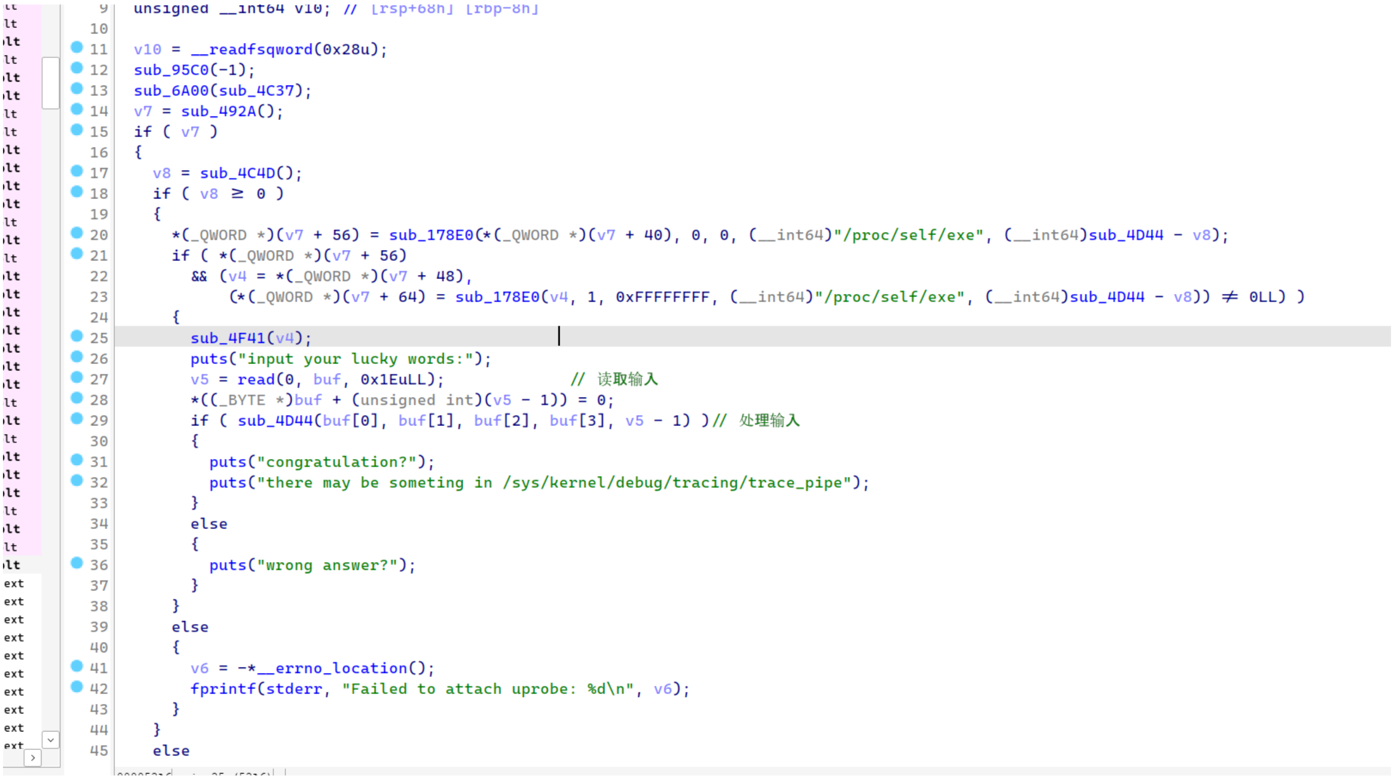
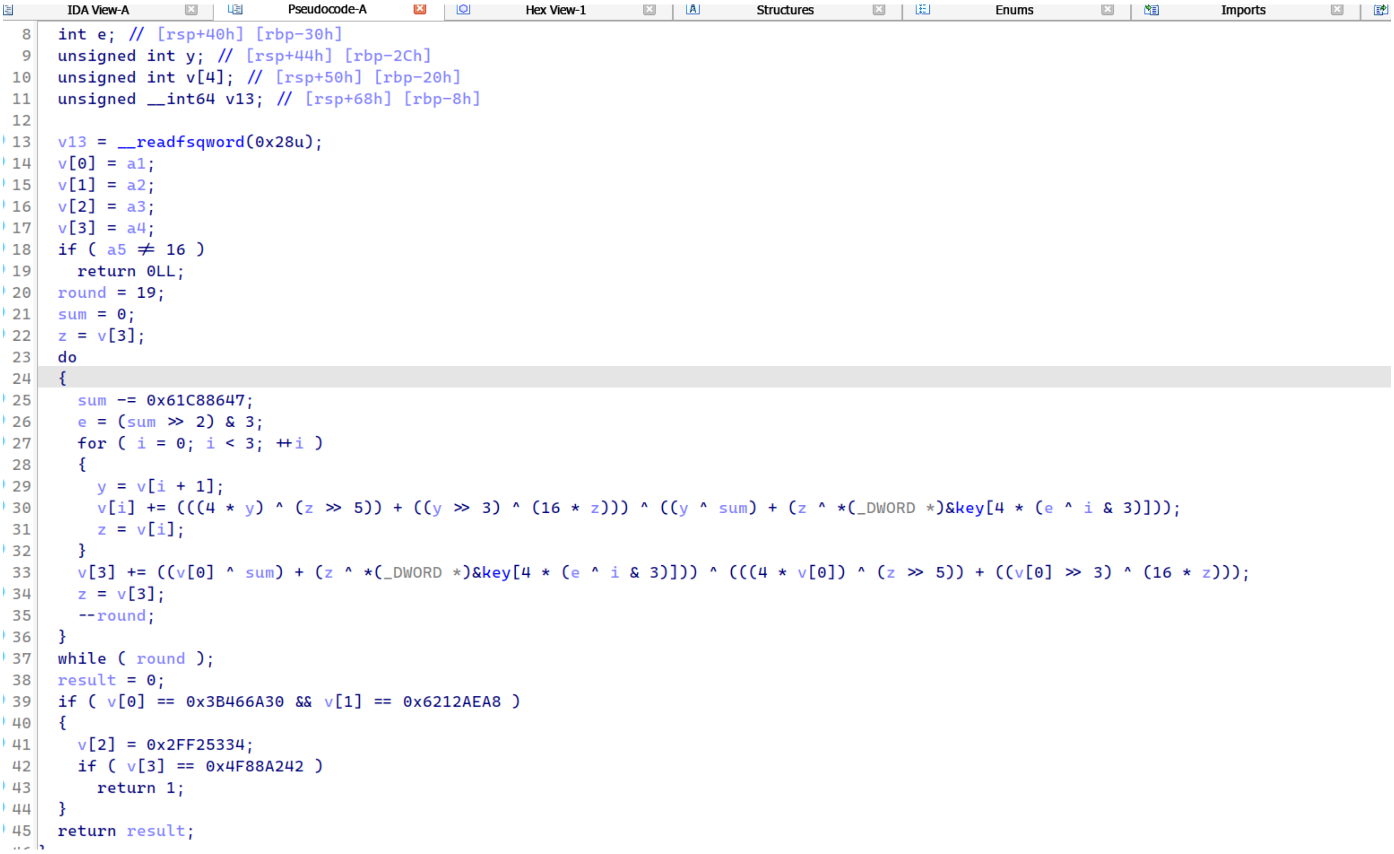
1. IDA静态分析

查看main函数找到处理输入的函数



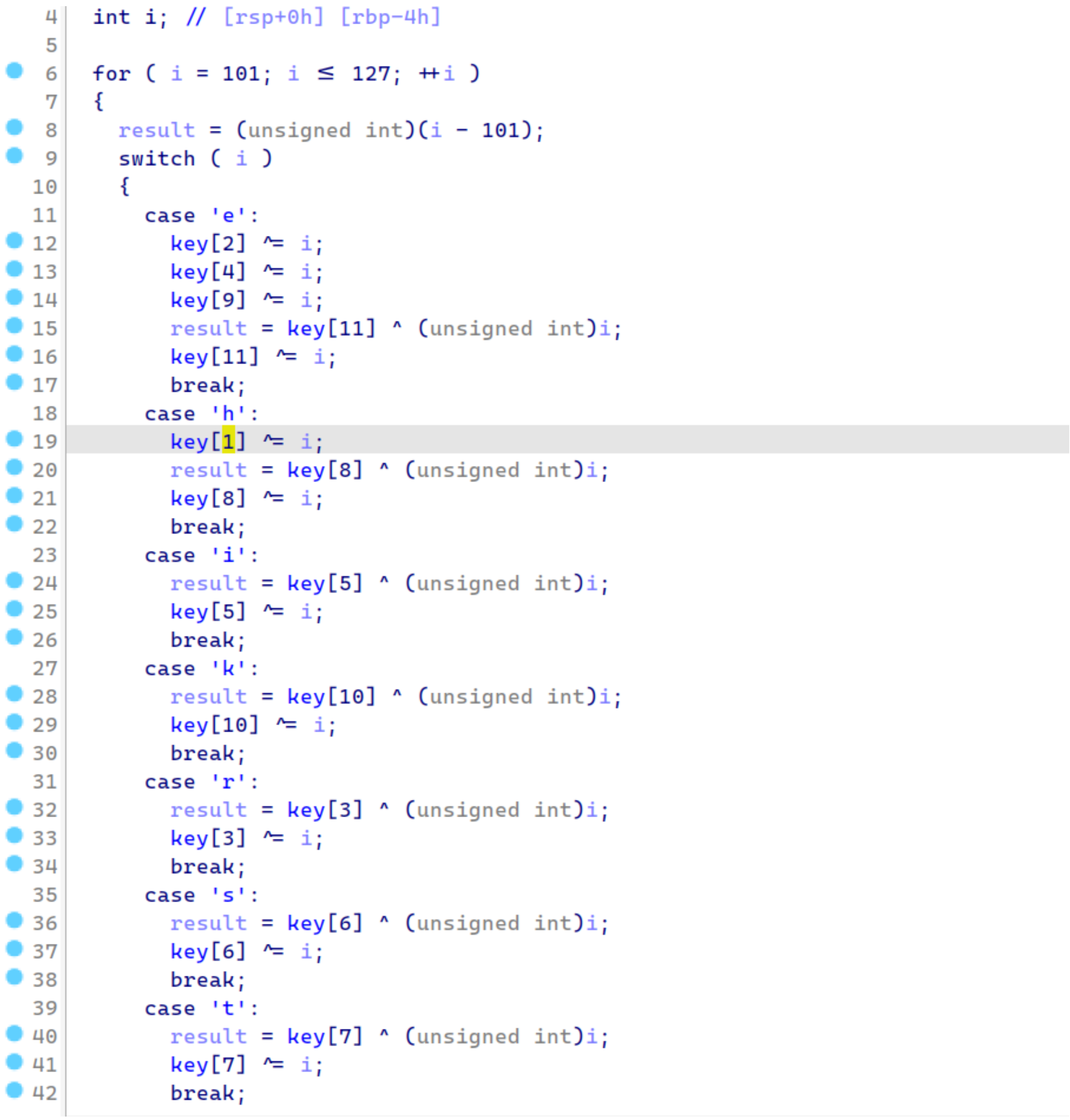
```
unsigned __int64 v10; // [rsp+68h] [rbp-8h]

v10 = __readfsqword(0x28u);
sub_95C0(-1);
sub_6A00(sub_4C37);
v7 = sub_492A();
if ( v7 )
{
  v8 = sub_4C4D();
  if ( v8 ≥ 0 )
  {
    *(_QWORD *)(v7 + 56) = sub_178E0(*(_QWORD *)(v7 + 40), 0, 0, (__int64)"/proc/self/exe", (__int64)sub_4D44 - v8);
    if ( *(_QWORD *)(v7 + 56)
      && (v4 = *(_QWORD *)(v7 + 48),
          (*(_QWORD *)(v7 + 64) = sub_178E0(v4, 1, 0xFFFFFFFF, (__int64)"/proc/self/exe", (__int64)sub_4D44 - v8)) ≠ 0LL) )
    {
      sub_4F41(v4);
      puts("input your lucky words:");
      v5 = read(0, buf, 0x1EuLL);             // 读取输入
      *((_BYTE *)buf + (unsigned int)(v5 - 1)) = 0;
      if ( sub_4D44(buf[0], buf[1], buf[2], buf[3], v5 - 1) )// 处理输入
      {
        puts("congratulation?");
        puts("there may be someting in /sys/kernel/debug/tracing/trace_pipe");
      }
      else
      {
        puts("wrong answer?");
      }
    }
    else
    {
      v6 = -*__errno_location();
      fprintf(stderr, "Failed to attach uprobe: %d\n", v6);
    }
  }
  else
```

