

Задача 3: hocus-pocus

Дизассемблируем командой:

```
objdump -d hocus-pocus
```

```
0000000000001189 <hocus_pocus>:
```

1189:	f3 0f 1e fa	endbr64
118d:	55	push %rbp
118e:	48 89 e5	mov %rsp,%rbp
1191:	48 89 7d e8	mov %rdi,-0x18(%rbp)
1195:	48 89 75 e0	mov %rsi,-0x20(%rbp)
1199:	48 8b 45 e8	mov -0x18(%rbp),%rax
119d:	8b 00	mov (%rax),%eax
119f:	89 45 fc	mov %eax,-0x4(%rbp)
11a2:	48 8b 45 e0	mov -0x20(%rbp),%rax
11a6:	8b 10	mov (%rax),%edx
11a8:	48 8b 45 e8	mov -0x18(%rbp),%rax
11ac:	89 10	mov %edx,(%rax)
11ae:	48 8b 45 e0	mov -0x20(%rbp),%rax
11b2:	8b 55 fc	mov -0x4(%rbp),%edx
11b5:	89 10	mov %edx,(%rax)
11b7:	90	nop
11b8:	5d	pop %rbp
11b9:	c3	ret

```
00000000000011ba <holy_function>:
```

11ba:	f3 0f 1e fa	endbr64
11be:	55	push %rbp
11bf:	48 89 e5	mov %rsp,%rbp
11c2:	48 83 ec 18	sub \$0x18,%rsp
11c6:	89 7d ec	mov %edi,-0x14(%rbp)
11c9:	89 75 e8	mov %esi,-0x18(%rbp)
11cc:	8b 55 ec	mov -0x14(%rbp),%edx
11cf:	8b 45 e8	mov -0x18(%rbp),%eax
11d2:	0f af c2	imul %edx,%eax
11d5:	89 45 fc	mov %eax,-0x4(%rbp)
11d8:	8b 55 ec	mov -0x14(%rbp),%edx
11db:	8b 45 e8	mov -0x18(%rbp),%eax
11de:	39 c2	cmp %eax,%edx
11e0:	7d 36	jge 1218 <holy_function+0x5e>
11e2:	48 8d 55 e8	lea -0x18(%rbp),%rdx
11e6:	48 8d 45 ec	lea -0x14(%rbp),%rax
11ea:	48 89 d6	mov %rdx,%rsi

```

11ed:    48 89 c7                mov     %rax,%rdi
11f0:    e8 94 ff ff ff          call    1189 <hocus_pocus>
11f5:    eb 21                   jmp     1218 <holy_function+0x5e>
11f7:    8b 45 ec                mov     -0x14(%rbp),%eax
11fa:    8b 4d e8                mov     -0x18(%rbp),%ecx
11fd:    99                       cltd
11fe:    f7 f9                   idiv    %ecx
1200:    89 d0                   mov     %edx,%eax
1202:    89 45 ec                mov     %eax,-0x14(%rbp)
1205:    48 8d 55 e8             lea     -0x18(%rbp),%rdx
1209:    48 8d 45 ec             lea     -0x14(%rbp),%rax
120d:    48 89 d6                mov     %rdx,%rsi
1210:    48 89 c7                mov     %rax,%rdi
1213:    e8 71 ff ff ff          call    1189 <hocus_pocus>
1218:    8b 45 e8                mov     -0x18(%rbp),%eax
121b:    85 c0                   test    %eax,%eax
121d:    75 d8                   jne     11f7 <holy_function+0x3d>
121f:    8b 4d ec                mov     -0x14(%rbp),%ecx
1222:    8b 45 fc                mov     -0x4(%rbp),%eax
1225:    99                       cltd
1226:    f7 f9                   idiv    %ecx
1228:    c9                       leave
1229:    c3                       ret

```

000000000000122a <main>:

```

122a:    f3 0f 1e fa             endbr64
122e:    55                       push    %rbp
122f:    48 89 e5                mov     %rsp,%rbp
1232:    48 83 ec 20             sub     $0x20,%rsp
1236:    64 48 8b 04 25 28 00    mov     %fs:0x28,%rax
123d:    00 00
123f:    48 89 45 f8             mov     %rax,-0x8(%rbp)
1243:    31 c0                   xor     %eax,%eax
1245:    48 8d 55 f0             lea     -0x10(%rbp),%rdx
1249:    48 8d 45 ec             lea     -0x14(%rbp),%rax
124d:    48 89 c6                mov     %rax,%rsi
1250:    48 8d 05 ad 0d 00 00    lea     0xdad(%rip),%rax      # 2004
<_IO_stdin_used+0x4>
1257:    48 89 c7                mov     %rax,%rdi
125a:    b8 00 00 00 00          mov     $0x0,%eax
125f:    e8 2c fe ff ff          call    1090 <__isoc99_scanf@plt>
1264:    8b 55 f0                mov     -0x10(%rbp),%edx
1267:    8b 45 ec                mov     -0x14(%rbp),%eax
126a:    89 d6                   mov     %edx,%esi
126c:    89 c7                   mov     %eax,%edi

