

Phase 03

To begin with 3rd phase, first of all, break point was set to avoid exploding of bomb in phase 3. The program was run and random value was inputted. Then the program code was disassembled and first function was `<_isoc99_sscanf@plt>` and basically through this function we can understand that we have to provide user input. Before executing the function, there is a value stored in hexadecimal form i.e., `$0x4025af`. To view this value `"x/s 0x4025af"` command was executed and it displayed `"%d %d"`. From this we can understand that this program asks user to input two integers.

```
Terminal
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...
(gdb) b phase_3
Breakpoint 1 at 0x400f15
(gdb) r text.txt
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003/bomb text.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?
0 1 1 2 3 5
That's number 2. Keep going!
hiiii

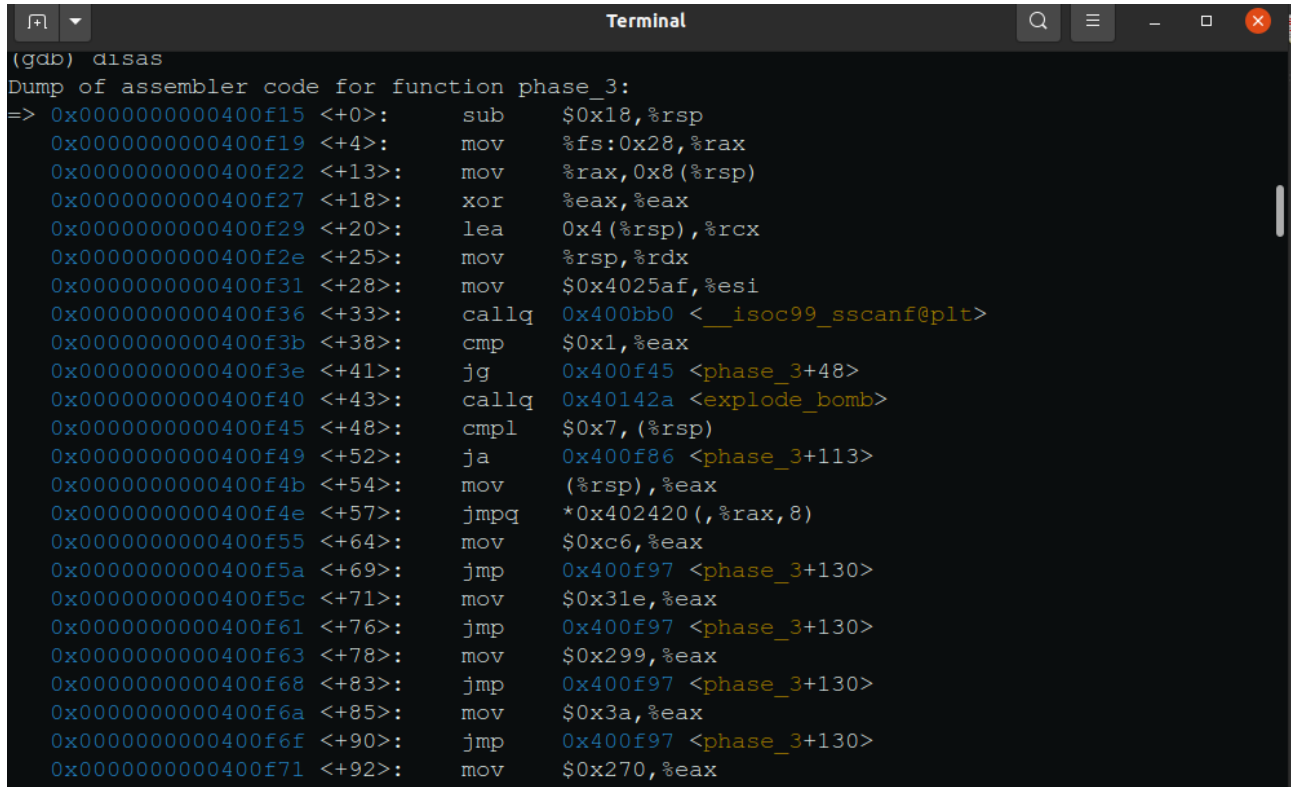
Breakpoint 1, 0x000000000400f15 in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
=> 0x000000000400f15 <+0>:      sub     $0x18,%rsp
```

```
Terminal
0x000000000400f12 <+13>:      callq  0x400800 <__stack_chk_guard@plt>
0x000000000400fb7 <+162>:     add     $0x18,%rsp
0x000000000400fbb <+166>:     retq

End of assembler dump.
(gdb) x/s 0x4025af
0x4025af:      "%d %d"
(gdb) r text.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003/bomb text.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?
0 1 1 2 3 5
That's number 2. Keep going!
2 899

Breakpoint 1, 0x000000000400f15 in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
=> 0x000000000400f15 <+0>:      sub     $0x18,%rsp
0x000000000400f19 <+4>:      mov     %fs:0x28,%rax
0x000000000400f22 <+13>:     mov     %rax,0x8(%rsp)
0x000000000400f27 <+18>:     xor     %eax,%eax
0x000000000400f29 <+20>:     lea     0x4(%rsp),%rcx
0x000000000400f2e <+25>:     mov     %rsp,%rdx
```

