



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.

```
C:\Users\rop\Pictures\rev\upx-4.2.3-win64>vmm.exe
St
'St
rtu
er
jis
jj1
:
rtup
VMM - Virtual Machine Monster (C)
Just enter the flag, you know it.
>> qwe
This is a false flag :(
Bye!
C:\Users\rop\Pictures\rev\upx-4.2.3-win64>
```

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.

40	00 00 00 00 00	80 00 00 E0	.....€..à
40	00 60 00 00 00	00 90 01 00	UPX1.....`.....
40	00 00 00 00 00	00 00 00 00 00	.`.....
40	55 50 58 32 00	00 00 00 00 00	....@..àUPX2....
40	00 02 00 00 00	00 62 00 00 00	....đ.....b..
40	00 00 00 00 00	40 00 00 C0	.....@..À
48	21 0D 24 02 08	D8 F9 44	4.23.UPX!.\$..ØùD
41	00 F7 5B 00 00	B2 93 01	’þS9.ÄÈ..÷[. .”.
4E	FF C3 0F 1F 44	00 00 66	.I..^ûýnÿÄ..D..f
4C	28 48 8B 05 45	E5 0A C9	....Hfi(H..Eå.É
41	12 0E 46 0C DD	03 39 90	Ìö~1ÒÇ....F.Ý.9.

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.

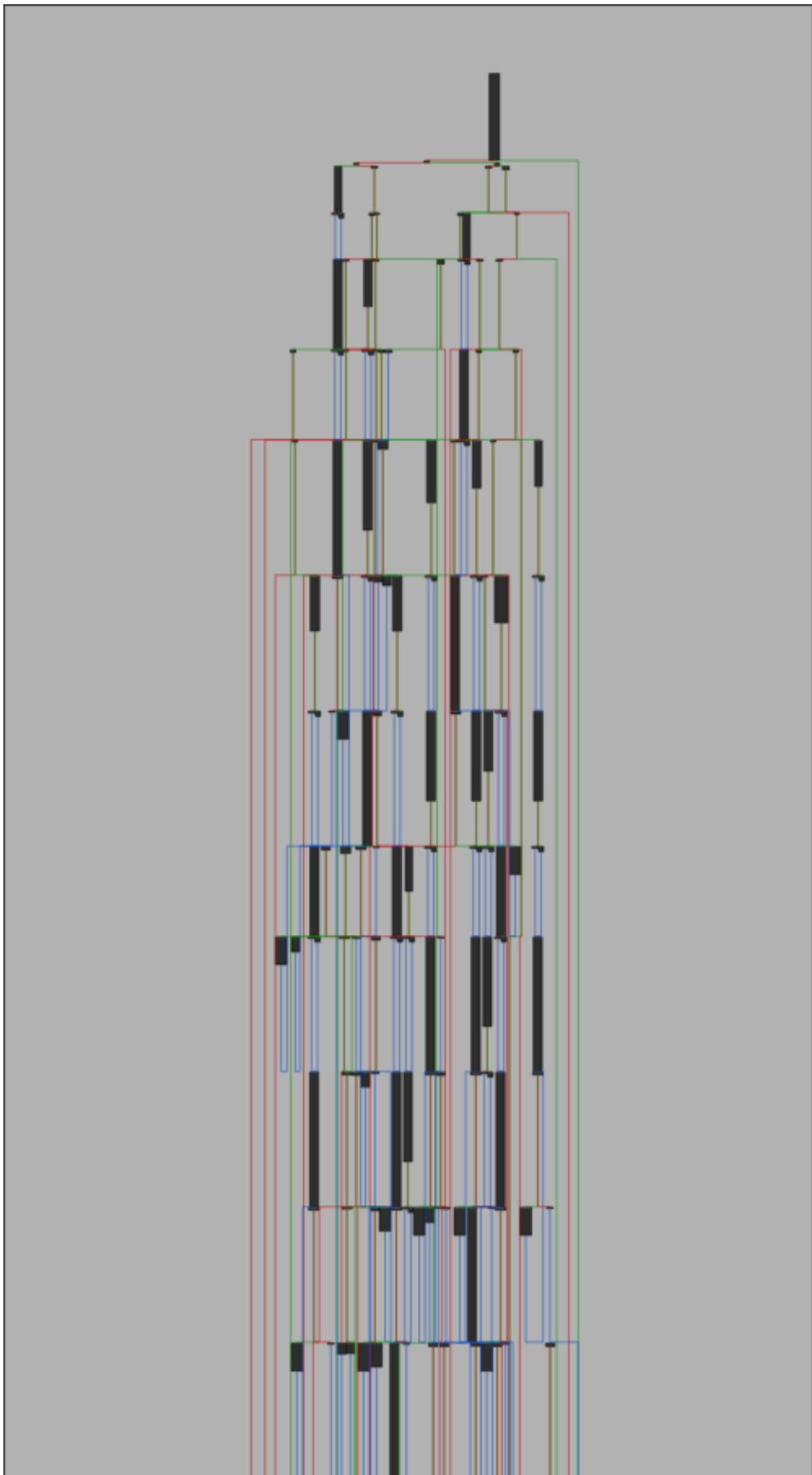
```
loc_401387:  
mov     qword ptr [rax], 0  
mov     cs:argv, rbp  
nop  
nop  
nop  
nop  
nop  
nop  
mov     rax, cs:_refptr__imp__initenv  
mov     rdx, cs:envp  
mov     ecx, cs:argc      ; argc  
mov     rax, [rax]  
mov     [rax], rdx  
mov     r8, cs:envp      ; envp  
mov     rdx, cs:argv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

Запатчили.

В `main` идёт выполнение кода VM.

```
loc_409C22:  
movzx    eax, cs:_8joooolakq  
movzx    eax, ax  
cdqe  
lea      rdx, [rax+rax]  
lea      rax, __912jie3kdlps  
movzx    eax, word ptr [rdx+rax]  
movzx    eax, ax  
mov      ecx, eax  
call    exec__912jie3kdlps|  
movzx    eax, cs:_8joooolakq  
add     eax, 1  
mov      cs:_8joooolakq, ax  
  
mov      eax, 0  
add     rsp, 20h  
pop     rbp  
retn  
main endp
```

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



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

```
PUSH, 13,  
  
OUT_MSG,  
SAVE_INPUT_TO_MEM,  
CAESAR_MEM, 13,  
  
SAVE____8joooolakq,  
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____8joooolakq,  
  
PRINT_FAIL,  
EXIT,  
PRINT_SUCCESS,  
EXIT
```

Правильный зашифрованный флаг:

```
0x88, 0x9a, 0x8b, 0x8a, 0x89, 0x84, 0xd5, 0x93, 0x6b, 0xaf, 0xb1, 0xb0, 0
x86, 0xa3, 0xb9, 0xa2, 0x98, 0xa8, 0x9f, 0x6b, 0x9b, 0xaa, 0xb9, 0xa2, 0x
68, 0xbb, 0xae, 0xb1, 0x8a, 0xaf, 0xd7
```

Дешифруем его в таком порядке:

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