查看处理输入的函数，可以看出输入的长度为16，并对输入进行了xxtea的操作，最后和数据进行了比较



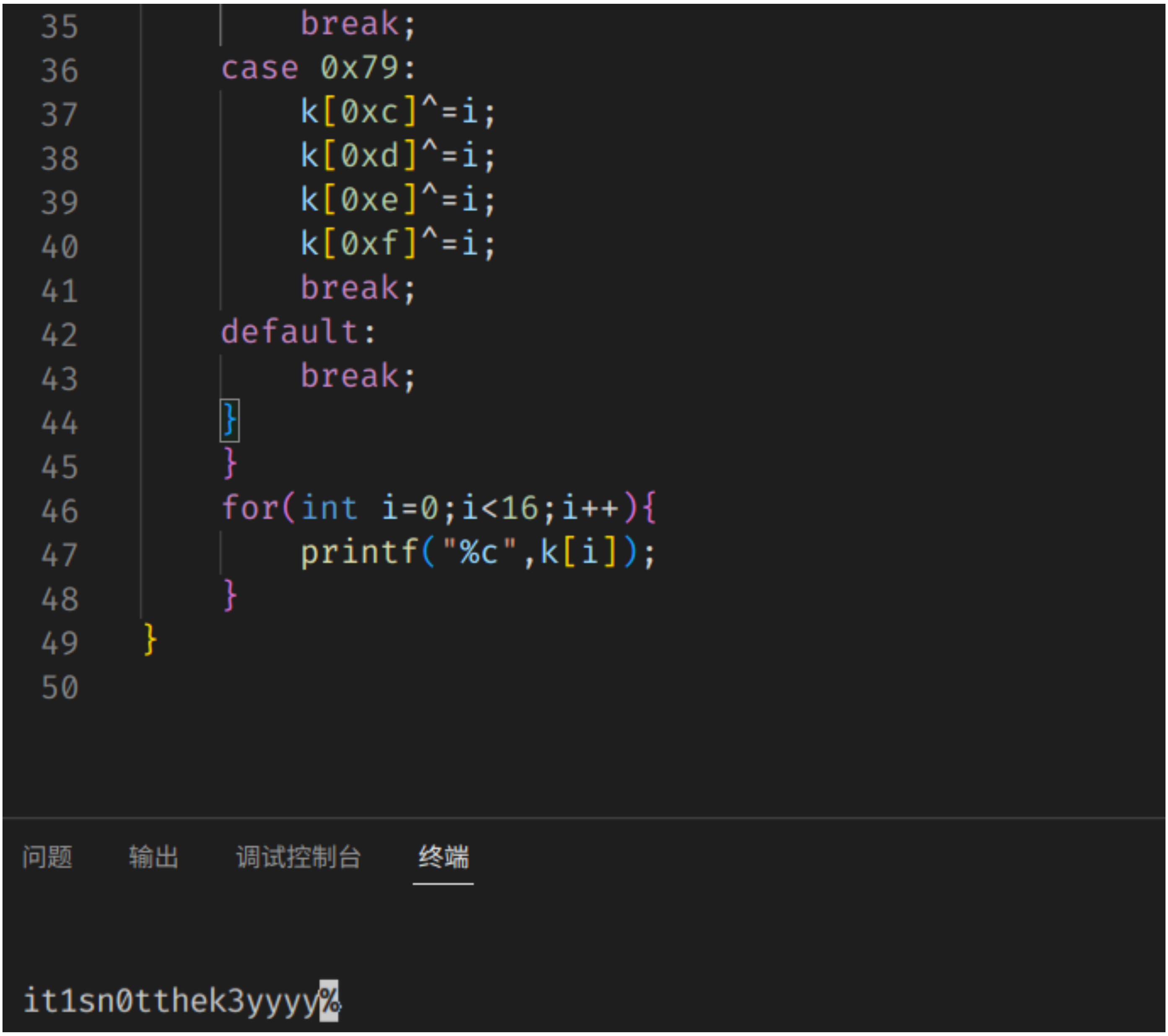
```
int e; // [rsp+40h] [rbp-30h]
unsigned int y; // [rsp+44h] [rbp-2Ch]
unsigned int v[4]; // [rsp+50h] [rbp-20h]
unsigned __int64 v13; // [rsp+68h] [rbp-8h]

v13 = __readfsqword(0x28u);
v[0] = a1;
v[1] = a2;
v[2] = a3;
v[3] = a4;
if ( a5 ≠ 16 )
  return 0LL;
round = 19;
sum = 0;
z = v[3];
do
{
  sum -= 0x61C88647;
  e = (sum >> 2) & 3;
  for ( i = 0; i < 3; ++i )
  {
    y = v[i + 1];
    v[i] += (((4 * y) ^ (z >> 5)) + ((y >> 3) ^ (16 * z))) ^ ((y ^ sum) + (z ^ *(_DWORD *)&key[4 * (e ^ i & 3)]));
    z = v[i];
  }
  v[3] += ((v[0] ^ sum) + (z ^ *(_DWORD *)&key[4 * (e ^ i & 3)])) ^ (((4 * v[0]) ^ (z >> 5)) + ((v[0] >> 3) ^ (16 * z)));
  z = v[3];
  --round;
}
while ( round );
result = 0;
if ( v[0] == 0x3B466A30 && v[1] == 0x6212AEA8 )
{
  v[2] = 0x2FF25334;
  if ( v[3] == 0x4F88A242 )
    return 1;
}
return result;
```

查看key的引用，会发现key被进行过写入操作，具体逻辑如下



```
int i; // [rsp+0h] [rbp-4h]

for ( i = 101; i ≤ 127; ++i )
{
  result = (unsigned int)(i - 101);
  switch ( i )
  {
    case 'e':
      key[2] ^= i;
      key[4] ^= i;
      key[9] ^= i;
      result = key[11] ^ (unsigned int)i;
      key[11] ^= i;
      break;
    case 'h':
      key[1] ^= i;
      result = key[8] ^ (unsigned int)i;
      key[8] ^= i;
      break;
    case 'i':
      result = key[5] ^ (unsigned int)i;
      key[5] ^= i;
      break;
    case 'k':
      result = key[10] ^ (unsigned int)i;
      key[10] ^= i;
      break;
    case 'r':
      result = key[3] ^ (unsigned int)i;
      key[3] ^= i;
      break;
    case 's':
      result = key[6] ^ (unsigned int)i;
      key[6] ^= i;
      break;
    case 't':
      result = key[7] ^ (unsigned int)i;
      key[7] ^= i;
      break;
```

其实就是将key与另一个字符串异或，换了一种形式。导出key的数据并复制运行处理key的代码逻辑，即可还原出key

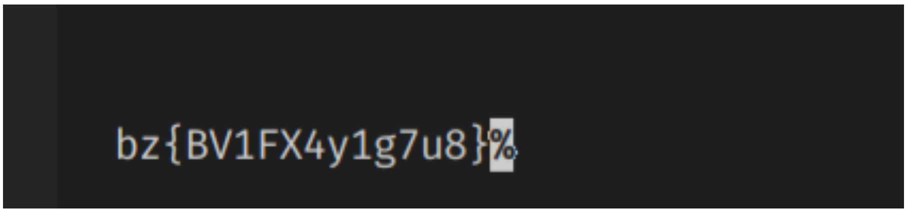


```
            break;
        case 0x79:
            k[0xc]^=i;
            k[0xd]^=i;
            k[0xe]^=i;
            k[0xf]^=i;
            break;
        default:
            break;
        }
        }
        for(int i=0;i<16;i++){
            printf("%c",k[i]);
        }
    }

