

# Лекция 2. Разбор домашнего задания

Александр Трифанов

Игорь Черватюк

Андрей Басарыгин

Москва, 2018

# Решение CrackmeLinux

```
push    rbp
mov     rbp, rsp
sub     rsp, 120h
mov     rax, 6F72745379726576h
mov     rdx, 6F7773736150676Eh
mov     qword ptr [rbp+s2], rax
mov     [rbp+var_18], rdx
mov     [rbp+var_10], 6472h
mov     [rbp+var_E], 0
lea     rsi, aEnterPassword ; "Enter password:\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
lea     rax, [rbp+s1]
mov     rsi, rax
lea     rdi, std::cin
call    std::operator>><char,std::char_traits<char>>
lea     rdx, [rbp+s2]
lea     rax, [rbp+s1]
mov     rsi, rdx           ; s2
mov     rdi, rax           ; s1
call    _strcmp
test    eax, eax
jnz     short loc_9A3
```

```
lea     rsi, aCongratulation ; "Congratulations!\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
jmp     short loc_9B6
```

```
loc_9A3:
lea     rsi, aWrongAnswer ; "Wrong answer\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
```

- Вводим пароль (сохраняется в s1)
- Видим что он сравнивается с s2
- s2 формируется как:
  - `mov rax, 6F7245379726576h`
  - `mov rdx, 6F7773736150676Eh`
  - Это похоже на строку...

# Решение CrackmeLinux

```
mov     rbp, rsp
sub     rsp, 120h
mov     rax, 'ortSyrev'
mov     rdx, 'owssaPgn'
mov     qword ptr [rbp+s2], rax
mov     qword ptr [rbp+s2+8], rdx
mov     [rbp+var_10], 6472h
mov     [rbp+var_E], 0
lea     rsi, aEnterPassword ; "Enter password:\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
lea     rax, [rbp+s1]
mov     rsi, rax
lea     rdi, std::cin
call    std::operator>><char,std::char_traits<char>>
lea     rdx, [rbp+s2]
lea     rax, [rbp+s1]
mov     rsi, rdx           ; s2
mov     rdi, rax           ; s1
call    _strcmp
test    eax, eax
jnz     short loc_9A3
```

```
lea     rsi, aCongratilation ; "Congratulations!\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
jmp     short loc_9B6
```

```
loc_9A3:
lea     rsi, aWrongAnswer ; "Wrong answer\n"
lea     rdi, std::cout
call    std::operator<<<std::char_traits<char>>
```

- Помогаем IDA понять, что в rax и rdx строки
- Вспоминаем что в процессорах Intel младший байт главный и строка будет интерпретироваться задом-наперёд:
- veryStrongPassword (rd получаем из var\_10)

# Решение CrackMe1

```
sub_40110B proc near
push    offset serial    ; lpString
call    strlenA
cmp     eax, 8
jbe     short loc_401120
```

```
loc_401120:
cmp     eax, 1
jnb     short loc_40112B
```

```
jmp     short loc_40112B
```

```
loc_40112B:
pusha
mov     esi, offset aThePasswordIs ; "The Password is: "
push    esi
xor     edi, edi
mov     esi, offset serial
mov     eax, 31333736h
cmp     eax, [esi]
jnz     short loc_401172
```

```
mov     eax, 37333234h
cmp     eax, [esi+4]
jnz     short loc_401172
```

Best luck the next time :- D

Congratulations!!

- Строки опять формируются через регистры
- Помним про обратный порядок байт, получаем серийный номер:

67314237

# Решение CrackMe2

```
push    [ebp+hWnd]
call    GetDlgItemTextA
call    sub_40117F
cmp     eax, 0F7C4h
jnz     short loc_40111F
```

Функция sub\_40117F решает правильный серийный номер или нет

```
loc_40113E:
cmp     eax, 6Fh
jnz     short loc_40115B
```

