|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《网络与系统安全》 实验报告** |
|  |
| 实验三  XSS 实验  学 院: 计算机科学与技术学院   |  |  | | --- | --- | | 姓 名: | 刘睿 | | 学 号: | 220110720 | | 专 业: | 计算机科学与技术 | | 日 期: | 2025年4月 | |

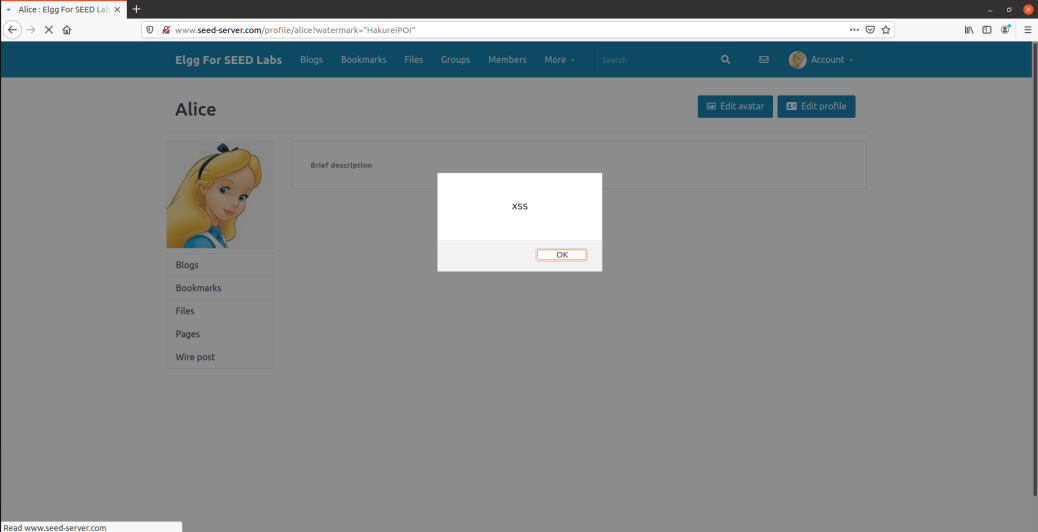
# 实验过程

任务一：发布恶意消息以显示警告窗口

登入Alice的账户后，在个人Profile界面修改Brief Description如下

<script>alert('XSS');</script>

在重新GET Alice的Profile界面时，JavaScript程序自动弹出了警告窗口

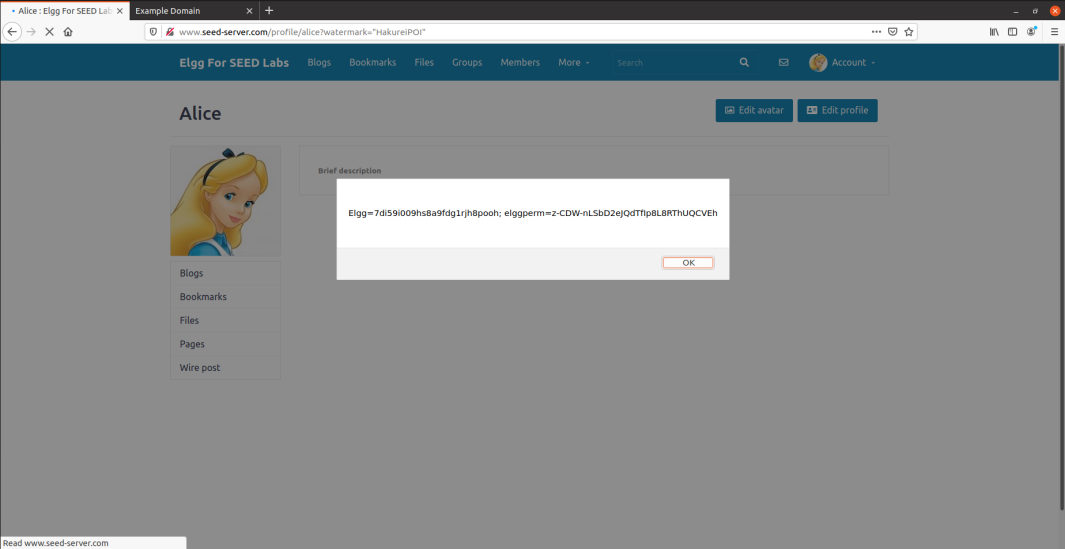


任务二：显示受害者的Cookie

修改任务一中嵌入的JavaScript程序如下：

<script>alert(document.cookie);</script>

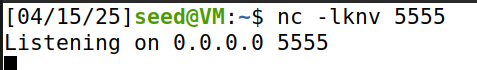
在重新加载Alice的Profile界面时，则自动弹出了用户的Cookie



