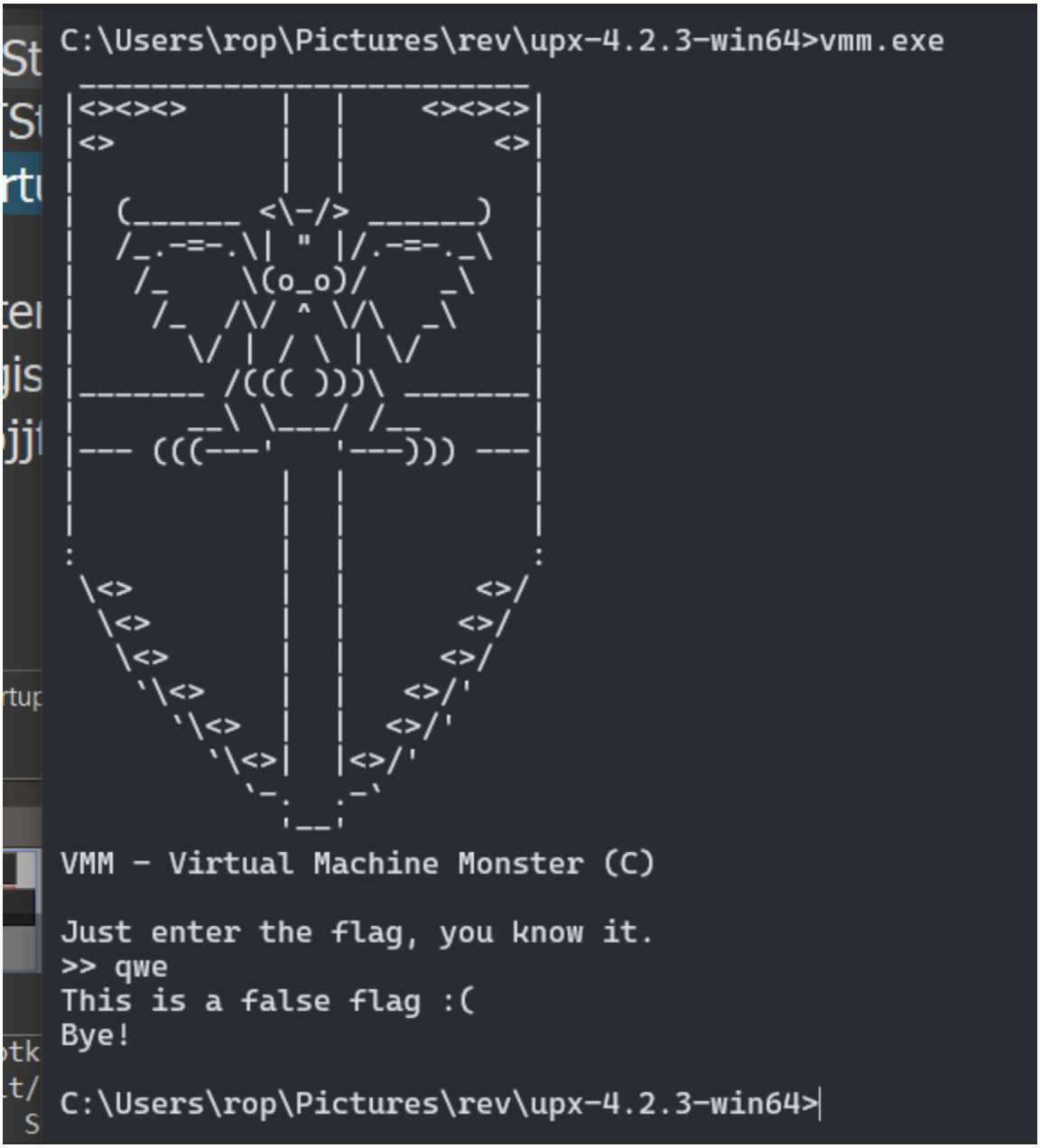
Название:	`VMM`
Категория:	`Реверс-инжиниринг`
Уровень:	`Сложный`
Очки:	`1000`
Описание:	`Эту виртуальную машину я купил у одного знакомого... Сможешь ли ты теперь получить флаг? :)`
Теги:	`Самописная виртуальная машина, Обфускация, Ложный код, Полиморфный код, Пропатченная системная функция, "Скрытая" функция, Побайтовое шифрование, UPX`
Автор:	`ROP`

Прохождение:

“Фичи” таска:

- Самописная виртуальная машина
- Обфускация
- Ложный код (для запутывания)
- Полиморфный код
- Пропатченная системная функция
- "Скрытая" функция для изменения ключей в xor, add, sub при наличии отладчика
- Побайтовое шифрование ключа через ROT13, XOR, ADD, SUB
- Накрыт UPX'ом с изменёнными Magic-байтами

Запускаем таск.