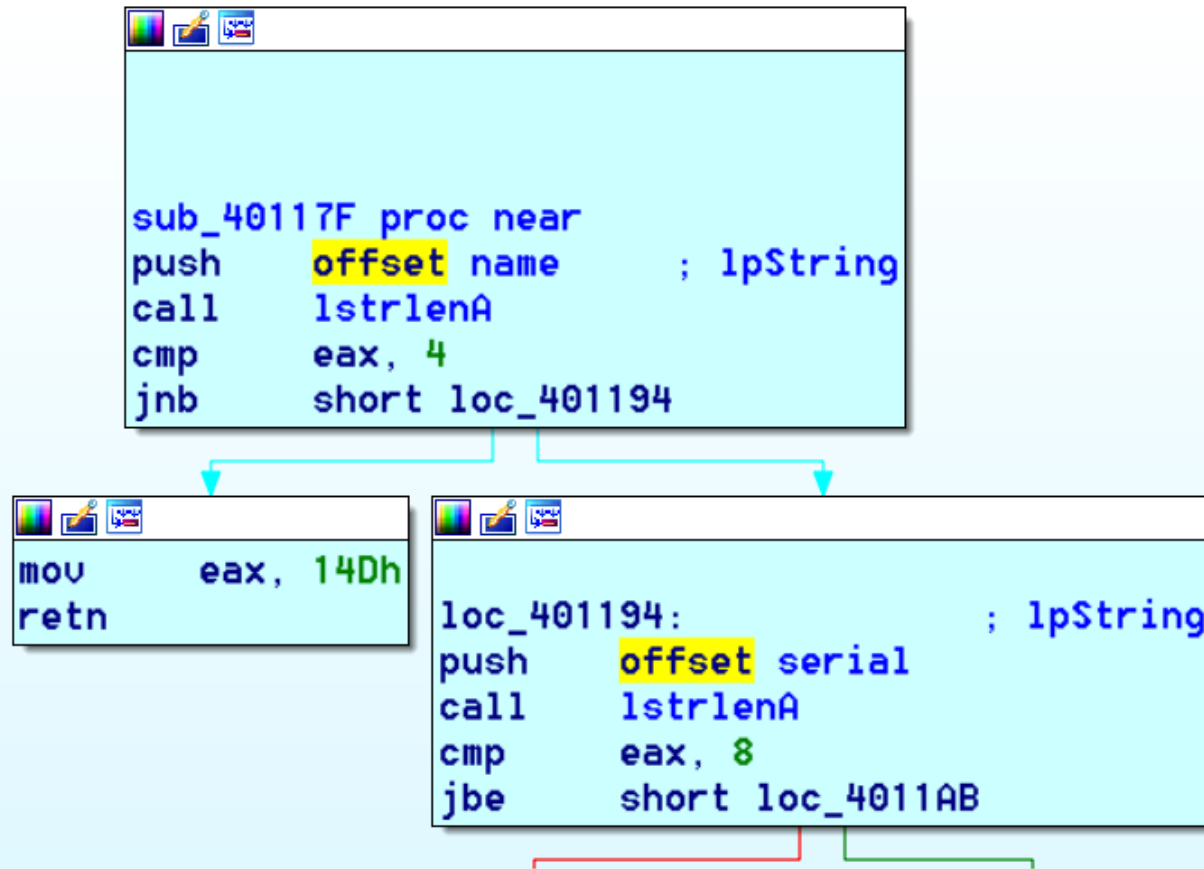
Подобные исходы удобно искать через  
список всех строк  
View->Subviews->Strings или Shift+F12

```
loc_40115B:                                ; uType
push    10h
push    offset aUhhhhhhhhhhhhhhhhhhhh ; "Uhhhhhhhhhhhhhhhhhhhh!!!!!!!!!!"
push    offset aBestLuckTheNex       ; "Best luck the next time :-D"
push    [ebp+hWnd]                   ; hWnd
call    MessageBoxA
```

```
push    30h                                ; uType
push    offset Caption                ; "The Password is: "
push    offset Text                   ; ""
push    [ebp+hWnd]                   ; hWnd
call    MessageBoxA
jmp     short loc_40116F
```

# Решение CrackMe2

Проверяющая функция:



сmp eax, 4

jnb ... jump if not below

Перейти если не меньше

**Имя >=4 символов**

Сmp eax,8

Jbe ... jump if below or equal

Перейти если меньше или равно

**Серийный номер <=8 символов**

# Генерация серийного номера

```
loc_4011B6:
pusha
xor     edi, edi
mov     esi, offset name
push    offset name      ; lpString
call    strlenA
mov     ecx, eax
mov     ebx, 2Dh
```

```
loc_4011CF:
mov     al, [esi]
and     eax, 0FFh
mul     ebx
inc     esi
add     edi, eax
inc     ebx
dec     ecx
jnz     short loc_4011CF
```

esi – указатель на имя

Цикл по длине имени (ecx)

edi=0;