任务三：从受害者的机器窃取Cookie

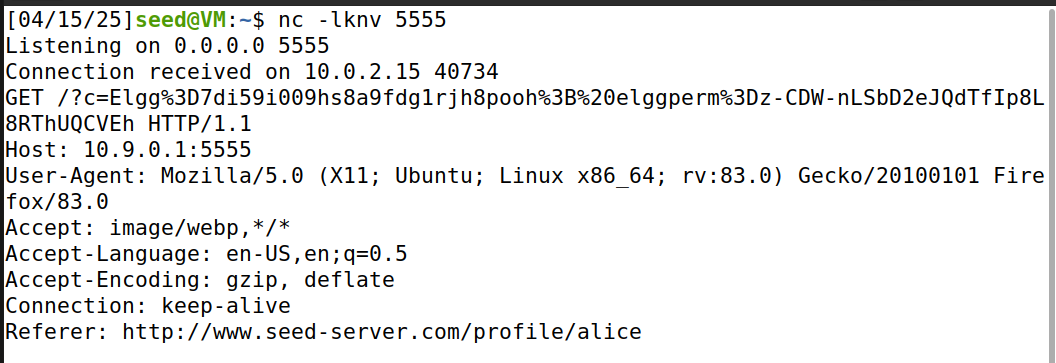
我们需要利用Netcat监听端口5555，并让Alice的Profile中嵌入的JavaScript程序将Alice的Cookie发送到攻击者机器的指定端口

首先打开监听端口



修改嵌入Alice Profile的程序

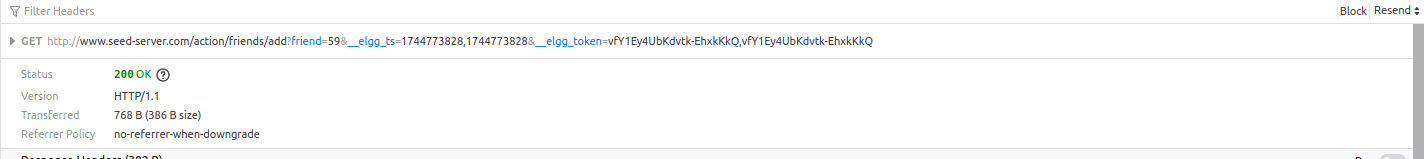
重新加载Profile页面后，发现我们的监听窗口接收到了一个HTTP GET请求，里面包含了用户的Cookie



任务四：成为受害者的“好友”

需要在黑客Samy的个人资料中嵌入一个JaveScript程序，受害者在打开Samy个人资料的时候就会自动发送一个添加Samy为好友的HTTP请求

我们使用Alice的账号向Samy手动发送一个好友请求，利用浏览器的开发者工具查看添加好友的HTTP请求如下

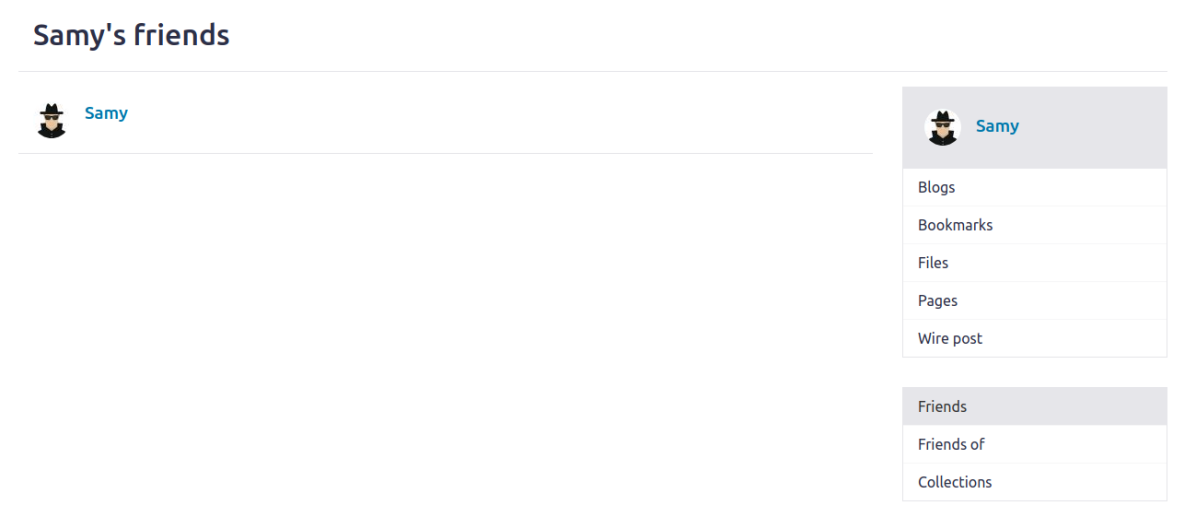


注意到添加好友的url是http://www.seed-server.com/action/friends/add，三个payload分别是friend（被添加对象的id，Samy是59），\_\_elgg\_ts，\_\_elgg\_token（时间和token的加密，但实验提供了接口，我们可以直接获得这两个字段的内容）

