Разбор заданий категории Reverse Engineering

Задание на 100 баллов
“DECODE”

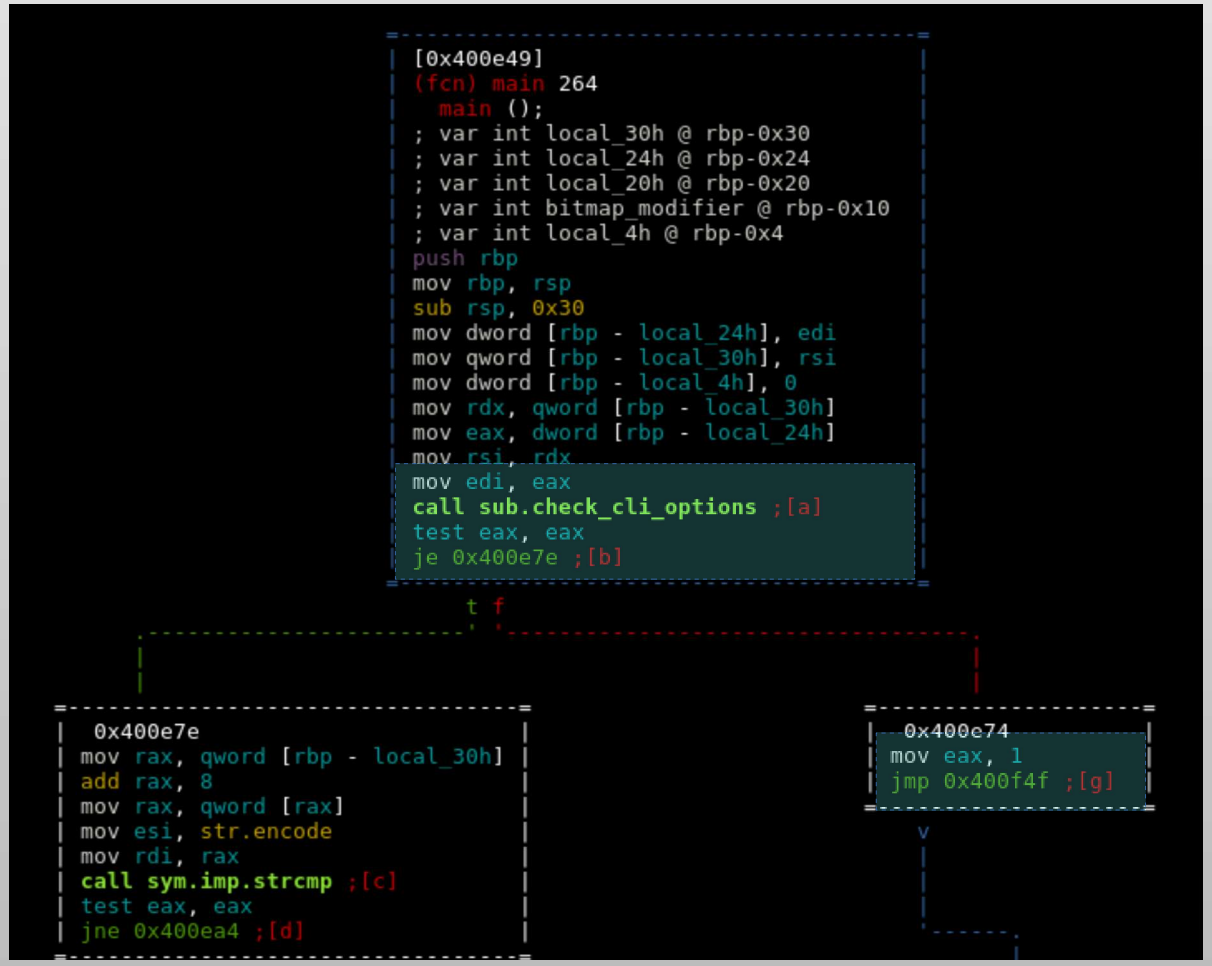
Участники получают пару объектов:

- исполняемый файл
- закодированный бинарный файл.

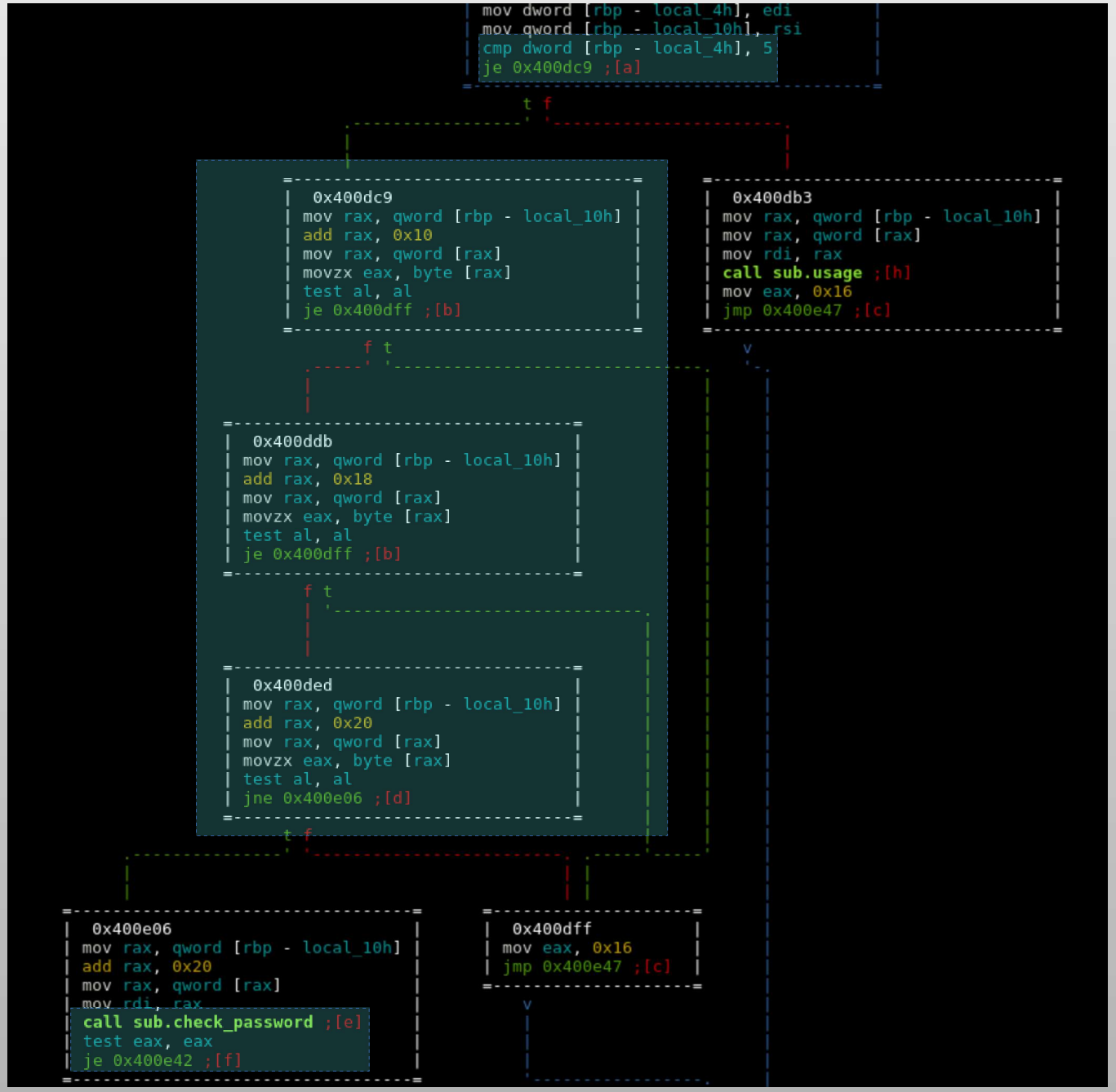
Задача:

Узнать ключ и декодировать файл

main


[0x400e49]
(fcn) main 264
main ();
; var int local_30h @ rbp-0x30
; var int local_24h @ rbp-0x24
; var int local_20h @ rbp-0x20
; var int bitmap_modifier @ rbp-0x10
; var int local_4h @ rbp-0x4
push rbp
mov rbp, rsp
sub rsp, 0x30
mov dword [rbp - local_24h], edi
mov qword [rbp - local_30h], rsi
mov dword [rbp - local_4h], 0
mov rdx, qword [rbp - local_30h]
mov eax, dword [rbp - local_24h]
mov rsi, rdx
mov edi, eax
call sub.check_cli_options ;[a]
test eax, eax
je 0x400e7e ;[b]
t f
0x400e7e
mov rax, qword [rbp - local_30h]
add rax, 8
mov rax, qword [rax]
mov esi, str.encode
mov rdi, rax
call sym.imp.strcmp ;[c]
test eax, eax
jne 0x400ea4 ;[d]
0x400e74
mov eax, 1
jmp 0x400f4f ;[g]

check_cli_options

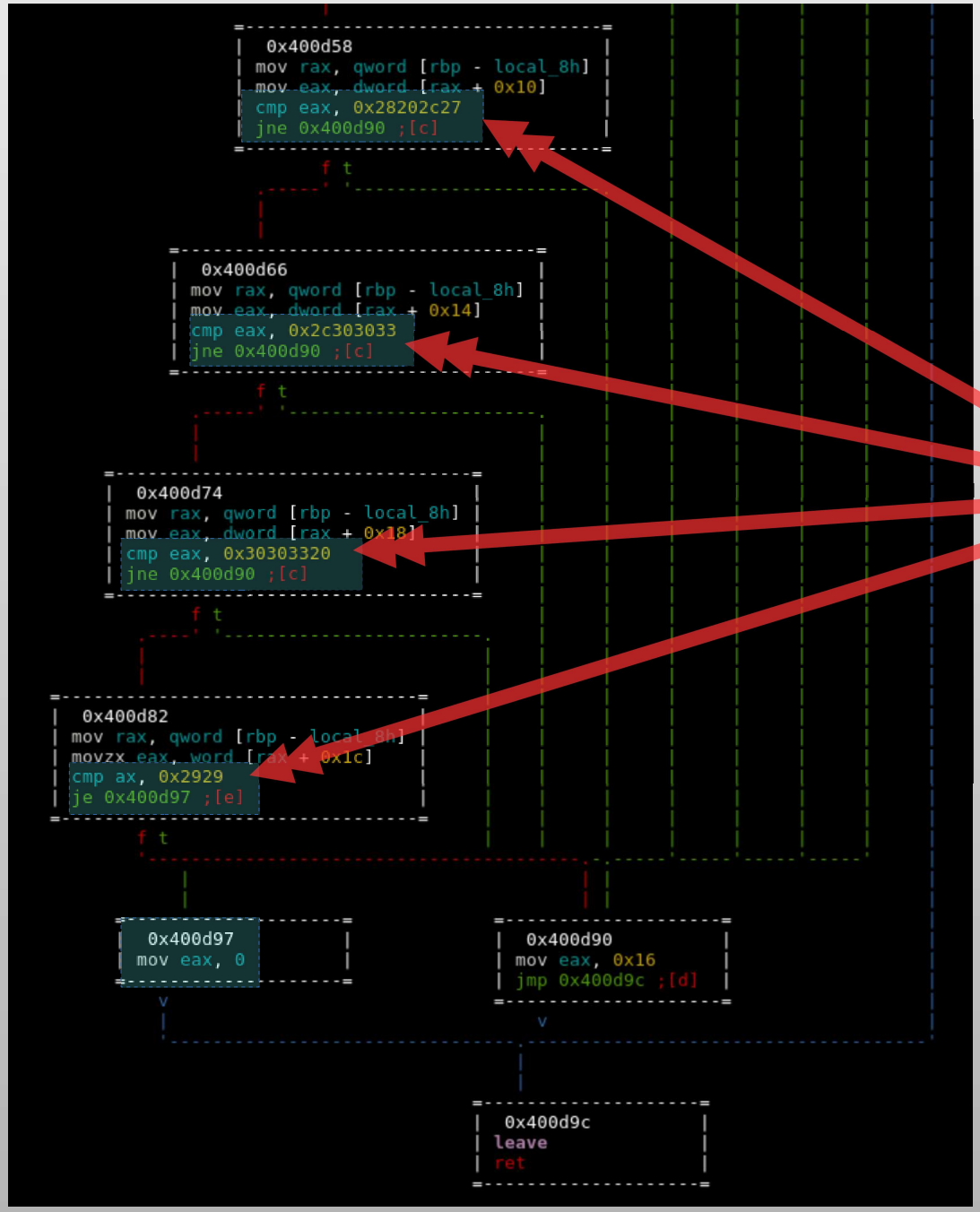


check_password

Длина ключа
0x1E

Ключ в открытом
виде

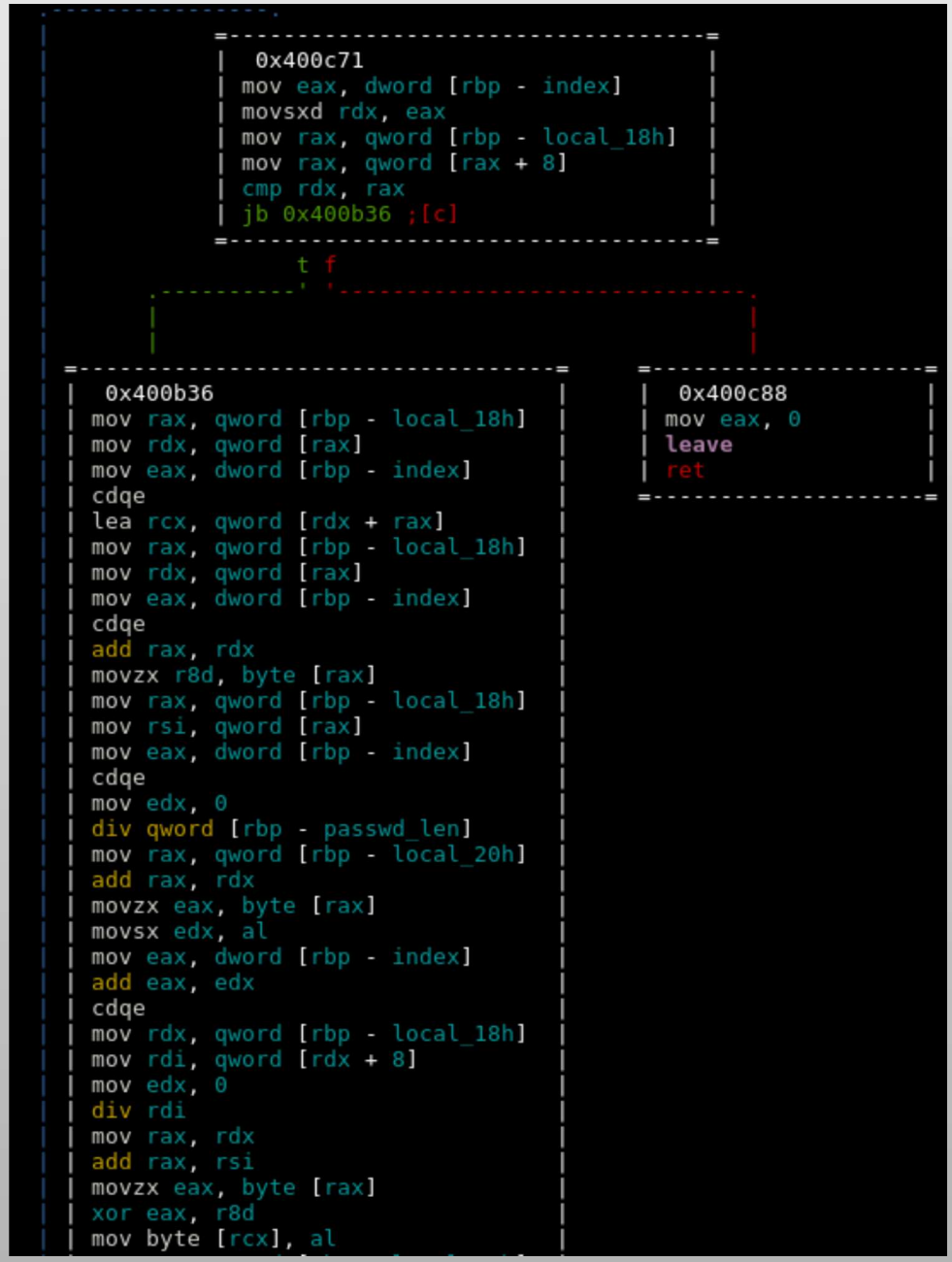
check_password (продолжение)



Ключ в открытом виде

Зная ключ, необходимо восстановить процедуру декодирования, обратить изображение и получить флаг

Преобразование изображения



Преобразование изображения (продолжение)