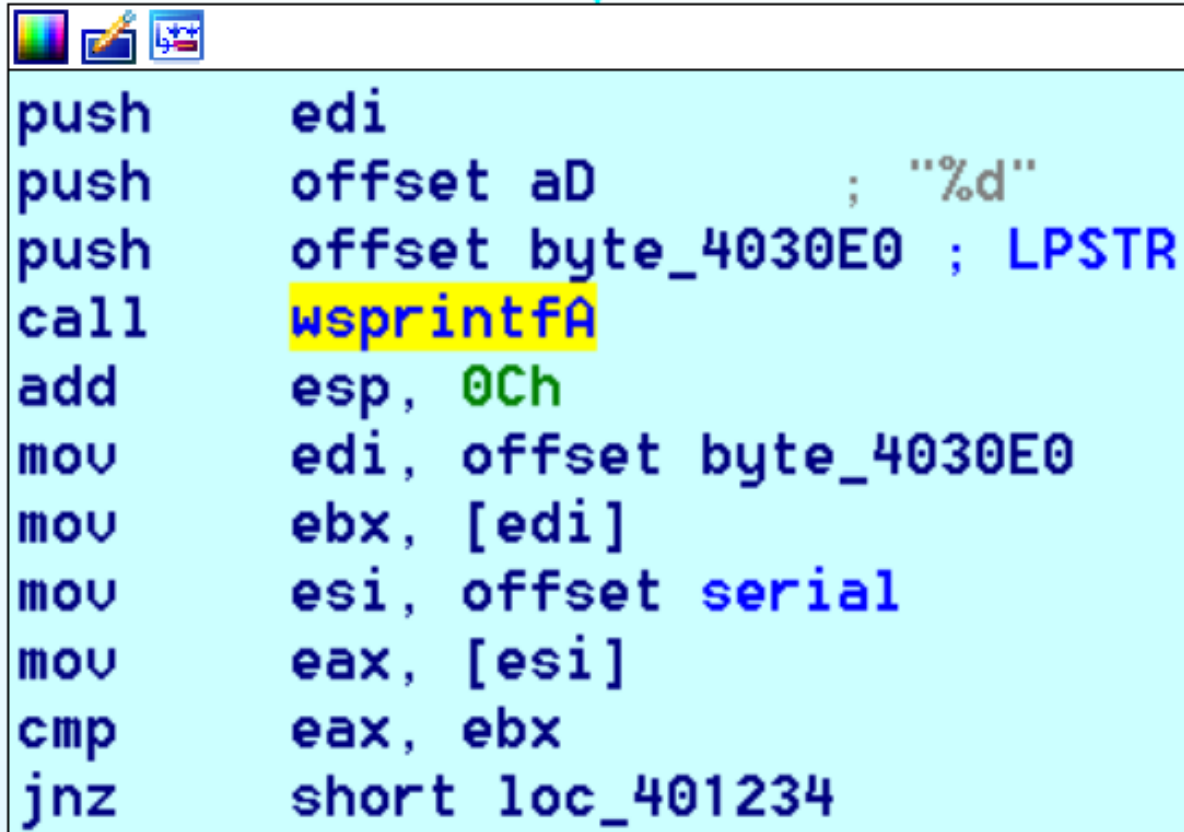
for(ecx=len(name);ecx>=0;ecx--){

edi+=(eax & 0xff)\*ebx;

ebx++;

}

# Последний шаг



```
push    edi
push    offset aD          ; "%d"
push    offset byte_4030E0 ; LPSTR
call    wsprintfA
add     esp, 0Ch
mov     edi, offset byte_4030E0
mov     ebx, [edi]
mov     esi, offset serial
mov     eax, [esi]
cmp     eax, ebx
jnz     short loc_401234
```

`wsprintf(s,edi,"%d");`

Допустим имя «**alex**»

Тогда:

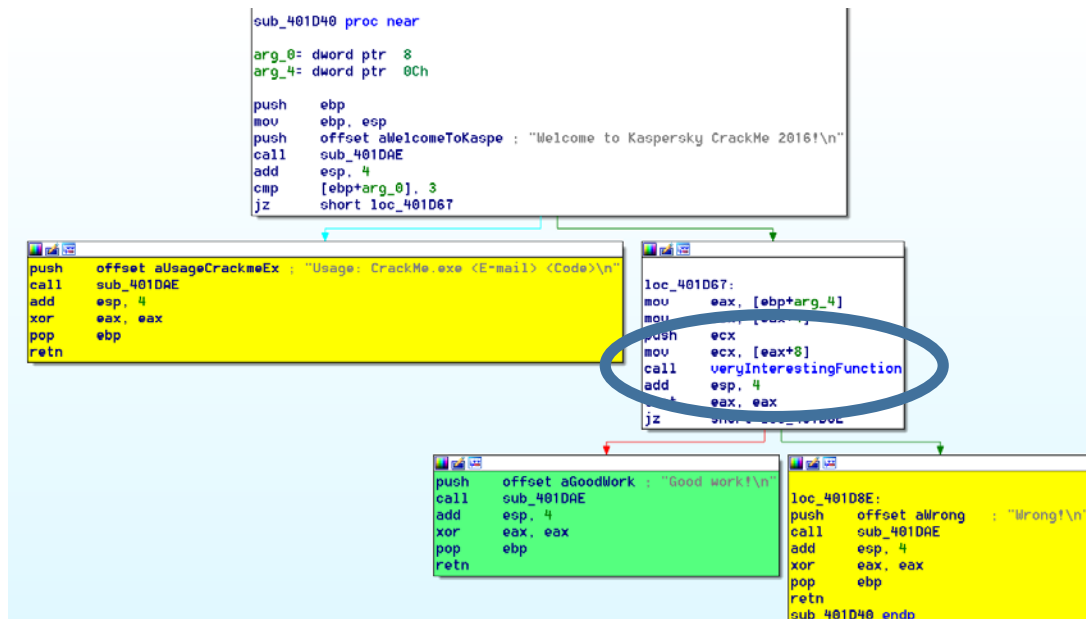
$edi = 61 * 2d + 6c * 2e + 65 * 2f + 78 * 30;$

$edi = 4D80$

$s = 19480$  – это 4D80 в десятичной системе и это мой серийный номер.



