**操作系统实验五实验报告**

**基本信息：**

完成人姓名：顾琰 学号：57119117 完成日期：2021 年 7 月 24 日

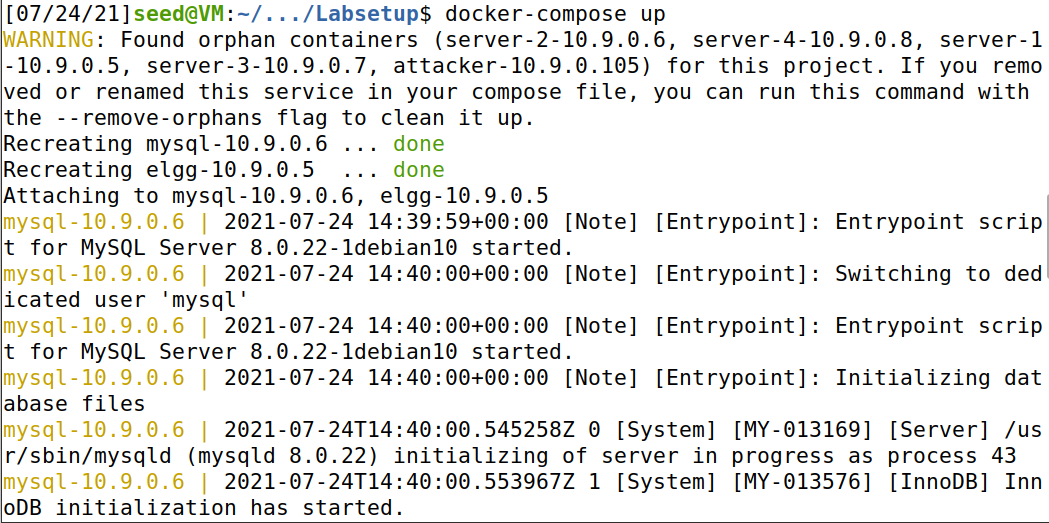
**实验内容：**

**环境配置：**

手动指定 DNS：



搭建实验环境(启动各个 Docker)：

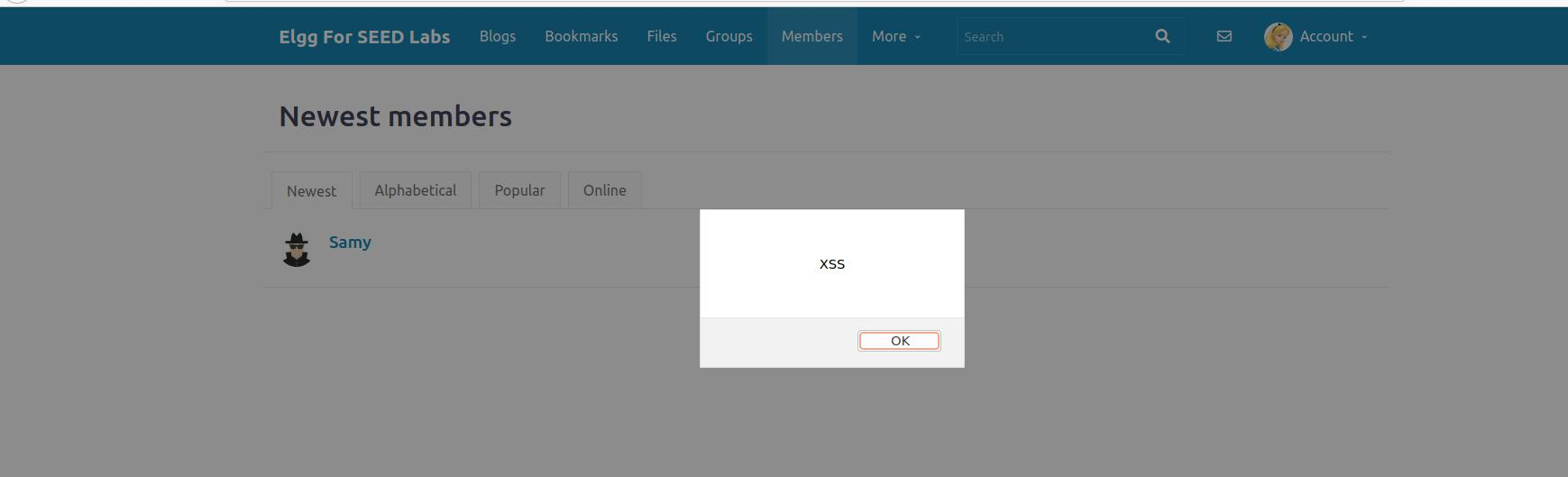


**Task 1: 发布恶意消息以显示警报窗口：**

访问我们搭建的 seed-server.com 并登录 samy 的账户输入恶意消息：

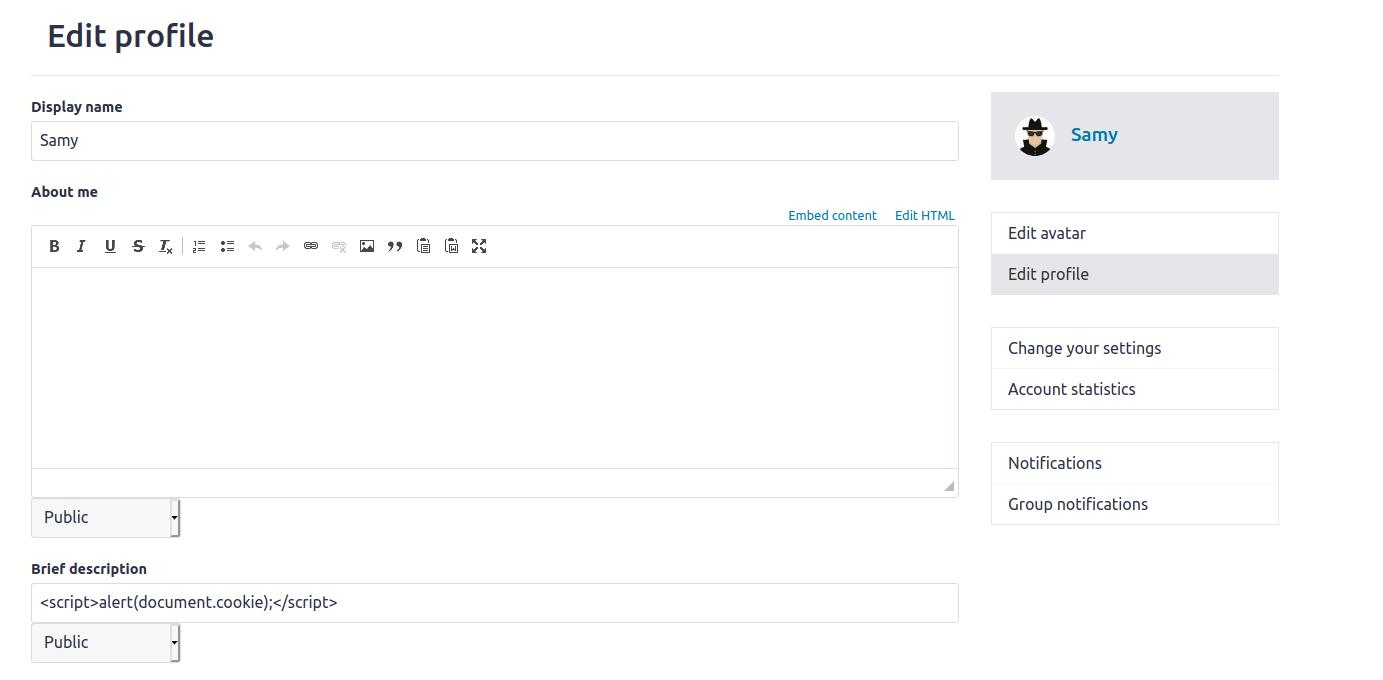


使用其他账户(Alice)访问 samy 的个人主页：

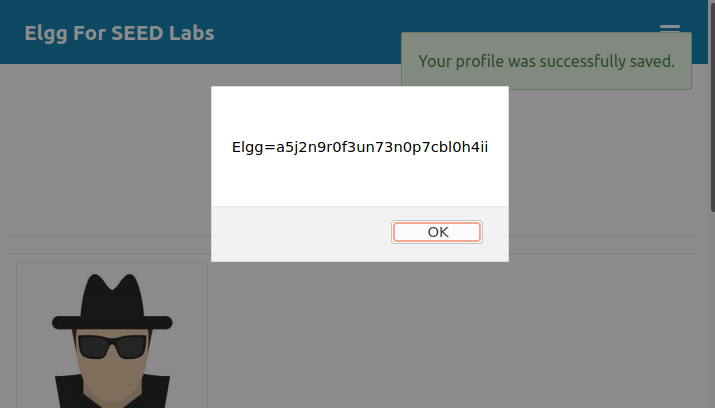


**Task 2: 发布恶意消息以显示用户 cookie：**

访问我们搭建的 seed-server.com 并登录 samy 的账户输入恶意消息：