```

```
it1sn0tthek3yyyy%
```

用现有的key和用来对比的数据，放到解xxtea的脚本里即可解密出原文

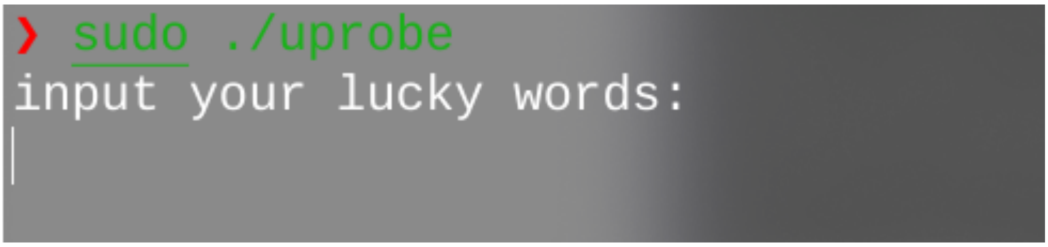
```
#include <stdio.h>
#include <stdint.h>
#define DELTA 0x9e3779b9
unsigned char k[17]="it1sn0tthek3yyyy";
void btea(uint32_t *v, int n)
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    unsigned int *key=(unsigned int *)k;
    if (n > 1)            /* Coding Part */
    {
        rounds = 6 + 52/n;
        sum = 0;
        z = v[n-1];
        do
        {
            sum += DELTA;
            e = (sum >> 2) & 3;
            for (p=0; p<n-1; p++)
            {
                y = v[p+1];
                z = v[p] += (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ 
z)));
            }
            y = v[0];
            z = v[n-1] += (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ 
z)));
        }
        while (--rounds);
    }
    else if (n < -1)      /* Decoding Part */
    {
        n = -n;
        rounds = 6 + 52/n;
        sum = rounds*DELTA;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p=n-1; p>0; p--)
            {
                z = v[p-1];
                y = v[p] -= (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ 
z)));
            }
            z = v[n-1];
            y = v[0] -= (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)));
            sum -= DELTA;
        }
        while (--rounds);
    }
}


int main()
{
    int n= 4;
    uint32_t v[4]={0x3B466A30,0x6212AEA8,0x2FF25334,0x4F88A242};
    btea(v, -n);
    for(int i=0;i<16;i++){
        printf("%c",((char*)v)[i]);
    }
    return 0;
}
```



```
bz{BV1FX4y1g7u8}%
```

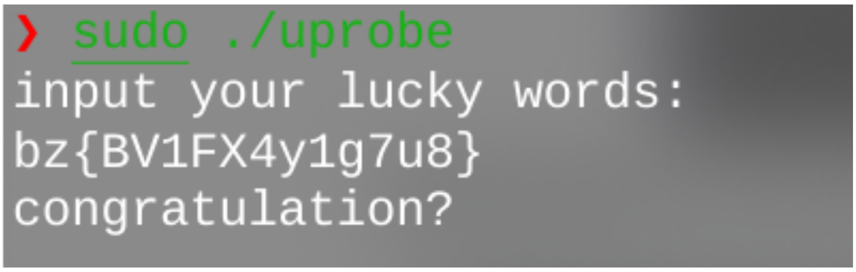
2. 动态运行

以root权限运行



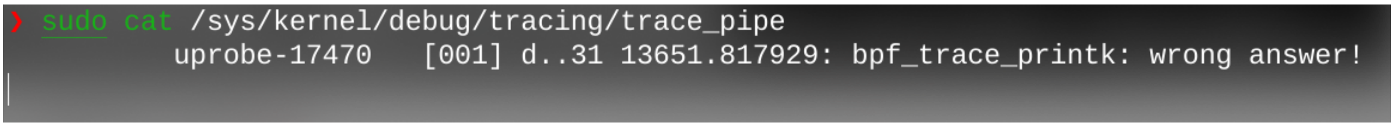
```
❯ sudo ./uprobe
input your lucky words:
```

输入解出的数据



```
❯ sudo ./uprobe
input your lucky words:
bz{BV1FX4y1g7u8}
congratulation?
```

查看/sys/kernel/debug/tracing/trace_pipe

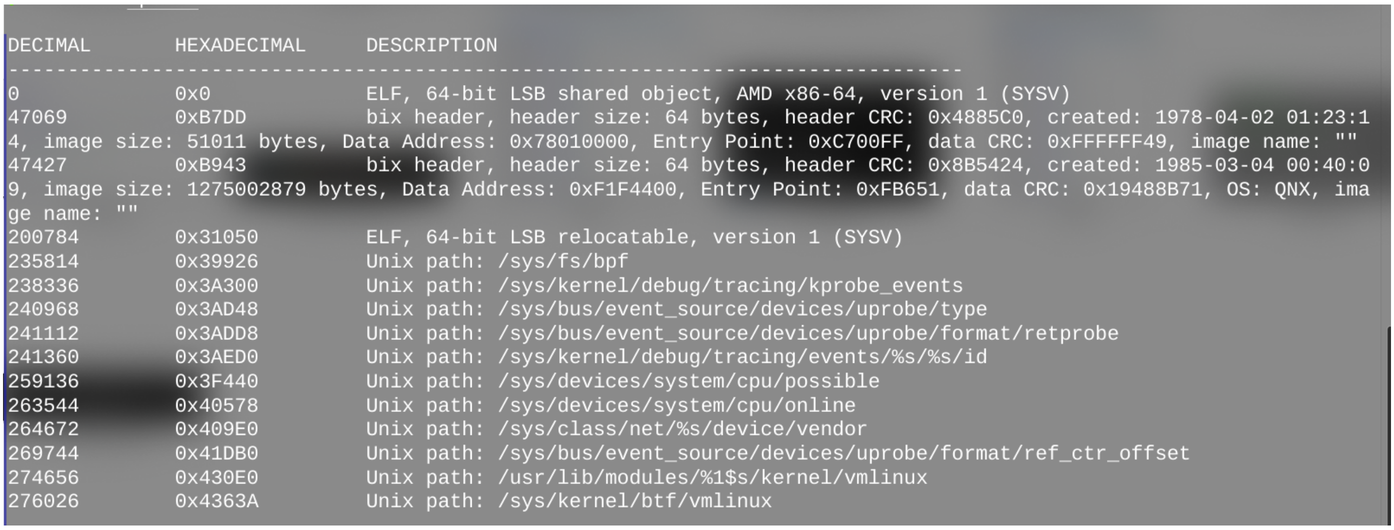


```
❯ sudo cat /sys/kernel/debug/tracing/trace_pipe
          uprobe-17470   [001] d..31 13651.817929: bpf_trace_printk: wrong answer!
```