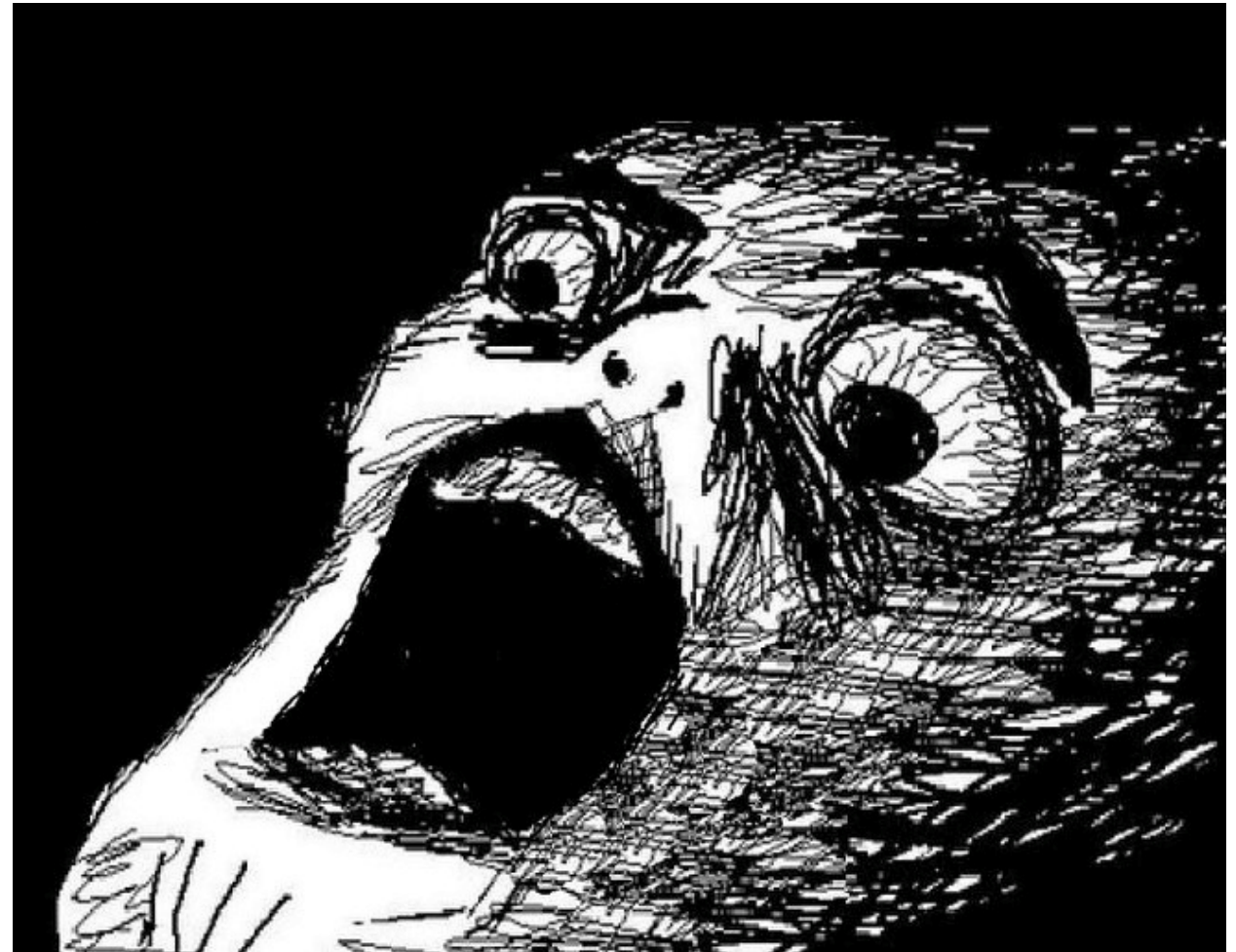
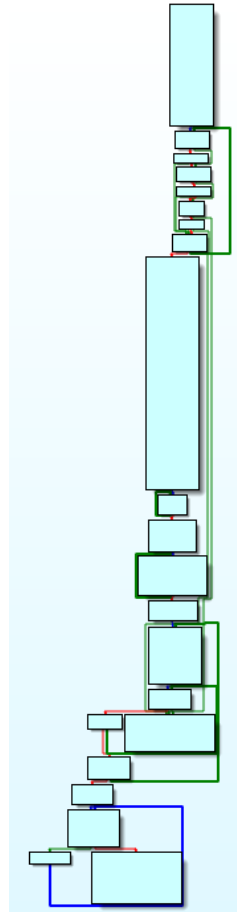
# CrackMe2016 от KasperskyLab



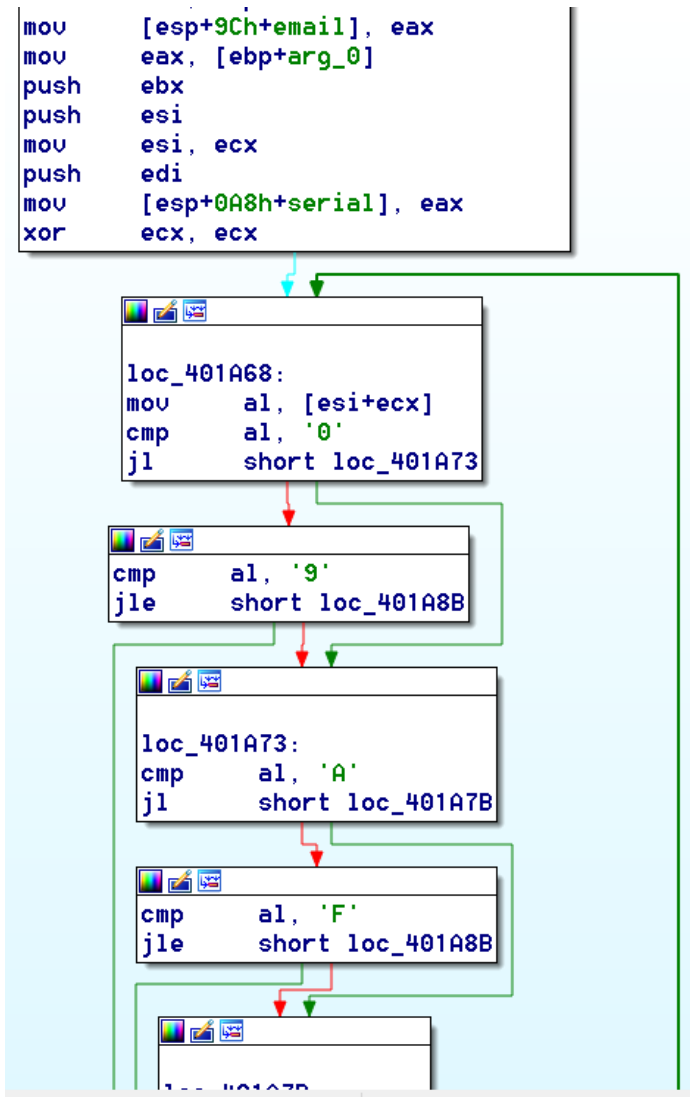
- Находим код, в котором выводятся надпись «wrong» и «Good work!»
- Видим функцию которая принимает решения
- Давайте заглянем в неё

