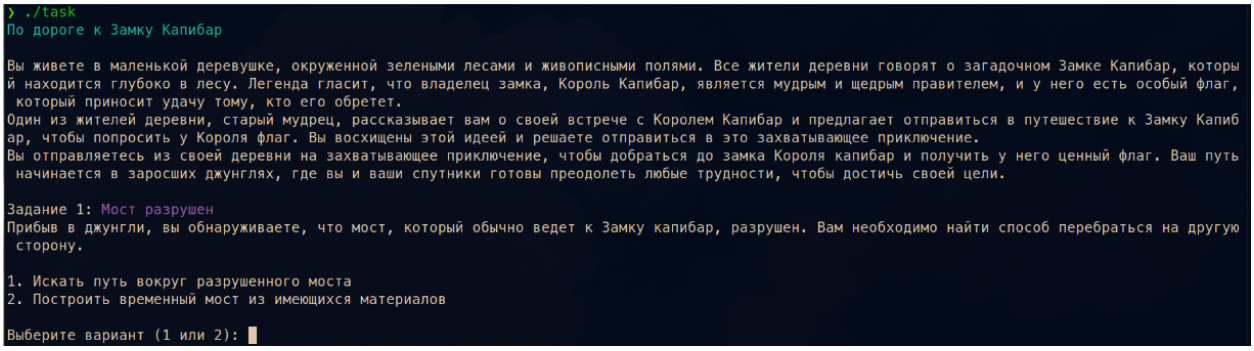
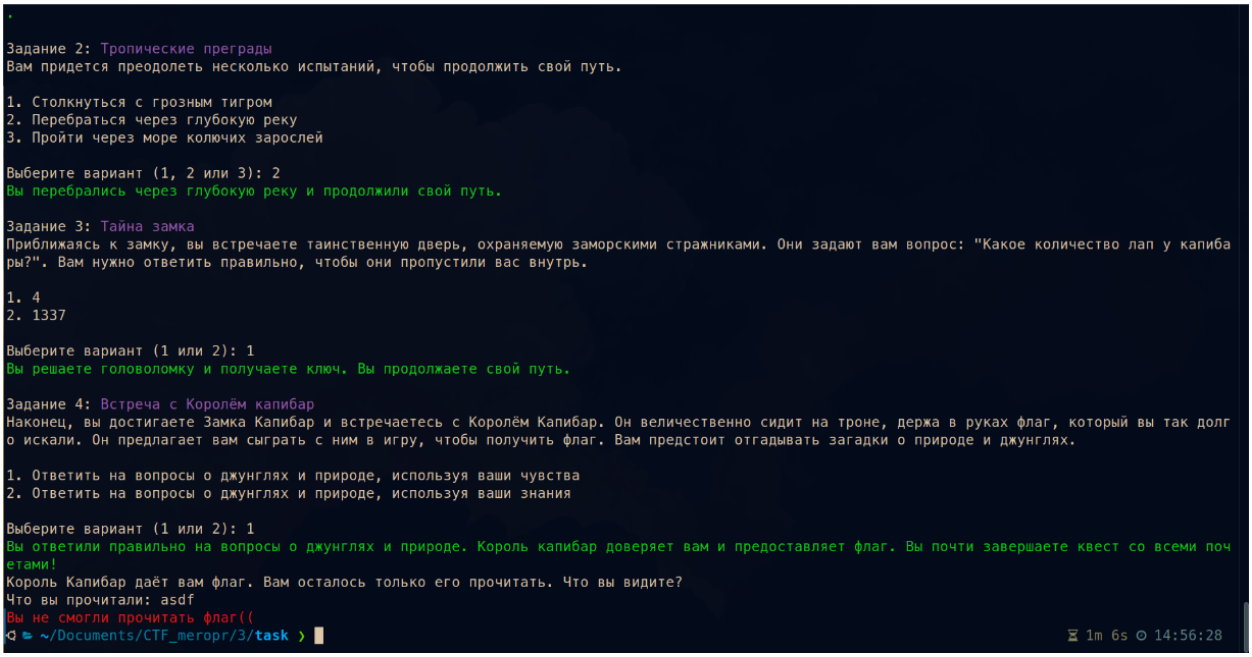
Название:	По дороге к Замку Капибар
Категория:	Реверс-инжиниринг
Уровень:	Средний
Очки:	700
Описание:	На пути к флагу!
Теги:	C
Автор:	ROP

Прохождение:

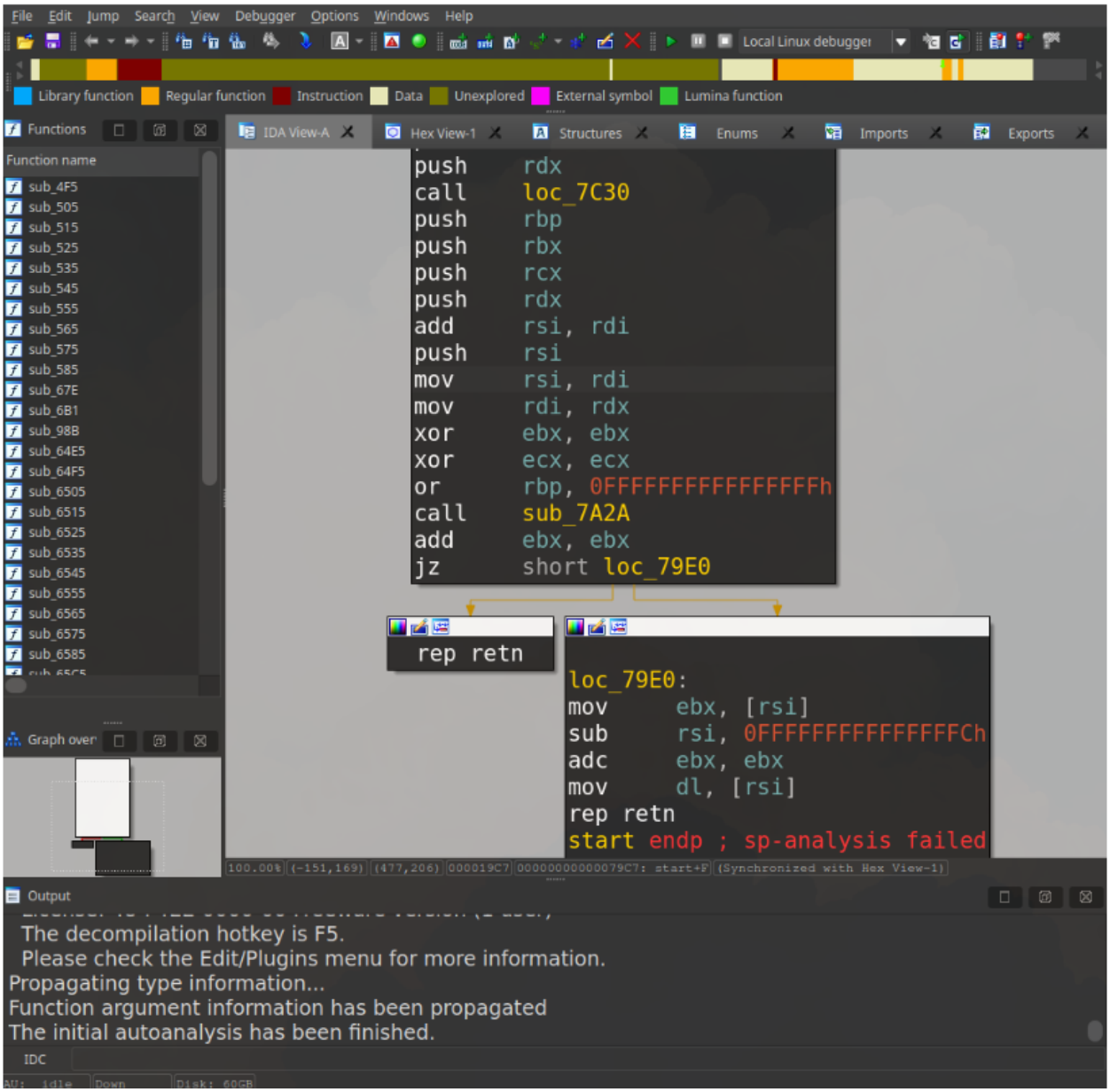
Попробуем запустить файл.



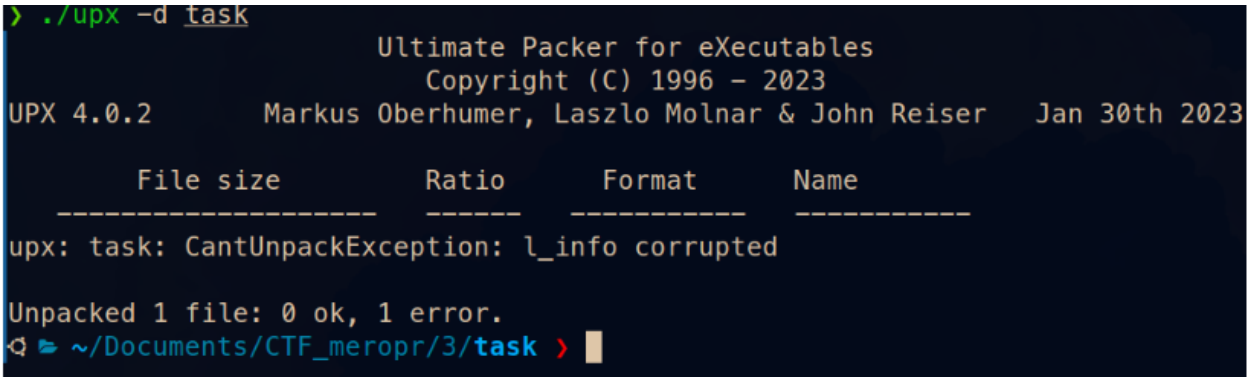
Какой-то квест. Попробуем пройти.



