**XSS总结**

1. XSS基本概念
2. **XSS的含义**

XSS(Cross-Site Scripting,跨站脚本)，是一种利用网页漏洞，将恶意代码注入网页中的攻击。当用户浏览网页时，这些代码得到执行，从而达成对用户的攻击。通过XSS，攻击者可以对用户进行Cookie窃取、会话劫持、钓鱼攻击等。

1. **XSS的分类**

XSS攻击根据攻击性质，一般分为以下三类：

* 反射型XSS

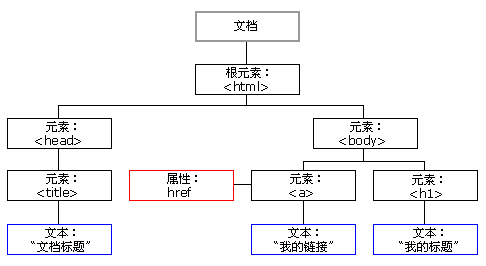
也称作非持久型XSS。反射型XSS的特点是攻击的语句表现在URL中，通过浏览器解析后形成攻击。这类攻击需要用户主动浏览带有XSS的URL才能实现，而且仅在用户浏览时触发一次，对于有一定网络安全意识的用户而言效果较差、危害性较小，即便是安全意识比较薄弱的用户，部分浏览器自带的XSS防护（例如Chrome的XSS Auditor）也会给出警告信息，阻止用户访问。虽然也有能绕过检测、骗过用户的方法，但仍然有较大的局限性。

* 存储型XSS

也称作持久型XSS。存储型XSS的特点是恶意代码会被上传至服务端，通过检测并保存在服务器中，当用户浏览到包含恶意代码的网页时执行。和反射型XSS不同的是，被上传至服务器中的恶意代码不会体现在URL中，对一般用户来说，漏洞服务器的哪些页面可能含有恶意代码是难以预料的，在正常浏览过程中也可能遭到攻击，这种隐蔽性使存储型XSS比反射型具有更大的危害性。

* 基于DOM的XSS

DOM(Document Object Model,文档对象模型)定义了网页中所有元素的对象、属性以及访问他们的方法。



这是一张常见的DOM树模型。可以看到HTML DOM将节点分为文档节点、元素节点、文本节点、属性节点。利用Javascript，可以对网页的DOM树节点进行修改，例如为元素添加src参数或onerror事件等。使用这种方法进行的攻击被称为基于DOM的XSS攻击，这类攻击可以在前端完成，不依赖与服务器数据的交换。

1. XSS解题思路

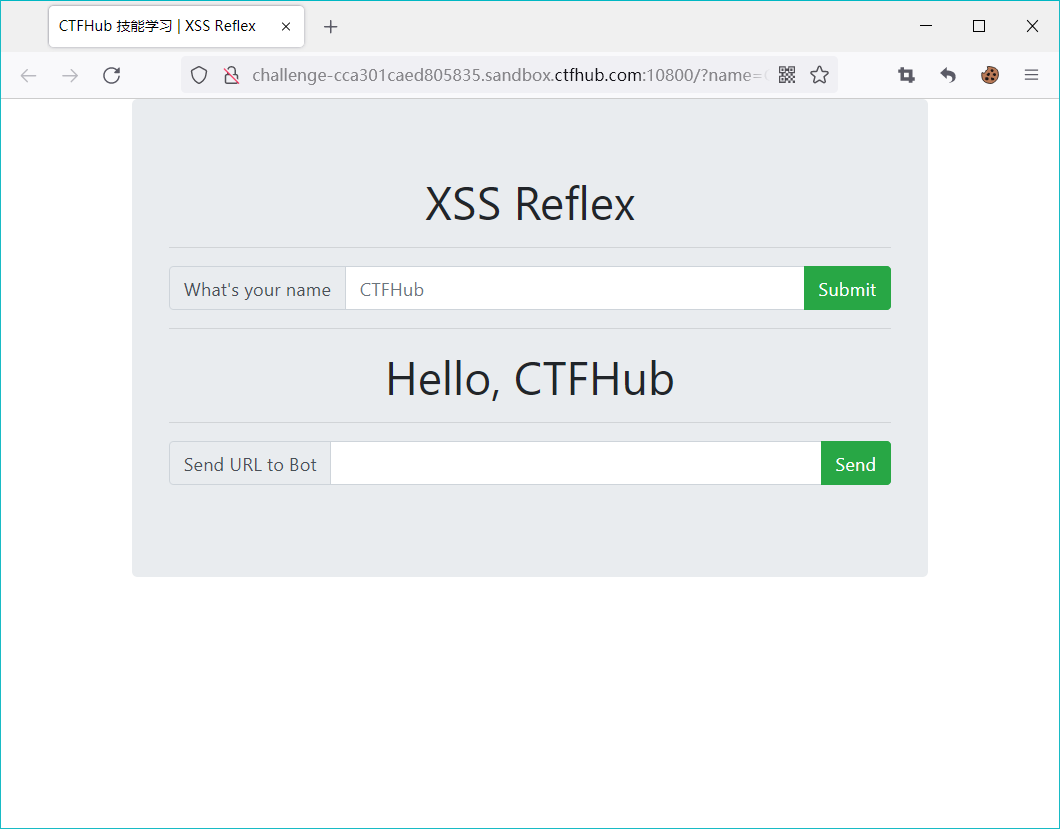
结合XSS攻击的特点，对于一般的XSS题目，可以大致以如下几个步骤进行：

1. 观察页面情况、结合URL表现，大致分析可能存在XSS漏洞的位置；
2. 尝试注入代码、结合页面源代码判断过滤情况；
3. 根据过滤情况，构造合适的payload绕过过滤。

