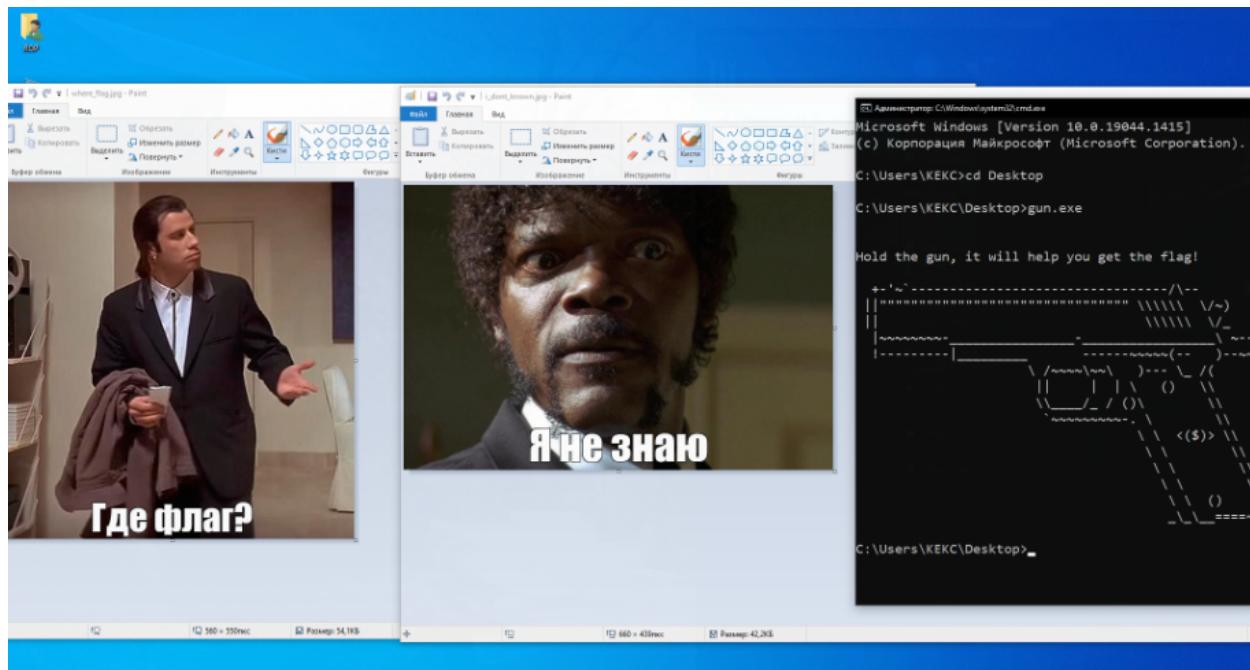




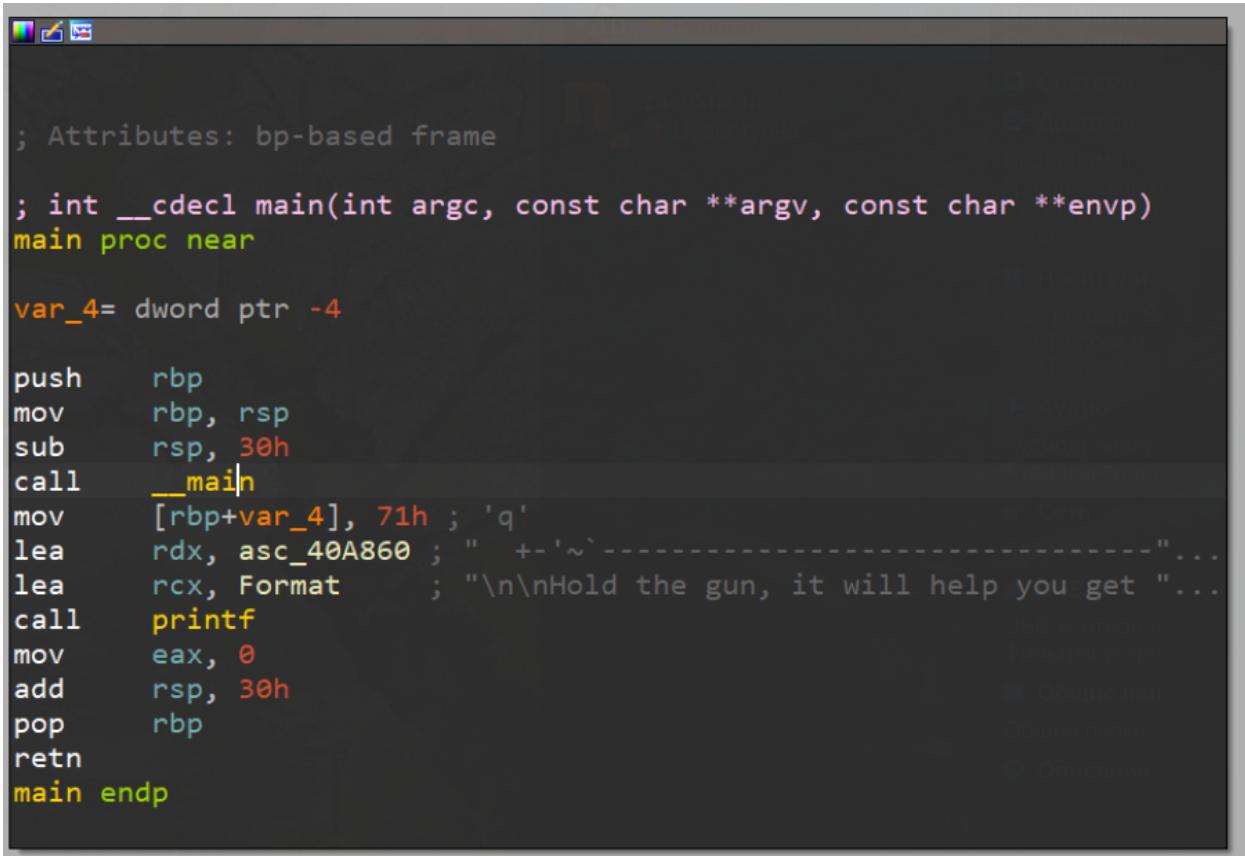
Название:	Где флаг?
Категория:	Реверс-инжиниринг
Уровень:	Средний
Очки:	400
Описание:	Я не знаю. Может быть ты сможешь найти его?
Теги:	C, патчинг
Автор:	ROP

Прохождение:

Распакуем архив, пробуем запустить.



Ничего необычного. Изучим таск в IDA.



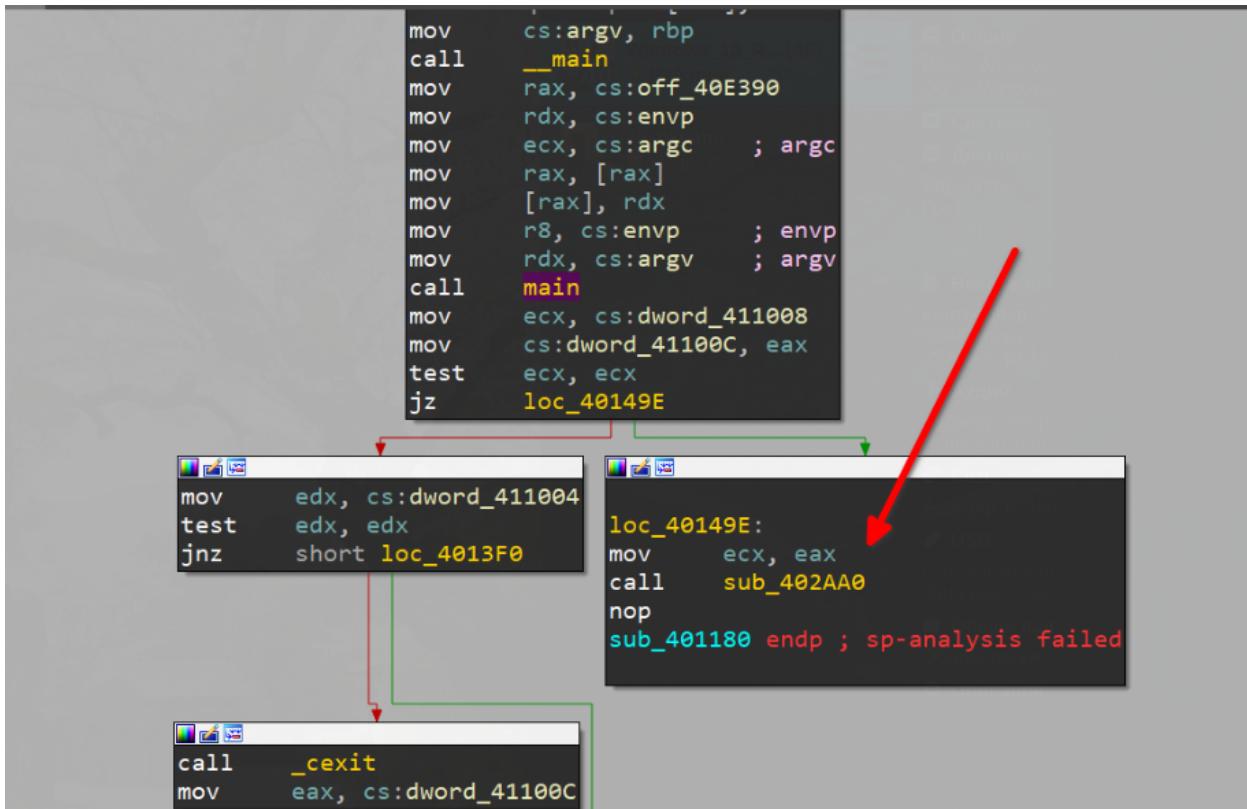
The screenshot shows the assembly view of the main function in IDA Pro. The code is as follows:

```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
main proc near

var_4= dword ptr -4

push    rbp
mov     rbp, rsp
sub    rsp, 30h
call    __main
mov     [rbp+var_4], 71h ; 'q'
lea     rdx, asc_40A860 ; " +-~`-----"
lea     rcx, Format      ; "\n\nHold the gun, it will help you get ..."
call    printf
mov     eax, 0
add    rsp, 30h
pop    rbp
retn
main endp
```