Чтение флага… Посмотрим в IDA.



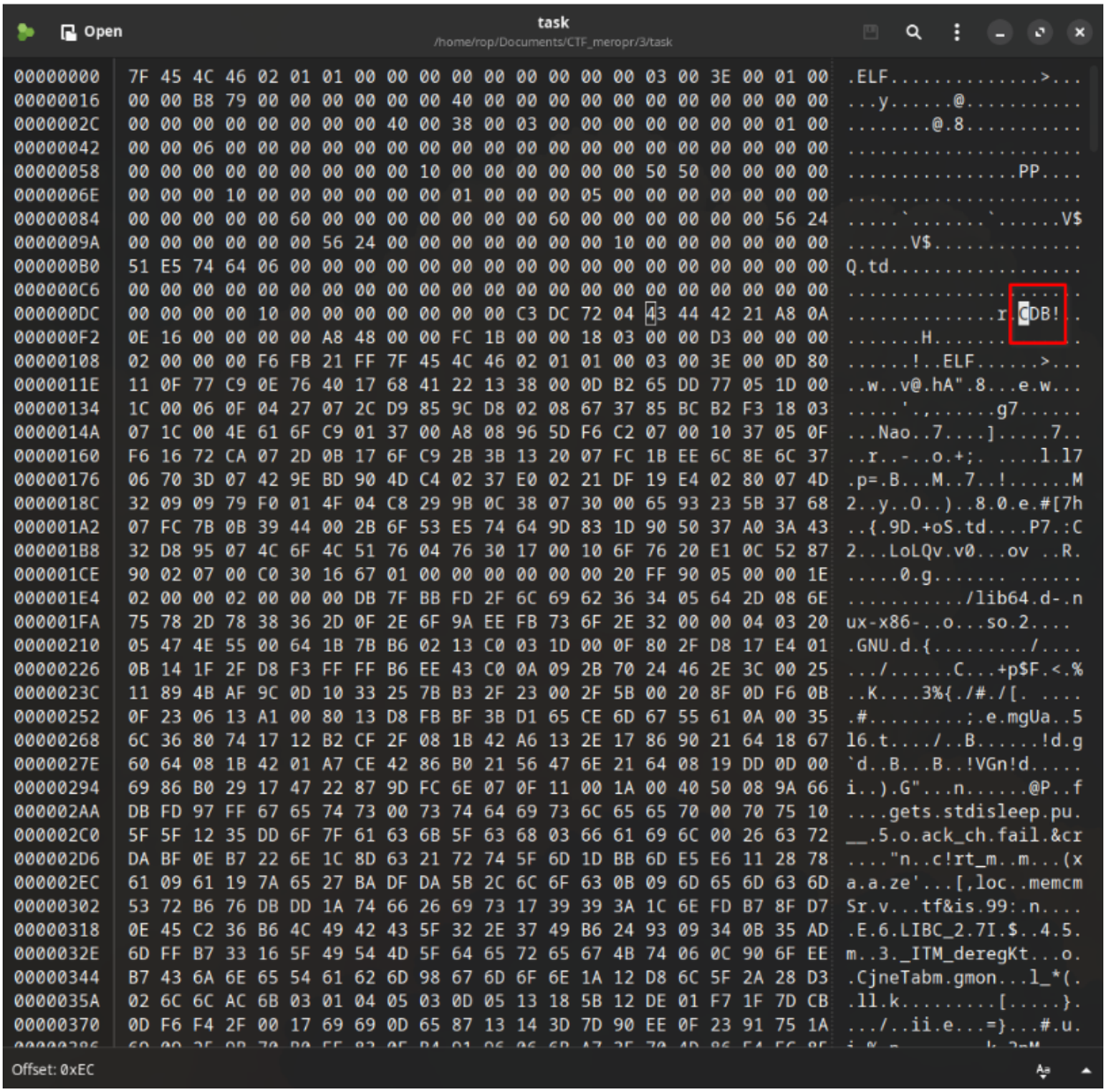
Вот это говорит о том, что файл запакован через UPX.