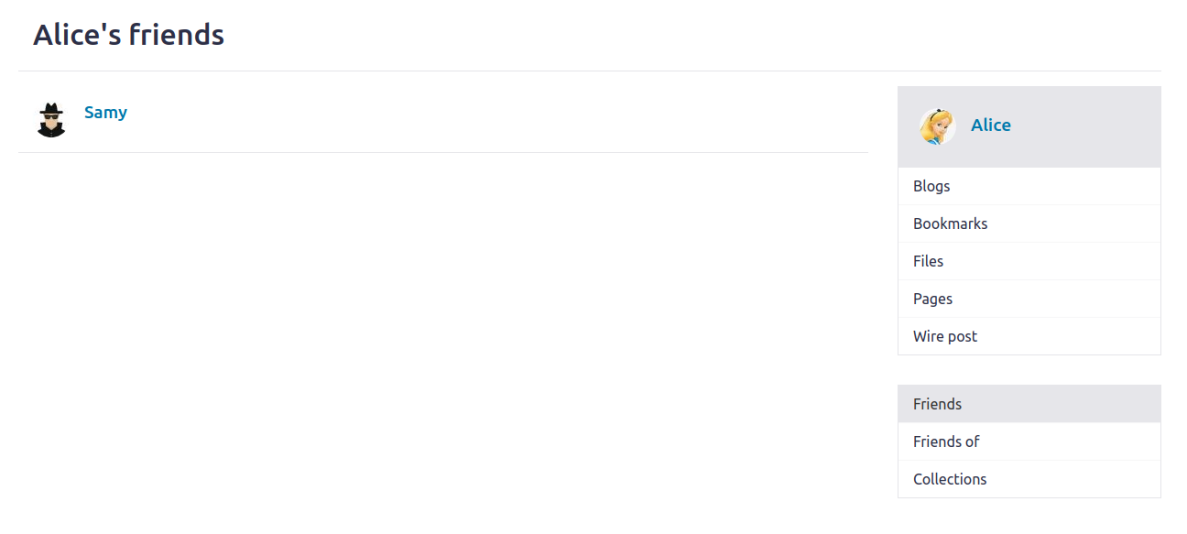
我们构造JavaScript程序如下：



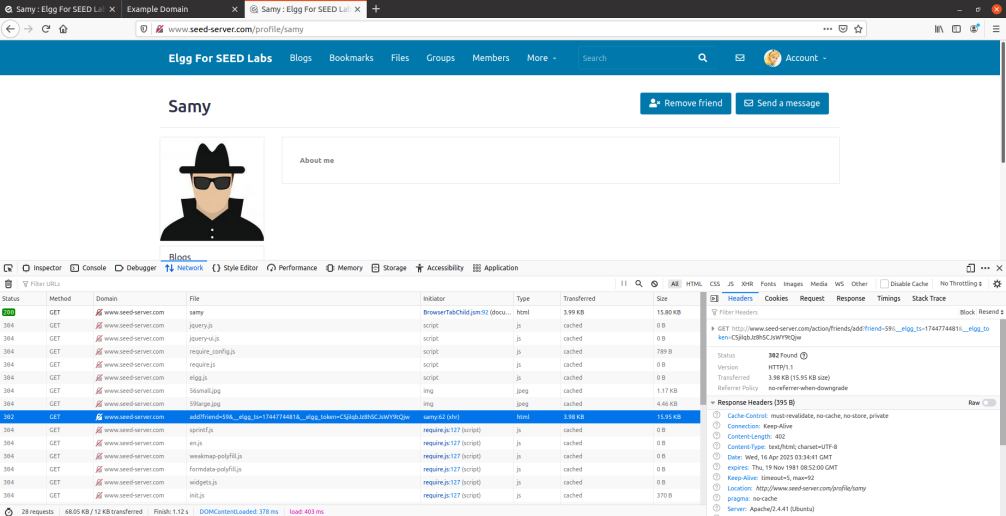
将其以文本模式嵌入Samy个人资料的“About Me”部分，重新加载Profile页面后发现，有趣的是，Samy自己将自己添加为了好友