**工具：**

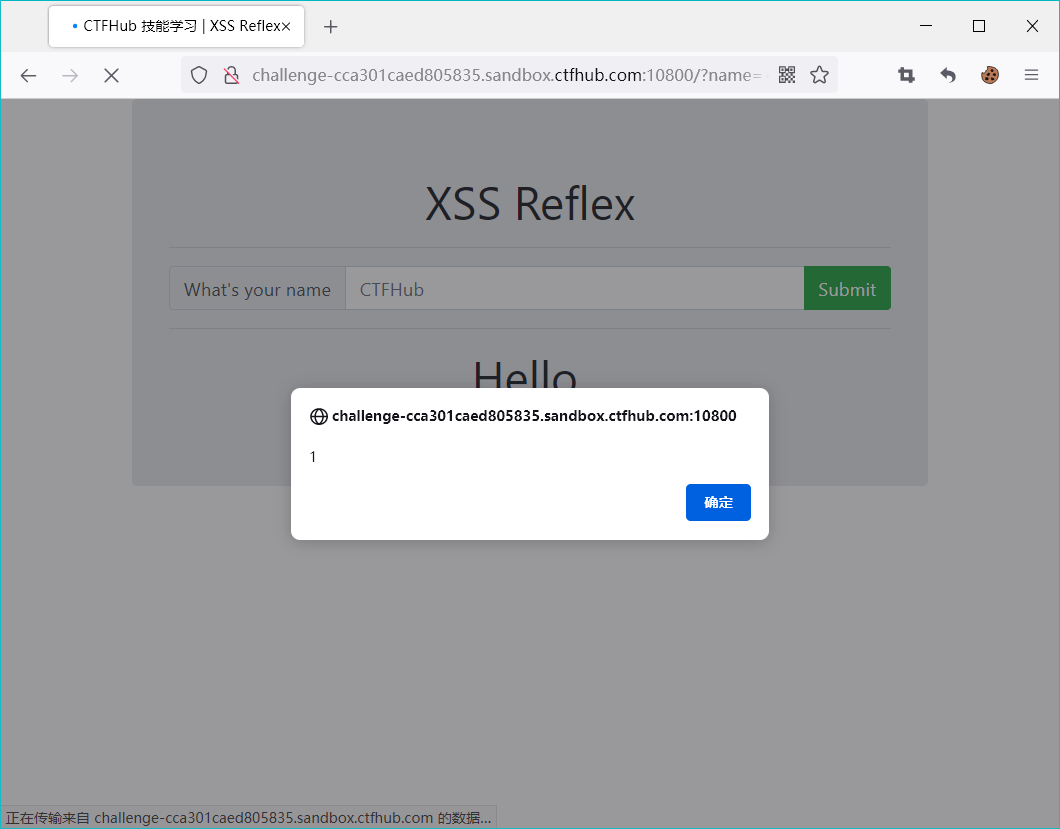
在XSS类型的解题中比较常用的工具是XSS平台，这一在线平台可以方便地模块化生成脚本、构造注入语句等。同时，利用XSS平台生成的脚本进行攻击，攻击的结果也会反馈到XSS平台上，便于查看。XSS平台有很多，也可自己搭建，使用方法大同小异。

**例题**：CTF Hub-XSS-反射型



靶机页面如图所示。

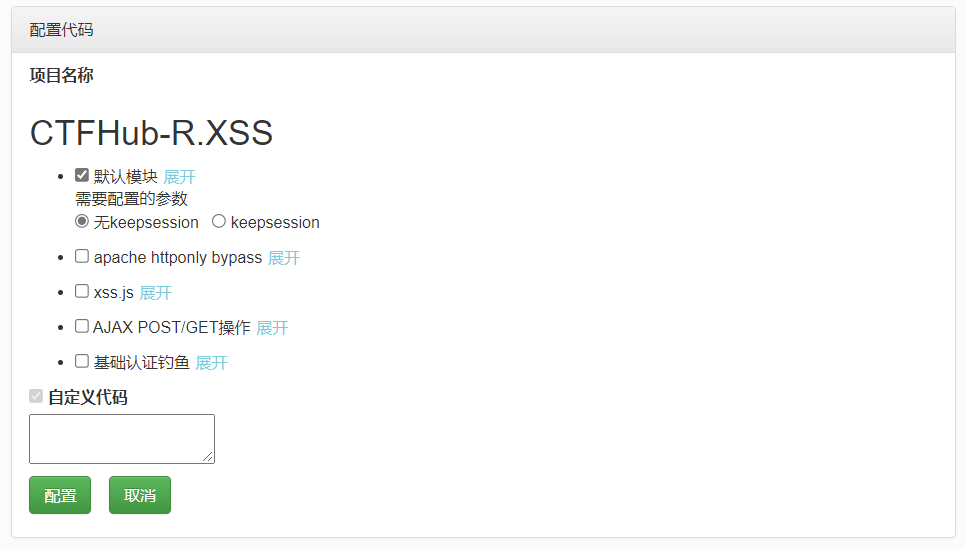
1. 观察页面：由两段文本、两个输入框构成。上方的输入框“What’s your name”，输入名字，下方的输入框“Send URL to Bot”，猜测是模拟反射型XSS中用户点击链接这一行为。中间的文本“Hello CTFHub”，可能与上方输入框中输入的名字有关。
2. 观察URL：...ctfhub.com:10800/?name=CTFHub，说明输入框中的name会被提交到URL中作为参数。
3. 尝试注入简单代码：<script>alert(1)</script>