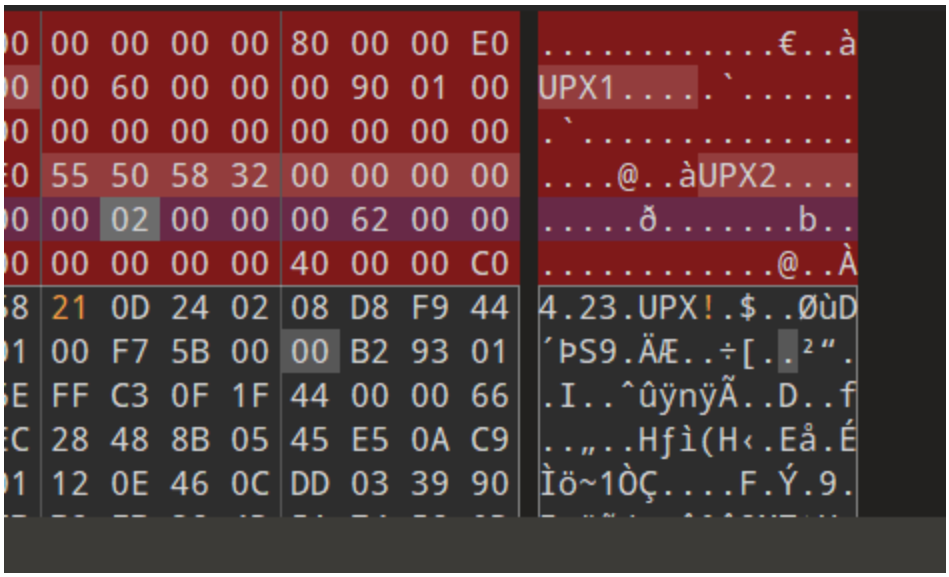
Через Detect It Easy определяем, что использован UPX.

```
C:\Users\rop\Pictures\rev\upx-4.2.3-win64>upx.exe -d vmm.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.3       Markus Oberhumer, Laszlo Molnar & John Reiser   Mar 27th 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: vmm.exe: CantUnpackException: file is possibly modified/hacked/protected; take care!

Unpacked 0 files.
```

При попытке распаковки получаем ошибку.



В HEX-редакторе патчим этот байт на восклицательный знак.

```
C:\Users\rop\Pictures\rev\upx-4.2.3-win64>upx.exe -d vmm.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.3       Markus Oberhumer, Laszlo Molnar & John Reiser   Mar 27th 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    103346 <-     55730   53.93%    win64/pe     vmm.exe

Unpacked 1 file.
```