可以确定还有一个判断逻辑运行在内核的ebpf上

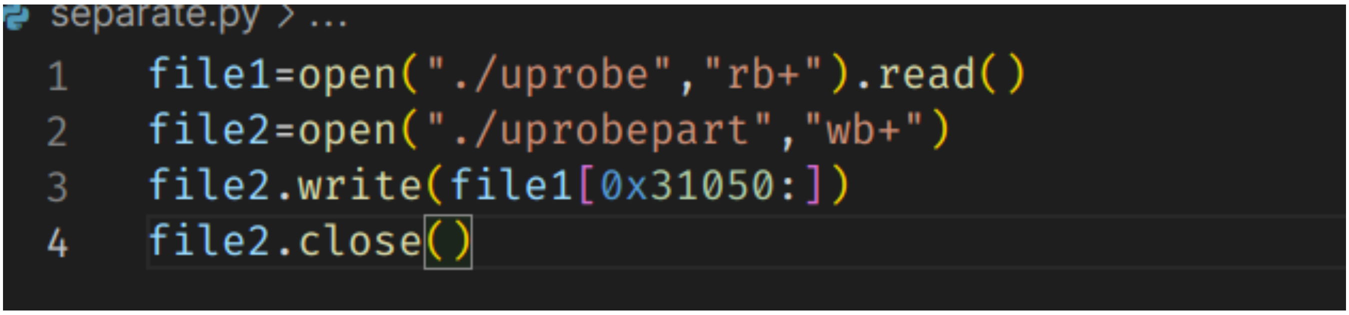
3. 分析ebpf程序的逻辑

用binwalk查看ELF文件



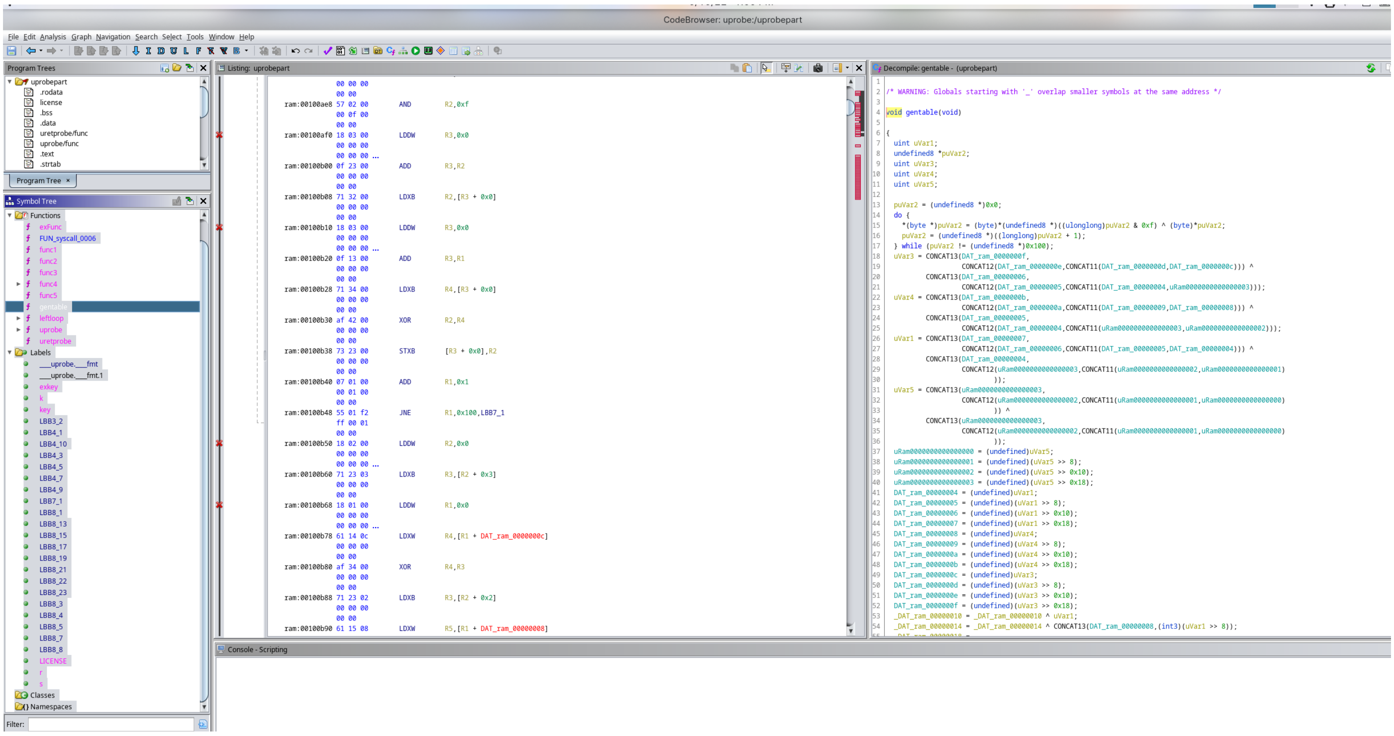
```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             ELF, 64-bit LSB shared object, AMD x86-64, version 1 (SYSV)
47069         0xB7DD          bix header, header size: 64 bytes, header CRC: 0x4885C0, created: 1978-04-02 01:23:1
4, image size: 51011 bytes, Data Address: 0x78010000, Entry Point: 0xC700FF, data CRC: 0xFFFFFF49, image name: ""
47427         0xB943          bix header, header size: 64 bytes, header CRC: 0x8B5424, created: 1985-03-04 00:40:0
9, image size: 1275002879 bytes, Data Address: 0xF1F4400, Entry Point: 0xFB651, data CRC: 0x19488B71, OS: QNX, ima
ge name: ""
200784        0x31050         ELF, 64-bit LSB relocatable, version 1 (SYSV)
235814        0x39926         Unix path: /sys/fs/bpf
238336        0x3A300         Unix path: /sys/kernel/debug/tracing/kprobe_events
240968        0x3AD48         Unix path: /sys/bus/event_source/devices/uprobe/type
241112        0x3ADD8         Unix path: /sys/bus/event_source/devices/uprobe/format/retprobe
241360        0x3AED0         Unix path: /sys/kernel/debug/tracing/events/%s/%s/id
259136        0x3F440         Unix path: /sys/devices/system/cpu/possible
263544        0x40578         Unix path: /sys/devices/system/cpu/online
264672        0x409E0         Unix path: /sys/class/net/%s/device/vendor
269744        0x41DB0         Unix path: /sys/bus/event_source/devices/uprobe/format/ref_ctr_offset
274656        0x430E0         Unix path: /usr/lib/modules/%1$s/kernel/vmlinux
276026        0x4363A         Unix path: /sys/kernel/btf/vmlinux
```

用python将偏移为0x31050的ELF提取出来



```python
file1=open("./uprobe","rb+").read()
file2=open("./uprobepart","wb+")
file2.write(file1[0x31050:])
file2.close()
```

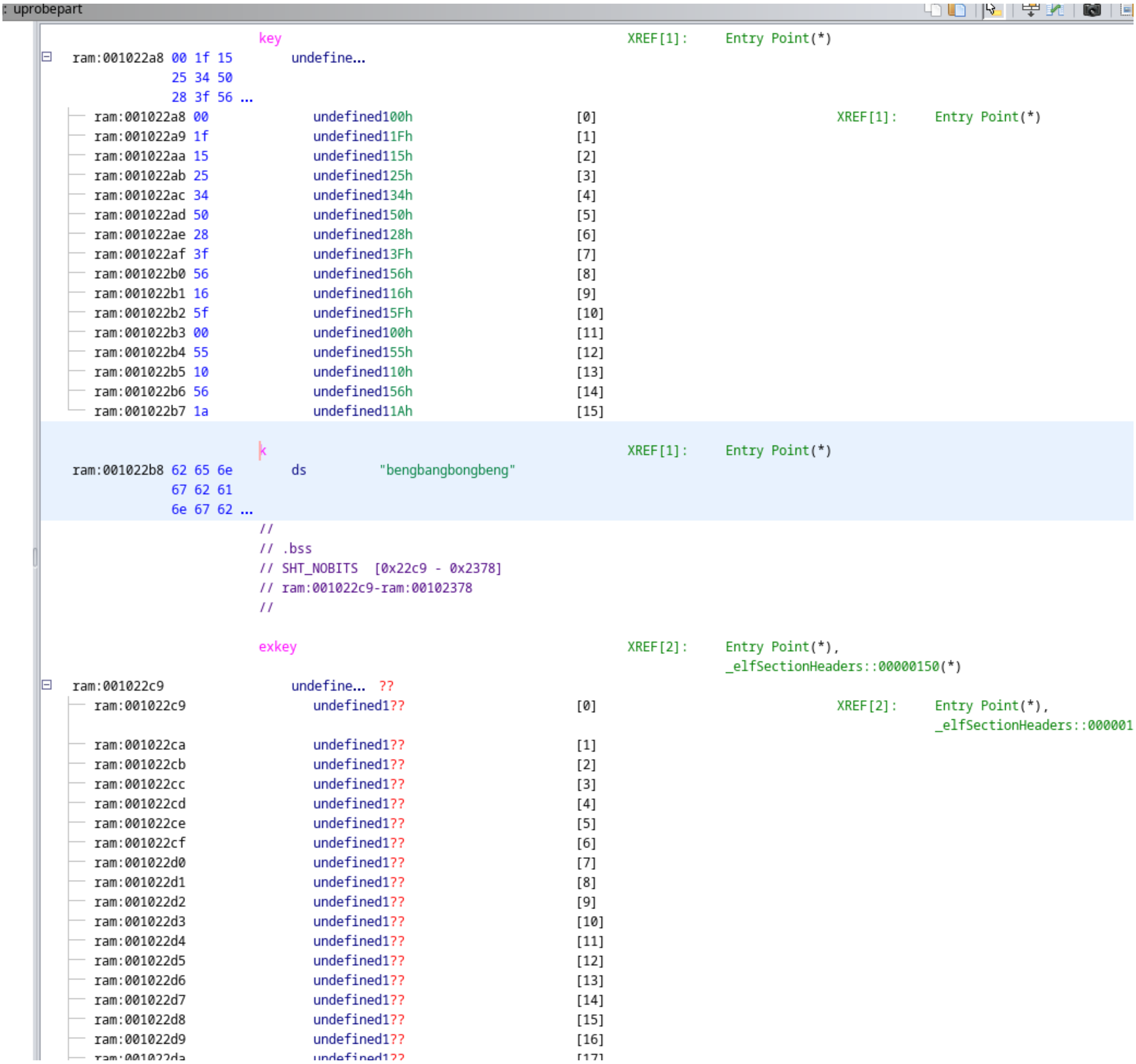
用Ghidra打开分离出的ELF，可以直接查看函数伪代码和全局变量。也可以用llvm-objdump反汇编ebpf字节码，不过不能反编译（Ghidra反编译效果也不好）。