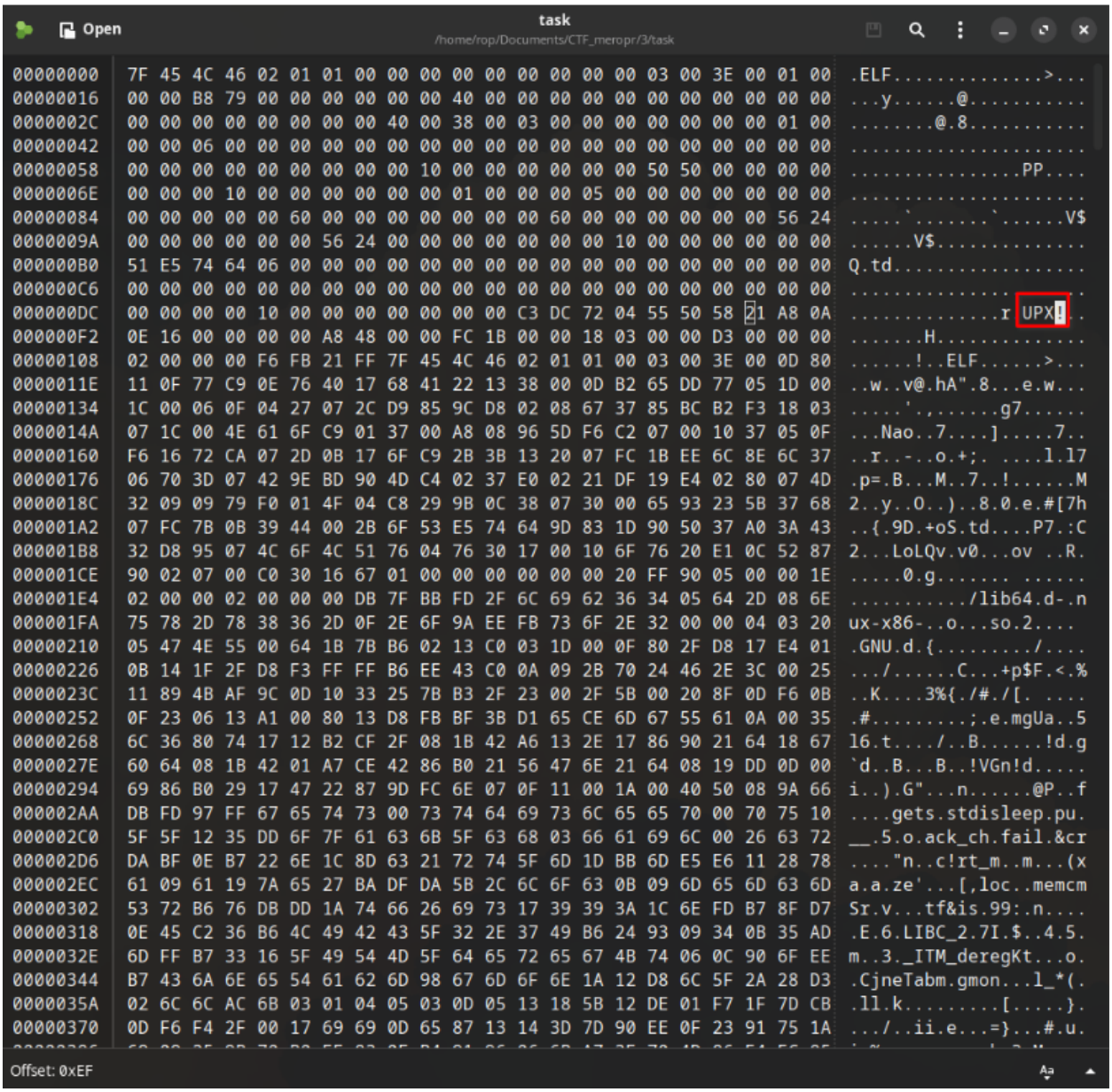
Попробуем распаковать через upx -d task .Ошибка. Если игрок знает структуру UPX, то зайдёт в HEX-

редактор и поменяет подменённый magic UPX'a на UPX! .

Было.


