(gdb) b phase\_5

Breakpoint 1 at 0x401081

(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!

So you got that one. Try this one.

5 115

Breakpoint 1, 0x0000000000401081 in phase\_5 ()

(gdb) disas

Dump of assembler code for function phase\_5:

=> 0x0000000000401081 <+0>: sub $0x18,%rsp //rsp = rsp - 24

0x0000000000401085 <+4>: mov %fs:0x28,%rax

0x000000000040108e <+13>: mov %rax,0x8(%rsp)

0x0000000000401093 <+18>: xor %eax,%eax // eax = eax ^ eax

0x0000000000401095 <+20>: lea 0x4(%rsp),%rcx //rcx = rsp - 0x4

0x000000000040109a <+25>: mov %rsp,%rdx // rdx = rsp

0x000000000040109d <+28>: mov $0x4025cf,%esi //esi = "%d %d"

0x00000000004010a2 <+33>: callq 0x400bb0 <\_\_isoc99\_sscanf@plt>

0x00000000004010a7 <+38>: cmp $0x1,%eax

0x00000000004010aa <+41>: jg 0x4010b1 <phase\_5+48> //(eax > 1) it basically checks whether user input is greater 1 or not. if user input is less than eax < 1 then bomb explode.

0x00000000004010ac <+43>: callq 0x40143d <explode\_bomb>

0x00000000004010b1 <+48>: mov (%rsp),%eax // eax = rsp

0x00000000004010b4 <+51>: and $0xf,%eax // eax = 0xf & eax

0x00000000004010b7 <+54>: mov %eax,(%rsp) // rsp = eax

0x00000000004010ba <+57>: cmp $0xf,%eax

0x00000000004010bd <+60>: je 0x4010ee <phase\_5+109> // if(0xf == eax) then bomb explode

0x00000000004010bf <+62>: mov $0x0,%ecx // ecx = 0

0x00000000004010c4 <+67>: mov $0x0,%edx // edx = 0

0x00000000004010c9 <+72>: add $0x1,%edx //edx = 1

0x00000000004010cc <+75>: cltq //sign extend eax to quadword (rax)

0x00000000004010ce <+77>: mov 0x402480(,%rax,4),%eax

0x00000000004010d5 <+84>: add %eax,%ecx // ecx = ecx + eax

0x00000000004010d7 <+86>: cmp $0xf,%eax

0x00000000004010da <+89>: jne 0x4010c9 <phase\_5+72> // if (eax != 0xf) goto phase\_5+72 (inc edx)

0x00000000004010dc <+91>: movl $0xf,(%rsp)

0x00000000004010e3 <+98>: cmp $0xf,%edx // if (edx != 15) explode\_bomb()

0x00000000004010e6 <+101>: jne 0x4010ee <phase\_5+109>

0x00000000004010e8 <+103>: cmp 0x4(%rsp),%ecx

0x00000000004010ec <+107>: je 0x4010f3 <phase\_5+114> if (ecx == \*(rsp + 4)) goto phase\_5+114

0x00000000004010ee <+109>: callq 0x40143d <explode\_bomb>

0x00000000004010f3 <+114>: mov 0x8(%rsp),%rax

0x00000000004010f8 <+119>: xor %fs:0x28,%rax

0x0000000000401101 <+128>: je 0x401108 <phase\_5+135>

0x0000000000401103 <+130>: callq 0x400b00 <\_\_stack\_chk\_fail@plt>

0x0000000000401108 <+135>: add $0x18,%rsp // rsp = rsp + 0x18

0x000000000040110c <+139>: retq // return

End of assembler dump.

(gdb) u\*0x000000000040109d

(gdb) x/s 0x4025cf

0x4025cf: "%d %d"

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) i r

rax 0xc 12

rbx 0x7ffffffee138 140737488281912

rcx 0xc 12

rdx 0x1 1

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 0x4010d7 0x4010d7 <phase\_5+86>

eflags 0x206 [ PF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) i r

rax 0x2 2

rbx 0x7ffffffee138 140737488281912

rcx 0x50 80

rdx 0xc 12

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 0x4010d7 0x4010d7 <phase\_5+86>

eflags 0x216 [ PF AF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) u\*0x00000000004010d7

0x00000000004010d7 in phase\_5 ()

(gdb) i r

rax 0xf 15

rbx 0x7ffffffee138 140737488281912

rcx 0x73 115

rdx 0xf 15

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 0x4010d7 0x4010d7 <phase\_5+86>

eflags 0x212 [ AF IF ]

cs 0x33 51

ss 0x2b 43

ds 0x0 0

es 0x0 0

fs 0x0 0

gs 0x0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Solution code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int numArray[15] = {10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6};

if(eax > 0x1){

eax = rsp

eax = 0xf & eax

rsp = eax

if(0xf != eax){

ecx = 0

edx = 0

do {

edx = (edx + 0x1)

eax = numArray[eax]

ecx = (ecx + eax)

}while (eax != 0xf)

if(0xf != eax){

explode()

}

else{

if(ecx == \*(rsp+4)){

defuse()

}

else{

explode()

}

}

}

else{

explode()

}

}

else{

explode()

}