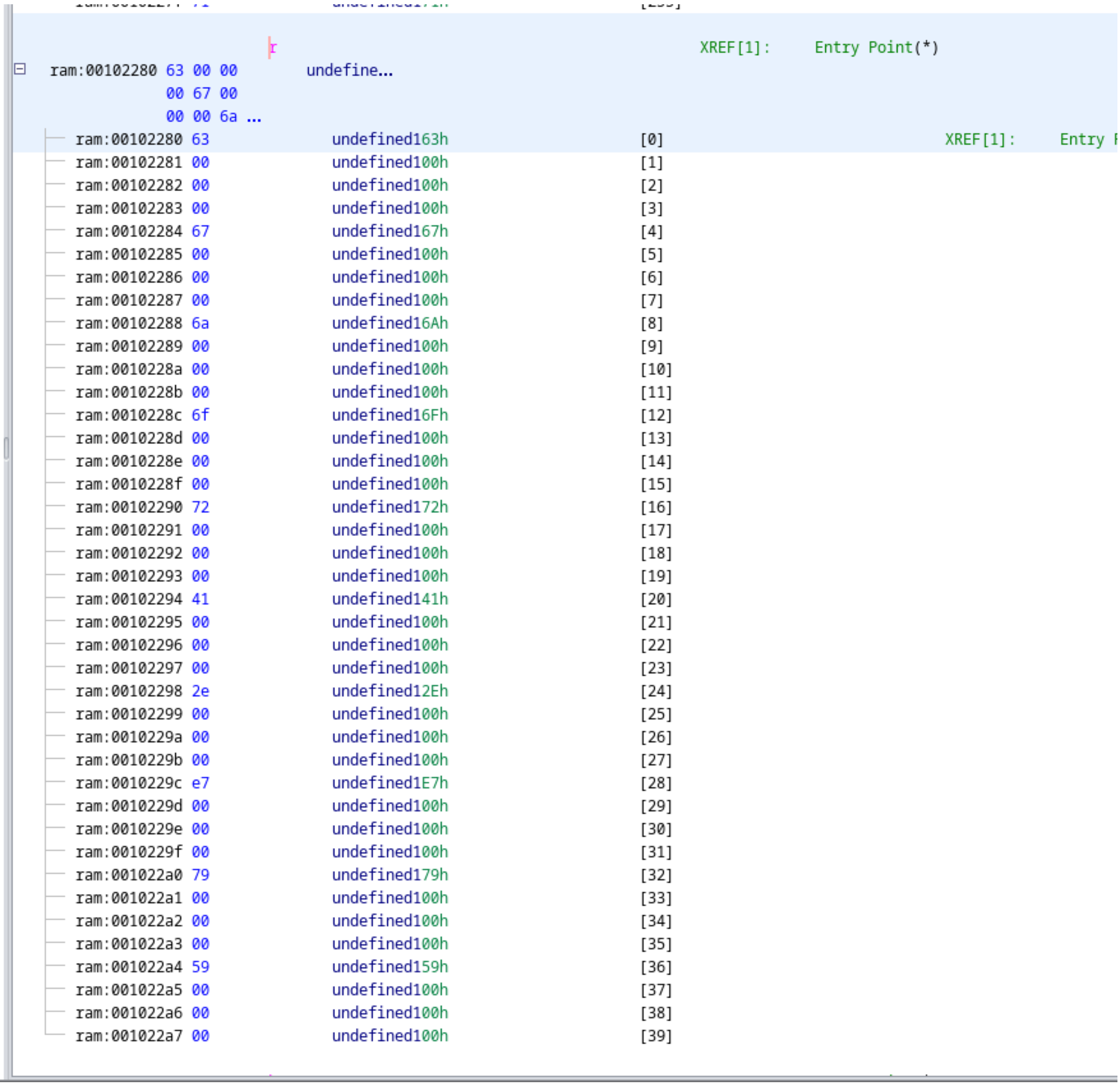
```
                    00 00 00
                    00 00
     ram:00100ae8 57 02 00        AND        R2,0xf
                    00 0f 00
                    00 00
     ram:00100af0 18 03 00        LDDW       R3,0x0
                    00 00 00
                    00 00 00 ...
     ram:00100b00 0f 23 00        ADD        R3,R2
                    00 00 00
                    00 00
     ram:00100b08 71 32 00        LDXB       R2,[R3 + 0x0]
                    00 00 00
                    00 00
     ram:00100b10 18 03 00        LDDW       R3,0x0
                    00 00 00
                    00 00 00 ...
     ram:00100b20 0f 13 00        ADD        R3,R1
                    00 00 00
                    00 00
     ram:00100b28 71 34 00        LDXB       R4,[R3 + 0x0]
                    00 00 00
                    00 00
     ram:00100b30 af 42 00        XOR        R2,R4
                    00 00 00
                    00 00
     ram:00100b38 73 23 00        STXB       [R3 + 0x0],R2
                    00 00 00
                    00 00
     ram:00100b40 07 01 00        ADD        R1,0x1
                    00 01 00
                    00 00
     ram:00100b48 55 01 f2        JNE        R1,0x100,LBB7_1
                    ff 00 01
                    00 00
     ram:00100b50 18 02 00        LDDW       R2,0x0
                    00 00 00
                    00 00 00 ...
     ram:00100b60 71 23 03        LDXB       R3,[R2 + 0x3]
                    00 00 00
                    00 00
     ram:00100b68 18 01 00        LDDW       R1,0x0
                    00 00 00
                    00 00 00 ...
     ram:00100b78 61 14 0c        LDXW       R4,[R1 + DAT_ram_0000000c]
                    00 00 00
                    00 00
     ram:00100b80 af 34 00        XOR        R4,R3
                    00 00 00
                    00 00
     ram:00100b88 71 23 02        LDXB       R3,[R2 + 0x2]
                    00 00 00
                    00 00
     ram:00100b90 61 15 08        LDXW       R5,[R1 + DAT_ram_00000008]
```

```c

/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void gentable(void)

{
  uint uVar1;
  undefined8 *puVar2;
  uint uVar3;
  uint uVar4;
  uint uVar5;

  puVar2 = (undefined8 *)0x0;
  do {
    *(byte *)puVar2 = (byte)*(undefined8 *)((ulonglong)puVar2 & 0xf) ^ (byte)*puVar2;
    puVar2 = (undefined8 *)((longlong)puVar2 + 1);
  } while (puVar2 != (undefined8 *)0x100);
  uVar3 = CONCAT13(DAT_ram_0000000f,
                   CONCAT12(DAT_ram_0000000e,CONCAT11(DAT_ram_0000000d,DAT_ram_0000000c))) ^
          CONCAT13(DAT_ram_00000006,
                   CONCAT12(DAT_ram_00000005,CONCAT11(DAT_ram_00000004,uRam0000000000000003)));
  uVar4 = CONCAT13(DAT_ram_0000000b,
                   CONCAT12(DAT_ram_0000000a,CONCAT11(DAT_ram_00000009,DAT_ram_00000008))) ^
          CONCAT13(DAT_ram_00000005,
                   CONCAT12(DAT_ram_00000004,CONCAT11(uRam0000000000000003,uRam0000000000000002)));
  uVar1 = CONCAT13(DAT_ram_00000007,
                   CONCAT12(DAT_ram_00000006,CONCAT11(DAT_ram_00000005,DAT_ram_00000004))) ^
          CONCAT13(DAT_ram_00000004,
                   CONCAT12(uRam0000000000000003,CONCAT11(uRam0000000000000002,uRam0000000000000001)
                           ));
  uVar5 = CONCAT13(uRam0000000000000003,
                   CONCAT12(uRam0000000000000002,CONCAT11(uRam0000000000000001,uRam0000000000000000)
                           )) ^
          CONCAT13(uRam0000000000000003,
                   CONCAT12(uRam0000000000000002,CONCAT11(uRam0000000000000001,uRam0000000000000000)
                           ));
  uRam0000000000000000 = (undefined)uVar5;
  uRam0000000000000001 = (undefined)(uVar5 >> 8);
  uRam0000000000000002 = (undefined)(uVar5 >> 0x10);
  uRam0000000000000003 = (undefined)(uVar5 >> 0x18);
  DAT_ram_00000004 = (undefined)uVar1;
  DAT_ram_00000005 = (undefined)(uVar1 >> 8);
  DAT_ram_00000006 = (undefined)(uVar1 >> 0x10);
  DAT_ram_00000007 = (undefined)(uVar1 >> 0x18);
  DAT_ram_00000008 = (undefined)uVar4;
  DAT_ram_00000009 = (undefined)(uVar4 >> 8);
  DAT_ram_0000000a = (undefined)(uVar4 >> 0x10);
  DAT_ram_0000000b = (undefined)(uVar4 >> 0x18);
  DAT_ram_0000000c = (undefined)uVar3;
  DAT_ram_0000000d = (undefined)(uVar3 >> 8);
  DAT_ram_0000000e = (undefined)(uVar3 >> 0x10);
  DAT_ram_0000000f = (undefined)(uVar3 >> 0x18);
  _DAT_ram_00000010 = _DAT_ram_00000010 ^ uVar1;
  _DAT_ram_00000014 = _DAT_ram_00000014 ^ CONCAT13(DAT_ram_00000008,(int3)(uVar1 >> 8));
```

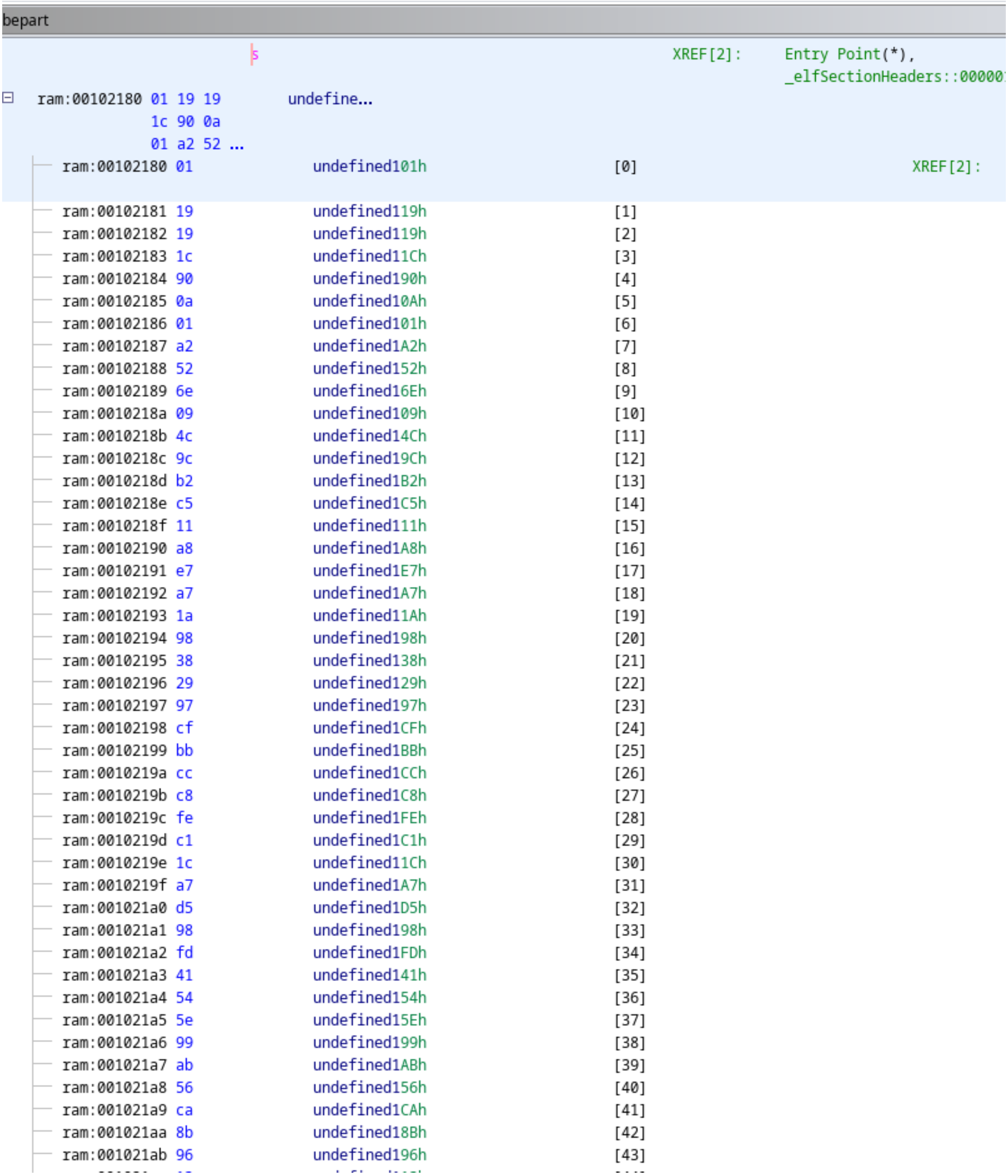
全局变量有176字节的未初始化变量exkey、明文k、16字节的key、40字节的r和256字节的s



```
                          key                                 XREF[1]:     Entry Point(*)
    ram:001022a8 00 1f 15        undefine…
                 25 34 50
                 28 3f 56 …
       ram:001022a8 00               undefined100h          [0]                           XREF[1]:     Entry Point(*)
       ram:001022a9 1f               undefined11Fh          [1]
       ram:001022aa 15               undefined115h          [2]
       ram:001022ab 25               undefined125h          [3]
       ram:001022ac 34               undefined134h          [4]
       ram:001022ad 50               undefined150h          [5]
       ram:001022ae 28               undefined128h          [6]
       ram:001022af 3f               undefined13Fh          [7]
       ram:001022b0 56               undefined156h          [8]
       ram:001022b1 16               undefined116h          [9]
       ram:001022b2 5f               undefined15Fh          [10]
       ram:001022b3 00               undefined100h          [11]
       ram:001022b4 55               undefined155h          [12]
       ram:001022b5 10               undefined110h          [13]
       ram:001022b6 56               undefined156h          [14]
       ram:001022b7 1a               undefined11Ah          [15]

                          k                                   XREF[1]:     Entry Point(*)
    ram:001022b8 62 65 6e        ds          "bengbangbongbeng"
                 67 62 61
                 6e 67 62 …
                          //
                          // .bss
                          // SHT_NOBITS  [0x22c9 - 0x2378]
                          // ram:001022c9-ram:00102378
                          //

                          exkey                               XREF[2]:     Entry Point(*),
                                                                           _elfSectionHeaders::00000150(*)
    ram:001022c9                 undefine…  ??
       ram:001022c9                  undefined1??           [0]                           XREF[2]:     Entry Point(*),
                                                                                                       _elfSectionHeaders::000001

       ram:001022ca                  undefined1??           [1]
       ram:001022cb                  undefined1??           [2]
       ram:001022cc                  undefined1??           [3]
       ram:001022cd                  undefined1??           [4]
       ram:001022ce                  undefined1??           [5]
       ram:001022cf                  undefined1??           [6]
       ram:001022d0                  undefined1??           [7]
       ram:001022d1                  undefined1??           [8]
       ram:001022d2                  undefined1??           [9]
       ram:001022d3                  undefined1??           [10]
       ram:001022d4                  undefined1??           [11]
       ram:001022d5                  undefined1??           [12]
       ram:001022d6                  undefined1??           [13]
       ram:001022d7                  undefined1??           [14]
       ram:001022d8                  undefined1??           [15]
       ram:001022d9                  undefined1??           [16]
       ram:001022da                  undefined1??           [17]
```