# CrackMe2016 от KasperskyLab

- Общая структура проверки выглядит вот так →→→
- Будем анализировать всё по порядку

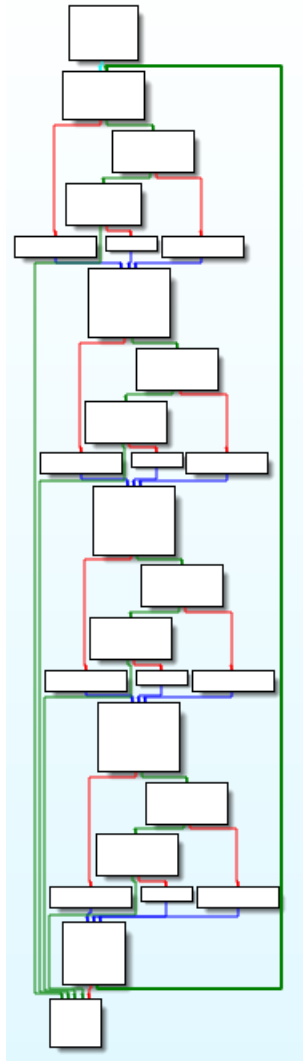


# CrackMe2016 от KasperskyLab



- Проверяем является ли серийный номер строковым представлением шестнадцатеричного числа.
- Чуть ниже – что серийный номер состоит из 32х СИМВОЛОВ

# Внутри функции sub\_401960



- Алгоритм:
  - Если символ – цифра, то вычесть из него 0x30h
  - Если большая буква, то вычесть из него 0x41h
  - Если маленькая буква, то вычесть из него 0x61h
- 0x30h='0'
- 0x41h='A'
- 0x61h='a'
- Функция преобразует строку в число.

# «Магические» числа в коде

```
mov     eax, edx
mov     [esp+0A8h+var_28], esi
mov     [esp+0A8h+var_8C], ebx
mov     [esp+0A8h+var_90], ebx
mov     [esp+0A8h+var_88], 67452301h
mov     [esp+0A8h+var_84], 0EFCDA89h
mov     [esp+0A8h+var_80], 98BADCFEh
mov     [esp+0A8h+var_7C], 10325476h
lea     esi, [eax+1]
```

Это просто md5 от электронного адреса

Branch: master ▼

[arc\\_conv](#) / [include](#) / [md5.asm](#)