出现弹窗，基本可以判断该网页不存在针对XSS的过滤。同时查看URL：

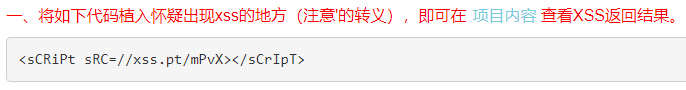
说明该输入框中提交的参数确实直接体现在URL中。

1. 打开XSS平台，创建项目：



模块可以自由搭配，不过本题由于构成简单，只要使用默认模块即可。

创建项目后，平台会给出使用方法：

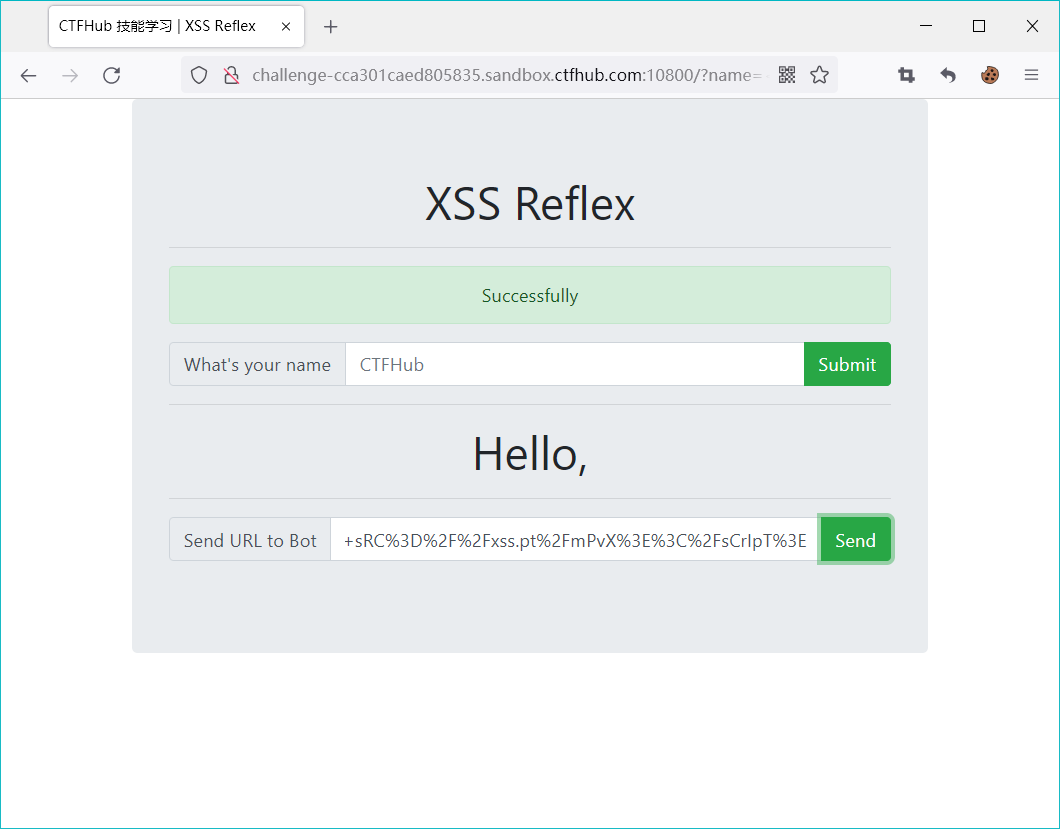


该页面中会给出若干形式不同的代码语句。根据具体过滤情况不同，可以选用不同的形式。

1. 复制代码并在第一个输入框中提交后，观察URL：



此时的URL已经附上了攻击语句。将该URL提交给bot:



网页提示Successfully。

1. 返回XSS平台，查看项目内容：



其中上方的一条为bot访问的结果，cookie栏中即为所求flag，下方一条为我们提交的时候刷新页面导致自己进行了一次访问，如果要避免这次访问，也可以不通过网页提交攻击语句，直接在URL的参数中进行构造。

1. Flag：ctfhub{4d517e1e4b1c1f57f5df0c89}

当然例题本身属于比较简单的题目，因为它没有针对XSS攻击进行任何防护措施，是用来熟悉反射型XSS漏洞的形式以及XSS平台工具的使用的。

1. XSS常用过滤绕过方法

XSS靶场常见的形式为“闯关”，靶场中有一系列“关卡”，过关的方法是为网页注入脚本，并使网页弹窗。

常见的XSS闯关靶场：

xss-quiz.int21h.jp

xss.test.tv

xss.haozi.me

等等。

利用这些闯关靶场可以接触大量的XSS过滤方法。

1. **闭合**

**例题1**：xss-quiz.int21h.jp/stage2.php



要求弹窗显示document.domain

直接尝试script脚本：



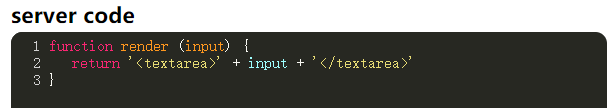
检查页面源码：



输入的内容在value的双引号中作为字符串被解析。构造payload闭合双引号及input标签：

"><script>alert(document.domain)</script>

**例题2：**xss.haozi.me/#/0x01



Server code显示输入的内容会被<textarea>标签包裹。

\*<textarea>：HTML中的文本框控件，被此标签包裹的脚本不会被解析。

构造payload：

</textarea><script>alert(1)</script>

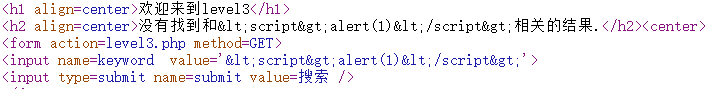
这样注入语句就在textarea标签之外，能够执行。

1. **利用on事件**

**例题1:**尖括号过滤 test.ctf8.com/level3.php



注入脚本<script>alert(1)</script>，没有效果。查看源码：



除了被单引号包裹以外，所有的尖括号均作了转义处理。

但是考虑到在input标签内，可以利用on事件触发脚本。构造payload：

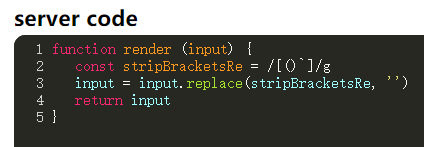
' onclick = ' alert(1)