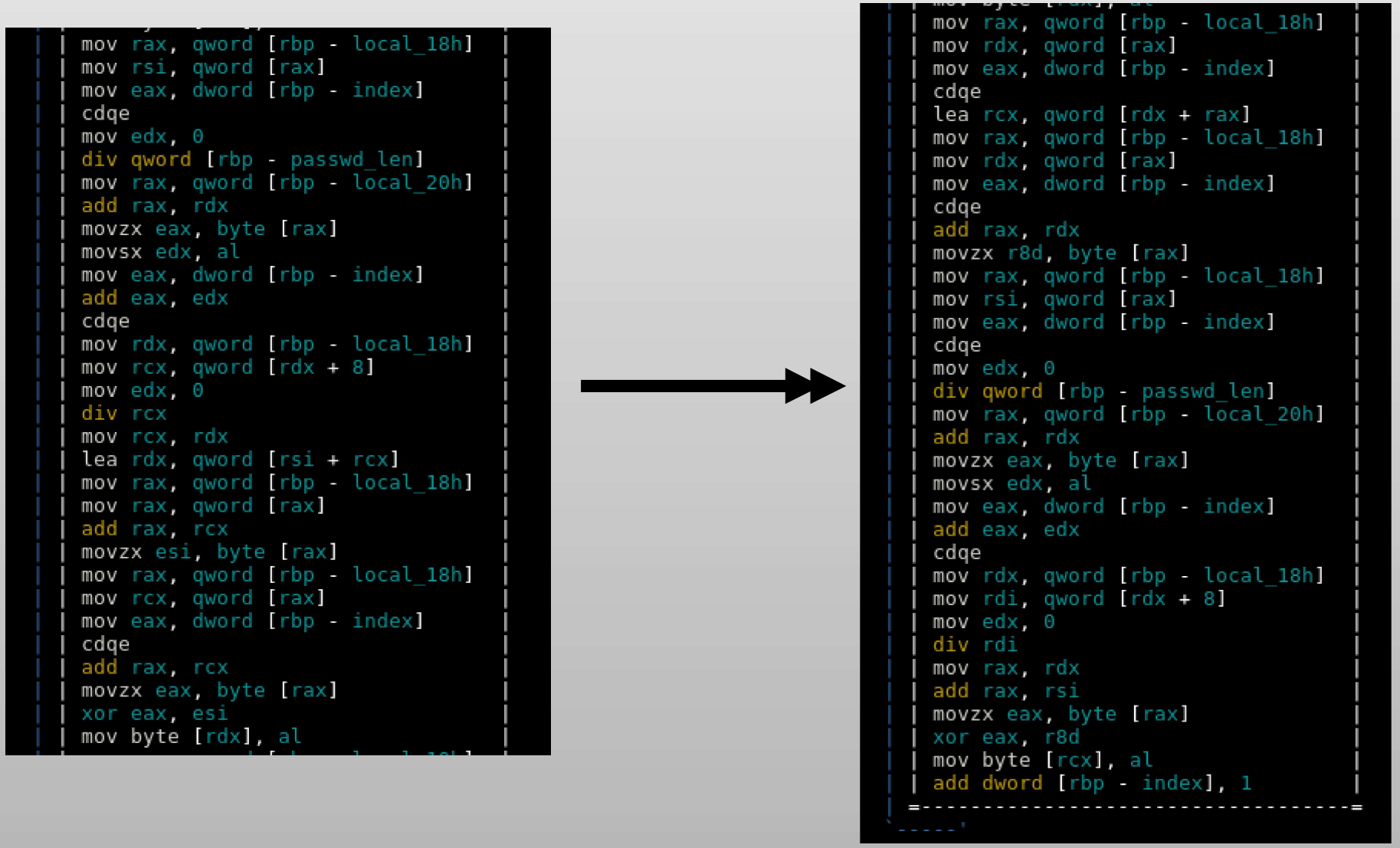


Схема кодирования:

Декодируем изображение и
получаем флаг

Ключ является подсказкой к размеру и
цветовой модели изображения

Задание на 200 баллов
“PUZZLE”

Задача:

Найти корректную ключевую фразу

main

Длина пароля
0x10

```
0x00401267      55              push rbp
0x00401268      4889e5          mov rbp, rsp
0x0040126b      4883ec10        sub rsp, 0x10
0x0040126f      897dfc          mov dword [rbp - local_4h], edi
0x00401272      488975f0        mov qword [rbp - local_10h], rsi
0x00401276      837dfc02        cmp dword [rbp - local_4h], 2 ; [0x2:4]=0x102464c
0x0040127a      7519            jne 0x401295
0x0040127c      488b45f0        mov rax, qword [rbp - local_10h]
0x00401280      4883c008        add rax, 8
0x00401284      488b00          mov rax, qword [rax]
0x00401287      4889c7          mov rdi, rax
0x0040128a      e841f5ffff      call sym.imp.strlen
0x0040128f      4883f810        cmp rax, 0x10
0x00401293      740a            je 0x40129f
; JMP XREF from 0x0040127a (main)
0x00401295      b801000000      mov eax, 1
0x0040129a      e9d9000000      jmp 0x401378
```

Функция будет выполнена
в новом процессе

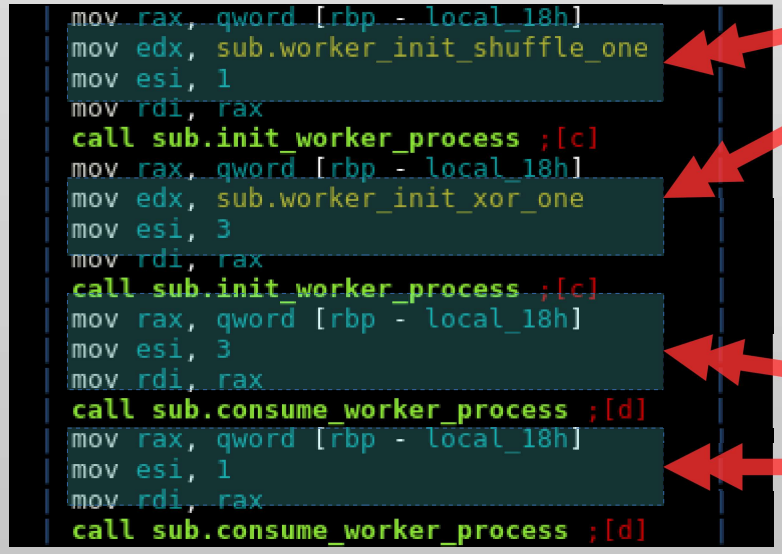
```
0x004012e2      488b45f0        mov rax, qword [rbp - local_10h]
0x004012e6      4883c008        add rax, 8
0x004012ea      488b00          mov rax, qword [rax]
0x004012ed      488b15b40d20.   mov rdx, qword [0x006020a8] ; [0
0x004012f4      488d8a180100.   lea rcx, qword [rdx + 0x118] ; 0
0x004012fb      ba10000000      mov edx, 0x10
0x00401300      4889c6          mov rsi, rax
0x00401303      4889cf          mov rdi, rcx
0x00401306      e8a5f4ffff      call sym.imp.strncpy        ; cha
0x0040130b      488b05960d20.   mov rax, qword [0x006020a8] ; [0
0x00401312      bab2104000      mov edx, 0x4010b2
0x00401317      be00000000      mov esi, 0
0x0040131c      4889c7          mov rdi, rax
0x0040131f      e872f6ffff      call sub.init_worker_process
0x00401324      488b057d0d20.   mov rax, qword [0x006020a8] ; [0
0x0040132b      be00000000      mov esi, 0
0x00401330      4889c7          mov rdi, rax
0x00401333      e875f7ffff      call sub.consume_worker_process
0x00401338      488b05690d20.   mov rax, qword [0x006020a8] ; [0
0x0040133f      4889c7          mov rdi, rax
0x00401342      e8aefeffff      call sub.is_password_correct
0x00401347      84c0            test al, al
0x00401349      7414            je 0x40135f
0x0040134b      ba03000000      mov edx, 3
0x00401350      be04144000      mov esi, 0x401404
0x00401355      bf01000000      mov edi, 1
0x0040135a      e861f4ffff      call sym.imp.write
```