 dsp2003 Upload r55

1 contributor

122 lines (117 sloc) | 1.78 KB

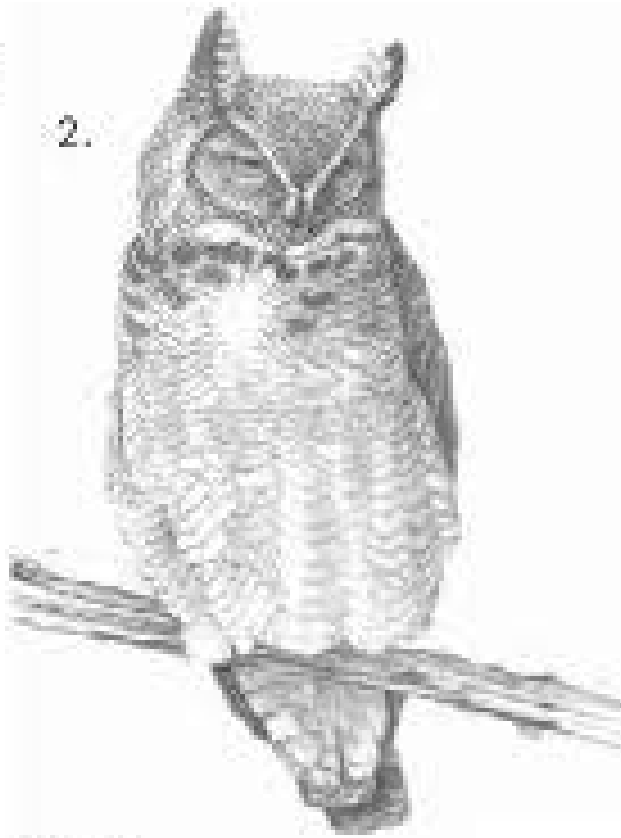
```
1
2  @md5_size = 58h
3
4  _md5_init@4 PROC          ; ctx
5      pop     eax
6      pop     ecx
7      push    eax
8      mov     eax, 67452301h
9      mov     edx, 0EFCDA89h
```

# Разбираемся с остальными проверками

Как нарисовать сову



2.

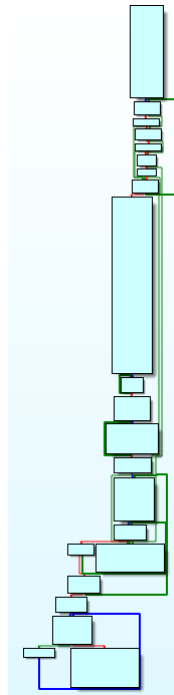


1. Рисуем кружочки

2. Рисуем остаток совы

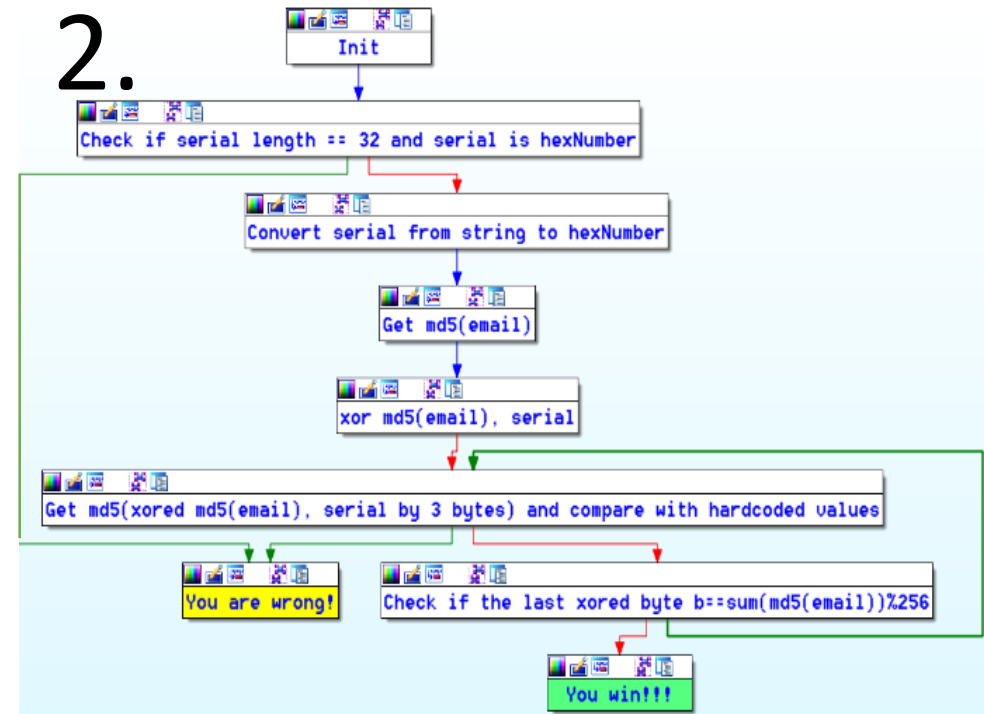
Как восстановить алгоритм работы программы

1.