Open
task
/home/rop/Documents/CTF_meropr/3/task
00000000 7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00 03 00 3E 00 01 00 .ELF.....>...
00000016 00 00 B8 79 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 ...y......@.....
0000002C 00 00 00 00 00 00 00 00 40 00 38 00 03 00 00 00 00 00 00 00 01 00 ....@.8.....
00000042 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000058 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 50 50 00 00 00 00 ....PP....
0000006E 00 00 00 10 00 00 00 00 00 00 01 00 00 00 05 00 00 00 00 00 00 00 ....
00000084 00 00 00 00 00 60 00 00 00 00 00 00 00 00 60 00 00 00 00 00 00 56 24 .....`.....`.....V$
0000009A 00 00 00 00 00 00 56 24 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 ......V$..........
000000B0 51 E5 74 64 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Q.td..........
000000C6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000000DC 00 00 00 00 10 00 00 00 00 00 00 00 C3 DC 72 04 43 44 42 21 A8 0A ....r.CDB!..
000000F2 0E 16 00 00 00 00 A8 48 00 00 FC 1B 00 00 18 03 00 00 D3 00 00 00 ....H........
00000108 02 00 00 00 F6 FB 21 FF 7F 45 4C 46 02 01 01 00 03 00 3E 00 0D 80 ......!..ELF......>...
0000011E 11 0F 77 C9 0E 76 40 17 68 41 22 13 38 00 0D B2 65 DD 77 05 1D 00 ..w..v@.hA".8...e.w...
00000134 1C 00 06 0F 04 27 07 2C D9 85 9C D8 02 08 67 37 85 BC B2 F3 18 03 .....',.....g7.....
0000014A 07 1C 00 4E 61 6F C9 01 37 00 A8 08 96 5D F6 C2 07 00 10 37 05 0F ...Nao..7....]....7..
00000160 F6 16 72 CA 07 2D 0B 17 6F C9 2B 3B 13 20 07 FC 1B EE 6C 8E 6C 37 ..r..-..o.+;. l.l7
00000176 06 70 3D 07 42 9E BD 90 4D C4 02 37 E0 02 21 DF 19 E4 02 80 07 4D .p=.B...M..7..!.....M
0000018C 32 09 09 79 F0 01 4F 04 C8 29 9B 0C 38 07 30 00 65 93 23 5B 37 68 2..y..O..)..8.0.e.#[7h
000001A2 07 FC 7B 0B 39 44 00 2B 6F 53 E5 74 64 9D 83 1D 90 50 37 A0 3A 43 ..{.9D.+oS.td....P7.:C
000001B8 32 D8 95 07 4C 6F 4C 51 76 04 76 30 17 00 10 6F 76 20 E1 0C 52 87 2...LoLQv.v0...ov ..R.
000001CE 90 02 07 00 C0 30 16 67 01 00 00 00 00 00 00 20 FF 90 05 00 00 1E .....0.g....... .....
000001E4 02 00 00 02 00 00 00 DB 7F BB FD 2F 6C 69 62 36 34 05 64 2D 08 6E lib64.d-.n
000001FA 75 78 2D 78 38 36 2D 0F 2E 6F 9A EE FB 73 6F 2E 32 00 00 04 03 20 ux-x86-..o...so.2....
00000210 05 47 4E 55 00 64 1B 7B B6 02 13 C0 03 1D 00 0F 80 2F D8 17 E4 01 .GNU.d.{........./....
00000226 0B 14 1F 2F D8 F3 FF FF B6 EE 43 C0 0A 09 2B 70 24 46 2E 3C 00 25 .../......C...+p$F.<.%
0000023C 11 89 4B AF 9C 0D 10 33 25 7B B3 2F 23 00 2F 5B 00 20 8F 0D F6 0B ..K....3%{./#./[. ....
00000252 0F 23 06 13 A1 00 80 13 D8 FB BF 3B D1 65 CE 6D 67 55 61 0A 00 35 .#.........;.e.mgUa..5
00000268 6C 36 80 74 17 12 B2 CF 2F 08 1B 42 A6 13 2E 17 86 90 21 64 18 67 l6.t..../..B......!d.g
0000027E 60 64 08 1B 42 01 A7 CE 42 86 B0 21 56 47 6E 21 64 08 19 DD 0D 00 `d..B...B..!VGn!d.....
00000294 69 86 B0 29 17 47 22 87 9D FC 6E 07 0F 11 00 1A 00 40 50 08 9A 66 i..).G"...n......@P..f
000002AA DB FD 97 FF 67 65 74 73 00 73 74 64 69 73 6C 65 65 70 00 70 75 10 ....gets.stdisleep.pu.
000002C0 5F 5F 12 35 DD 6F 7F 61 63 6B 5F 63 68 03 66 61 69 6C 00 26 63 72 __.5.o.ack_ch.fail.&cr
000002D6 DA BF 0E B7 22 6E 1C 8D 63 21 72 74 5F 6D 1D BB 6D E5 E6 11 28 78 ....."n..c!rt_m..m...(x
000002EC 61 09 61 19 7A 65 27 BA DF DA 5B 2C 6C 6F 63 0B 09 6D 65 6D 63 6D a.a.ze'...[,loc..memcm
00000302 53 72 B6 76 DB DD 1A 74 66 26 69 73 17 39 39 3A 1C 6E FD B7 8F D7 Sr.v...tf&is.99:.n....
00000318 0E 45 C2 36 B6 4C 49 42 43 5F 32 2E 37 49 B6 24 93 09 34 0B 35 AD .E.6.LIBC_2.7I.$..4.5.
0000032E 6D FF B7 33 16 5F 49 54 4D 5F 64 65 72 65 67 4B 74 06 0C 90 6F EE m..3._ITM_deregKt...o.
00000344 B7 43 6A 6E 65 54 61 62 6D 98 67 6D 6F 6E 1A 12 D8 6C 5F 2A 28 D3 .CjneTabm.gmon...l_*(.
0000035A 02 6C 6C AC 6B 03 01 04 05 03 0D 05 13 18 5B 12 DE 01 F7 1F 7D CB .ll.k.........[.....}.
00000370 0D F6 F4 2F 00 17 69 69 0D 65 87 13 14 3D 7D 90 EE 0F 23 91 75 1A .../..ii.e...=}...#.u.
Offset: 0xEC