当该payload提交后，页面对应的源代码变为：

<input name=keyword value='' onclick = 'alert(1)'>

此时点击输入框，就可以触发脚本实现弹窗。

**例题2：**圆括号过滤 xss.haozi.me/#/0x04



Server code显示，所有的圆括号和反引号会被删去。如果只删去圆括号，可以使用反引号代替括号绕过过滤，参考同靶场0x03。



直接输入脚本，alert(1)会变为alert1。这种过滤可以利用onerror事件配合html编码绕过。构造payload：

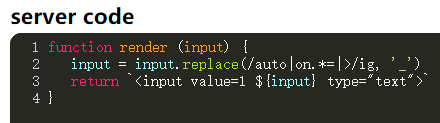
<img src="1" onerror=alert&#40;1&#41;>

该语句表示从”1”这个URL加载图片，如果出错则执行alert(1)。因为1这个地址肯定不能加载出图片，所以onerror事件必定触发。

HTML中的on事件有很多种，其中比较常用的有onerror、onclick、onload、onmouseover等。on事件后面的参数可以是字符串，也可以是脚本语句，都能解析执行,友好。

1. **换行绕过**

**例题1：**xss.haozi.me/#/0x06



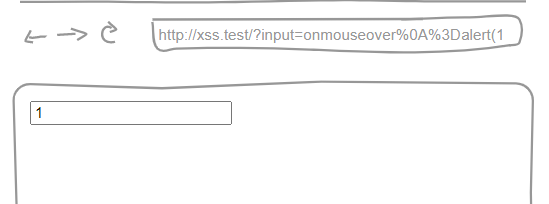
Server code显示所有以auto、on开头，且以=或>结尾的串会被替换为下划线，并且在返回值中也控制了格式，输入框中的value为1，type为text。

因为关键字过滤，不能直接使用on事件进行绕过，但是在on事件和=之间插入换行，可以绕过正则表达式，不被过滤，而javascript脚本依然能够解析执行。构造payload：

onmouseover

=alert(1);

由于server code显示网页输入框中的1是固定存在的：



在提交该payload后，将鼠标移入包含1的输入框内即可触发事件，执行弹窗。

**例题2：**xss.haozi.me/#/0x08



Server code显示，输入的数据会被包裹在<style>标签中，且包含</style>的部分会被替换成 /\* 坏人 \*/



但是如果在/style与括号之间插入任意空字符，就能绕过过滤。构造payload:

</style

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

1. **Javascript伪协议**

与实际存在的网络协议不同，伪协议一般用于关联应用程序。Javascript伪协议的形式为javascript:，在加载该伪协议的URL时，会执行冒号后方的javascript代码，利用此特性可以绕过部分过滤。

**例题1：**test.ctf8.com/level5.php



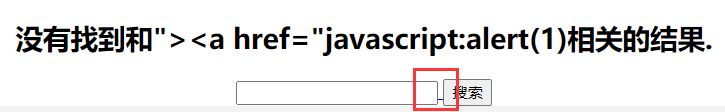
这一关过滤了所有script标签以及on关键字：





这个时候可以通过添加<a>标签，利用href参数传入javascript伪协议。构造payload:

"><a href="javascript:alert(1)



这里会出现一个很短的超链接，点击即可弹窗。

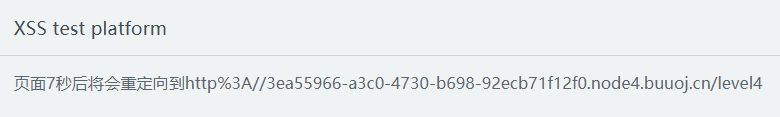
如果不喜欢这样，也可以修改payload：

"><a href="javascript:alert(1)">xss

这样超链接就会有对应的文字：

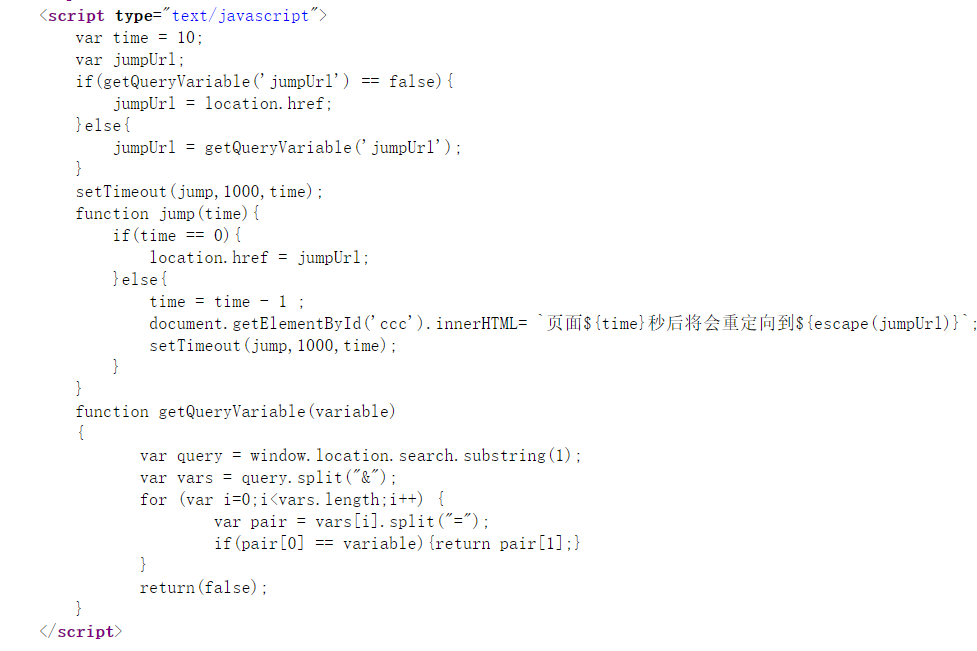


**例题2：**[BUUOJ XSS闯关](https://buuoj.cn/challenges" \l "[%E7%AC%AC%E4%BA%8C%E7%AB%A0%20web%E8%BF%9B%E9%98%B6]XSS%E9%97%AF%E5%85%B3) Level 4