init_worker_process

```
> 0x004009bb      488d45f0        lea rax, qword [rbp - local_10h]
  0x004009bf      4889c7          mov rdi, rax
  0x004009c2      e899feffff      call sym.imp.pthread_barrierattr_init
  0x004009c7      488d45f0        lea rax, qword [rbp - local_10h]
  0x004009cb      be01000000      mov esi, 1
  0x004009d0      4889c7          mov rdi, rax
  0x004009d3      e828feffff      call sym.imp.pthread_barrierattr_setpshared
  0x004009d8      488b55e0        mov rdx, qword [rbp - local_20h]
  0x004009dc      4889d0          mov rax, rdx
  0x004009df      48c1e002        shl rax, 2
  0x004009e3      4801d0          add rax, rdx                ; '('
  0x004009e6      48c1e003        shl rax, 3
  0x004009ea      488b55e8        mov rdx, qword [rbp - local_18h]
  0x004009ee      4801d0          add rax, rdx                ; '('
  0x004009f1      488d4808        lea rcx, qword [rax + 8]    ; 0x8
  0x004009f5      488d45f0        lea rax, qword [rbp - local_10h]
  0x004009f9      ba02000000      mov edx, 2
  0x004009fe      4889c6          mov rsi, rax
  0x00400a01      4889cf          mov rdi, rcx
  0x00400a04      e817feffff      call sym.imp.pthread_barrier_init
  0x00400a09      488d45f0        lea rax, qword [rbp - local_10h]
  0x00400a0d      4889c7          mov rdi, rax
  0x00400a10      e8fbfdffff      call sym.imp.pthread_barrierattr_destroy
  0x00400a15      e876feffff      call sym.imp.fork
  0x00400a1a      89c6            mov esi, eax
  0x00400a1c      488b4de8        mov rcx, qword [rbp - local_18h]
  0x00400a20      488b55e0        mov rdx, qword [rbp - local_20h]
  0x00400a24      4889d0          mov rax, rdx
  0x00400a27      48c1e002        shl rax, 2
  0x00400a2b      4801d0          add rax, rdx                ; '('
  0x00400a2e      48c1e003        shl rax, 3
  0x00400a32      4801c8          add rax, rcx                ; '&'
  0x00400a35      8930            mov dword [rax], esi
  0x00400a37      488b4de8        mov rcx, qword [rbp - local_18h]
  0x00400a3b      488b55e0        mov rdx, qword [rbp - local_20h]
  0x00400a3f      4889d0          mov rax, rdx
  0x00400a42      48c1e002        shl rax, 2
  0x00400a46      4801d0          add rax, rdx                ; '('
  0x00400a49      48c1e003        shl rax, 3
  0x00400a4d      4801c8          add rax, rcx                ; '&'
  0x00400a50      8b00            mov eax, dword [rax]
  0x00400a52      85c0            test eax, eax
< 0x00400a54      751b            jne 0x400a71
  0x00400a56      488b4de0        mov rcx, qword [rbp - local_20h]
  0x00400a5a      488b55e8        mov rdx, qword [rbp - local_18h]
  0x00400a5e      488b45d8        mov rax, qword [rbp - local_28h]
  0x00400a62      4889ce          mov rsi, rcx
  0x00400a65      4889d7          mov rdi, rdx
  0x00400a68      ffd0            call rax
  0x00400a6a      89c7            mov edi, eax
  0x00400a6c      e8fffdffff      call sym.imp.exit           ; void exit(int st
```

Функция передана третьим аргументом

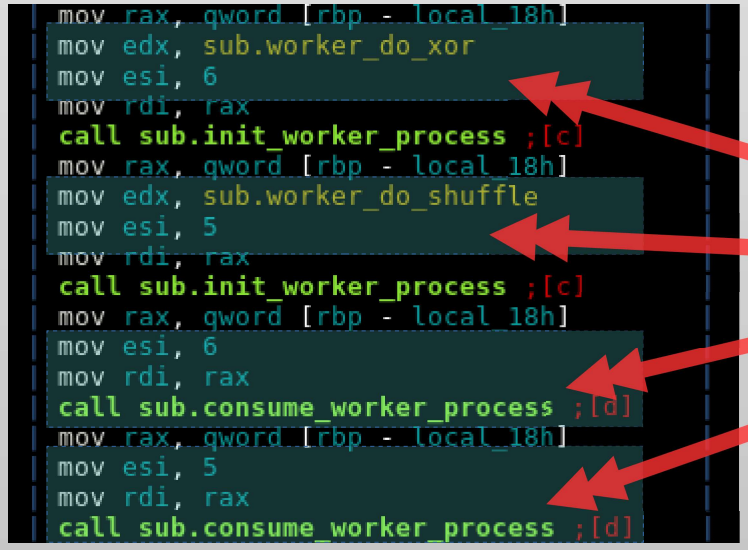
main_worker_process (0x4010B2)



Инициализация процессов-модификаторов состояния

Модификация состояния происходит в обратном порядке

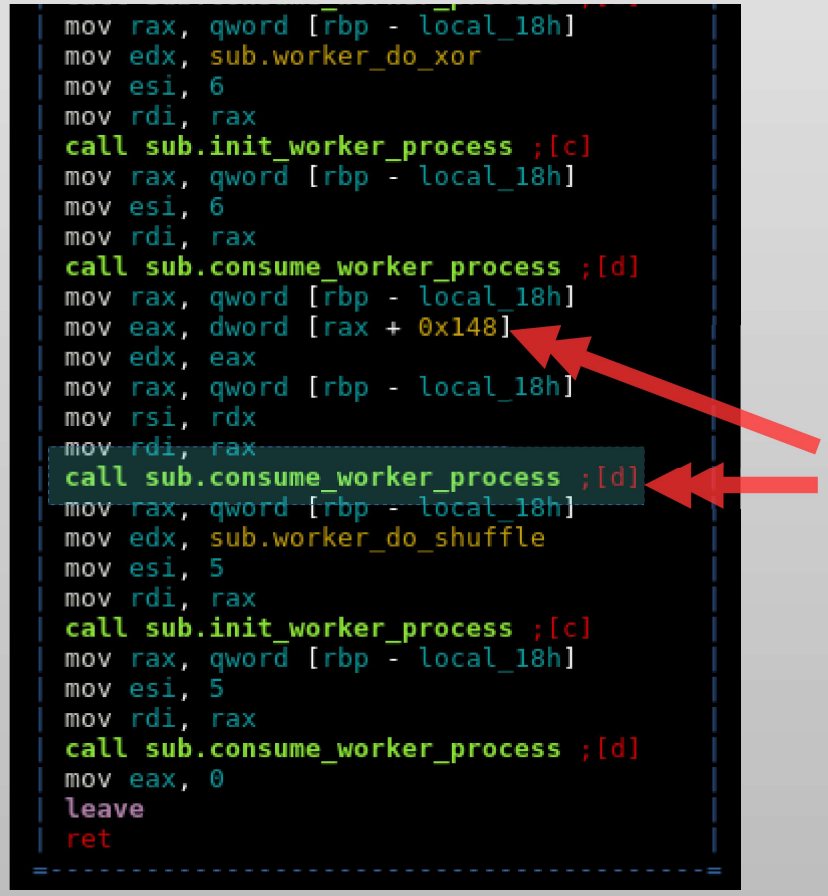
main_worker_process (продолжение)