尝试用Alice的账号访问Samy的Profile，成功添加Samy为好友了



注意到在访问Samy的Profile的时候，自动发送了一个添加Samy为好友的HTTP请求，如下图：



问题回答：

第①行和第②行的目的是获得加密后的时间戳和令牌，是我们payload的一部分。时间戳参数用于验证请求的时效性，防止旧的恶意请求被重放攻击；\_\_elgg\_token 是服务器用来校验用户请求是否合法的重要参数。如果请求中不包含该 token，服务器会认为该请求不是用户主动发起的，从而拒绝处理。该行确保恶意脚本发送的请求通过了服务器的 CSRF 验证

通常情况下难以发起攻击，编辑器模式会对JavaScript进行过滤转义或清洗，除非存在一些漏洞可以让我们绕过编辑器模式的过滤，让我们以某种手段插入精心设计的脚本

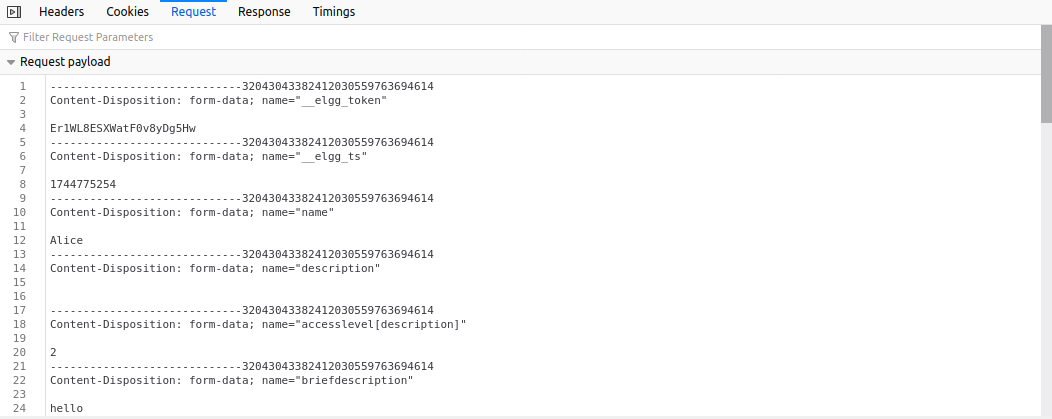
任务5：修改受害者的资料

需要在任务4的基础上，让嵌入脚本的作用改为修改受害者的资料

同样，我们先去尝试一下修改Profile具体发送了一个怎样的请求

编辑Profile的url是http://www.seed-server.com/profile/alice/edit，

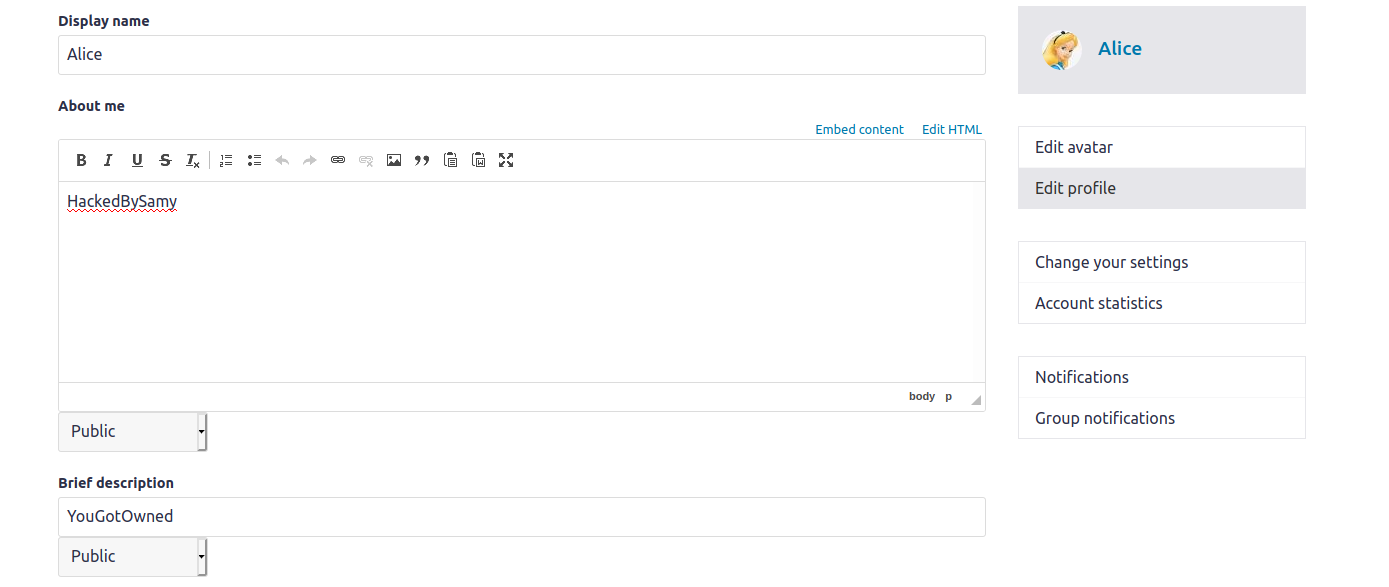
提交表单后，注意到我们的payload形式如下：



这个payload是一个form-data的形式，根据上述内容构造url如下



将其嵌入Samy的“About Me”中，尝试用Alice访问Samy的个人资料，重新打开Alice资料后发现已经被修改了（虽然不知道为什么对name的修改没有起作用）