В `main` ничего необычного. Сам файл **не** статический, то есть кода функций в нём нет. Проверим, что происходит после `main`.



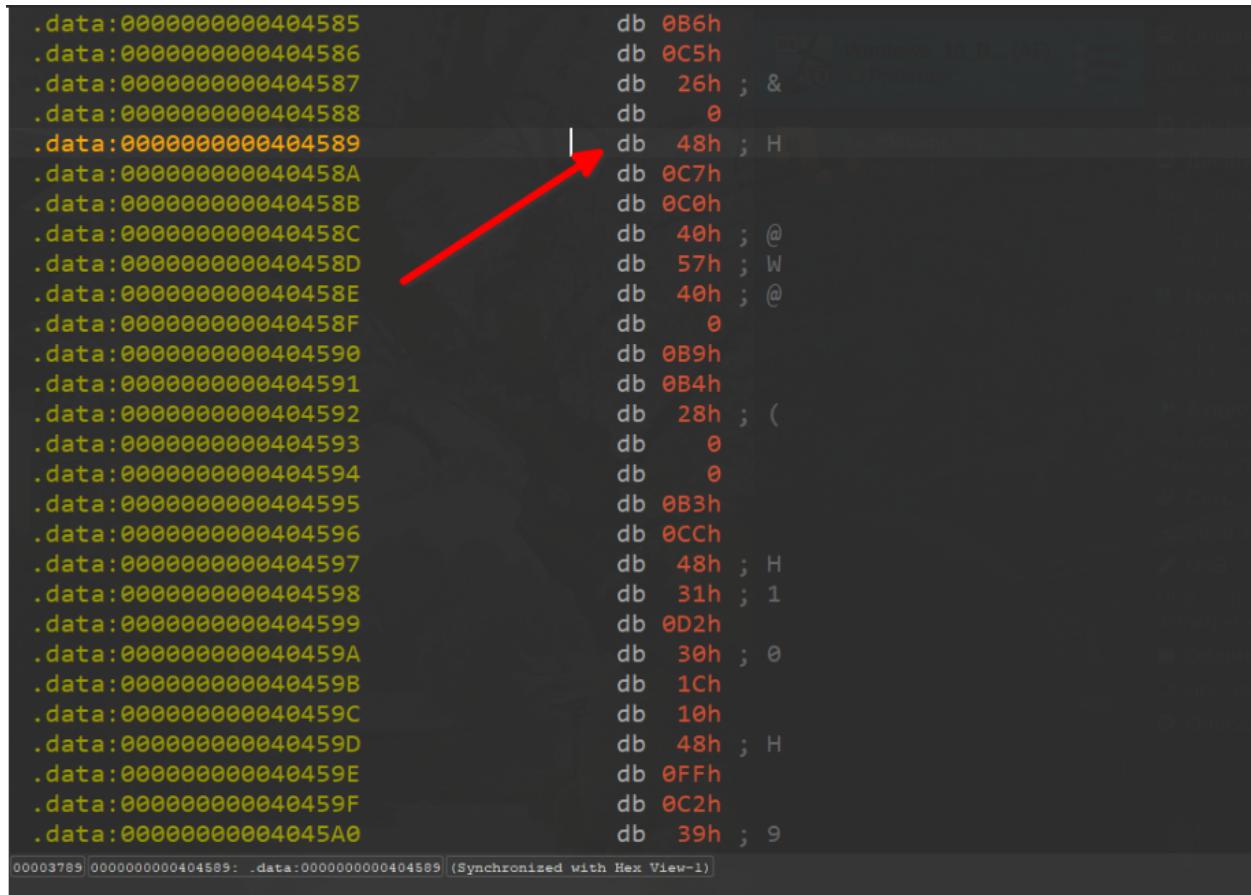
Тут должен быть `exit(0)`, но сейчас какая-то функция. Изучим её.

```

sub_402AA0 proc near
push    404589h
pop    |    rax
jmp    |    rax
sub_402AA0 endp
    
```