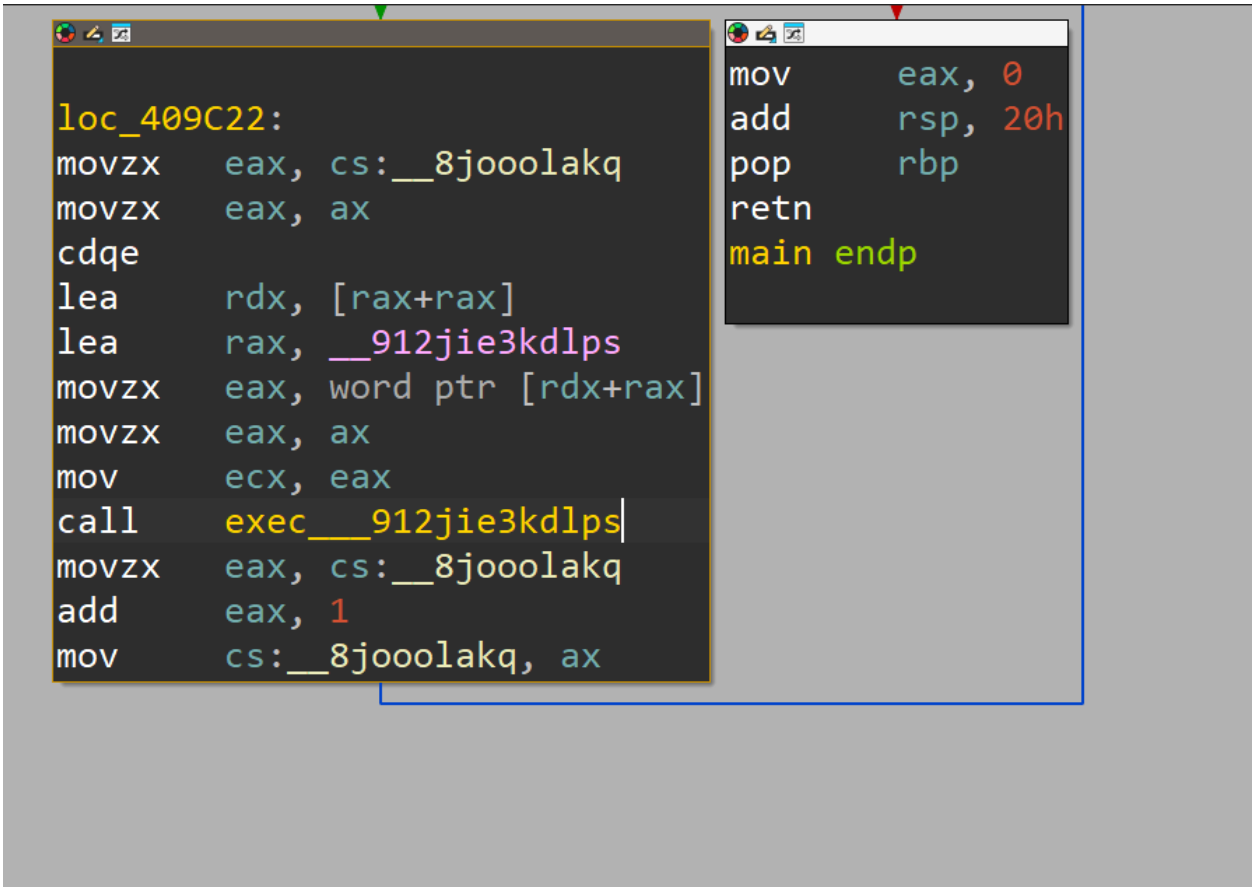
Откроем таск в IDA.

В функции `__tmainCRTStartup` убираем инструкцию `call __mingw_chkstk` - скрытая функция для изменения ключей шифрования XOR, ADD, SUB при включённом отладчике.