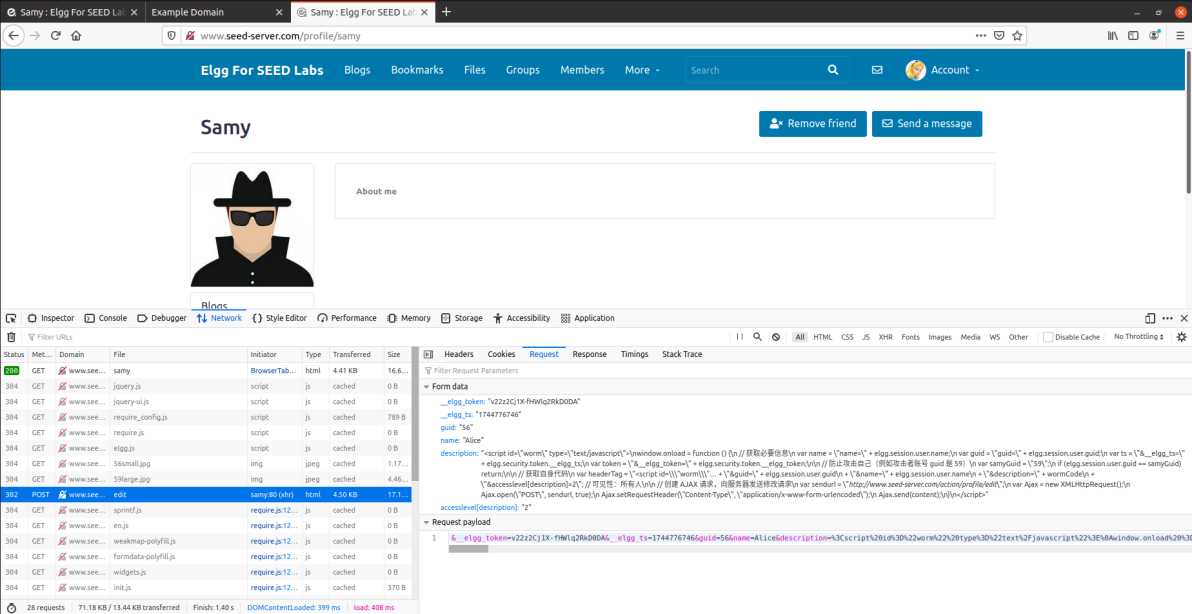


任务6：编写自我传播的XSS蠕虫

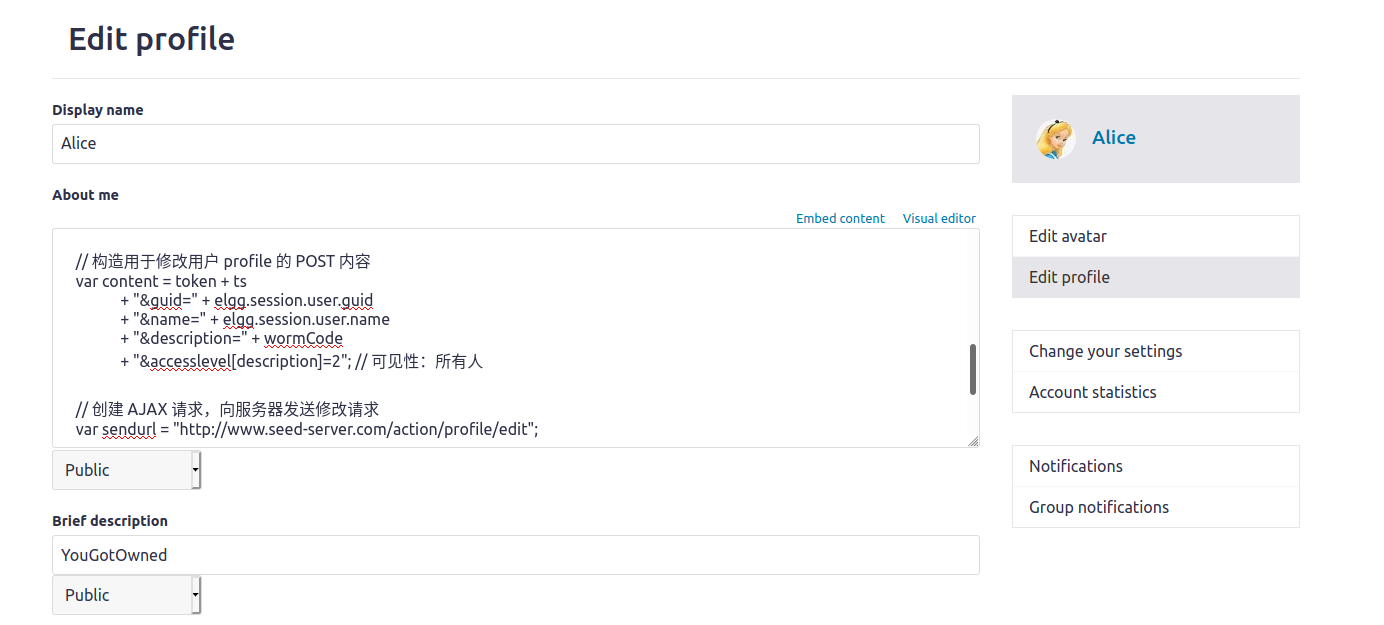
任务6需要在任务5的基础上，使用DOM方法实现XSS蠕虫的自我复制，自我传播，我们对代码进行一定的修改如下：