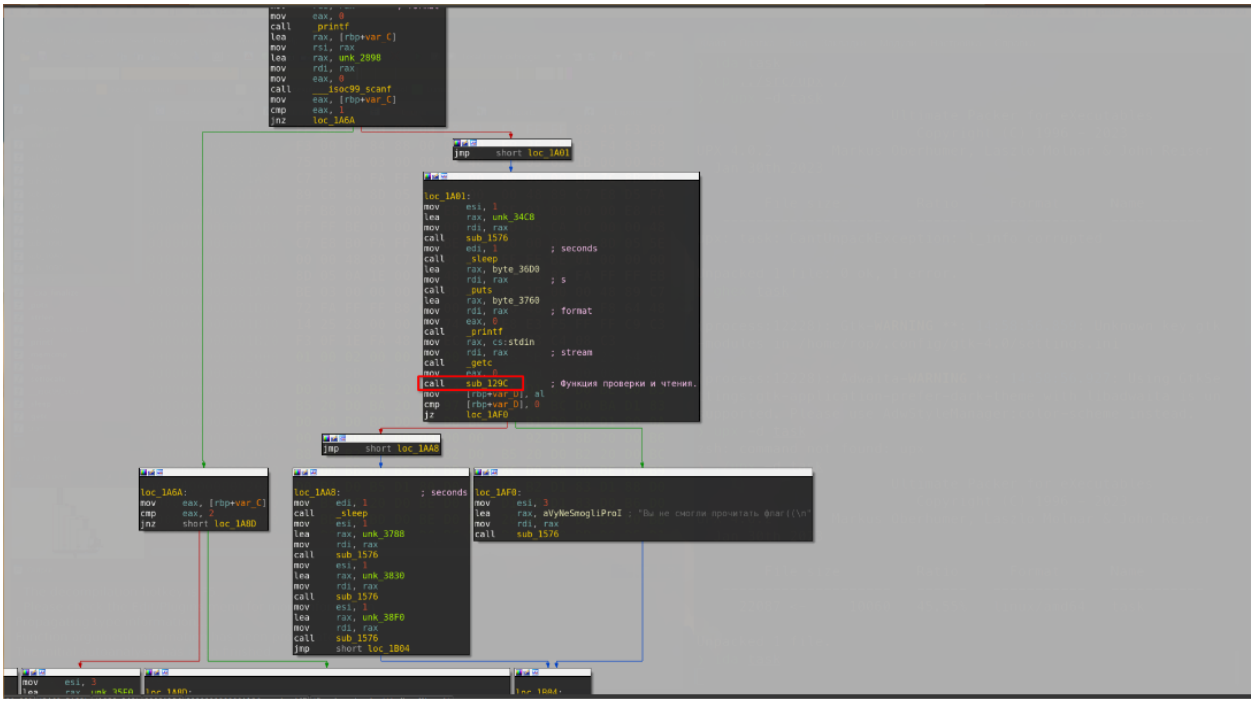
Стало.