Модификация
пользовательского ввода

Таблицы перестановки и XOR
были инициализированы
на предыдущем шаге

main_worker_process (продолжение)



```
mov rax, qword [rbp - local_18h]
mov edx, sub.worker_do_xor
mov esi, 6
mov rdi, rax
call sub.init_worker_process ;[c]
mov rax, qword [rbp - local_18h]
mov esi, 6
mov rdi, rax
call sub.consume_worker_process ;[d]
mov rax, qword [rbp - local_18h]
mov eax, dword [rax + 0x148]
mov edx, eax
mov rax, qword [rbp - local_18h]
mov rsi, rdx
mov rdi, rax
call sub.consume_worker_process ;[d]
mov rax, qword [rbp - local_18h]
mov edx, sub.worker_do_shuffle
mov esi, 5
mov rdi, rax
call sub.init_worker_process ;[c]
mov rax, qword [rbp - local_18h]
mov esi, 5
mov rdi, rax
call sub.consume_worker_process ;[d]
mov eax, 0
leave
ret
```

Завершение процесса без предварительной инициализации
Процесс инициализируется внутри одного из воркеров?
(смещение 0x148 содержит идентификатор процесса)

Описание процессов-модификаторов состояния

Инициализация XOR-таблицы

```
push rbp
mov rbp, rsp
sub rsp, 0x20
mov qword [rbp - local_18h], rdi
mov qword [rbp - local_20h], rsi
mov rdx, qword [rbp - local_20h]
mov rax, qword [rbp - local_18h]
mov rsi, rdx
mov rdi, rax
call sub.load_worker_id ;[a]
mov qword [rbp - local_8h], rax
mov rax, qword [rbp - local_8h]
add rax, 8
mov rdi, rax
call sym.imp.pthread_barrier_wait ;[b]
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x138], 0xfe
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x139], 0x51
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13a], 7
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13b], 0x3d
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13c], 0x5d
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13d], 0x85
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13e], 0xeb
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13f], 0x60
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x140], 0x7a
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x141], 0xc1
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x142], 0x4b
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x143], 0x54
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x144], 0x7a
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x145], 0xa0
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x146], 0xdd
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x147], 0x99
mov eax, 0
leave
ret
```

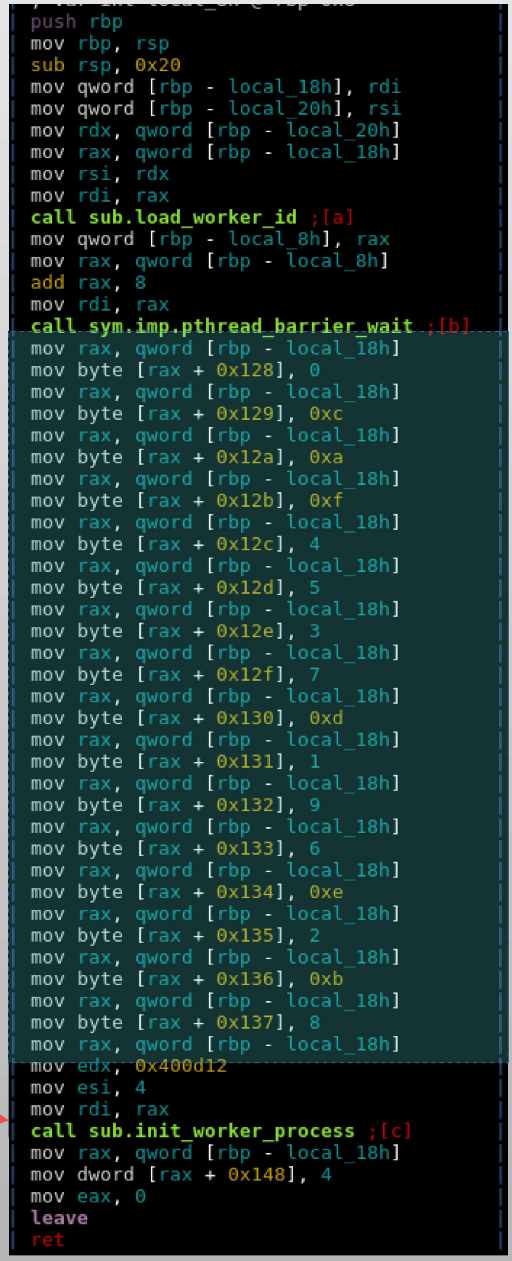
Синхронизация с управляющим процессом

Фиксированное значение элементов таблицы (смещение XOR-таблицы составляет 0x138)

Инициализация таблицы перестановок

Фиксированное значение элементов таблицы (смещение таблицы перестановок составляет 0x128)

Инициализация вложенного процесса-модификатора (0x400D12; 4)



Повторная инициализация XOR-таблицы (функция по адресу 0x400D12)

```
push rbp
mov rbp, rsp
sub rsp, 0x20
mov qword [rbp - local_18h], rdi
mov qword [rbp - local_20h], rsi
mov rdx, qword [rbp - local_20h]
mov rax, qword [rbp - local_18h]
mov rsi, rdx
mov rdi, rax
call sub.load_worker_id ;[a]
mov qword [rbp - local_8h], rax
mov rax, qword [rbp - local_8h]
add rax, 8
mov rdi, rax
call sym.imp.pthread_barrier_wait ;[b]
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x138], 0x49
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x139], 0xc8
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13a], 5
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13b], 0x10
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13c], 0xf
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13d], 0x1a
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13e], 0xd4
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x13f], 0xf
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x140], 0xea
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x141], 0x1e
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x142], 0xa
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x143], 0x7c
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x144], 0xd9
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x145], 6
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x146], 0xfc
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x147], 0x14
mov rax, qword [rbp - local_18h]
mov edx, sub.worker_init_shuffle_two
mov esi, 2
mov rdi, rax
call sub.init_worker_process ;[c]
mov rax, qword [rbp - local_18h]
mov dword [rax + 0x148], 2
mov eax, 0
leave
ret
```

