tshering@TD:/mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001$ gedit answers.txt

Unable to init server: Could not connect: Connection refused

(gedit:2084): Gtk-WARNING \*\*: 20:50:42.690: cannot open display:

tshering@TD:/mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001$ gedit answers.txt

Unable to init server: Could not connect: Connection refused

(gedit:2086): Gtk-WARNING \*\*: 20:51:01.082: cannot open display:

tshering@TD:/mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001$ nano answers.txt

Use "fg" to return to nano.

[2]+ Stopped nano answers.txt

tshering@TD:/mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001$ clear

tshering@TD:/mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001$ gdb bomb

GNU gdb (Ubuntu 8.1.1-0ubuntu1) 8.1.1

Copyright (C) 2018 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:

<http://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 bomb...done.

(gdb) b phase\_3

Breakpoint 1 at 0x400f15

(gdb)

Note: breakpoint 1 also set at pc 0x400f15.

Breakpoint 2 at 0x400f15

(gdb) b explode\_bomb

Breakpoint 3 at 0x40143d

(gdb) r

Starting program: /mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001/bomb

Welcome to my fiendish little bomb. You have 6 phases with

which to blow yourself up. Have a nice day!

The moon unit will be divided into two divisions.

Phase 1 defused. How about the next one?

0 1 1 2 3 5

That's number 2. Keep going!

Hi **// test input**

Breakpoint 1, 0x0000000000400f15 in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

=> 0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi **// it stores the input format so we take the address**

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) x/s 0x4025cf **// we want to get the string stored in the address 0x4025cf so we use x/s followed by the address.**

0x4025cf: "%d %d" **// it shows that input consist of 2 integer**

(gdb) r

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001/bomb

Welcome to my fiendish little bomb. You have 6 phases with

which to blow yourself up. Have a nice day!

^V

^C

Program received signal SIGINT, Interrupt.

0x00007fffff110151 in \_\_GI\_\_\_libc\_read (fd=0, buf=0x605270, nbytes=4096) at ../sysdeps/unix/sysv/linux/read.c:27

27 ../sysdeps/unix/sysv/linux/read.c: No such file or directory.

(gdb) r

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001/bomb

Welcome to my fiendish little bomb. You have 6 phases with

which to blow yourself up. Have a nice day!

The moon unit will be divided into two divisions.

Phase 1 defused. How about the next one?

0 1 1 2 3 5

That's number 2. Keep going!

2 98

Breakpoint 1, 0x0000000000400f15 in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

=> 0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax **// address of eax is taken since it stores the correct input. Cmp as we want want to compare the correct input.**

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) until \* 0x0000000000400f3b **//To get inside this particular address that is eax as it stores the correct input**

0x0000000000400f3b in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

=> 0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) i r

rax 0x2 2 **//The first correct input is 2 as eax 32 bit stores the input**

rbx 0x0 0

rcx 0x0 0

rdx 0x7ffffffee324 140737488282404

rsi 0x0 0

rdi 0x7ffffffedcb0 140737488280752

rbp 0x4021f0 0x4021f0 <\_\_libc\_csu\_init>

rsp 0x7ffffffee320 0x7ffffffee320

r8 0x0 0

r9 0x0 0

r10 0x7fffff19ec40 140737473276992

r11 0x4025d4 4203988

r12 0x400c60 4197472

r13 0x7ffffffee420 140737488282656

r14 0x0 0

r15 0x0 0

rip 0x400f3b 0x400f3b <phase\_3+38>

eflags 0x206 [ PF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) ni

0x0000000000400f3e in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

=> 0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) ni

0x0000000000400f45 in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

=> 0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) ni

0x0000000000400f49 in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

=> 0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) ni

0x0000000000400f4b in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

=> 0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax **// address of eax is taken since it stores the correct input**

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) until \*0x0000000000400fb6

0x0000000000400fb6 in phase\_3 ()

(gdb) disas

Dump of assembler code for function phase\_3:

0x0000000000400f15 <+0>: sub $0x18,%rsp

0x0000000000400f19 <+4>: mov %fs:0x28,%rax

0x0000000000400f22 <+13>: mov %rax,0x8(%rsp)

0x0000000000400f27 <+18>: xor %eax,%eax

0x0000000000400f29 <+20>: lea 0x4(%rsp),%rcx

0x0000000000400f2e <+25>: mov %rsp,%rdx

0x0000000000400f31 <+28>: mov $0x4025cf,%esi

0x0000000000400f36 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000400f3b <+38>: cmp $0x1,%eax

0x0000000000400f3e <+41>: jg 0x400f45 <phase\_3+48>

0x0000000000400f40 <+43>: callq 0x40143d <explode\_bomb>

0x0000000000400f45 <+48>: cmpl $0x7,(%rsp)

0x0000000000400f49 <+52>: ja 0x400fa6 <phase\_3+145>

0x0000000000400f4b <+54>: mov (%rsp),%eax

0x0000000000400f4e <+57>: jmpq \*0x402440(,%rax,8)