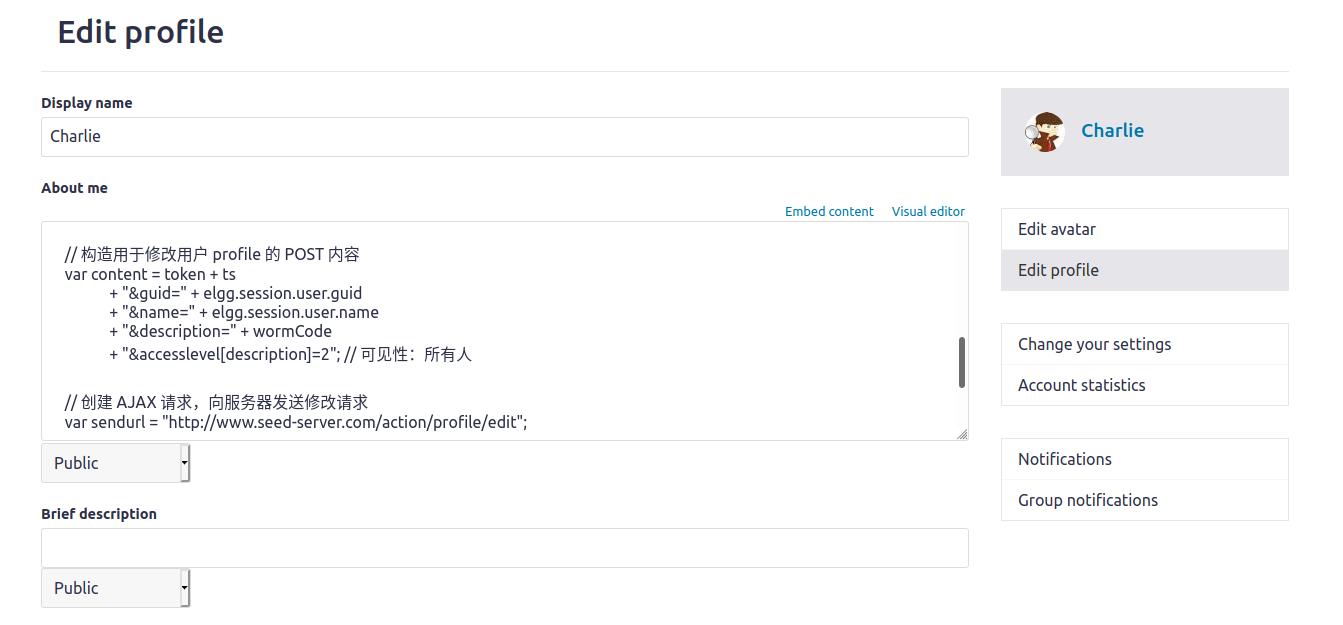
将其以文本形式嵌入Samy的“About Me”中，我们尝试用Alice账号访问Samy的Profile，发现自动发出了一个用于编辑自己Profile的POST请求，payload中description部分嵌入了完整的蠕虫程序