网页每隔10s会重定向至自身一次。

查看网页源代码：

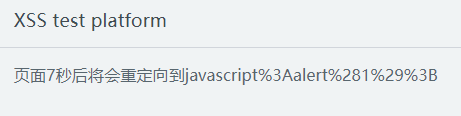


可以看出，最终网页会跳转到变量jumpUrl。

利用伪协议构造payload:

?jumpUrl=javascript:alert(1)

页面提示：



10s后即弹窗。

任何可以插入url的地方都可能可以利用伪协议运行脚本，不要错过咯。

1. **大小写绕过**

**例题：**test.ctf8.com/level6.php



比level 5多过滤了href关键字。



但是竟然没有进行大小写过滤。构造payload：

"><a HrEf="javascript:alert(1)

1. **嵌套绕过**

**例题：**test.ctf8.com/level7.php



过滤了script关键字，所有的script关键字会被删去：



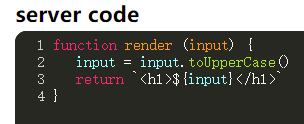
构造payload：

"><scrscriptipt>alert(1)</scrscriptipt>

在删除了script关键字后，剩下的部分组成script关键字触发脚本。所以过滤要过滤干净。

1. **编码绕过**

**例题1：**xss.haozi.me/#/0x0B



Server code显示所有输入的数据会被转换为大写。虽然HTML标签不分大小写，但是javascript是区分大小写的，这里被转大写就导致脚本不能运行。

可以利用html编码绕过大写转换，构造payload：

<img src="1" onerror=alert(1)>

编码后为：

<img src="1" onerror=&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;>

**例题2：**test.ctf8.com/level8.php



友情链接。猜测还是使用伪协议。然而这关会过滤javascript关键字：



利用html编码绕过过滤，构造payload:

&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;:alert(1)

1. **结合简单代码审计的XSS**

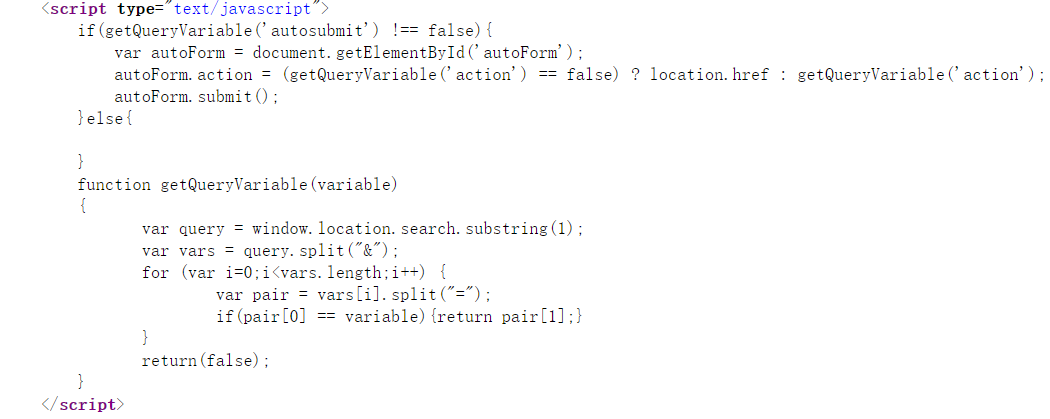
**例题：**[BUUOJ XSS闯关](https://buuoj.cn/challenges#[%E7%AC%AC%E4%BA%8C%E7%AB%A0%20web%E8%BF%9B%E9%98%B6]XSS%E9%97%AF%E5%85%B3) Level 5



页面显示为自动提交表单。但是无论输入什么提交查询均返回以下结果：



查看源代码：



可以看出网页可以接收两个参数：autosubmit和action。

当autosubmit和action均有值时，autoForm.action会被赋值为action，这就意味着自动表单会执行action指定的动作。因此，在构造payload时脚本语句应当作为action的值传入。

构造payload:

?autosubmit=1&action=javascript:alert(1);

1. 扩展
2. **KeepSession**

KeepSession是XSS平台提供的功能，其作用是延长Session的生命周期，实际操作是每隔一段时间会携带获取的cookie访问服务器，这样可以避免Session过期。参考：

[XSS平台中keepsession分析\_威客新闻\_多人维 (duorenwei.com)](http://www.duorenwei.com/news/5048.html)

1. **CSRF**

CSRF(Cross-Site Request Forgery， 跨站请求伪造)是利用网站漏洞，冒充用户发起请求，进而实现一些违背用户意愿的操作的攻击手段。CSRF有多种实现方法，一般而言满足“冒充用户”和“非本意操作”两点的攻击行为可以被认为是CSRF。XSS常用于获取网站管理员的Cookie，从而获取管理员权限对网站进行恶意操作，因此利用XSS可以实现CSRF，对网站造成比较严重的危害。

1. **SVG**

SVG(Scalable Vector Graphics, 可缩放矢量图形)是一种用XML定义的描述图形的语言，SVG图像在改变尺寸的情况下不会损失图形质量。

<svg>标签有时用于编码绕过的技巧中，在脚本前添加<svg>标签可以使后面的<script>中被html编码的部分解码，从而触发脚本运行。利用svg本身用代码描述图形的特性，也有许多插入XSS脚本的技巧。参考：