使用其他账户(Alice)访问 samy 的个人主页：

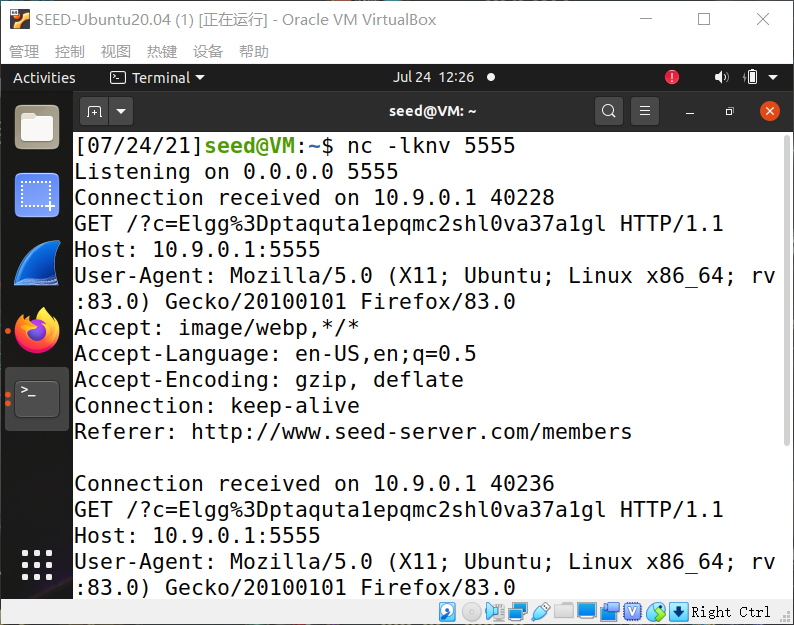


**Task 3: 从受害者的电脑中窃取 cookie：**

访问我们搭建的 seed-server.com 并登录 samy 的账户输入恶意消息：



使用其他账户访问 samy 的个人主页：



Samy 获得了 Alice 的 cookie攻击成功

**Task 4: 成为被害者的朋友：**

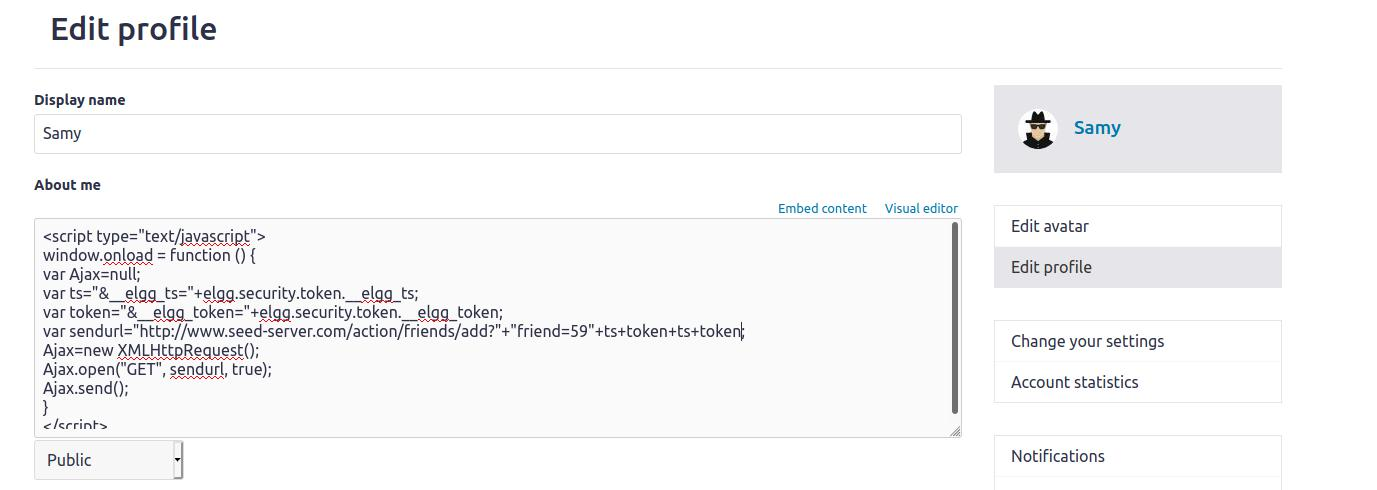
获取添加 samy 为好友的 get 请求：



补充完整攻击代码，并添加到 samy 的 aboutme 个人页面中(注意改变输入模式为 Edit

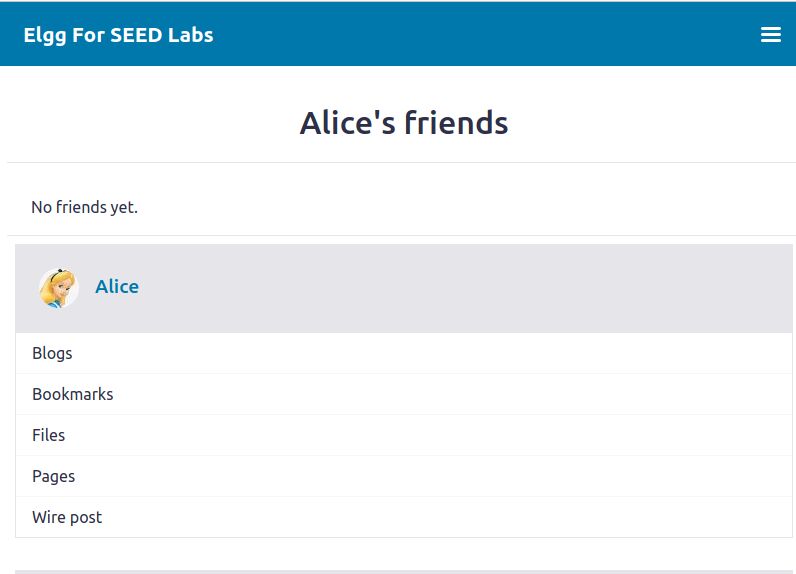
HTML)：



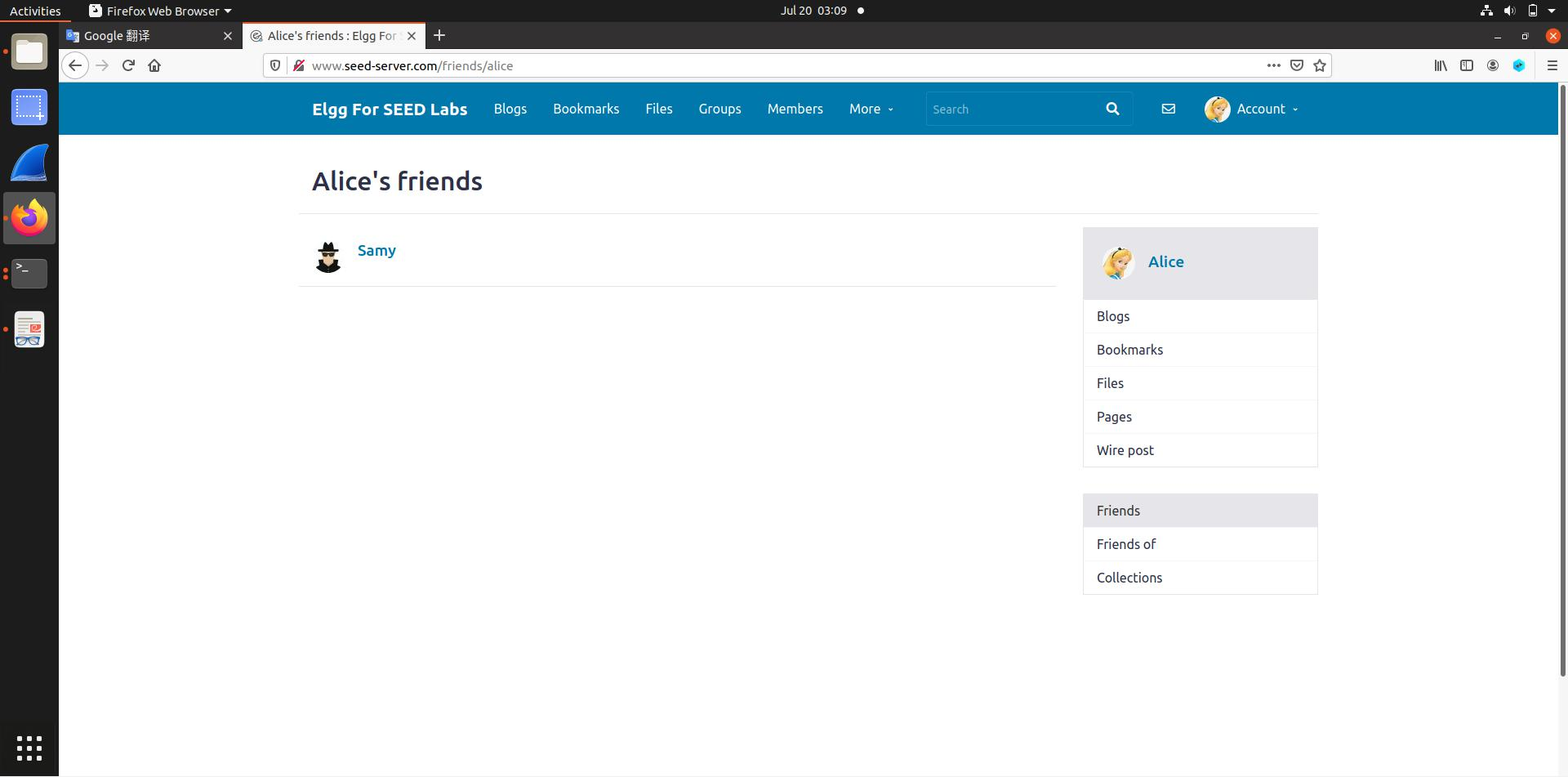


使用其他账户(Alice)访问 samy 的个人主页：

Alice 没有好友：