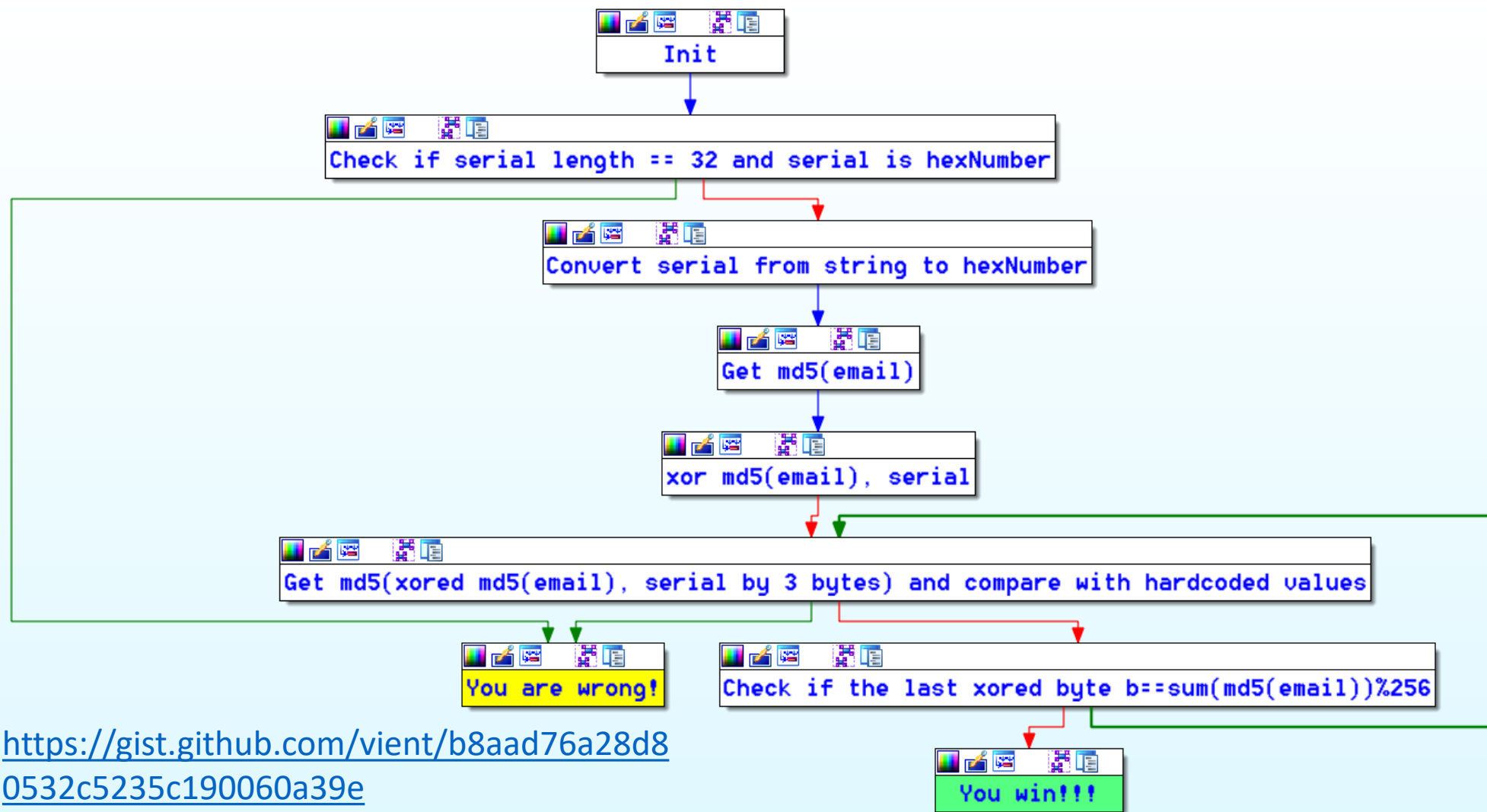
1. Анализируем  
фрагменты  
кода в IDA

2.



2. Получаем алгоритм  
работы

## 2. Восстанавливаем общий вид функции проверки😊



# Спасибо! Вопросы?!

Александр Трифанов

Игорь Черватюк

Андрей Басарыгин

Москва, 2018