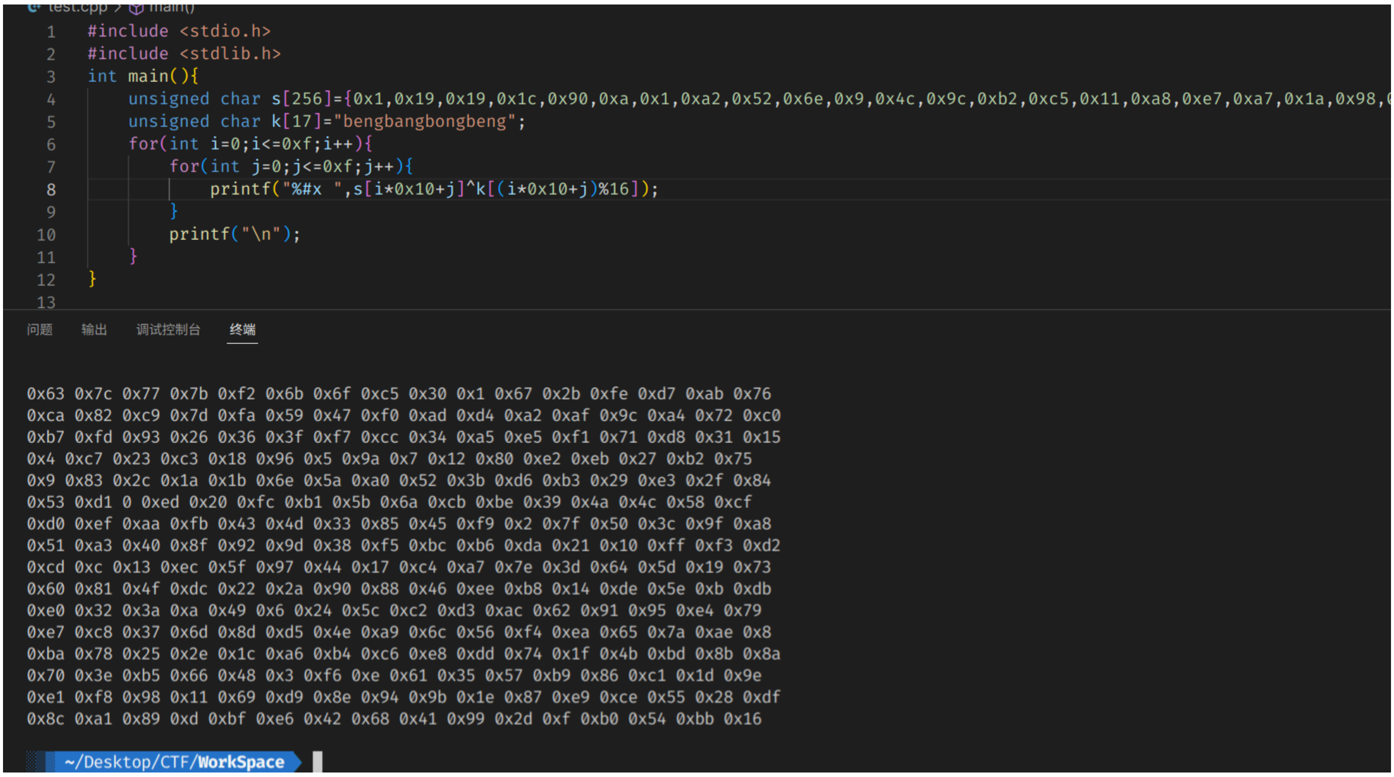
```
      ram:0010227f 71          undefined171h          [255]

                          r                                    XREF[1]:     Entry Point(*)
⊟   ram:00102280 63 00 00      undefine…
                 00 67 00
                 00 00 6a …
   ├── ram:00102280 63          undefined163h          [0]                    XREF[1]:     Entry P
   ├── ram:00102281 00          undefined100h          [1]
   ├── ram:00102282 00          undefined100h          [2]
   ├── ram:00102283 00          undefined100h          [3]
   ├── ram:00102284 67          undefined167h          [4]
   ├── ram:00102285 00          undefined100h          [5]
   ├── ram:00102286 00          undefined100h          [6]
   ├── ram:00102287 00          undefined100h          [7]
   ├── ram:00102288 6a          undefined16Ah          [8]
   ├── ram:00102289 00          undefined100h          [9]
   ├── ram:0010228a 00          undefined100h          [10]
   ├── ram:0010228b 00          undefined100h          [11]
   ├── ram:0010228c 6f          undefined16Fh          [12]
   ├── ram:0010228d 00          undefined100h          [13]
   ├── ram:0010228e 00          undefined100h          [14]
   ├── ram:0010228f 00          undefined100h          [15]
   ├── ram:00102290 72          undefined172h          [16]
   ├── ram:00102291 00          undefined100h          [17]
   ├── ram:00102292 00          undefined100h          [18]
   ├── ram:00102293 00          undefined100h          [19]
   ├── ram:00102294 41          undefined141h          [20]
   ├── ram:00102295 00          undefined100h          [21]
   ├── ram:00102296 00          undefined100h          [22]
   ├── ram:00102297 00          undefined100h          [23]
   ├── ram:00102298 2e          undefined12Eh          [24]
   ├── ram:00102299 00          undefined100h          [25]
   ├── ram:0010229a 00          undefined100h          [26]
   ├── ram:0010229b 00          undefined100h          [27]
   ├── ram:0010229c e7          undefined1E7h          [28]
   ├── ram:0010229d 00          undefined100h          [29]
   ├── ram:0010229e 00          undefined100h          [30]
   ├── ram:0010229f 00          undefined100h          [31]
   ├── ram:001022a0 79          undefined179h          [32]
   ├── ram:001022a1 00          undefined100h          [33]
   ├── ram:001022a2 00          undefined100h          [34]
   ├── ram:001022a3 00          undefined100h          [35]
   ├── ram:001022a4 59          undefined159h          [36]
   ├── ram:001022a5 00          undefined100h          [37]
   ├── ram:001022a6 00          undefined100h          [38]
   └── ram:001022a7 00          undefined100h          [39]
```



```
bepart
                          s                                    XREF[2]:     Entry Point(*),
                                                                            _elfSectionHeaders::00000
⊟  ram:00102180 01 19 19        undefine...
                1c 90 0a
                01 a2 52 ...
   ram:00102180 01              undefined101h               [0]                          XREF[2]:
   ram:00102181 19              undefined119h               [1]
   ram:00102182 19              undefined119h               [2]
   ram:00102183 1c              undefined11Ch               [3]
   ram:00102184 90              undefined190h               [4]
   ram:00102185 0a              undefined10Ah               [5]
   ram:00102186 01              undefined101h               [6]
   ram:00102187 a2              undefined1A2h               [7]
   ram:00102188 52              undefined152h               [8]
   ram:00102189 6e              undefined16Eh               [9]
   ram:0010218a 09              undefined109h               [10]
   ram:0010218b 4c              undefined14Ch               [11]
   ram:0010218c 9c              undefined19Ch               [12]
   ram:0010218d b2              undefined1B2h               [13]
   ram:0010218e c5              undefined1C5h               [14]
   ram:0010218f 11              undefined111h               [15]
   ram:00102190 a8              undefined1A8h               [16]
   ram:00102191 e7              undefined1E7h               [17]
   ram:00102192 a7              undefined1A7h               [18]
   ram:00102193 1a              undefined11Ah               [19]
   ram:00102194 98              undefined198h               [20]
   ram:00102195 38              undefined138h               [21]
   ram:00102196 29              undefined129h               [22]
   ram:00102197 97              undefined197h               [23]
   ram:00102198 cf              undefined1CFh               [24]
   ram:00102199 bb              undefined1BBh               [25]
   ram:0010219a cc              undefined1CCh               [26]
   ram:0010219b c8              undefined1C8h               [27]
   ram:0010219c fe              undefined1FEh               [28]
   ram:0010219d c1              undefined1C1h               [29]
   ram:0010219e 1c              undefined11Ch               [30]
   ram:0010219f a7              undefined1A7h               [31]
   ram:001021a0 d5              undefined1D5h               [32]
   ram:001021a1 98              undefined198h               [33]
   ram:001021a2 fd              undefined1FDh               [34]
   ram:001021a3 41              undefined141h               [35]
   ram:001021a4 54              undefined154h               [36]
   ram:001021a5 5e              undefined15Eh               [37]
   ram:001021a6 99              undefined199h               [38]
   ram:001021a7 ab              undefined1ABh               [39]
   ram:001021a8 56              undefined156h               [40]
   ram:001021a9 ca              undefined1CAh               [41]
   ram:001021aa 8b              undefined18Bh               [42]
   ram:001021ab 96              undefined196h               [43]
```