[使用SVG格式的xss注入 - heycomputer - 博客园 (cnblogs.com)](https://www.cnblogs.com/heycomputer/articles/10229723.html)

[xss link&svg黑魔法 · LoRexxar's Blog](https://lorexxar.cn/2015/11/19/xss-link/)

1. **CSP**

CSP(Content Security Policy, 内容安全策略)是一种安全防护手段，可以用来防护包括XSS在内的一部分攻击。启用CSP的网页存在一个CSP头部，或是一个包含CSP策略的<meta>标签。利用CSP可以指定网站内部资源的来源，包括脚本。XSS作为跨站脚本，受到这种安全策略的影响较大，不过，仍然有许多方法可以绕过CSP策略，具体可以参考：

[CSP策略及绕过方法 - 简书 (jianshu.com)](https://www.jianshu.com/p/f1de775bc43e)

[CSP的知识总结和绕过 | 大专栏 (dazhuanlan.com)](https://www.dazhuanlan.com/2020/03/17/5e70313b6e0f1/)

1. **Angular JS**

Angular JS是一个Javascript框架。它有“模板”和“表达式”等特性。利用Angular JS的特性可以进行XSS攻击，参考：

[AngularJS客户端模板注入（XSS）|NOSEC安全讯息平台 - 白帽汇安全研究院](https://nosec.org/home/detail/4153.html)

[AngularJs XSS沙箱绕过payload - Eleven\_Liu - 博客园 (cnblogs.com)](https://www.cnblogs.com/Eleven-Liu/p/12600961.html)

1. **HttpOnly**

HttpOnly是为cookie附加的标识，包含HttpOnly标识的cookie将不能被客户端脚本访问，这意味着XSS无法直接取得用户cookie，增加了cookie的安全性。

尽管不能直接取得用户cookie，XSS攻击仍然可以获取一些其它的信息。在HttpOnly的情况下XSS的发挥可以参考：

[Xss之HttpOnly下的攻击手法 - linuxsec - 博客园 (cnblogs.com)](https://www.cnblogs.com/linuxsec/articles/12684226.html)

[XSS遇到后台登录限制或者HTTPONLY时\_xysoul的专栏-CSDN博客](https://blog.csdn.net/xysoul/article/details/45366695)

1. **AJAX跨域**

AJAX(Asynchronous Javascript And XML,异步javascript与XML)是一种用于创建快速动态网页的技术，能够在不重新加载整个网页的情况下，更新网页部分内容。浏览器的同源政策限制了AJAX所使用的XMLHttpRequest对象不能访问外源网站，但是仍然有一些方法可以绕过该限制。参考：

[浏览器同源政策及其规避方法 - 阮一峰的网络日志 (ruanyifeng.com)](http://www.ruanyifeng.com/blog/2016/04/same-origin-policy.html)

[1.7 xss之同源策略与跨域访问 - bmjoker - 博客园 (cnblogs.com)](https://www.cnblogs.com/bmjoker/p/9500374.html)

1. 综合-窃取管理员权限

通过XSS平台，我们可以获取访问含有恶意脚本页面的用户的Cookie信息。利用获取的Cookie信息，可以伪装成正常用户甚至管理员，从而获取\*做点什么\*的权限。

**例题：**[BUUCTF [CISCN 2019 华东北赛区]Web2](https://buuoj.cn/challenges#[CISCN2019%20%E5%8D%8E%E4%B8%9C%E5%8C%97%E8%B5%9B%E5%8C%BA]Web2)

**题目描述：** 靶机无法访问外网，请使用 [xss.buuoj.cn](http://xss.buuoj.cn)

反馈时请使用 web 来代替地址中的靶机域名。

**工具：**

|  |  |  |  |
| --- | --- | --- | --- |
| 工具类型 | 工具名称 | 版本 | 备注 |
| 基础环境 | Firefox浏览器 | 90.0.2 |  |
| Python | 2.7.16 |  |
| 工具 | XSS Platform |  | xss.buuoj.cn |
| XSS’OR | \ | xssor.io  用于编码 |
| 脚本 | submd5.py | \ | 计算Md5验证码 |
|  | \ |  |

**脚本**：submd5.py

# -\*- coding: utf-8 -\*-

import multiprocessing

import hashlib

import random

import string

import sys

CHARS = string.letters + string.digits

def cmp\_md5(substr, stop\_event, str\_len, start=0, size=20):

    global CHARS

    while not stop\_event.is\_set():

        rnds = ''.join(random.choice(CHARS) for \_ in range(size))

        md5 = hashlib.md5(rnds)

        if md5.hexdigest()[start: start+str\_len] == substr:

            print rnds

            stop\_event.set()

if \_\_name\_\_ == '\_\_main\_\_':

    substr = sys.argv[1].strip()

    start\_pos = int(sys.argv[2]) if len(sys.argv) > 1 else 0

    str\_len = len(substr)

    cpus = multiprocessing.cpu\_count()

    stop\_event = multiprocessing.Event()

    processes = [multiprocessing.Process(target=cmp\_md5, args=(substr,

                                         stop\_event, str\_len, start\_pos))

                 for i in range(cpus)]

    for p in processes:

        p.start()

    for p in processes:

        p.join()