访问 samy 主页后：



Alice 只是访问了 samy 的个人主页便自动添加了 samy 为好友,攻击成功。

Question 1：代码中①②行的作用：

获取被攻击者的浏览器\_\_elgg\_ts 和\_\_elgg\_token 值，用于取得服务器验证。

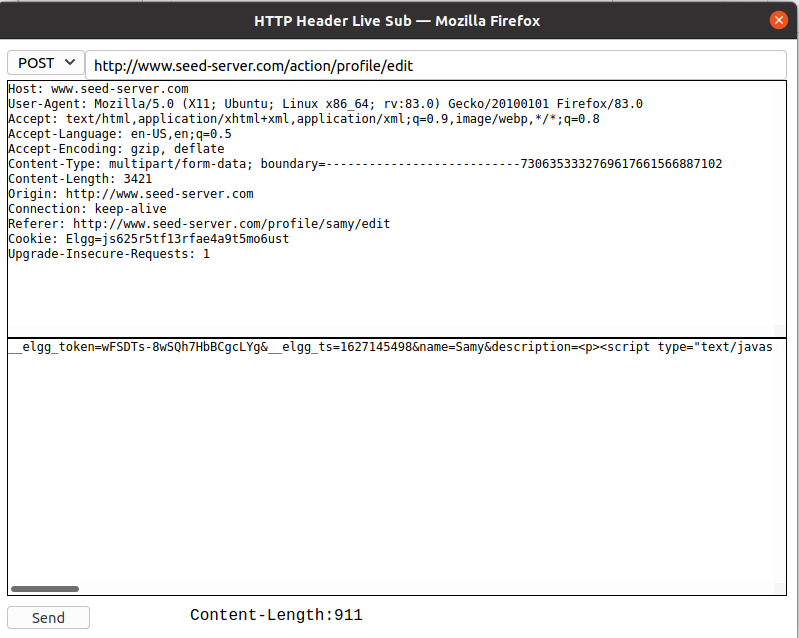
Question 2：如果 elgg 网站的 aboutme 功能只提供 Editor mode，你仍能成功发起攻击吗？

这时，需要通过查看页面源码的形式知晓网站对我们的输入做了怎样的处理，然后根据

网站的处理我们对我们的输入恶意代码进行调整。

**Task 5: 修改被攻击者的个人资料：**

获取用户修改个人资料的 post 报文：



构造 JavaScript 代码(同上个任务一样，需要在 Edit HTML 模式下输入)：



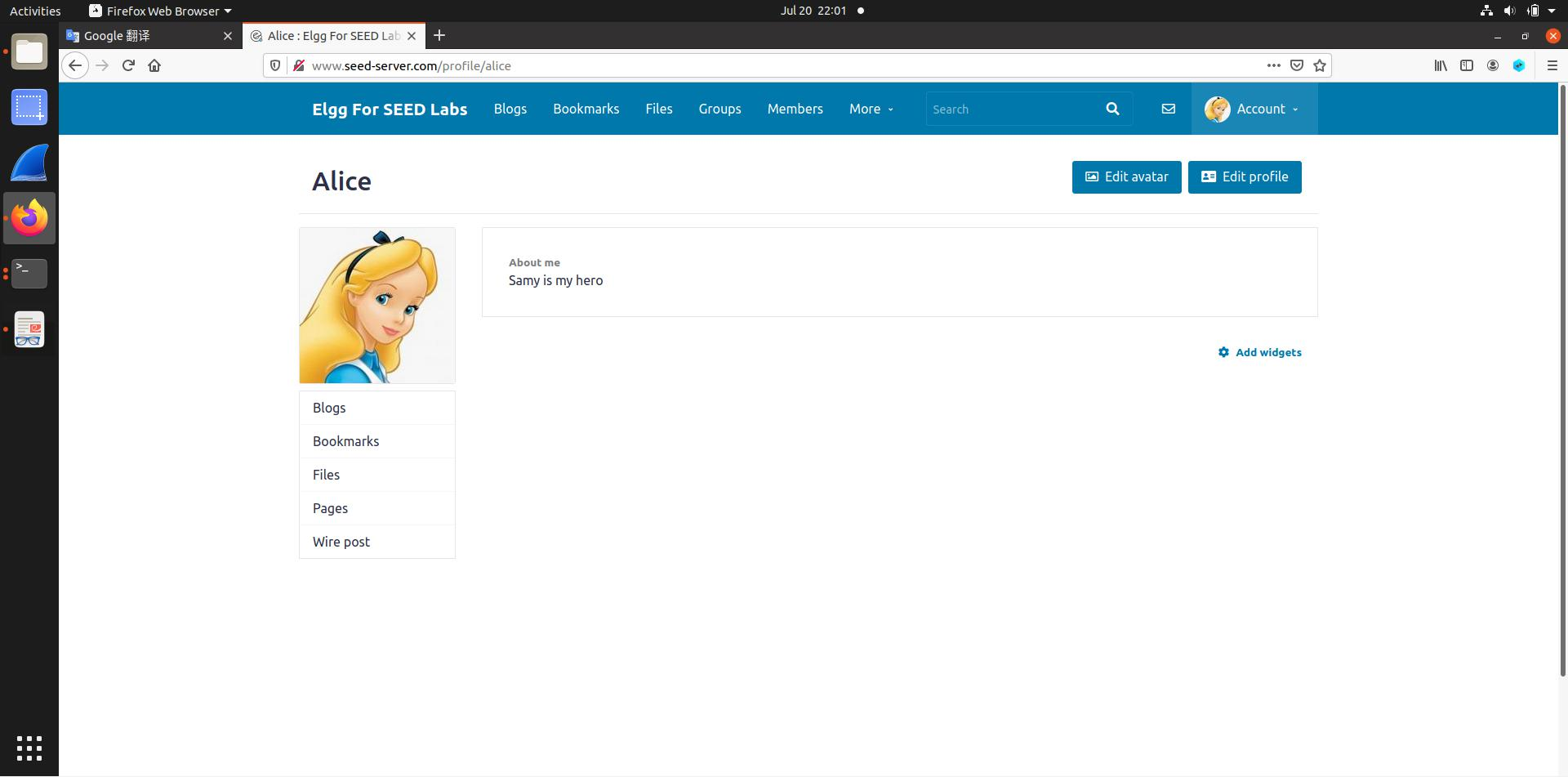
使用其他账户(Alice 和 Boby)访问 samy 的个人主页：

Alice:

访问前：

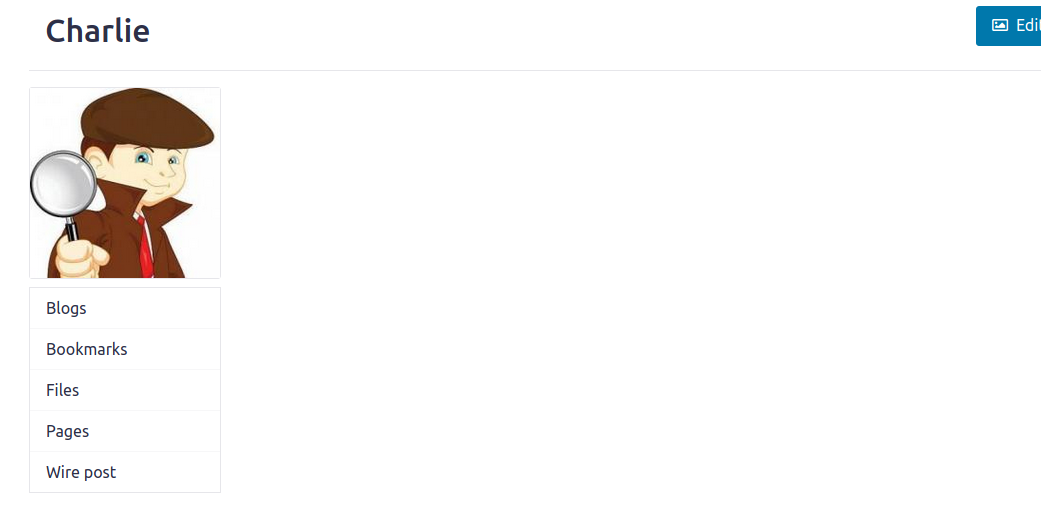


访问后：

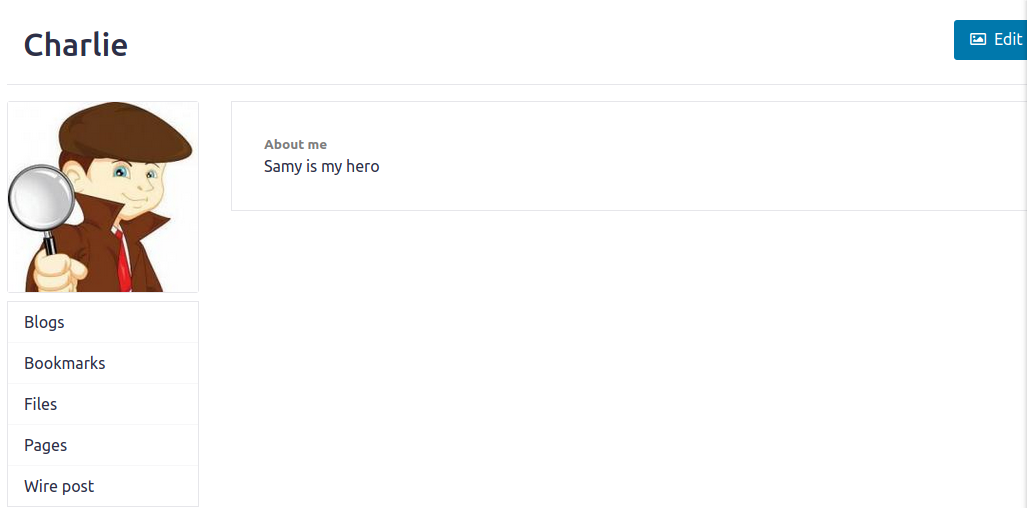


Charlie：

访问前：



访问后：



Alice 和 Charlie 都只是访问了 samy 的个人主页便自动修改了他们的 aboutme 为 “Samy is my hero”，攻击成功。

Question 3： 代码中①的作用？

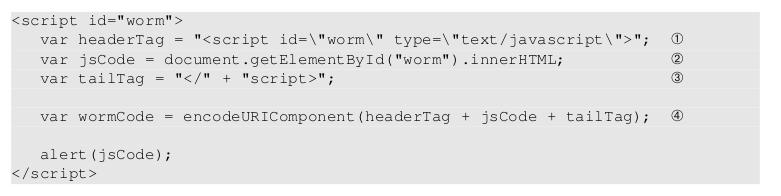
防止攻击到自己，当访问用户的 guid 是我们攻击者的 guid 时不进行相关攻击。

