(gdb) b phase\_4

Breakpoint 1 at 0x40100e

(gdb) r defuse.txt

Starting program: /mnt/e/GCIT/Semester V/ITP304-CSI/Assignment/Assignment1/Assignment 1\_2/Assignment 1/bomb001/bomb defuse.txt

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

which to blow yourself up. Have a nice day!

Phase 1 defused. How about the next one?

That's number 2. Keep going!

Halfway there!

8 35

Breakpoint 1, 0x000000000040100e in phase\_4 ()

(gdb) disas

Dump of assembler code for function phase\_4:

=> 0x000000000040100e <+0>: sub $0x18,%rsp //makes stack frame

0x0000000000401012 <+4>: mov %fs:0x28,%rax

0x000000000040101b <+13>: mov %rax,0x8(%rsp)

0x0000000000401020 <+18>: xor %eax,%eax // eax = eax ^ eax

0x0000000000401022 <+20>: lea 0x4(%rsp),%rcx // rcx = rsp - 0x4

0x0000000000401027 <+25>: mov %rsp,%rdx // rdx = rsp

0x000000000040102a <+28>: mov $0x4025cf,%esi //format of answer: %d %d

0x000000000040102f <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x0000000000401034 <+38>: cmp $0x2,%eax // if NOT equal to 2 jump to explode bomb

0x0000000000401037 <+41>: jne 0x40103f <phase\_4+49>

0x0000000000401039 <+43>: cmpl $0xe,(%rsp) //0xe =14

0x000000000040103d <+47>: jbe 0x401044 <phase\_4+54> //if first input(Destination) < 14(Source), skip bomb

0x000000000040103f <+49>: callq 0x40143d <explode\_bomb>

0x0000000000401044 <+54>: mov $0xe,%edx //%edx = 14

0x0000000000401049 <+59>: mov $0x0,%esi //%esi = 0

0x000000000040104e <+64>: mov (%rsp),%edi //%edi = first input

0x0000000000401051 <+67>: callq 0x400fdb <func4> //calls <func4>

0x0000000000401056 <+72>: cmp $0x23,%eax

0x0000000000401059 <+75>: jne 0x401062 <phase\_4+84> // if NOT equal to $0x23 jump to explode bomb

0x000000000040105b <+77>: cmpl $0x23,0x4(%rsp)

0x0000000000401060 <+82>: je 0x401067 <phase\_4+89> // if NOT equal to $0x23 jump to explode bomb

0x0000000000401062 <+84>: callq 0x40143d <explode\_bomb>

0x0000000000401067 <+89>: mov 0x8(%rsp),%rax // rax = 0x8(%rsp)

0x000000000040106c <+94>: xor %fs:0x28,%rax // %rax = %fs ^ 0x28

0x0000000000401075 <+103>: je 0x40107c <phase\_4+110>

0x0000000000401077 <+105>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x000000000040107c <+110>: add $0x18,%rsp // %rsp = $0x18 + %rsp

0x0000000000401080 <+114>: retq

End of assembler dump.

(gdb) u\*0x000000000040102a

(gdb) x/s 0x4025cf

0x4025cf: "%d %d"

(gdb) i r

rax 0x2 2

rbx 0x7ffffffee138 140737488281912

rcx 0x0 0

rdx 0x7ffffffee024 140737488281636

rsi 0x0 0

rdi 0x7ffffffed9d0 140737488280016

rbp 0x0 0x0

rsp 0x7ffffffee020 0x7ffffffee020

r8 0xffffffff 4294967295

r9 0x0 0

r10 0x7fffff74eac0 140737479240384

r11 0x0 0

r12 0x400c60 4197472

r13 0x7ffffffee130 140737488281904

r14 0x0 0

r15 0x0 0

rip 0x401034 0x401034 <phase\_4+38>

eflags 0x206 [ PF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) x/d 0x7ffffffee020

0x7ffffffee020: 8

(gdb) si

0x0000000000400fdb in func4 ()

(gdb) disas

Dump of assembler code for function func4:

=> 0x0000000000400fdb <+0>: push %rbx //make stack frame

0x0000000000400fdc <+1>: mov %edx,%eax // eax = edx

0x0000000000400fde <+3>: sub %esi,%eax // eax = eax - esi

0x0000000000400fe0 <+5>: mov %eax,%ebx // ebx = eax

0x0000000000400fe2 <+7>: shr $0x1f,%ebx //0x1f = 31; ebx = 14 >>31; returns most sigfig bit --> 0

0x0000000000400fe5 <+10>: add %ebx,%eax // eax = eax + ebx

0x0000000000400fe7 <+12>: sar %eax //eax = eax >> 1; 14 >> 1 --> same as eax/2

0x0000000000400fe9 <+14>: lea (%rax,%rsi,1),%ebx //ebx = eax + rsi --> ebx = 7

0x0000000000400fec <+17>: cmp %edi,%ebx //first input VS ecx

0x0000000000400fee <+19>: jle 0x400ffc <func4+33> //if 7 <= first input, jump pass recursion call

0x0000000000400ff0 <+21>: lea -0x1(%rbx),%edx //edx = rbx - 1

0x0000000000400ff3 <+24>: callq 0x400fdb <func4> //back to beginning

0x0000000000400ff8 <+29>: add %ebx,%eax // eax = eax + ebx

0x0000000000400ffa <+31>: jmp 0x40100c <func4+49>

0x0000000000400ffc <+33>: mov %ebx,%eax // eax = ebx

0x0000000000400ffe <+35>: cmp %edi,%ebx first input vs 7

0x0000000000401000 <+37>: jge 0x40100c <func4+49> //%eax >= edi --> 7 >= first input

0x0000000000401002 <+39>: lea 0x1(%rbx),%esi

0x0000000000401005 <+42>: callq 0x400fdb <func4>

0x000000000040100a <+47>: add %ebx,%eax

0x000000000040100c <+49>: pop %rbx

0x000000000040100d <+50>: retq

End of assembler dump.

(gdb) i r

rax 0x7 7

rbx 0x7 7

rcx 0x0 0

rdx 0xe 14

rsi 0x0 0

rdi 0x8 8

rbp 0x0 0x0

rsp 0x7ffffffee010 0x7ffffffee010

r8 0xffffffff 4294967295

r9 0x0 0

r10 0x7fffff74eac0 140737479240384

r11 0x0 0

r12 0x400c60 4197472

r13 0x7ffffffee130 140737488281904

r14 0x0 0

r15 0x0 0

rip 0x400fec 0x400fec <func4+17>

eflags 0x202 [ IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) p/x \*(int\*)(0x4+$rsp)

$1 = 0x23s

(gdb) u\*0x0000000000401080

main (argc=<optimized out>, argv=<optimized out>) at bomb.c:96

96 phase\_defused();

(gdb)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Conculsion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(0x2 != eax){

explode()

}

else{

if(0xe > rsp){

edx = 14

esi = 0

func(){

if(ebc <= edi){

eax = ebx

if(ebx >= edi){

eax = eax + ebx

return eax

}

}

else{

edx = rbx - 1

func()

}

}

return eax

if(eax != 0x23){

explode()

}

else{

if(0x23 == (0x4+rsp)){

defuse()

}

else{

explode()

}

}

}

else{

explode()

}

}