使用方法：控制台命令 >python submd5.py "指定字符串" 起始位置

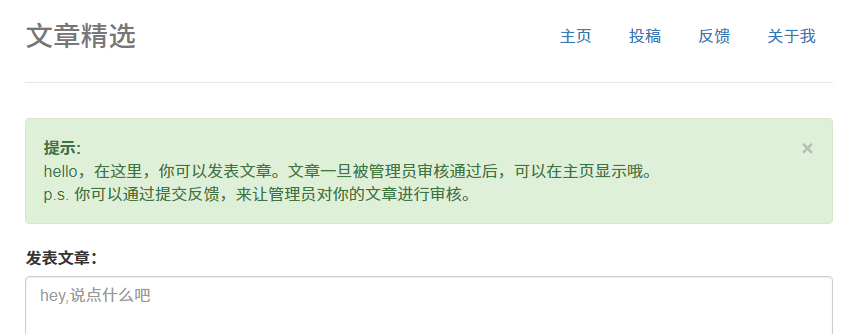
会返回一个长度20的字符串。

**解题步骤：**



靶机主页：文章投稿平台。右上角显示其它tag：投稿、反馈、关于我。初步判断该题应当是存储型XSS，利用投稿功能上传脚本。

点击投稿，提示需要注册，就随意了。注册完后查看页面：



“可以通过提交反馈，让管理员对文章进行审核。”说明这个网站是有管理员的，可能需要利用XSS来获取管理员的Session。

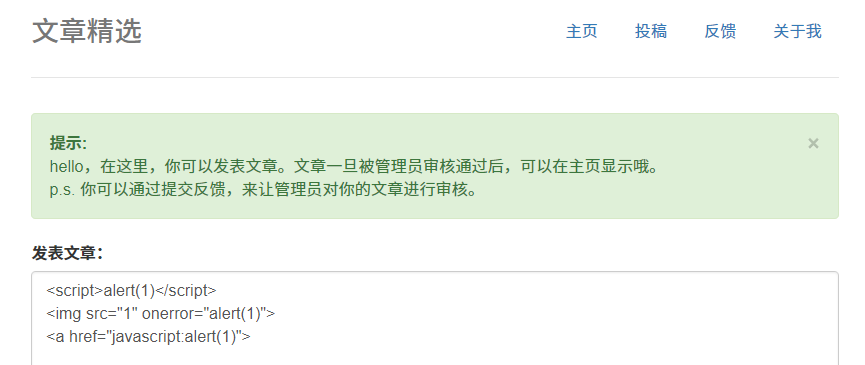
查看反馈页面：



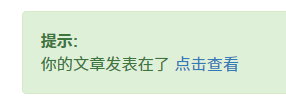
“我会亲自查看。”说明管理员会访问提交的URL，那么只要想办法将恶意代码投稿，再利用反馈来让管理员访问，就可以获取管理员的Session。这里还有一个验证码，是一个md5截断验证，要求提交的验证码的md5前六位为指定字符串。

基本思路：用投稿功能上传脚本，并利用反馈功能结合XSS获取管理员信息。

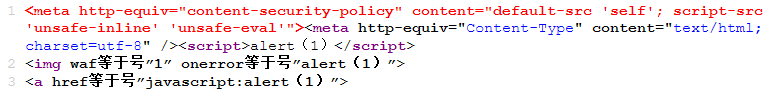
那么先尝试在投稿页上传一些简单的脚本代码：



提交后会提示文章上传的位置：



当然，点进去并不会弹窗。查看页面源代码：



注意到该网页开启了CSP，同时设置了将()转义为（）、src转义为waf、=转义为等于号等一系列转义规则。结合之前在闯关靶场遇到的各种转义，考虑用编码绕过。

CSP具体的策略内容：default-src为self，意味着在没有单独设置策略的情况下，图片、脚本等内容来源只能是本站；script-src有unsafe-inline和unsafe-eval两项。其中unsafe-inline允许页面执行内联代码。在这种情况下，虽然<script src="xss"></script>这种直接引入脚本的方法不可用，但是由<script>标签所包裹的脚本代码，比如：

<script>alert(1)</script>

是可以执行的。

unsafe-eval则允许通过eval()方法从字符串创建代码，比如：

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

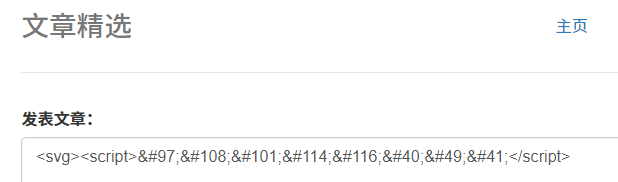
是可以执行的。不过这题并不需要用到这一点。

那么攻击思路就是：利用<script>标签，配合编码绕过网页过滤，上传恶意代码。

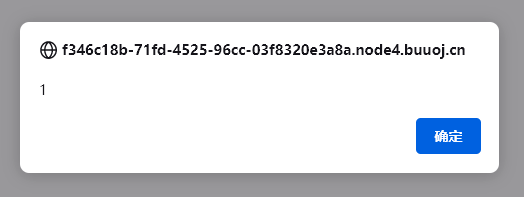
先进行一个尝试：上传alert(1)。将代码转换为html编码：

&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;

实际上传的内容为：

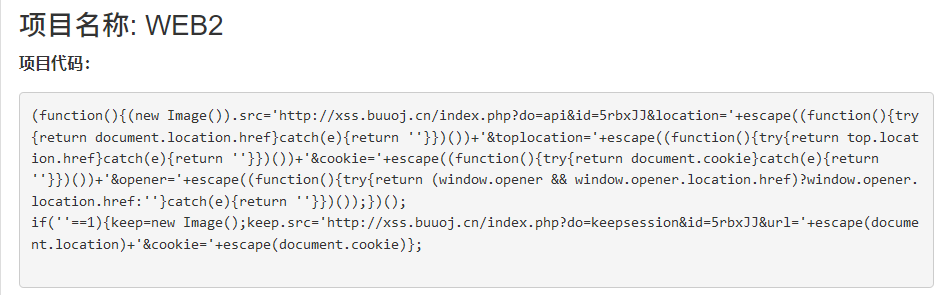


这里使用<svg>标签可以对html编码进行解码触发脚本。提交后查看页面，能够弹窗：



那么接下来只要注入XSS平台的代码即可。

打开XSS平台，创建项目：

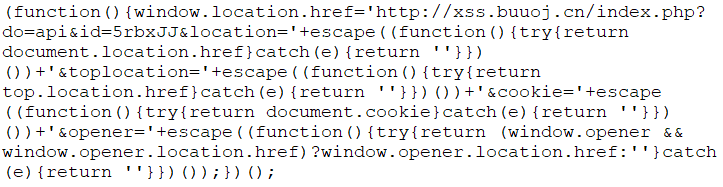


可以看到代码有两段，下面的一段是keepsession的部分，可以先不用管，重点是上面这段代码，它用于捕获访问者的信息，只要将这段代码上传到服务器，就可以通过XSS平台查看访问者的cookie了。