观察gentable函数，可以发现全局变量进行了大量异或操作，尝试将明文key与s数组相异或，得出AES的s-box

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    unsigned char s[256]={
0x1,0x19,0x19,0x1c,0x90,0xa,0x1,0xa2,0x52,0x6e,0x9,0x4c,0x9c,0xb2,0xc5,0x11,
0xa8,0xe7,0xa7,0x1a,0x98,0x38,0x29,0x97,0xcf,0xbb,0xcc,0xc8,0xfe,0xc1,0x1c,0xa7,
0xd5,0x98,0xfd,0x41,0x54,0x5e,0x99,0xab,0x56,0xca,0x8b,0x96,0x13,0xbd,0x5f,0x72,
0x66,0xa2,0x4d,0xa4,0x7a,0xf7,0x6b,0xfd,0x65,0x7d,0xee,0x85,0x89,0x42,0xdc,0x12,
0x6b,0xe6,0x42,0x7d,0x79,0xf,0x34,0xc7,0x30,0x54,0xb8,0xd4,0x4b,0x86,0x41,0xe3,
0x31,0xb4,0x6e,0x8a,0x42,0x9d,0xdf,0x3c,0x8,0xa4,0xd0,0x5e,0x28,0x29,0x36,0xa8,
0xb2,0x8a,0xc4,0x9c,0x21,0x2c,0x5d,0xe2,0x27,0x96,0x6c,0x18,0x32,0x59,0xf1,0xcf,
0x33,0xc6,0x2e,0xe8,0xf0,0xfc,0x56,0x92,0xde,0xd9,0xb4,0x46,0x72,0x9a,0x9d,0xb5,
0xaf,0x69,0x7d,0x8b,0x3d,0xf6,0x2a,0x70,0xa6,0xc8,0x10,0x5a,0x6,0x38,0x77,0x14,
0x2,0xe4,0x21,0xbb,0x40,0x4b,0xfe,0xef,0x24,0x81,0xd6,0x73,0xbc,0x3b,0x65,0xbc,
0x82,0x57,0x54,0x6d,0x2b,0x67,0x4a,0x3b,0xa0,0xbc,0xc2,0x5,0xf3,0xf0,0x8a,0x1e,
0x85,0xad,0x59,0xa,0xef,0xb4,0x20,0xce,0xe,0x39,0x9a,0x8d,0x7,0x1f,0xc0,0x6f,
0xd8,0x1d,0x4b,0x49,0x7e,0xc7,0xda,0xa1,0x8a,0xb2,0x1a,0x78,0x29,0xd8,0xe5,0xed,
0x12,0x5b,0xdb,0x1,0x2a,0x62,0x98,0x69,0x3,0x5a,0x39,0xde,0xe4,0xa4,0x73,0xf9,
0x83,0x9d,0xf6,0x76,0xb,0xb8,0xe0,0xf3,0xf9,0x71,0xe9,0x8e,0xac,0x30,0x46,0xb8,
0xee,0xc4,0xe7,0x6a,0xdd,0x87,0x2c,0xf,0x23,0xf6,0x43,0x68,0xd2,0x31,0xd5,0x71
};
    unsigned char k[17]="bengbangbongbeng";
    for(int i=0;i<=0xf;i++){
        for(int j=0;j<=0xf;j++){
            printf("%#x ",s[i*0x10+j]^k[(i*0x10+j)%16]);
        }
        printf("\n");
    }
}
```



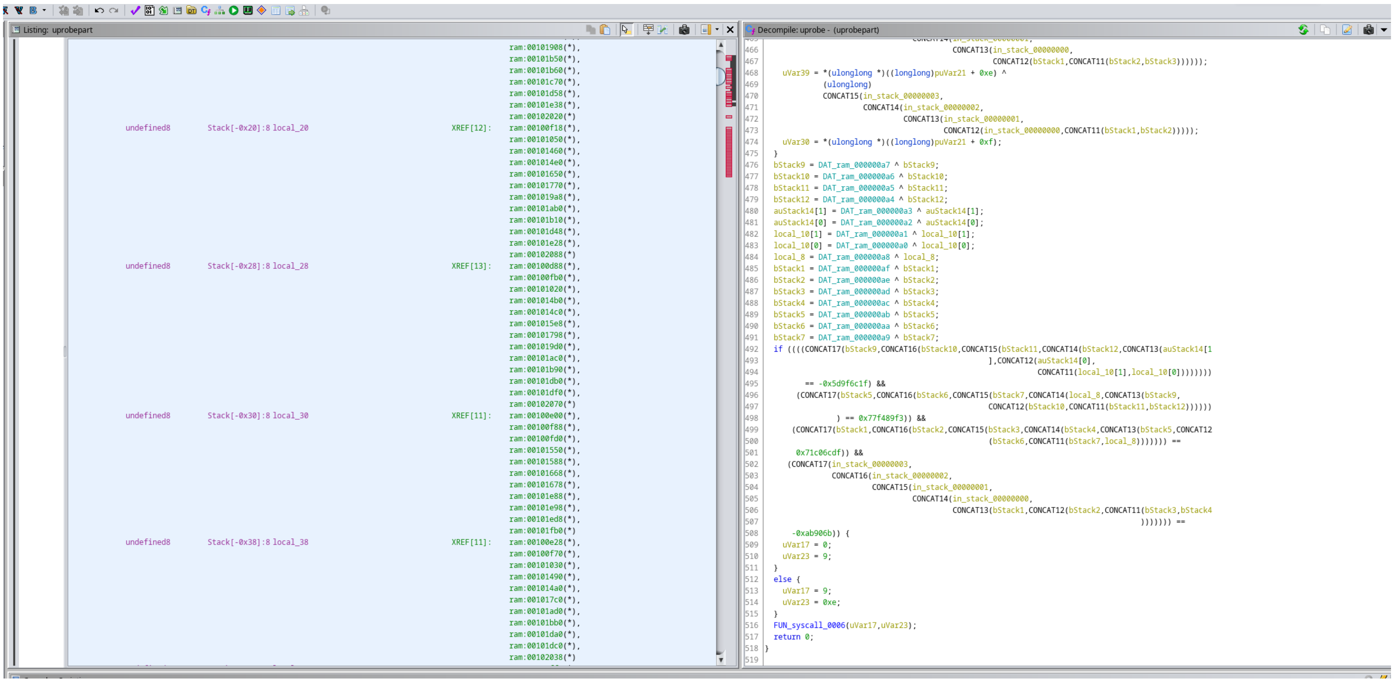
```
#include <stdio.h>
#include <stdlib.h>
int main(){
    unsigned char s[256]={0x1,0x19,0x19,0x1c,0x90,0xa,0x1,0xa2,0x52,0x6e,0x9,0x4c,0x9c,0xb2,0xc5,0x11,0xa8,0xe7,0xa7,0x1a,0x98,
    unsigned char k[17]="bengbangbongbeng";
    for(int i=0;i<=0xf;i++){
        for(int j=0;j<=0xf;j++){
            printf("%#x ",s[i*0x10+j]^k[(i*0x10+j)%16]);
        }
        printf("\n");
    }
}
```

```
0x63 0x7c 0x77 0x7b 0xf2 0x6b 0x6f 0xc5 0x30 0x1 0x67 0x2b 0xfe 0xd7 0xab 0x76
0xca 0x82 0xc9 0x7d 0xfa 0x59 0x47 0xf0 0xad 0xd4 0xa2 0xaf 0x9c 0xa4 0x72 0xc0
0xb7 0xfd 0x93 0x26 0x36 0x3f 0xf7 0xcc 0x34 0xa5 0xe5 0xf1 0x71 0xd8 0x31 0x15
0x4 0xc7 0x23 0xc3 0x18 0x96 0x5 0x9a 0x7 0x12 0x80 0xe2 0xeb 0x27 0xb2 0x75
0x9 0x83 0x2c 0x1a 0x1b 0x6e 0x5a 0xa0 0x52 0x3b 0xd6 0xb3 0x29 0xe3 0x2f 0x84
0x53 0xd1 0 0xed 0x20 0xfc 0xb1 0x5b 0x6a 0xcb 0xbe 0x39 0x4a 0x4c 0x58 0xcf
0xd0 0xef 0xaa 0xfb 0x43 0x4d 0x33 0x85 0x45 0xf9 0x2 0x7f 0x50 0x3c 0x9f 0xa8
0x51 0xa3 0x40 0x8f 0x92 0x9d 0x38 0xf5 0xbc 0xb6 0xda 0x21 0x10 0xff 0xf3 0xd2
0xcd 0xc 0x13 0xec 0x5f 0x97 0x44 0x17 0xc4 0xa7 0x7e 0x3d 0x64 0x5d 0x19 0x73
0x60 0x81 0x4f 0xdc 0x22 0x2a 0x90 0x88 0x46 0xee 0xb8 0x14 0xde 0x5e 0xb 0xdb
0xe0 0x32 0x3a 0xa 0x49 0x6 0x24 0x5c 0xc2 0xd3 0xac 0x62 0x91 0x95 0xe4 0x79
0xe7 0xc8 0x37 0x6d 0x8d 0xd5 0x4e 0xa9 0x6c 0x56 0xf4 0xea 0x65 0x7a 0xae 0x8
0xba 0x78 0x25 0x2e 0x1c 0xa6 0xb4 0xc6 0xe8 0xdd 0x74 0x1f 0x4b 0xbd 0x8b 0x8a
0x70 0x3e 0xb5 0x66 0x48 0x3 0xf6 0xe 0x61 0x35 0x57 0xb9 0x86 0xc1 0x1d 0x9e
0xe1 0xf8 0x98 0x11 0x69 0xd9 0x8e 0x94 0x9b 0x1e 0x87 0xe9 0xce 0x55 0x28 0xdf
0x8c 0xa1 0x89 0xd 0xbf 0xe6 0x42 0x68 0x41 0x99 0x2d 0xf 0xb0 0x54 0xbb 0x16