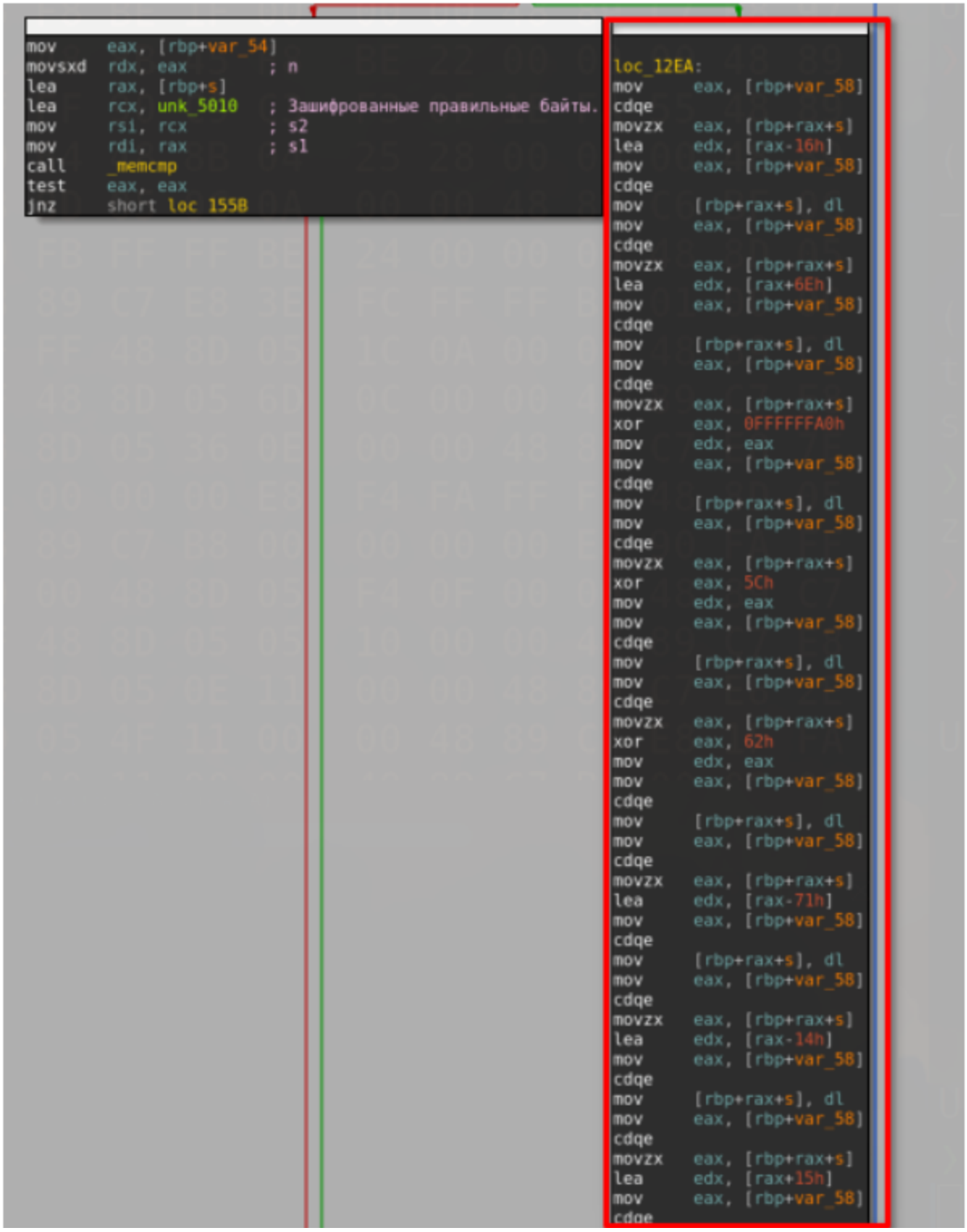
Теперь можно распаковать.

```
❯ ./upx -d task
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.0.2        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 30th 2023

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     22087 <-     10060   45.55%   linux/amd64   task

Unpacked 1 file.
 ~/Documents/CTF_meropr/3/task ❯
```