不过在这段代码中使用了(new Image()).src的方法来引入脚本，而CSP限制了不能使用外源图片，这会导致后面的脚本不能被加载，所以这里要用另一种方法来触发脚本。

一个比较简单的替代方案：使用window.location.href来代替(new Image()).src。window.location.href的作用是跳转到指定url，这样可以保证在访问后加载我们上传的脚本，同时又不会因为引入外站资源被CSP限制。

那么实际代码就变成：



编码后配合<svg><script>上传到服务器：



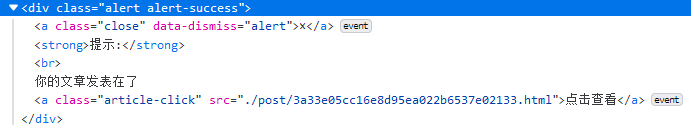
很长，好在文章没有字数限制。上传成功后查看文章，会直接跳回首页。

查看XSS平台项目内容：



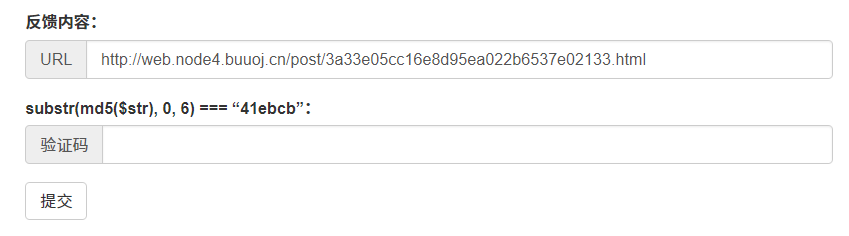
出现了我们自己访问的痕迹。

前往反馈页面，需要提交页面URL，但是闪得太快了来不及复制，可以在投稿成功的页面检查元素：



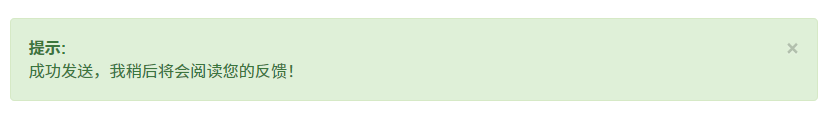
这里的src后面的链接即是文章位置。

复制链接，并将靶机域名按照题目要求改为web。利用脚本来运算验证码并提交：





会提示成功发送。



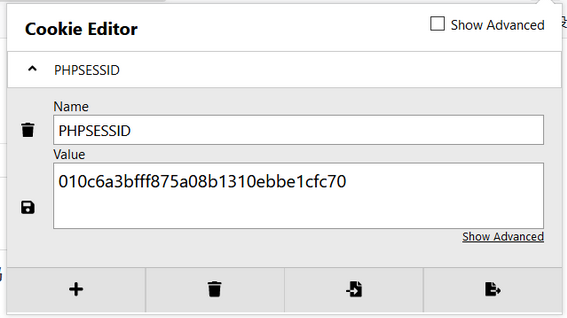
到这里，理论上XSS平台就会收到管理员访问的信息，不过这里的靶机环境似乎有些问题，导致平台无法收到管理员的Cookie，所以到这里就没法进行下一步了。好在大部分XSS的工作已经完成，剩下的部分是从网上的WP得来的。

如果靶机环境没有问题的话，大概会收到以下内容：



从location中的web可以判断，这是管理员访问的痕迹，cookie中的PHPSESSID就是管理员的Session。

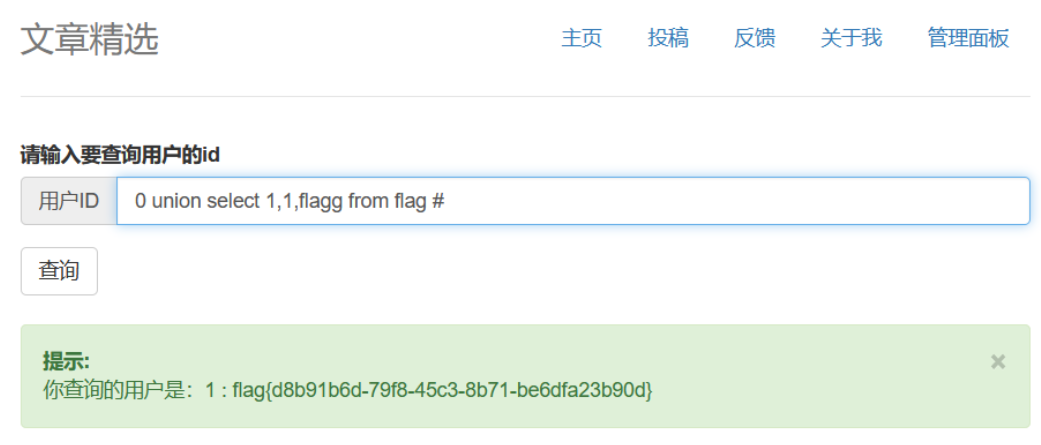
在管理员页面(/admin.php，好猜，猜不到的话在主页修改cookie可能也会出现管理页面的选项)，修改cookie：



页面就会显示一个查询框。



接下来的部分是一个很简单的SQL注入：



取得flag，结束。