~/Desktop/CTF/WorkSpace
```

同理可得r为扩展密钥用的rcon数组，所以key为AES的key、exkey为扩展密钥、k为加密全局变量的变量、s为s-box、r为rcon。

> AES也可以通过其他函数的算法特征识别出来，比如func5中的异或0x1b

查看主函数uprobe，在最后的判断中进行了数据比对

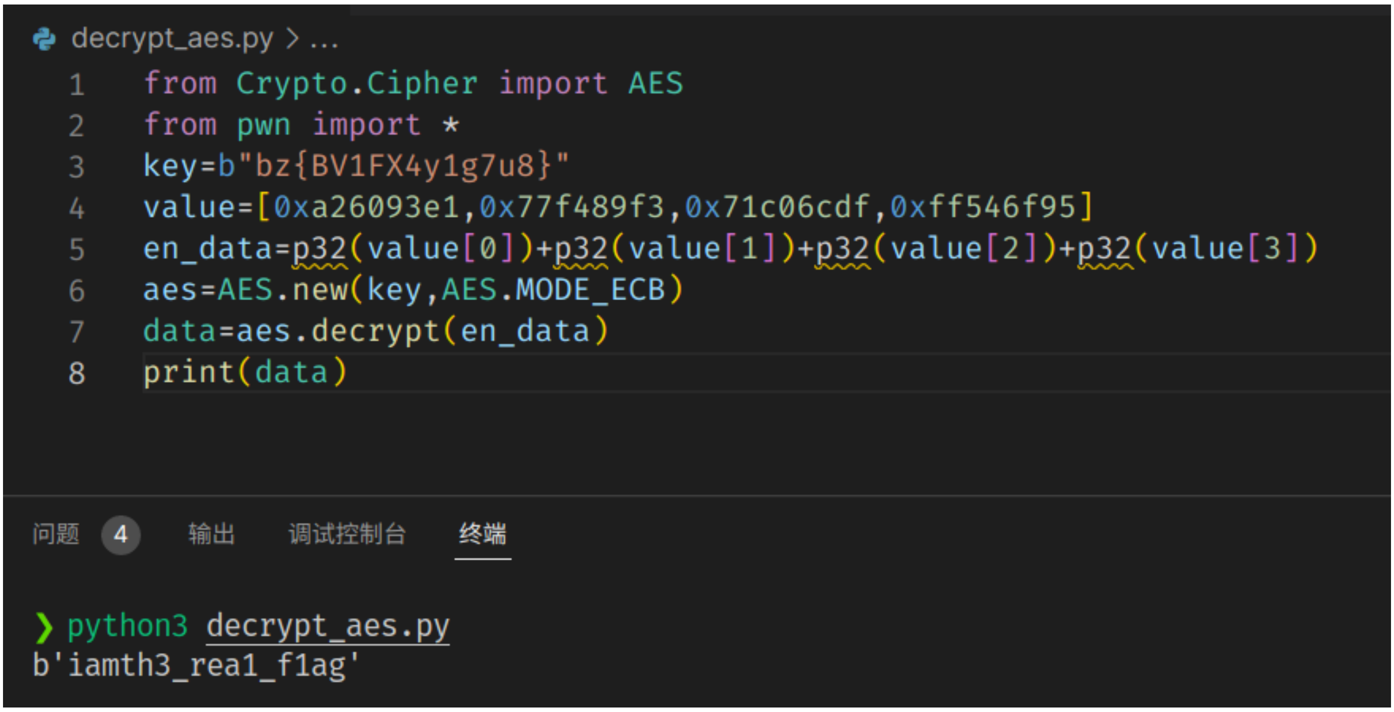


```
                                                          ram:00101908(*),
                                                          ram:00101b50(*),
                                                          ram:00101b60(*),
                                                          ram:00101c70(*),
                                                          ram:00101d58(*),
                                                          ram:00101e38(*),
                                                          ram:00102020(*)
     undefined8     Stack[-0x20]:8 local_20      XREF[12]: ram:00100f18(*),
                                                          ram:00101050(*),
                                                          ram:00101460(*),
                                                          ram:001014e0(*),
                                                          ram:00101650(*),
                                                          ram:00101770(*),
                                                          ram:001019a8(*),
                                                          ram:00101ab0(*),
                                                          ram:00101b10(*),
                                                          ram:00101d48(*),
                                                          ram:00101e28(*),
                                                          ram:00102088(*)
     undefined8     Stack[-0x28]:8 local_28      XREF[13]: ram:00100d88(*),
                                                          ram:00100fb0(*),
                                                          ram:00101020(*),
                                                          ram:001014b0(*),
                                                          ram:001014c0(*),
                                                          ram:001015e8(*),
                                                          ram:00101798(*),
                                                          ram:001019d0(*),
                                                          ram:00101ac0(*),
                                                          ram:00101b90(*),
                                                          ram:00101db0(*),
                                                          ram:00101df0(*),
                                                          ram:00102070(*)
     undefined8     Stack[-0x30]:8 local_30      XREF[11]: ram:00100e00(*),
                                                          ram:00100f88(*),
                                                          ram:00100fd0(*),
                                                          ram:00101550(*),
                                                          ram:00101588(*),
                                                          ram:00101668(*),
                                                          ram:00101678(*),
                                                          ram:00101e88(*),
                                                          ram:00101e98(*),
                                                          ram:00101ed8(*),
                                                          ram:00101fb0(*)
     undefined8     Stack[-0x38]:8 local_38      XREF[11]: ram:00100e28(*),
                                                          ram:00100f70(*),
                                                          ram:00101030(*),
                                                          ram:00101490(*),
                                                          ram:001014a0(*),
                                                          ram:001017c0(*),
                                                          ram:00101ad0(*),
                                                          ram:00101bb0(*),
                                                          ram:00101da0(*),
                                                          ram:00101dc0(*),
                                                          ram:00102038(*)
```

```c
                              CONCAT14(in_stack_00000001,
                                       CONCAT13(in_stack_00000000,
                                                CONCAT12(bStack1,CONCAT11(bStack2,bStack3))))));
    uVar39 = *(ulonglong *)((longlong)puVar21 + 0xe) ^
             (ulonglong)
             CONCAT15(in_stack_00000003,
                      CONCAT14(in_stack_00000002,
                               CONCAT13(in_stack_00000001,
                                        CONCAT12(in_stack_00000000,CONCAT11(bStack1,bStack2)))));
    uVar30 = *(ulonglong *)((longlong)puVar21 + 0xf);
  }
  bStack9 = DAT_ram_000000a7 ^ bStack9;
  bStack10 = DAT_ram_000000a6 ^ bStack10;
  bStack11 = DAT_ram_000000a5 ^ bStack11;
  bStack12 = DAT_ram_000000a4 ^ bStack12;
  auStack14[1] = DAT_ram_000000a3 ^ auStack14[1];
  auStack14[0] = DAT_ram_000000a2 ^ auStack14[0];
  local_10[1] = DAT_ram_000000a1 ^ local_10[1];
  local_10[0] = DAT_ram_000000a0 ^ local_10[0];
  local_8 = DAT_ram_000000a8 ^ local_8;
  bStack1 = DAT_ram_000000af ^ bStack1;
  bStack2 = DAT_ram_000000ae ^ bStack2;
  bStack3 = DAT_ram_000000ad ^ bStack3;
  bStack4 = DAT_ram_000000ac ^ bStack4;
  bStack5 = DAT_ram_000000ab ^ bStack5;
  bStack6 = DAT_ram_000000aa ^ bStack6;
  bStack7 = DAT_ram_000000a9 ^ bStack7;
  if ((((CONCAT17(bStack9,CONCAT16(bStack10,CONCAT15(bStack11,CONCAT14(bStack12,CONCAT13(auStack14[1
                                                ],CONCAT12(auStack14[0],
                                                           CONCAT11(local_10[1],local_10[0])))))))
        == -0x5d9f6c1f) &&
      (CONCAT17(bStack5,CONCAT16(bStack6,CONCAT15(bStack7,CONCAT14(local_8,CONCAT13(bStack9,
                                                CONCAT12(bStack10,CONCAT11(bStack11,bStack12))))))
              ) == 0x77f489f3)) &&
     (CONCAT17(bStack1,CONCAT16(bStack2,CONCAT15(bStack3,CONCAT14(bStack4,CONCAT13(bStack5,CONCAT12
                                                (bStack6,CONCAT11(bStack7,local_8))))))) ==
      0x71c06cdf)) &&
    (CONCAT17(in_stack_00000003,
              CONCAT16(in_stack_00000002,
                       CONCAT15(in_stack_00000001,
                                CONCAT14(in_stack_00000000,
                                         CONCAT13(bStack1,CONCAT12(bStack2,CONCAT11(bStack3,bStack4
                                                                                   ))))))) ==
     -0xab906b)) {
    uVar17 = 0;
    uVar23 = 9;
  }
  else {
    uVar17 = 9;
    uVar23 = 0xe;
  }
  FUN_syscall_0006(uVar17,uVar23);
  return 0;
}
```

提取出数据，异或解密出密钥，用python求解AES即可获得正确的flag

```python
from Crypto.Cipher import AES
from pwn import *
key=b"bz{BV1FX4y1g7u8}"
value=[0xa26093e1,0x77f489f3,0x71c06cdf,0xff546f95]
en_data=p32(value[0])+p32(value[1])+p32(value[2])+p32(value[3])
aes=AES.new(key,AES.MODE_ECB)
data=aes.decrypt(en_data)
print(data)
```



```python
from Crypto.Cipher import AES
from pwn import *
key=b"bz{BV1FX4y1g7u8}"
value=[0xa26093e1,0x77f489f3,0x71c06cdf,0xff546f95]
en_data=p32(value[0])+p32(value[1])+p32(value[2])+p32(value[3])
aes=AES.new(key,AES.MODE_ECB)
data=aes.decrypt(en_data)
print(data)
```

```
❯ python3 decrypt_aes.py
b'iamth3_rea1_f1ag'
```