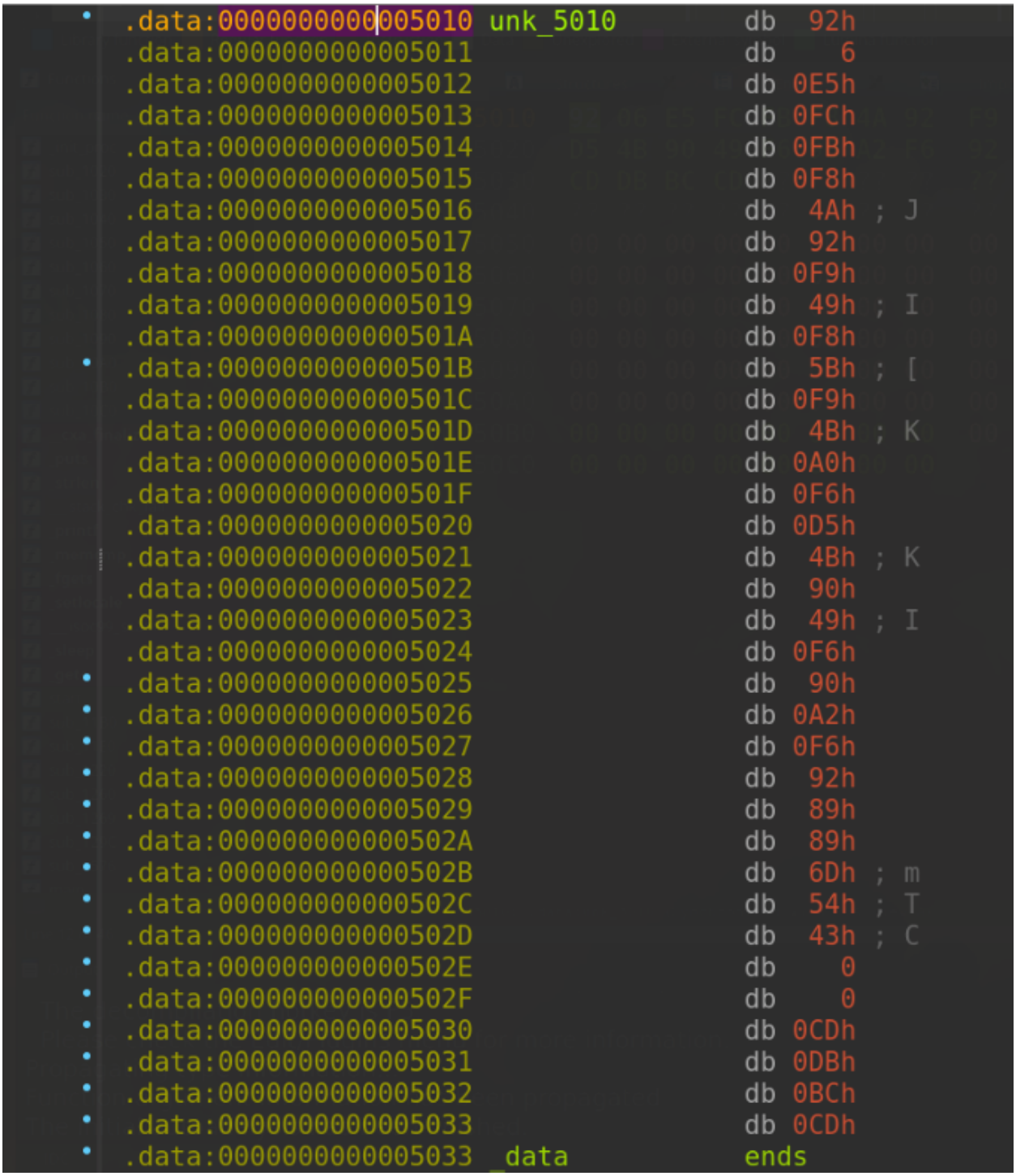
И уже в IDA можно увидеть исходный код.



Зайдём в функцию.


mov eax, [rbp+var_54]
movsxd rdx, eax ; n
lea rax, [rbp+s]
lea rcx, unk_5010 ; Зашифрованные правильные байты.
mov rsi, rcx ; s2
mov rdi, rax ; s1
call _memcmp
test eax, eax
jnz short loc_155B
loc_12EA:
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
lea edx, [rax-16h]
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
lea edx, [rax+6Eh]
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
xor eax, 0FFFFFFA0h
mov edx, eax
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
xor eax, 5Ch
mov edx, eax
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
xor eax, 62h
mov edx, eax
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
lea edx, [rax-71h]
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
lea edx, [rax-14h]
mov eax, [rbp+var_58]
cdqe
mov [rbp+rax+s], dl
mov eax, [rbp+var_58]
cdqe
movzx eax, [rbp+rax+s]
lea edx, [rax+15h]
mov eax, [rbp+var_58]
cdqe

Видим алгоритм проверки и зашифрованные правильные байты (`unk_5010`).



Напишем дешифратор по алгоритму выше.

```
#include <stdio.h>

unsigned char check_pass[] = {
  146,
  6,
  229,
  252,
  251,
  248,
  74,
  146,
  249,
  73,
  248,
  91,
  249,
  75,
  160,
  246,
  213,
  75,
  144,
  73,
  246,
  144,
  162,
  246,
  146,
  137,
  137,
  109,
  84,
  67,
  0,
  0,
```

```
  205,
  219,
  188,
  205
};

int main() {

    int len_check = sizeof(check_pass) / sizeof(check_pass[0]);

    unsigned char i, j;

    char flag[50] = {0};

    for (j = 0; j < len_check; j++) {
      for (i = 0; i < 0x80; i++) {
        unsigned char temp = i;
        i -= 22;
        i += 110;
        i ^= 160;
        i ^= 92;
        i ^= 98;
        i += 143;
        i -= 20;
        i += 21;
        i -= 73;
        i ^= 76;
        i ^= 151;
        i ^= 56;
        i += 102;
        i -= 203;
        i += 86;
        i += 157;
        i -= 163;
        i += 198;
        i ^= 233;
```

```
        i -= 147;
        i += 28;
        i ^= 212;
        i -= 159;
        i ^= 252;
        i ^= 73;

        if (i == check_pass[j])
        {
          flag[j] = temp;
          break;
        }

        i = temp;
      }
    }

    printf("%s\n", flag);
}
```

Запускаем и получаем флаг!