A screenshot of a debugger showing the assembly code for the `sub_402AA0` function. The code consists of three instructions: a `push` instruction with the value `404589h`, a `pop` instruction into `rax`, and a `jmp` instruction to `rax`. A red arrow points to the `push` instruction, highlighting it.

Переход по этому адресу через
`jmp rax`. Смотрим, куда он ведёт.



```
.data:0000000000404585 db 0B6h
.data:0000000000404586 db 0C5h ; &
.data:0000000000404587 db 26h ; &
.data:0000000000404588 db 0
.data:0000000000404589 db 48h ; H
.data:000000000040458A db 0C7h
.data:000000000040458B db 0C0h
.data:000000000040458C db 40h ; @
.data:000000000040458D db 57h ; W
.data:000000000040458E db 40h ; @
.data:000000000040458F db 0
.data:0000000000404590 db 0B9h
.data:0000000000404591 db 0B4h
.data:0000000000404592 db 28h ; (
.data:0000000000404593 db 0
.data:0000000000404594 db 0
.data:0000000000404595 db 0B3h
.data:0000000000404596 db 0CCh
.data:0000000000404597 db 48h ; H
.data:0000000000404598 db 31h ; 1
.data:0000000000404599 db 0D2h
.data:000000000040459A db 30h ; 0
.data:000000000040459B db 1Ch
.data:000000000040459C db 10h
.data:000000000040459D db 48h ; H
.data:000000000040459E db 0FFh
.data:000000000040459F db 0C2h
.data:00000000004045A0 db 39h ; 9
```

00003789|0000000000404589: .data:0000000000404589 (Synchronized with Hex View-1)

Поставим курсор сюда и нажмём С.

```
sub_404589 proc near
mov      rax, 405740h
mov      ecx, 28B4h
mov      bl, 0CCh
xor      rdx, rdx
```