再打开Alice自己的Profile，发现自己的Profile已经被修改并嵌入了蠕虫程序

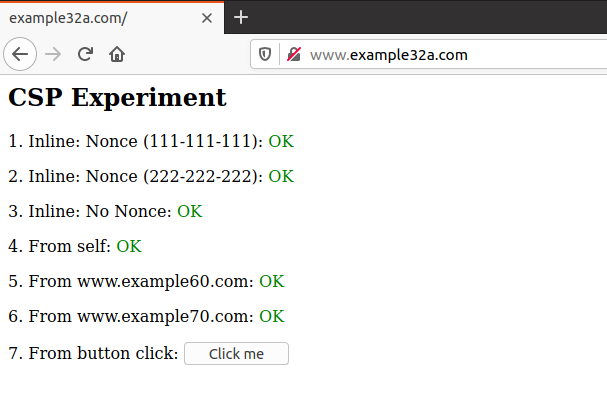


我们尝试用Charlie的账号再来访问Alice的Profile，发现Charlie的Profile也被修改并嵌入了蠕虫程序

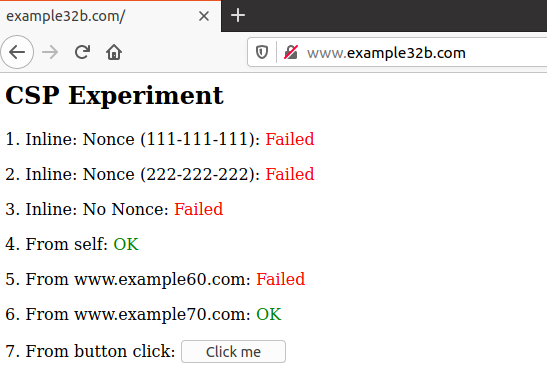


任务7：使用CSP防御XSS攻击

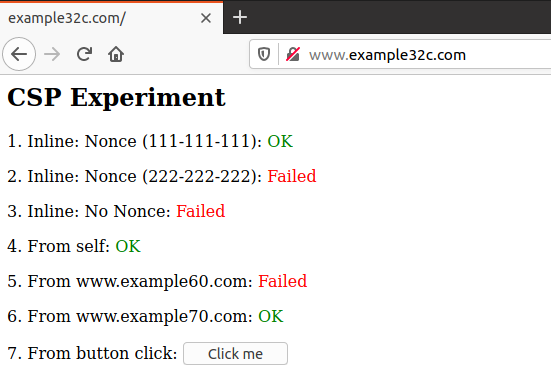
1. 访问三个网站



全部OK，没有限制，最容易受攻击

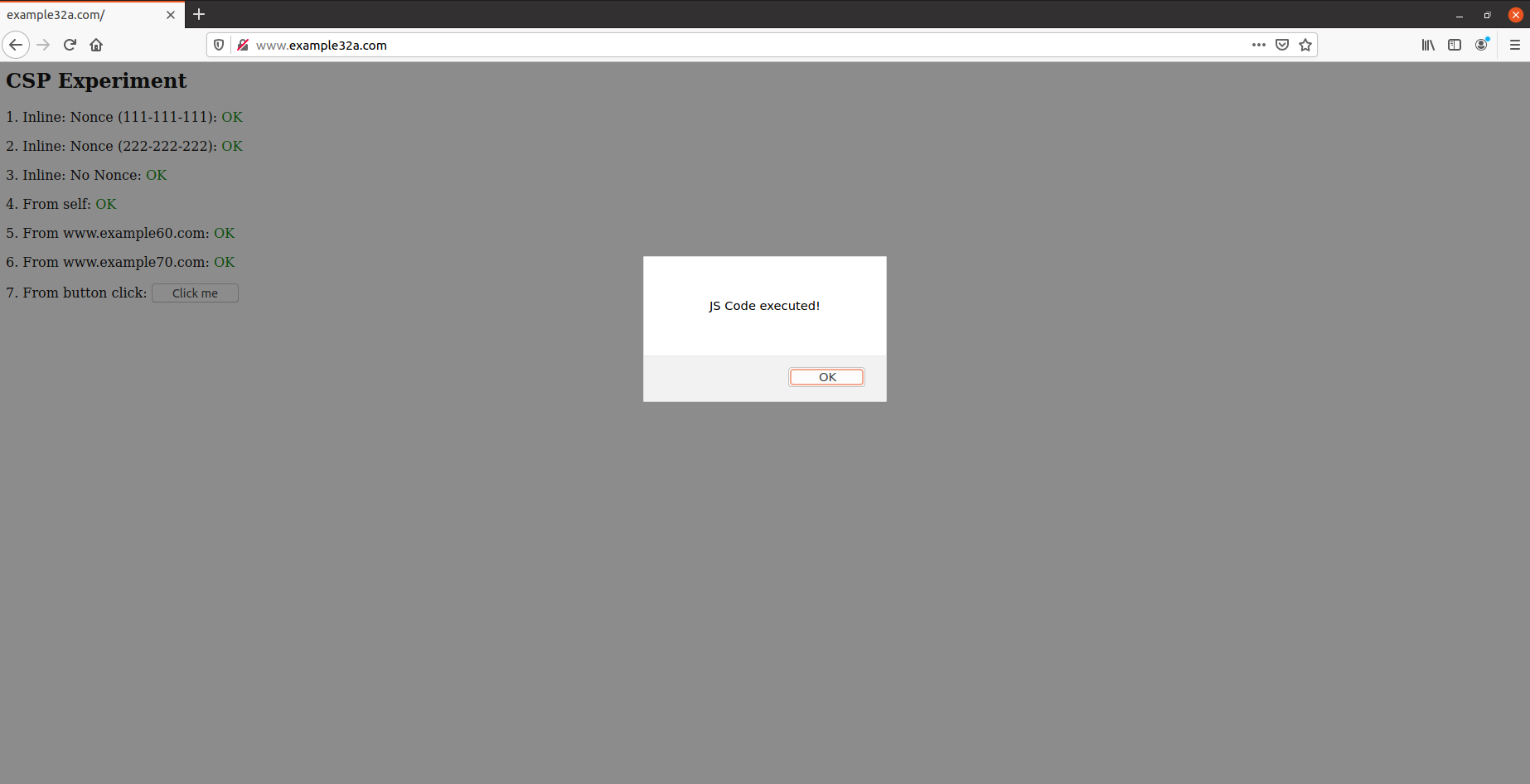


仅area4，area6 OK，严格限制inline和来源域名



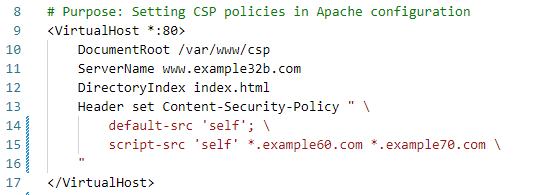
area1，area4，area6 OK，允许部分inline，控制更灵活