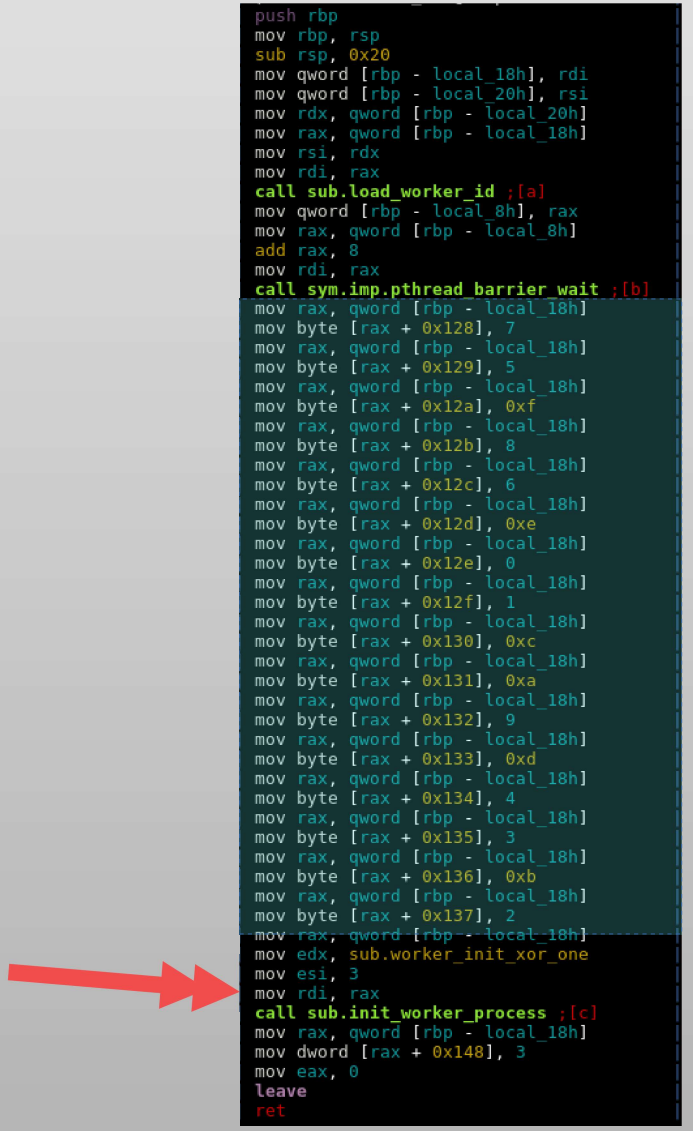
Инициализация XOR-таблицы повторно производится фиксированными значениями (отличающимися от первоначальных)

Очередной запуск вложенного процесса (0x400C00; 2)

Повторная инициализация таблицы перестановок (функция по адресу 0x400C00)

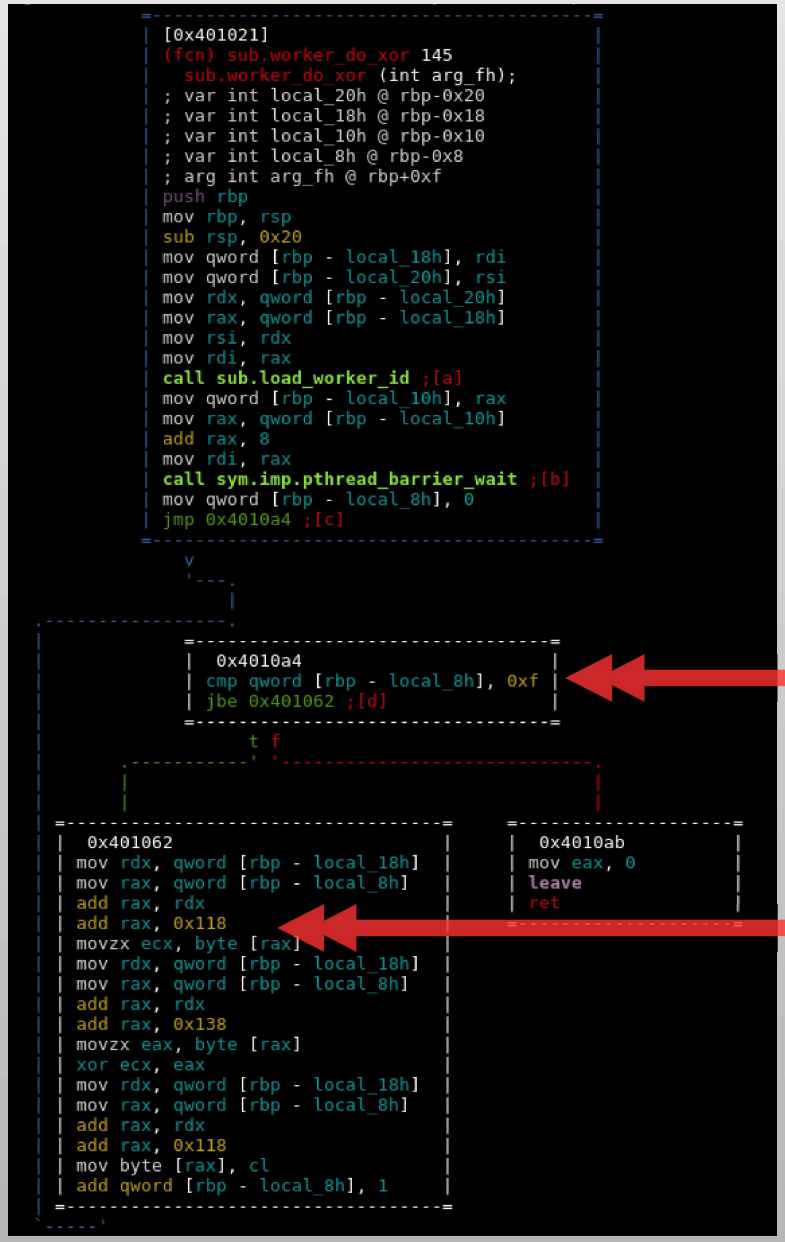
Переинициализация таблицы перестановок произойдет после повторной инициализации XOR-таблицы

Завершение данного процесса произойдет уже после финальной модификации пользовательского ввода



```
push rbp
mov rbp, rsp
sub rsp, 0x20
mov qword [rbp - local_18h], rdi
mov qword [rbp - local_20h], rsi
mov rdx, qword [rbp - local_20h]
mov rax, qword [rbp - local_18h]
mov rsi, rdx
mov rdi, rax
call sub.load_worker_id ;[a]
mov qword [rbp - local_8h], rax
mov rax, qword [rbp - local_8h]
add rax, 8
mov rdi, rax
call sym.imp.pthread_barrier_wait ;[b]
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x128], 7
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x129], 5
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12a], 0xf
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12b], 8
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12c], 6
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12d], 0xe
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12e], 0
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x12f], 1
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x130], 0xc
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x131], 0xa
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x132], 9
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x133], 0xd
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x134], 4
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x135], 3
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x136], 0xb
mov rax, qword [rbp - local_18h]
mov byte [rax + 0x137], 2
mov rax, qword [rbp - local_18h]
mov edx, sub.worker_init_xor_one
mov esi, 3
mov rdi, rax
call sub.init_worker_process ;[c]
mov rax, qword [rbp - local_18h]
mov dword [rax + 0x148], 3
mov eax, 0
leave
ret
```