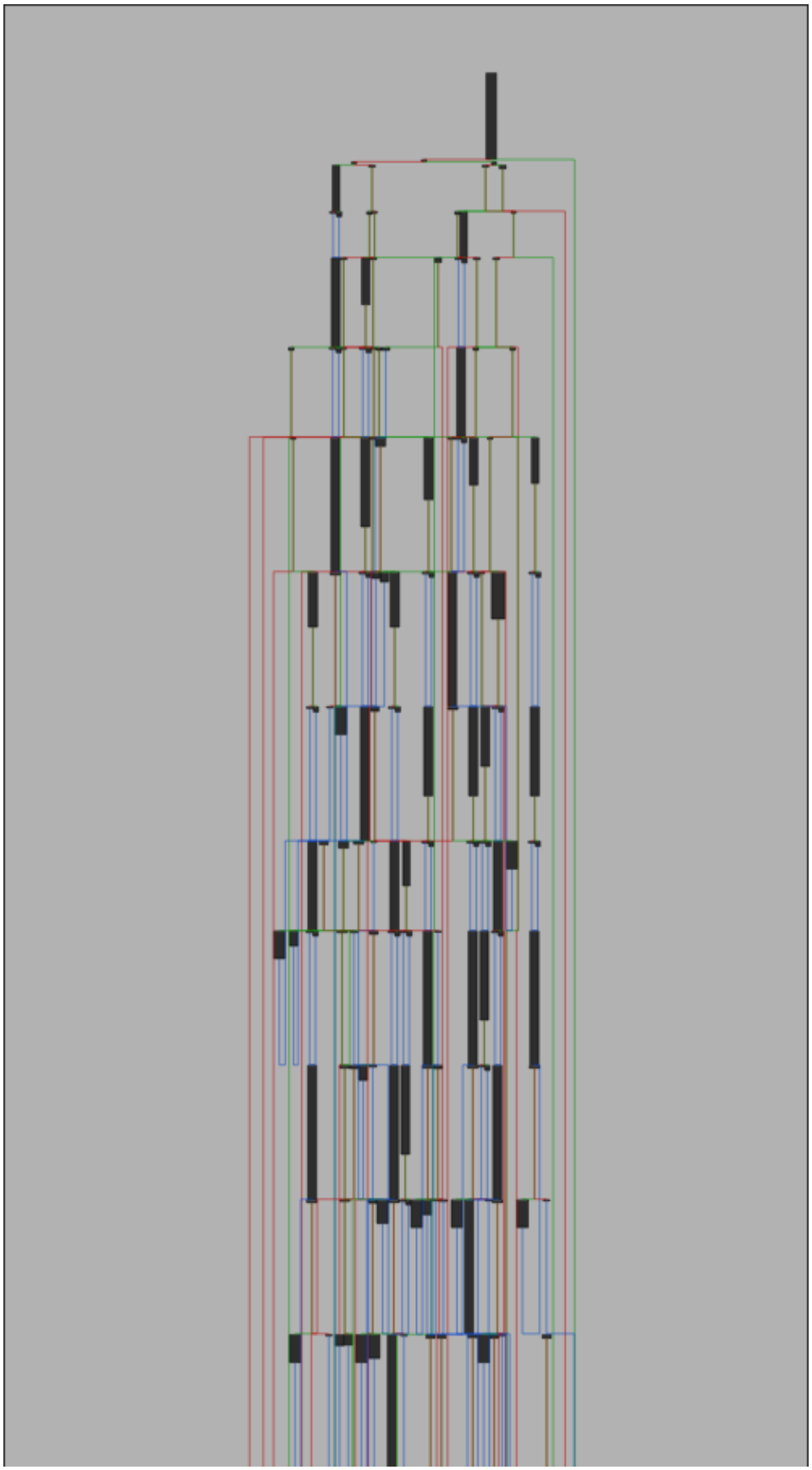
```
loc_401387:
mov     qword ptr [rax], 0
mov     cs:argv, rbp
nop
nop
nop
nop
nop
mov     rax, cs:_refptr___imp___initenv
mov     rdx, cs:envp
mov     ecx, cs:argc    ; argc
mov     rax, [rax]
mov     [rax], rdx
mov     r8, cs:envp     ; envp
mov     rdx, cs:argv    ; argv
call    main
mov     ecx, cs:managedapp
mov     cs:mainret, eax
test    ecx, ecx
jz      loc_40149E
```

Запатчили.

В `main` идёт выполнение кода VM.



Функция `exec` выбирает блок кода для исполнения по коду команды из VM.



Граф этой функции. Много обфускации и лишнего однотипного кода. Но при выполнении кода получаем такой байт-код:

```
    PUSH, 13,

    OUT_MSG,
    SAVE_INPUT_TO_MEM,
    CAESAR_MEM, 13,

    SAVE___8jooolakq,
    ENTER,
    PUSH, 0x65,
    ADD,
    PUSH, 0x99,
    XOR,
    PUSH, 0xa4,
    SUB,
    GET___89j243oljlajs_CHR,
    CMP,
    JNZ, 31,
    GET___892jpojolaja,
    PUSH, TRUE_LEN-1,
    CMP,
    JZ, 33,
    INC___892jpojolaja,
    POP,
    POP,
    LOAD___8jooolakq,

    PRINT_FAIL,
    EXIT,
    PRINT_SUCCESS,
    EXIT
```

Правильный зашифрованный флаг:

```
0x88,0x9a,0x8b,0x8a,0x89,0x84,0xd5,0x93,0x6b,0xaf,0xb1,0xb0,0x86,0xa3,0xb9,0xa2,0x98,0xa8,0x9f,0x6b,0x9b,0xaa,0xb9,0xa2,0x68,0xbb,0xae,0xb1,0x8a,0xaf,0xd7
```

Дешифруем его в таком порядке:

1. ADD 0xA4
2. XOR 0x99
3. SUB 0x65
4. ROT13