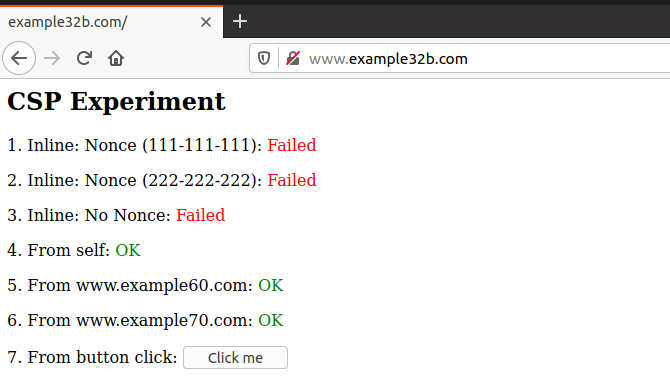
1. 点击来自这三个网站的网页上的按钮，描述并解释观察结果



仅example32a网站上的按钮click之后会弹出如上窗口，剩下两个没反应

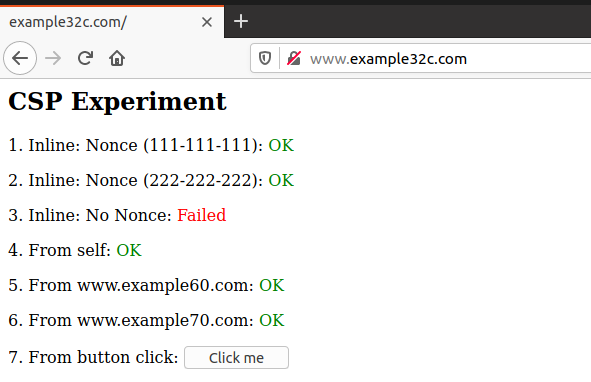
1. 修改example32b的配置，使得area5和area6显示OK





1. 修改example32c的配置，使得区域12456都显示为OK





1. 解释为什么 CSP 可以防止跨站脚本攻击

CSP 本质上将“代码”和“数据”分离，并赋予浏览器对脚本执行的主动管控能力。它显著提高了 Web 应用的安全性，是现代浏览器防御 XSS 的重要机制。通过合理设置 CSP，开发者可以有效遏制绝大多数 XSS 攻击路径，即便网站存在漏洞，攻击者的代码也难以真正执行

# 二、遇到问题及解决方法

# 三、对本次实验的建议