```

```

126e:      e8 47 ff ff ff      call    11ba <holy_function>
1273:      89 45 f4            mov     %eax, -0xc(%rbp)
1276:      8b 45 f4            mov     -0xc(%rbp), %eax
1279:      89 c6              mov     %eax, %esi
127b:      48 8d 05 88 0d 00 00 lea     0xd88(%rip), %rax      # 200a
<_IO_stdin_used+0xa>
1282:      48 89 c7            mov     %rax, %rdi
1285:      b8 00 00 00 00      mov     $0x0, %eax
128a:      e8 f1 fd ff ff      call    1080 <printf@plt>
128f:      b8 00 00 00 00      mov     $0x0, %eax
1294:      48 8b 55 f8          mov     -0x8(%rbp), %rdx
1298:      64 48 2b 14 25 28 00 sub     %fs:0x28, %rdx
129f:      00 00
12a1:      74 05                je      12a8 <main+0x7e>
12a3:      e8 c8 fd ff ff      call    1070 <__stack_chk_fail@plt>
12a8:      c9                  leave
12a9:      c3                  ret

```

Также можно посмотреть на код с помощью radare2:

```

└ 49: sym.hocus_pocus (int64_t arg1, int64_t arg2);
    ; arg int64_t arg1 @ rdi
    ; arg int64_t arg2 @ rsi
    ; var int64_t var_4h @ rbp-0x4
    ; var int64_t var_18h @ rbp-0x18
    ; var int64_t var_20h @ rbp-0x20
    0x00001189      f30f1efa      endbr64
    0x0000118d      55           push rbp
    0x0000118e      4889e5       mov rbp, rsp
    0x00001191      48897de8     mov qword [var_18h], rdi      ; arg1
    0x00001195      488975e0     mov qword [var_20h], rsi      ; arg2
    0x00001199      488b45e8     mov rax, qword [var_18h]
    0x0000119d      8b00         mov eax, dword [rax]
    0x0000119f      8945fc       mov dword [var_4h], eax
    0x000011a2      488b45e0     mov rax, qword [var_20h]
    0x000011a6      8b10         mov edx, dword [rax]
    0x000011a8      488b45e8     mov rax, qword [var_18h]
    0x000011ac      8910         mov dword [rax], edx
    0x000011ae      488b45e0     mov rax, qword [var_20h]
    0x000011b2      8b55fc       mov edx, dword [var_4h]
    0x000011b5      8910         mov dword [rax], edx
    0x000011b7      90           nop
    0x000011b8      5d           pop rbp
    0x000011b9      c3           ret

```

Функция hocus_pocus меняет местами arg1 и arg2

```
112: sym.holy_function (int64_t arg1, int64_t arg2);
    ; arg int64_t arg1 @ rdi
    ; arg int64_t arg2 @ rsi
    ; var int64_t var_4h @ rbp-0x4
    ; var int64_t var_14h @ rbp-0x14
    ; var int64_t var_18h @ rbp-0x18
    0x000011ba      f30f1efa      endbr64
    0x000011be      55              push rbp
    0x000011bf      4889e5          mov rbp, rsp
    0x000011c2      4883ec18        sub rsp, 0x18
    0x000011c6      897dec          mov dword [var_14h], edi    ; arg1
    0x000011c9      8975e8          mov dword [var_18h], esi    ; arg2
    0x000011cc      8b55ec          mov edx, dword [var_14h]
    0x000011cf      8b45e8          mov eax, dword [var_18h]
    0x000011d2      0fafc2          imul eax, edx
    0x000011d5      8945fc          mov dword [var_4h], eax
    0x000011d8      8b55ec          mov edx, dword [var_14h]
    0x000011db      8b45e8          mov eax, dword [var_18h]
    0x000011de      39c2            cmp edx, eax
    0x000011e0      7d36            jge 0x1218
    0x000011e2      488d55e8        lea rdx, [var_18h]
    0x000011e6      488d45ec        lea rax, [var_14h]
    0x000011ea      4889d6          mov rsi, rdx
    0x000011ed      4889c7          mov rdi, rax
    0x000011f0      e894ffffff      call sym.hocus_pocus
    0x000011f5      eb21            jmp 0x1218
    0x000011f7      8b45ec          mov eax, dword [var_14h]
    0x000011fa      8b4de8          mov ecx, dword [var_18h]
    0x000011fd      99              cdq
    0x000011fe      f7f9            idiv ecx
    0x00001200      89d0            mov eax, edx
    0x00001202      8945ec          mov dword [var_14h], eax
    0x00001205      488d55e8        lea rdx, [var_18h]
    0x00001209      488d45ec        lea rax, [var_14h]
    0x0000120d      4889d6          mov rsi, rdx
    0x00001210      4889c7          mov rdi, rax
    0x00001213      e871ffffff      call sym.hocus_pocus
    ; CODE XREF from sym.holy_function @ 0x11f5(x)
    0x00001218      8b45e8          mov eax, dword [var_18h]
    0x0000121b      85c0            test eax, eax
    0x0000121d      75d8            jne 0x11f7
    0x0000121f      8b4dec          mov ecx, dword [var_14h]
    0x00001222      8b45fc          mov eax, dword [var_4h]
    0x00001225      99              cdq
```