**TASK 6: 写一个自我复制的蠕虫 XSS 攻击：**

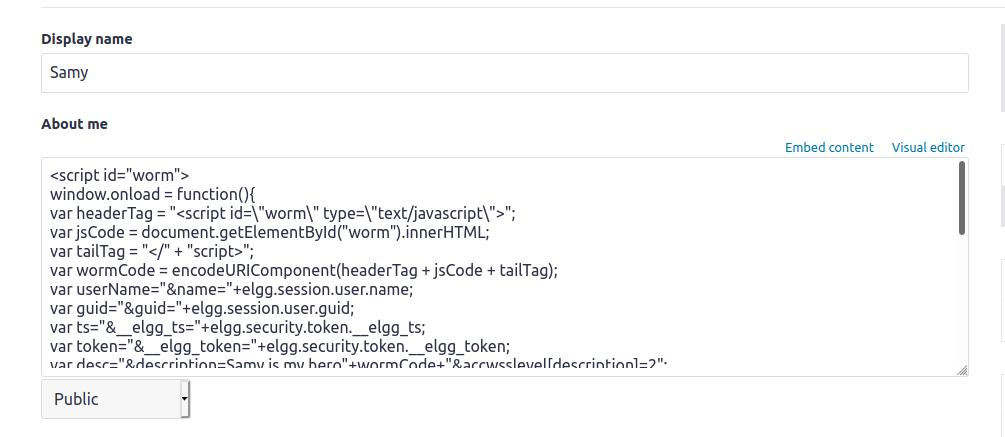
两种攻击方法：①连接法②DOM 法

DOM 法：

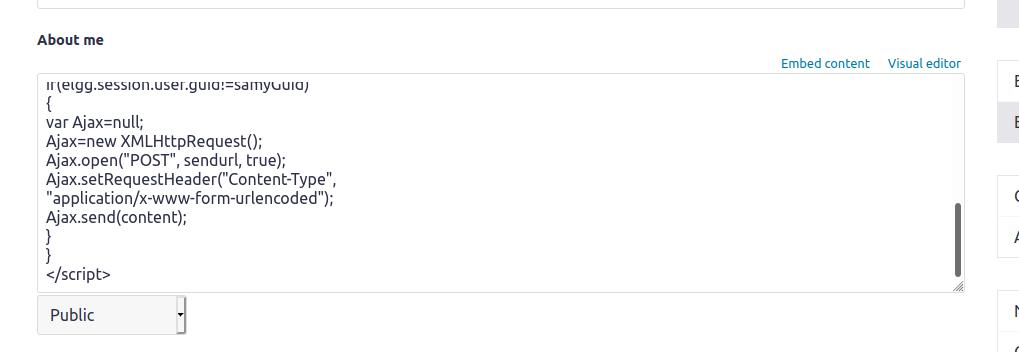
官方提供的复制代码：



将其加入我们在 task5 中写的代码：

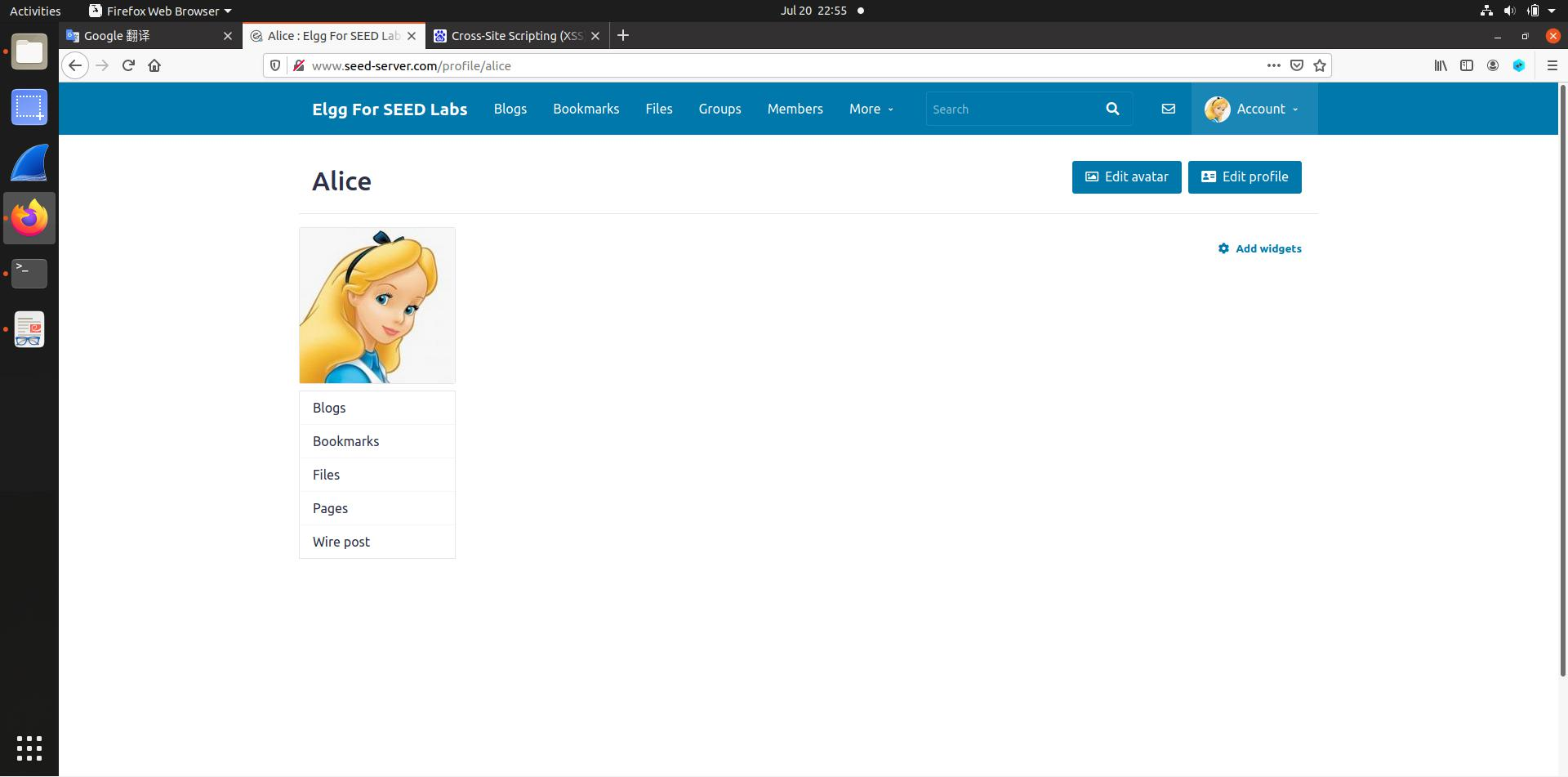


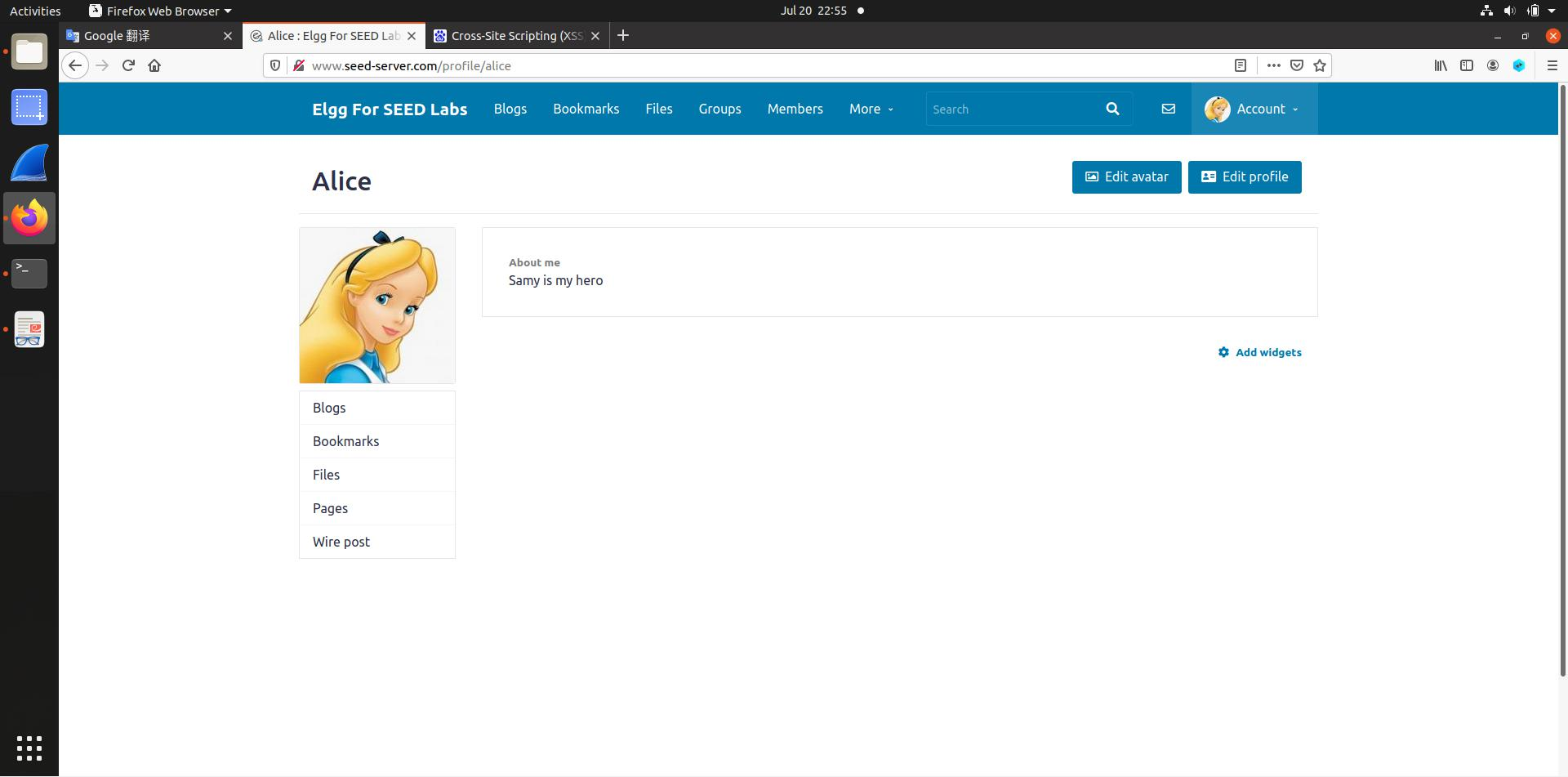




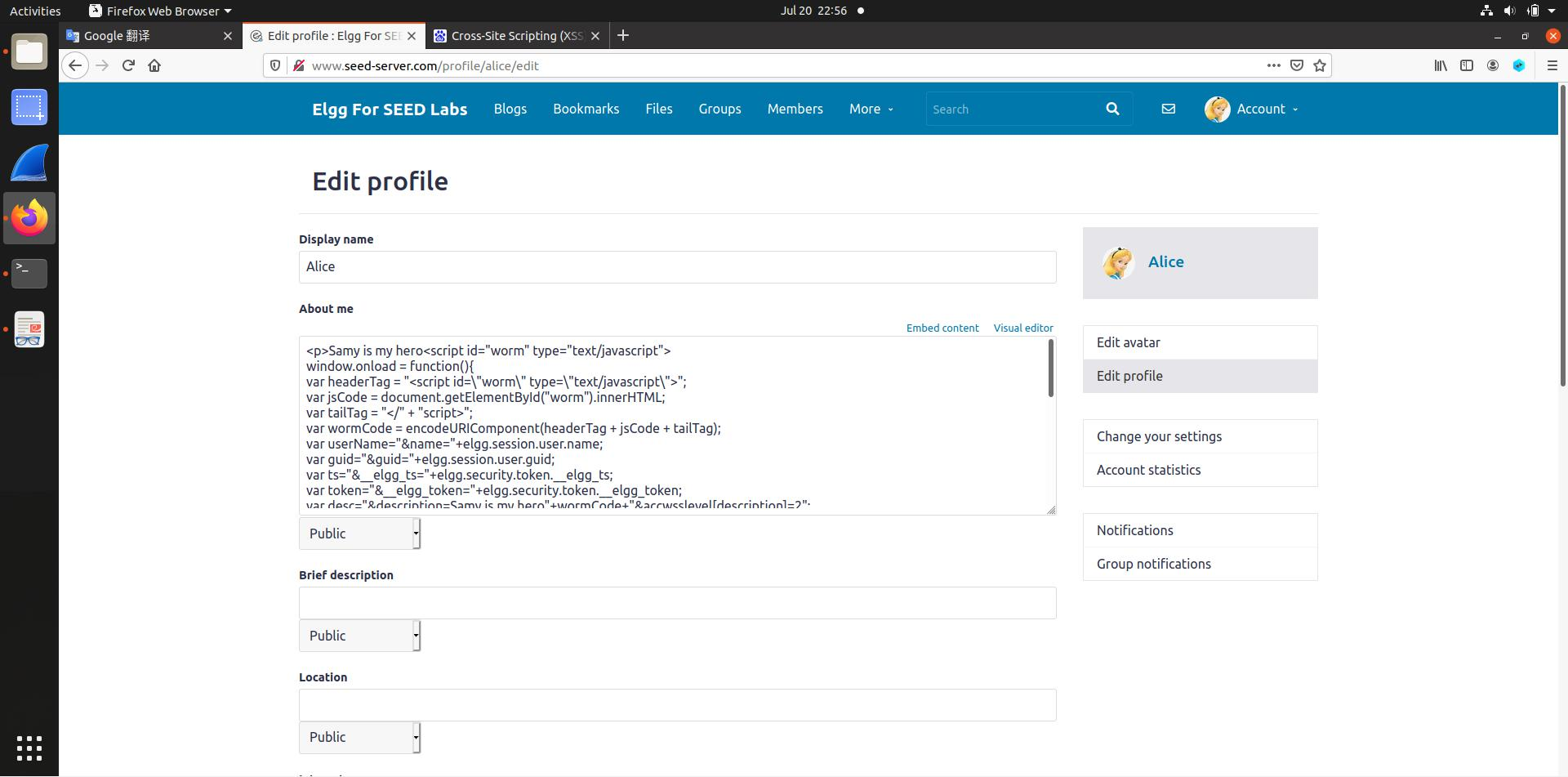
测试攻击：

Alice 访问 Samy:

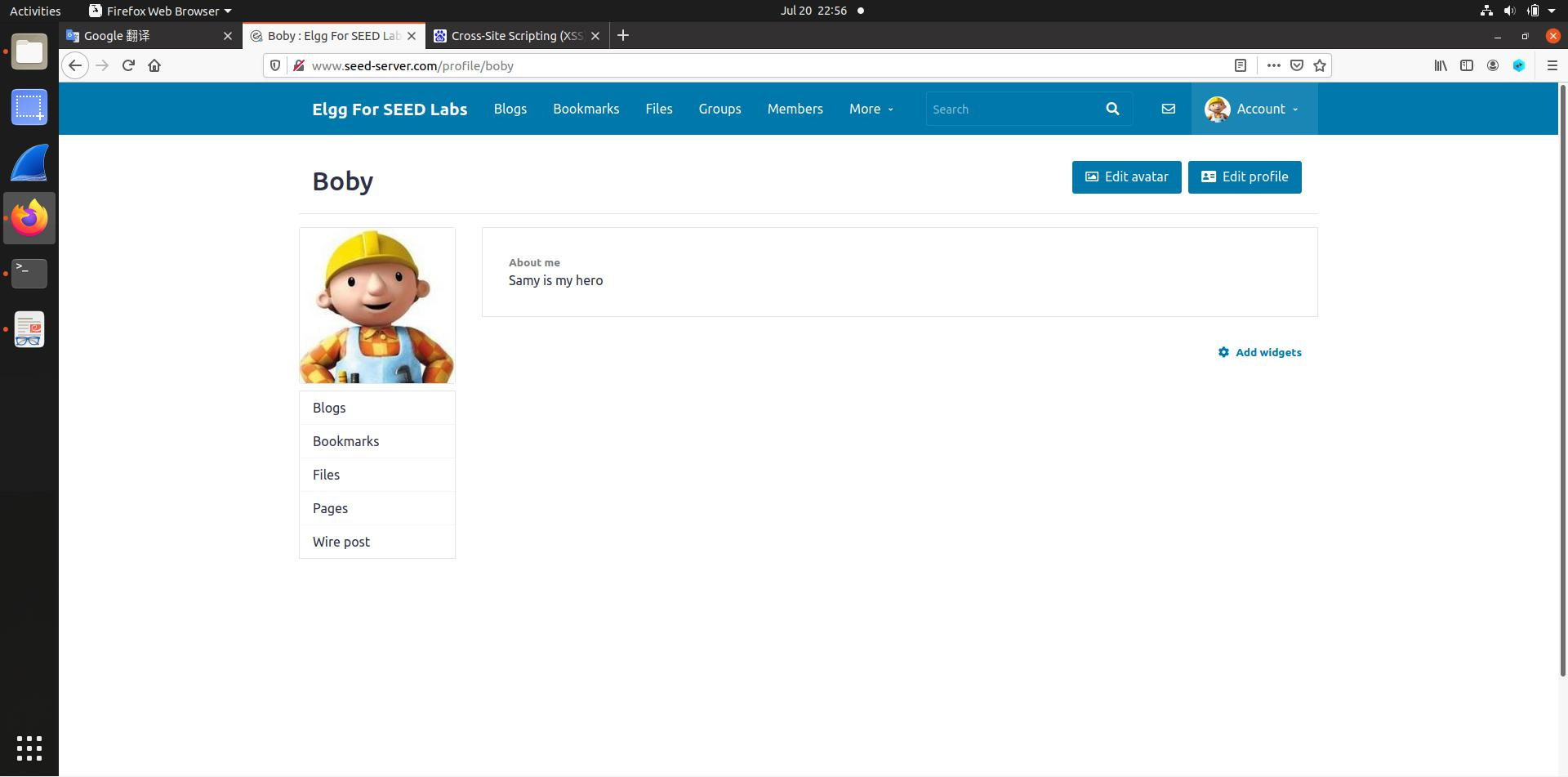




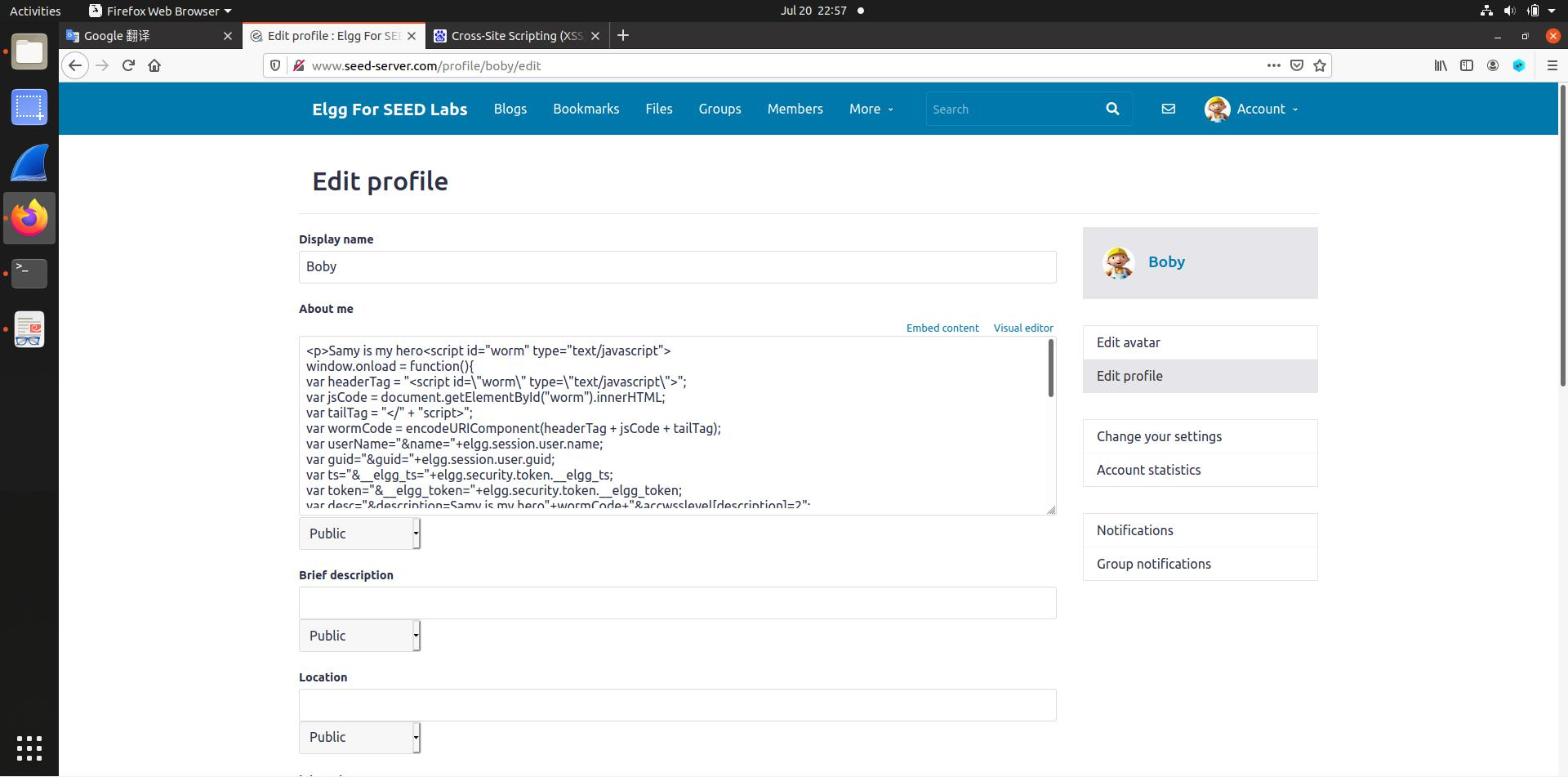
查看 Alice 的信息，其已经复制了 Samy 的蠕虫病毒：



Boby 访问 Alice：



可以查看 Boby 的信息，其已经复制了 Alice(Samy)的蠕虫病毒：

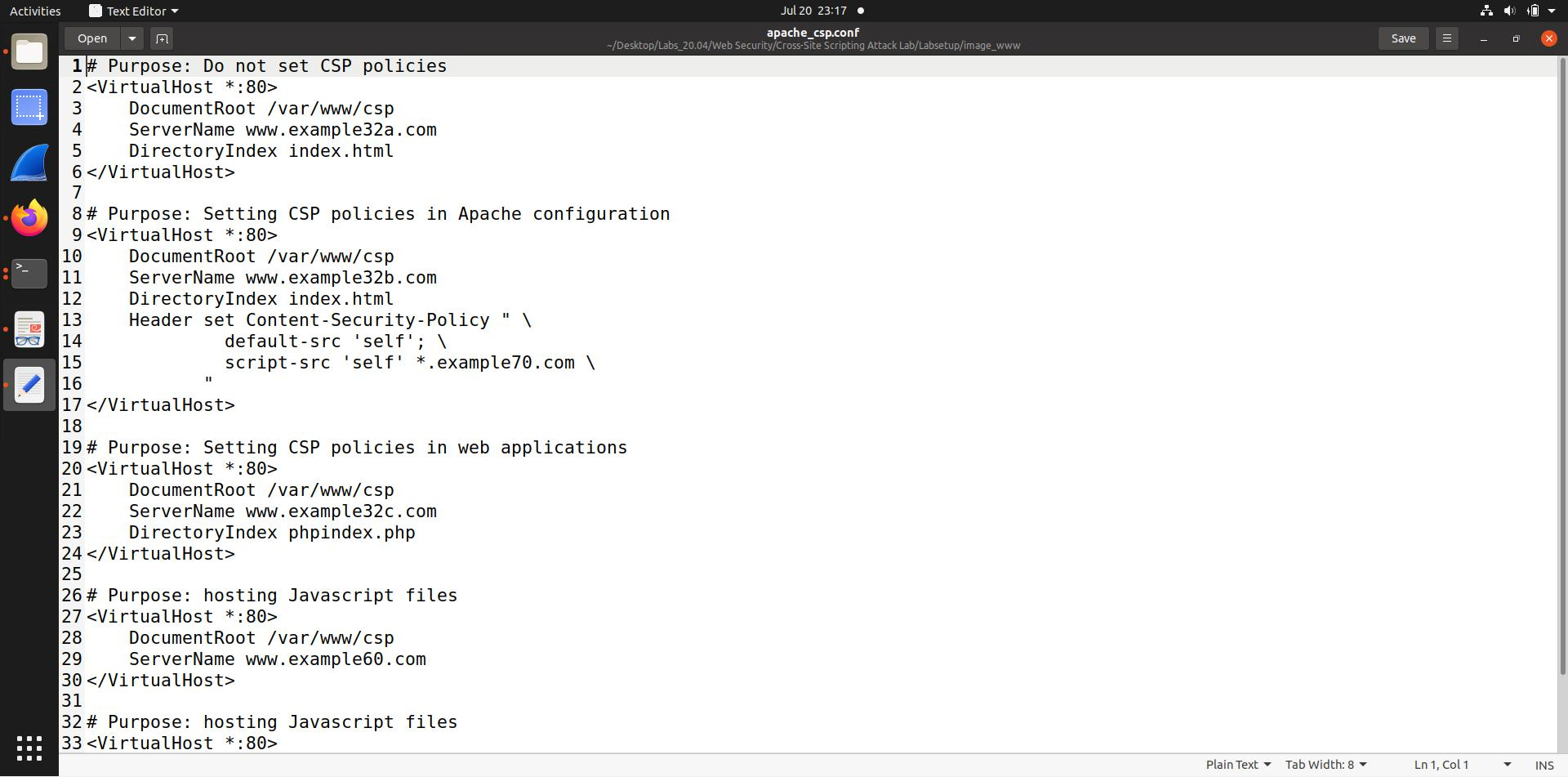


实现了 XSS 攻击的蠕虫病毒化，可在用户之间广泛传播。

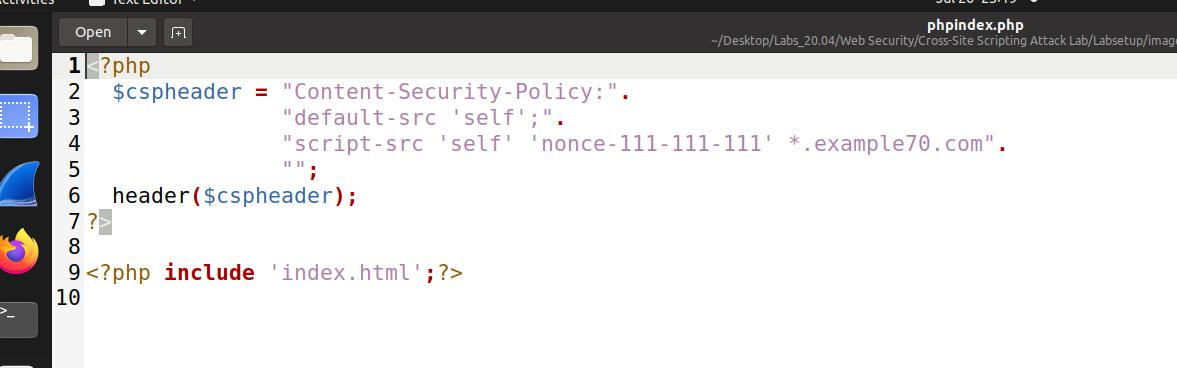
**TASK 7: 使用 CSP 对 XSS 攻击进行防御：**

两种设置 CSP 的方法：

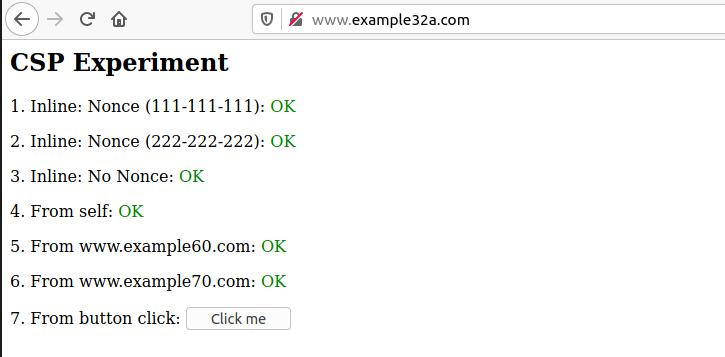
①通过 Apache 修改 CSP config 文件：