0x0000000000400f55 <+64>: mov $0x134,%eax

0x0000000000400f5a <+69>: jmp 0x400f61 <phase\_3+76>

0x0000000000400f5c <+71>: mov $0x0,%eax

0x0000000000400f61 <+76>: sub $0x85,%eax

0x0000000000400f66 <+81>: jmp 0x400f6d <phase\_3+88>

0x0000000000400f68 <+83>: mov $0x0,%eax

0x0000000000400f6d <+88>: add $0x201,%eax

0x0000000000400f72 <+93>: jmp 0x400f79 <phase\_3+100>

0x0000000000400f74 <+95>: mov $0x0,%eax

0x0000000000400f79 <+100>: sub $0x68,%eax

0x0000000000400f7c <+103>: jmp 0x400f83 <phase\_3+110>

0x0000000000400f7e <+105>: mov $0x0,%eax

0x0000000000400f83 <+110>: add $0x68,%eax

---Type <return> to continue, or q <return> to quit---

0x0000000000400f86 <+113>: jmp 0x400f8d <phase\_3+120>

0x0000000000400f88 <+115>: mov $0x0,%eax

0x0000000000400f8d <+120>: sub $0x68,%eax

0x0000000000400f90 <+123>: jmp 0x400f97 <phase\_3+130>

0x0000000000400f92 <+125>: mov $0x0,%eax

0x0000000000400f97 <+130>: add $0x68,%eax

0x0000000000400f9a <+133>: jmp 0x400fa1 <phase\_3+140>

0x0000000000400f9c <+135>: mov $0x0,%eax

0x0000000000400fa1 <+140>: sub $0x68,%eax

0x0000000000400fa4 <+143>: jmp 0x400fb0 <phase\_3+155>

0x0000000000400fa6 <+145>: callq 0x40143d <explode\_bomb>

0x0000000000400fab <+150>: mov $0x0,%eax

0x0000000000400fb0 <+155>: cmpl $0x5,(%rsp)

0x0000000000400fb4 <+159>: jg 0x400fbc <phase\_3+167>

=> 0x0000000000400fb6 <+161>: cmp 0x4(%rsp),%eax

0x0000000000400fba <+165>: je 0x400fc1 <phase\_3+172>

0x0000000000400fbc <+167>: callq 0x40143d <explode\_bomb>

0x0000000000400fc1 <+172>: mov 0x8(%rsp),%rax

0x0000000000400fc6 <+177>: xor %fs:0x28,%rax

0x0000000000400fcf <+186>: je 0x400fd6 <phase\_3+193>

0x0000000000400fd1 <+188>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000400fd6 <+193>: add $0x18,%rsp

0x0000000000400fda <+197>: retq

End of assembler dump.

(gdb) i r **// it is used to check the input register**

rax 0x199 409 **// Second correct input**

rbx 0x0 0

rcx 0x0 0

rdx 0x7ffffffee324 140737488282404

rsi 0x0 0

rdi 0x7ffffffedcb0 140737488280752

rbp 0x4021f0 0x4021f0 <\_\_libc\_csu\_init>

rsp 0x7ffffffee320 0x7ffffffee320

r8 0x0 0

r9 0x0 0

r10 0x7fffff19ec40 140737473276992

r11 0x4025d4 4203988

r12 0x400c60 4197472

r13 0x7ffffffee420 140737488282656

r14 0x0 0

r15 0x0 0

rip 0x400fb6 0x400fb6 <phase\_3+161>

eflags 0x293 [ CF AF SF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) r

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001/bomb

Welcome to my fiendish little bomb. You have 6 phases with

which to blow yourself up. Have a nice day!

The moon unit will be divided into two divisions.

Phase 1 defused. How about the next one?

0 1 1 2 3 5

That's number 2. Keep going!

2 409

Breakpoint 1, 0x0000000000400f15 in phase\_3 ()

(gdb) i b

Num Type Disp Enb Address What

1 breakpoint keep y 0x0000000000400f15 <phase\_3>

breakpoint already hit 1 time

2 breakpoint keep y 0x0000000000400f15 <phase\_3>

breakpoint already hit 1 time

3 breakpoint keep y 0x000000000040143d <explode\_bomb>

(gdb) del 1

(gdb) del 2

(gdb) i b

Num Type Disp Enb Address What

3 breakpoint keep y 0x000000000040143d <explode\_bomb>

(gdb) r

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /mnt/d/one drive/OneDrive/Desktop/12190094/Assignment 1/bomb001/bomb

Welcome to my fiendish little bomb. You have 6 phases with

which to blow yourself up. Have a nice day!

The moon unit will be divided into two divisions.

Phase 1 defused. How about the next one?

0 1 1 2 3 5

That's number 2. Keep going!

2 409

Halfway there! **// phase 3 defused**