XOR-преобразование пользовательского ввода



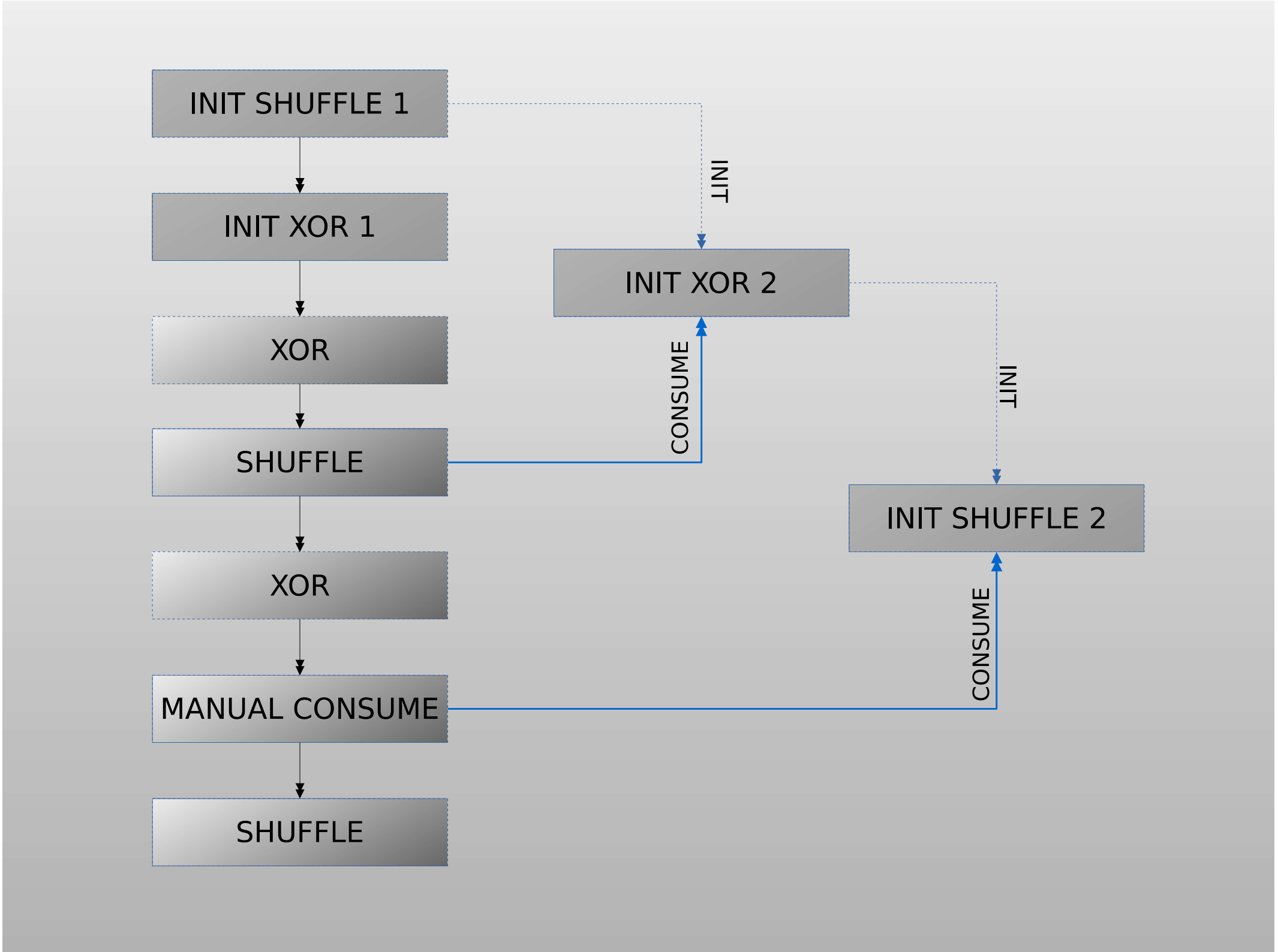
Размер массива 0x10

Смещение массива 0x118

Перестановка

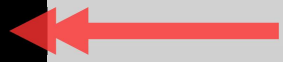
Запуск процесса-модификатора на исполнение (идентификатор процесса должен быть сохранен по смещению 0x148)

Общая схема модификации пользовательского ввода


INIT SHUFFLE 1
INIT XOR 1
XOR
SHUFFLE
XOR
MANUAL CONSUME
SHUFFLE
INIT
INIT XOR 2
CONSUME
INIT
INIT SHUFFLE 2
CONSUME

is_password_correct

```
push rbp
mov rbp, rsp
sub rsp, 0x20
mov qword [rbp - local_18h], rdi
mov byte [rbp - local_10h], 0x3c
mov byte [rbp - local_fh], 0x8e
mov byte [rbp - local_eh], 0x4f
mov byte [rbp - local_dh], 0x47
mov byte [rbp - local_ch], 0x96
mov byte [rbp - local_bh], 0x5e
mov byte [rbp - local_ah], 0x18
mov byte [rbp - local_9h], 0xde
mov byte [rbp - local_8h], 0x75
mov byte [rbp - local_7h], 0x28
mov byte [rbp - local_6h], 0x5d
mov byte [rbp - local_5h], 0x3a
mov byte [rbp - local_4h], 0xc6
mov byte [rbp - local_3h], 0xe6
mov byte [rbp - local_2h], 1
mov byte [rbp - 1], 0x85
mov rax, qword [rbp - local_18h]
lea rcx, qword [rax + 0x118]
lea rax, qword [rbp - local_10h]
mov edx, 0x10
mov rsi, rcx
mov rdi, rax
call sym.imp.memcmp ;[a]
test eax, eax
sete al
leave
ret
```



Проверка ключевой фразы заключается в сравнении модифицированного пользовательского ввода с эталонным значением

Произведя обратные перестановки и XOR-преобразования, придем к искомой ключевой фразе