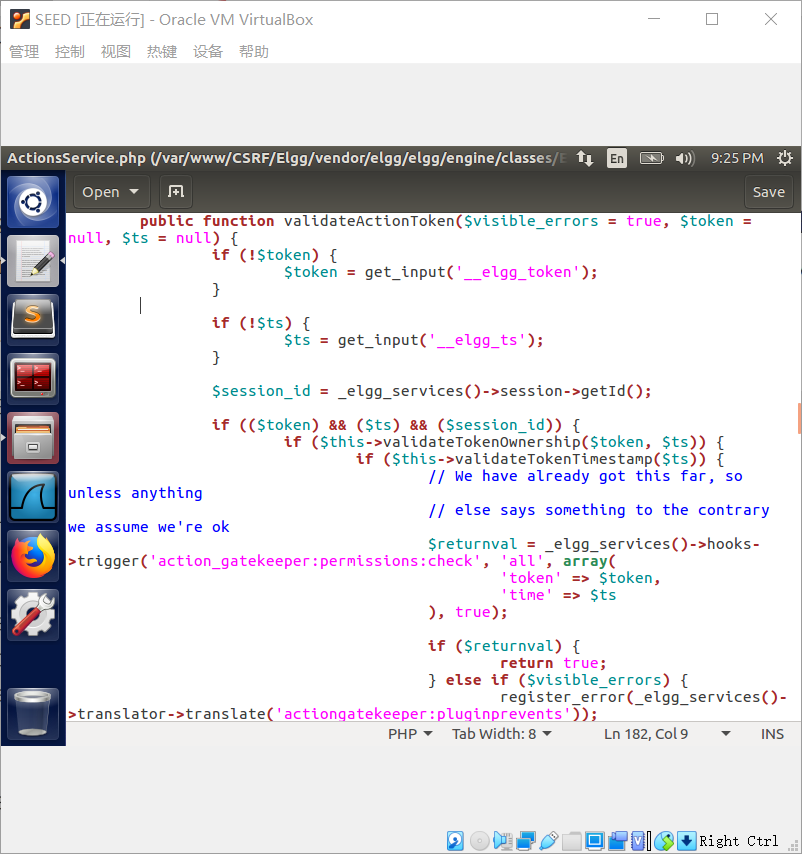


\_\_elgg\_ts和\_\_elgg\_token的值都是在/var/www/CSRF/Elgg/vendor/elgg/engine/classes/Elgg/ActionService.php里的generateActionToken中的函数生成的。Elgg安全令牌是站点机密值(从数据库检索)、时间戳、用户sessionID和随机生成的会话字符串的散列值(md5消息摘要)。通过防御CSRF的攻击。下面的代码显示了在Elgg中生成秘密令牌的过程。

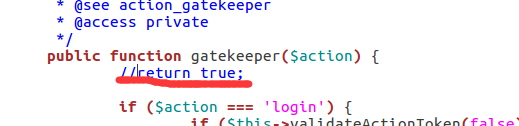


elgg web应用程序验证生成的令牌和时间戳防御CSRF的攻击。每个用户操作都调用验证操作令牌函数，该函数验证令牌。如果令牌不存在或无效，操作将被拒绝，用户将被重定向。

下面的代码片段显示了函数验证操作令牌。



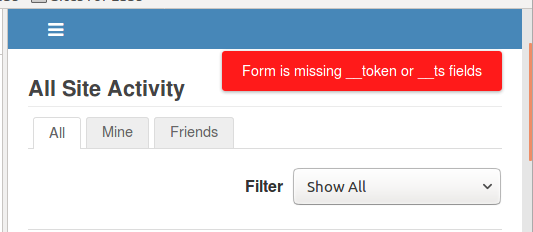
关闭/var/www/CSRF/ Elgg/vendor/elgg/elgg/engine/classes/Elgg中的gatekeeper（）函数中防御机制，注释语句：return true；

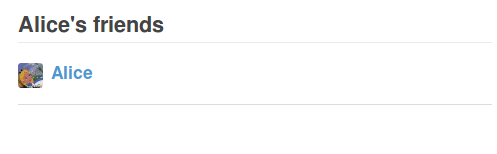


重复之前的攻击

1. Alice加Boby好友

首先删除Boby好友，再点击之前恶意网站链接：



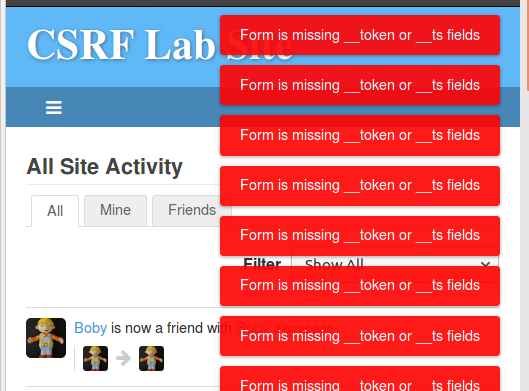


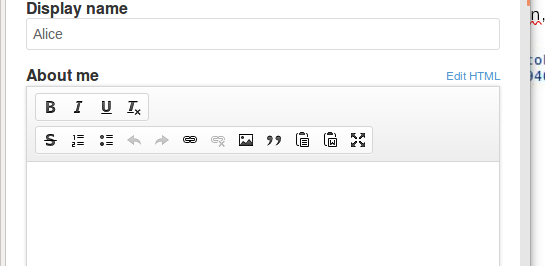
网页提示丢失\_\_elgg\_ts和\_\_elgg\_token，并且Alice没有成功添加Boby好友。



1. 编辑：Boby is my hero

出现内容和1一样，并且未编辑成功





失败原因：注释掉的语句是为了关闭Elgg的防护措施。之前实际上是让这个守护函数总是返回真，让所有的请求都通过验证。如果重复之前的攻击，攻击会失败。为了成功攻击，攻击者需要了解秘密令牌的值以及目标用户的Elgg页面内嵌的时间戳。然而，浏览器的访问控制机制会阻止攻击者网页的JavaScript代码访问Elgg页面的任何内容。

实验感想：

本次实验是体验了CSRF攻击和防御的过程。该过程让我深刻地体会到了在现实生活中一些恶意网站的操作，比如说短信，qq发送的陌生链接不能点击，差不多都是这个原理。在了解了CSRF攻击原理之后，我对网络攻防有了更深的了解。但是因为前期了解不足，所以做实验有很多困难，但是经过查找资料最终成功完成了实验。