0x00001226	f7f9	idiv ecx
0x00001228	c9	leave
0x00001229	c3	ret

Функция holy_function считает наименьшее общее кратное (НОК) с помощью подсчета наибольшего общего делителя (НОД)

```

128: int main (int argc, char **argv, char **envp);
    ; var int64_t var_8h @ rbp-0x8
    ; var int64_t var_ch @ rbp-0xc
    ; var int64_t var_10h @ rbp-0x10
    ; var int64_t var_14h @ rbp-0x14
    0x0000122a      f30f1efa      endbr64
    0x0000122e      55              push rbp
    0x0000122f      4889e5          mov rbp, rsp
    0x00001232      4883ec20        sub rsp, 0x20
    0x00001236      64488b042528.   mov rax, qword fs:[0x28]
    0x0000123f      488945f8        mov qword [var_8h], rax
    0x00001243      31c0            xor eax, eax
    0x00001245      488d55f0        lea rdx, [var_10h]
    0x00001249      488d45ec        lea rax, [var_14h]
    0x0000124d      4889c6          mov rsi, rax
    0x00001250      488d05ad0d00.   lea rax, str._d__d ; 0x2004 ; "%d %d"
    0x00001257      4889c7          mov rdi, rax
    0x0000125a      b800000000      mov eax, 0
    0x0000125f      e82cfeffff      call sym.imp.__isoc99_scanf ; scanf
    0x00001264      8b55f0          mov edx, dword [var_10h]
    0x00001267      8b45ec          mov eax, dword [var_14h]
    0x0000126a      89d6            mov esi, edx
    0x0000126c      89c7            mov edi, eax
    0x0000126e      e847ffffff      call sym.holy_function
    0x00001273      8945f4          mov dword [var_ch], eax
    0x00001276      8b45f4          mov eax, dword [var_ch]
    0x00001279      89c6            mov esi, eax
    0x0000127b      488d05880d00.   lea rax, [0x0000200a] ; "%d\n"
    0x00001282      4889c7          mov rdi, rax
    0x00001285      b800000000      mov eax, 0
    0x0000128a      e8f1fdffff      call sym.imp.printf ; printf
    0x0000128f      b800000000      mov eax, 0
    0x00001294      488b55f8        mov rdx, qword [var_8h]
    0x00001298      64482b142528.   sub rdx, qword fs:[0x28]
    < 0x000012a1      7405            je 0x12a8
    0x000012a3      e8c8fdffff      call sym.imp.__stack_chk_fail ; void
__stack_chk_fail(void)
    > 0x000012a8      c9              leave

```

```
L          0x000012a9      c3          ret
```

Функция main считывает arg1 и arg2, вызывает функцию holy_function() с параметрами arg1 и arg2. В конце с помощью printf печатается результат работы holy_function()

Таким образом, программа печатает НОК

Примеры входных данных:

input_03_42: 1 42 -> 42

input_03_0: 1 0 -> 0

input_03_int_min: 1 -2147483648 (INT_MIN) -> -2147483648 (INT_MIN)

input_03_terminate: 0 0 -> завершится аварийно

```
--- SIGFPE (Floating point exception) ---  
+++ killed by SIGFPE +++
```

Потому что произойдет попытка деления на 0

Статус, с которым завершится программа, можно узнать, например, так:

```
./hocus-pocus < input_03_terminate; echo $?
```

Получим:

```
[1]    932 floating point exception ./hocus-pocus < input_03_terminate  
136
```

Код завершения: 136

Эквивалентный код на си: hocus_pocus.c