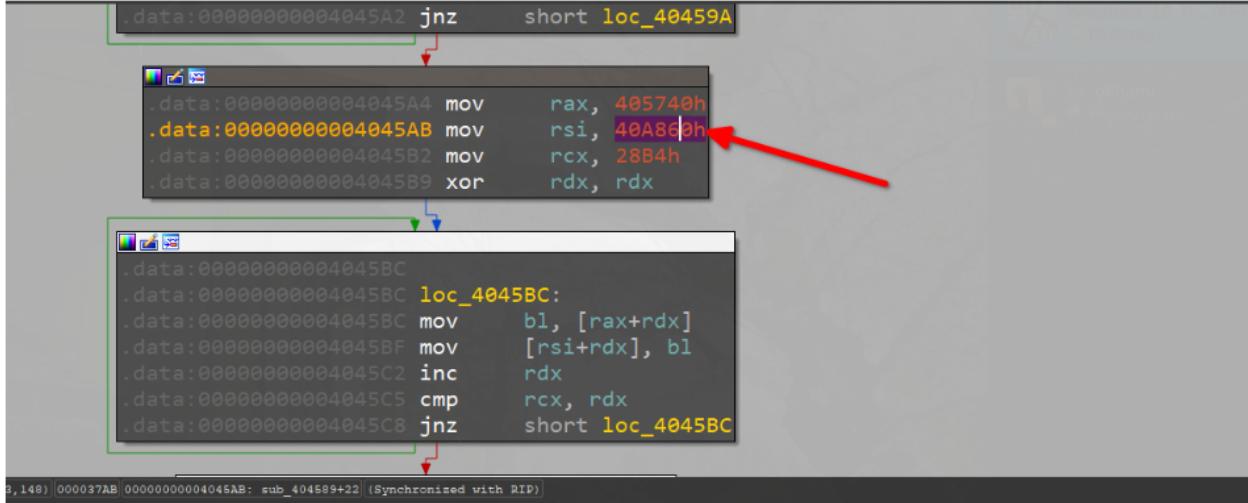
```
loc_40459A:
xor      [rax+rdx], bl
inc      rdx
cmp      edx, ecx
jnz      short loc_40459A
```

```
mov      rax, 405740h
mov      rsi, 40A860h
mov      rcx, 28B4h
xor      rdx, rdx
```

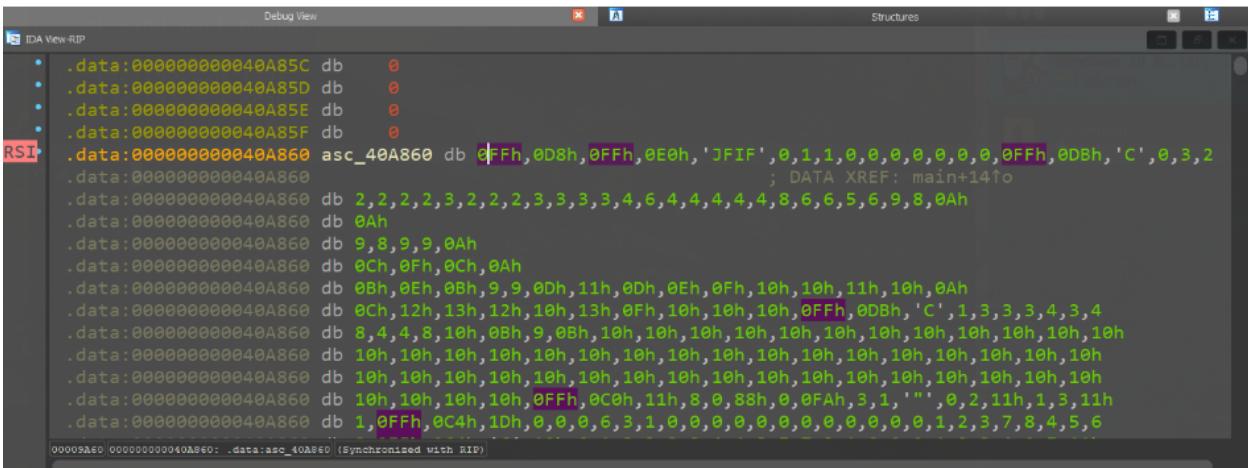
```
loc_4045BC:
mov      bl, [rax+rdx]
mov      [rsi+rdx], bl
inc      rdx
cmp      rcx, rdx
jnz      short loc_4045BC
```

```
nop
sub_404589 endp
```

Получили такой код. Он расшифровывает массив из секции `.data`. Поставим точку останова сюда и запустим отладчик.



Переходим по этому адресу - начало массива. В rcx будет его длина - 0x28B4.



Судя по тексту это файл формата JPEG. Сдампим его через Script Command (Shift+F2).

```
auto fname = "C:\\dump_mem.jpeg";
auto address = 0x40A860;    адрес начала
auto size = 0x28B4;    размер
auto file= fopen(fname, "wb");
savefile(file, 0, address, size);
fclose(file);
```

Открываем "C:\\dump_mem.jpeg". Там флаг.