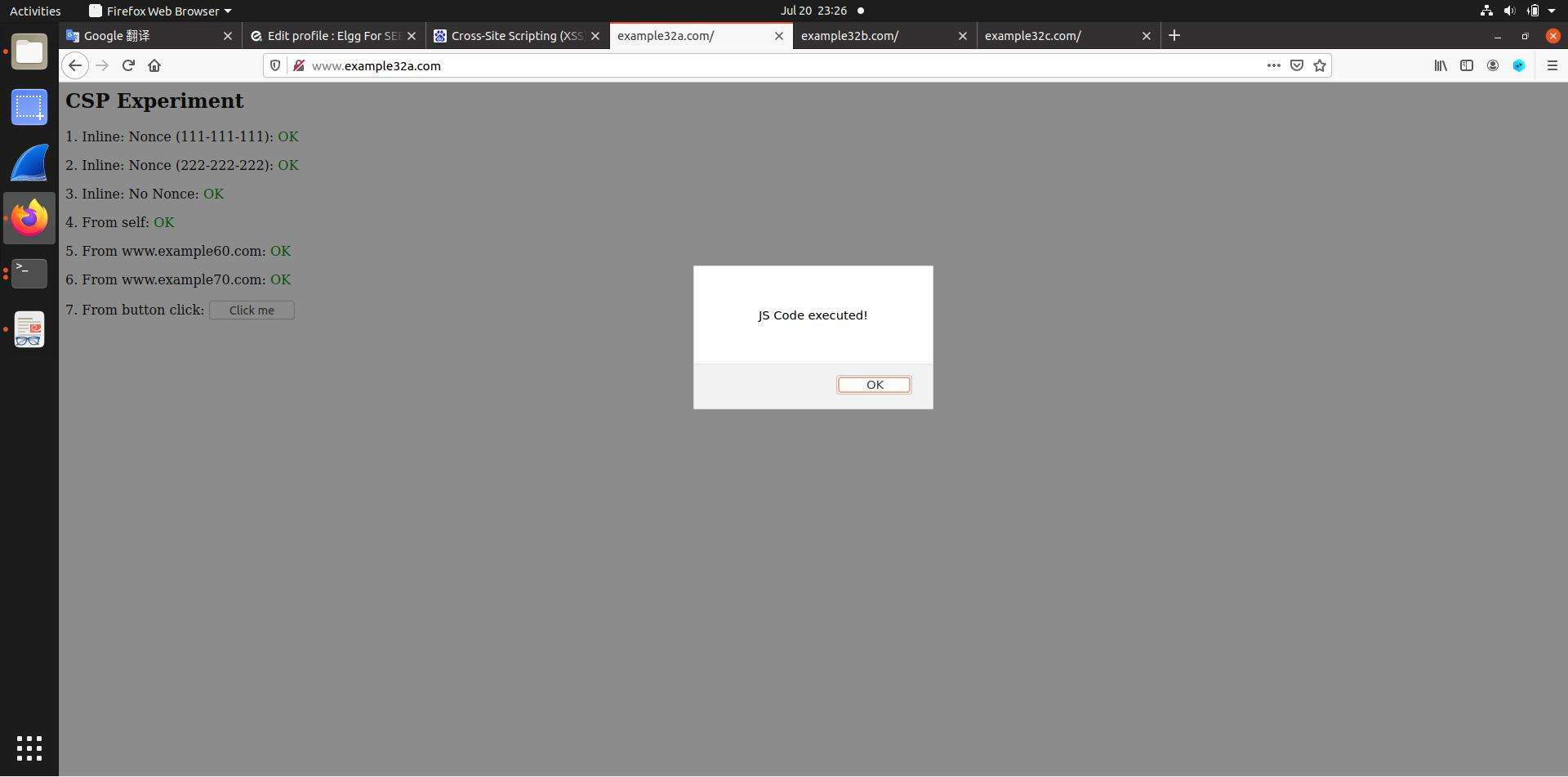


②通过网页应用修改 CSP config 文件

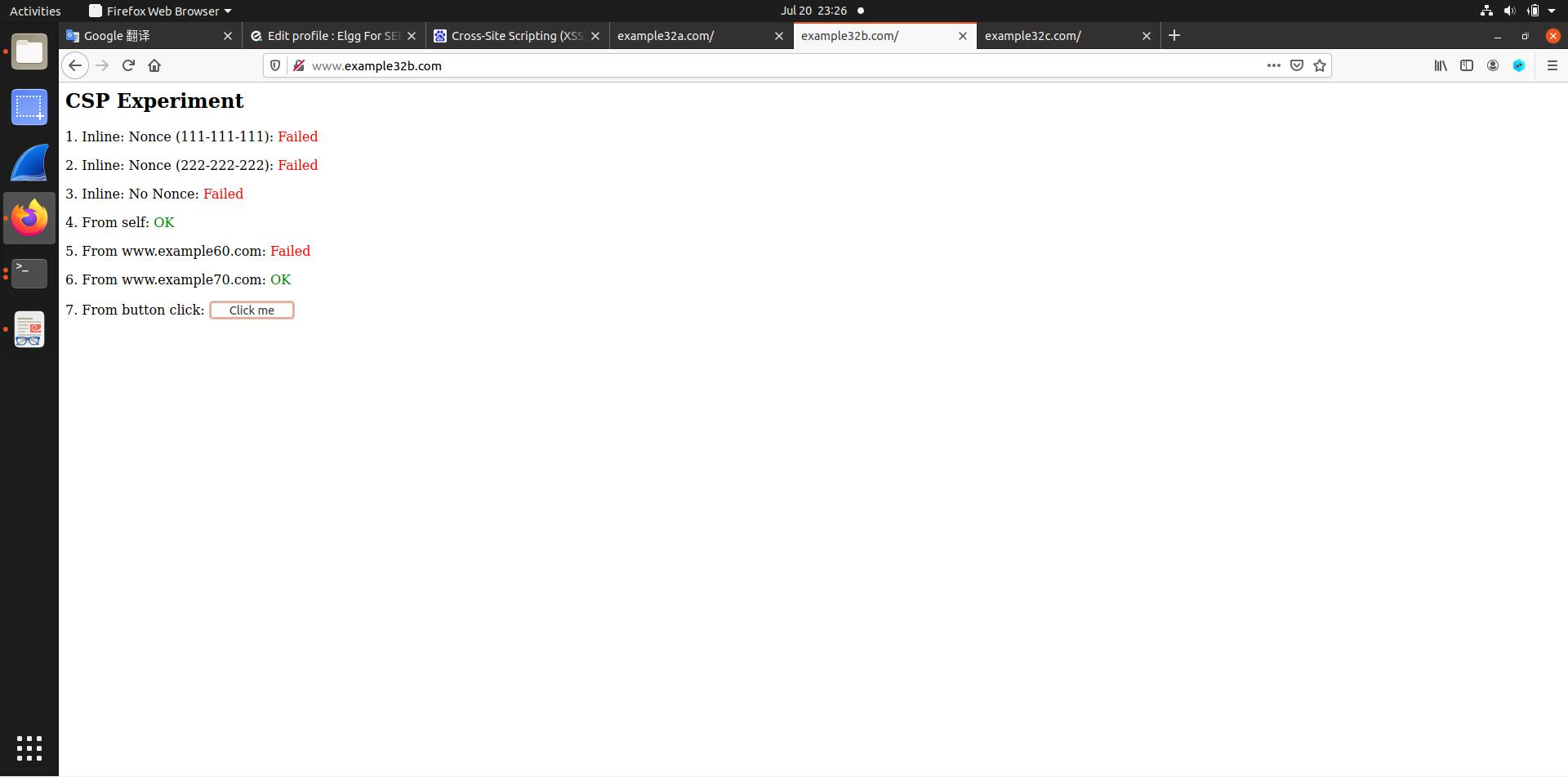


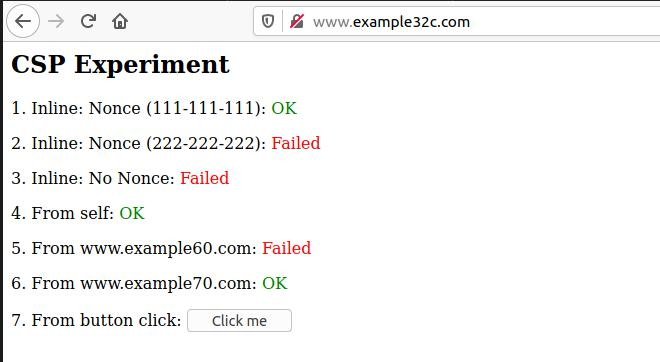
测试：

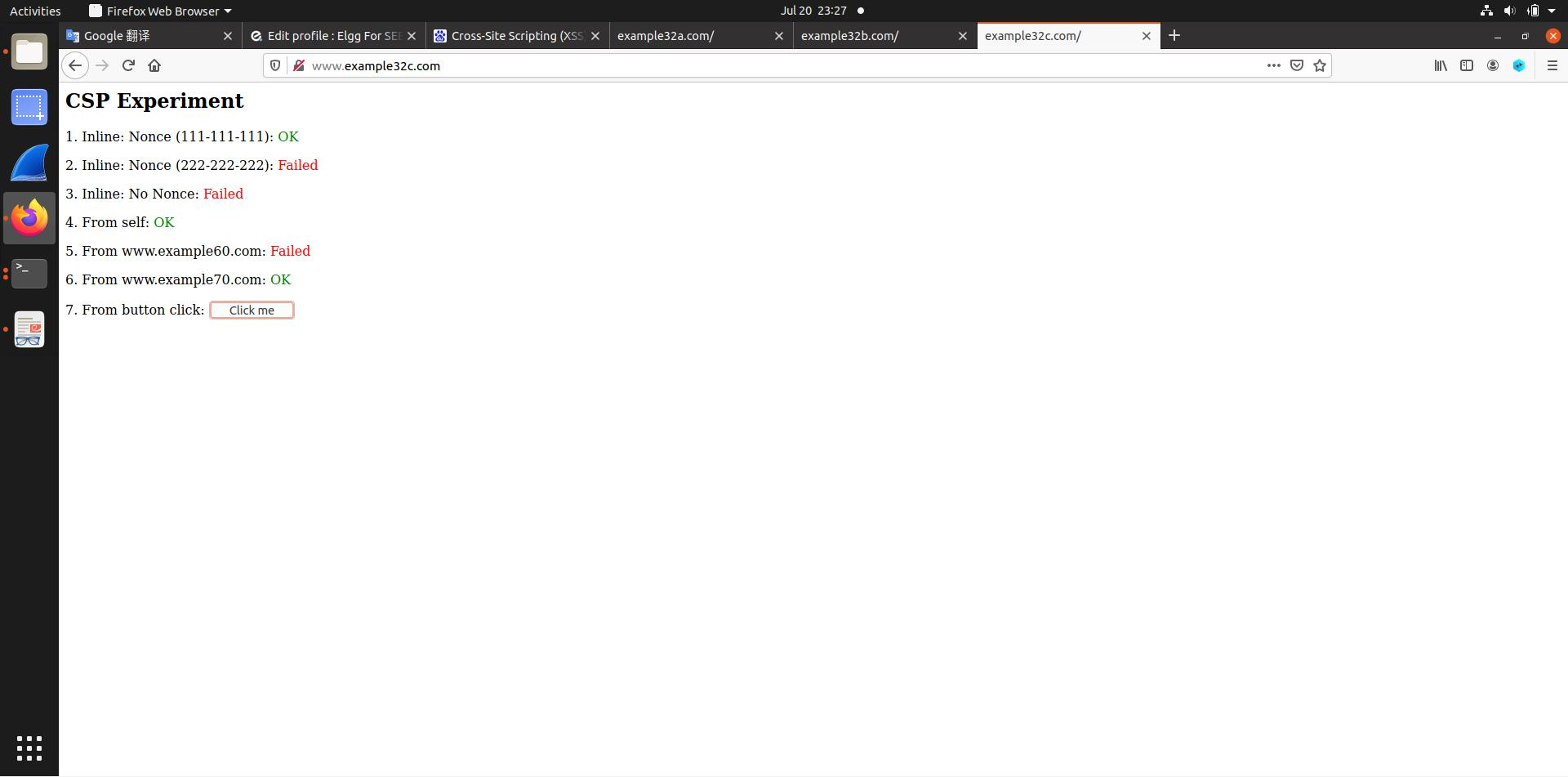












上述三个对比实验得出结论：

1、a 网站为启用 CSP 防御，所以所有 javascript 执行成功。

2、b 网站通过 Apache 方式设置 CSP 防御，只有受信的 4、6 两个源执行成功。

3、c 网站通过 php 方式设置 CSP 防御，只有受信的 1、4、6 三个源执行成功。

且点击按钮后也只有 a 网站执行成功，b、c 网站均不执行按钮操作，故 CSP 防御有效。 修改代码使相关源的 JavaScript 执行成功，只需要在 CSP 的设置中将其源设置为可信源

即可。

CSP 防御的有效性在于将网页中的执行代码和数据分开了，只执行来自可信源的代码，

从而使攻击者的代码不可执行。

**实验体会：**

本次实验是我们的第五次实验，经过本次实验，我总结了如下的知识点:

1、XSS的根源在于Javascript代码可以自然地与HTML数据混合。在XSS攻击中，一旦攻击者的代码进入目标用户的浏览器，代码可以代表此用户发送伪造请求，如删除用户的好友及篡改用户的个人资料。除此之外，恶意代码还能在目标用户的账号中存储一个自己的副本，感染用户数据。当其他用户访问被感染的数据时，恶意代码就能进一步感染其他用户，实际上就成了一个自我传播的蠕虫。

2、最好的防御方法是把JavaScript代码和数据分离，并用CSP规则来禁止执行来自不可信源的JavaScript代码。