Now let's discuss about function in the program code. In phase 3, predefined value i.e., 1 is being compared with the value of %eax i.e., 2. Followed by compare we have one condition that value of %eax should be greater than 1 and equal to 1. If this condition is not fulfilled then the bomb will get exploded. As condition happens to be true, we have move to next line i.e., 48. Again same value is compared with 7 but this time the value is stored in (%rsp). As soon as it is compared, it executes next line and we have condition stating that user input value should be equal 7 and less than 7.



```
(gdb) disas
Dump of assembler code for function phase_3:
=> 0x000000000400f15 <+0>:      sub     $0x18,%rsp
    0x000000000400f19 <+4>:      mov     %fs:0x28,%rax
    0x000000000400f22 <+13>:     mov     %rax,0x8(%rsp)
    0x000000000400f27 <+18>:     xor     %eax,%eax
    0x000000000400f29 <+20>:     lea     0x4(%rsp),%rcx
    0x000000000400f2e <+25>:     mov     %rsp,%rdx
    0x000000000400f31 <+28>:     mov     $0x4025af,%esi
    0x000000000400f36 <+33>:     callq  0x400bb0 <__isoc99_sscanf@plt>
    0x000000000400f3b <+38>:     cmp     $0x1,%eax
    0x000000000400f3e <+41>:     jg      0x400f45 <phase_3+48>
    0x000000000400f40 <+43>:     callq  0x40142a <explode_bomb>
    0x000000000400f45 <+48>:     cmpl    $0x7,(%rsp)
    0x000000000400f49 <+52>:     ja      0x400f86 <phase_3+113>
    0x000000000400f4b <+54>:     mov     (%rsp),%eax
    0x000000000400f4e <+57>:     jmpq    *0x402420(,%rax,8)
    0x000000000400f55 <+64>:     mov     $0xc6,%eax
    0x000000000400f5a <+69>:     jmp     0x400f97 <phase_3+130>
    0x000000000400f5c <+71>:     mov     $0x31e,%eax
    0x000000000400f61 <+76>:     jmp     0x400f97 <phase_3+130>
    0x000000000400f63 <+78>:     mov     $0x299,%eax
    0x000000000400f68 <+83>:     jmp     0x400f97 <phase_3+130>
    0x000000000400f6a <+85>:     mov     $0x3a,%eax
    0x000000000400f6f <+90>:     jmp     0x400f97 <phase_3+130>
    0x000000000400f71 <+92>:     mov     $0x270,%eax
```

From this program code we can understand the first integer should be between 1 and 7. if it does not meet this condition then the bomb will explode. As our condition is fulfilled, now it will execute next instruction. In next instruction program code have several operation like sub, mov, lea and then compare. After executing all the operation in program code, now second digit or integer is being compared with value of %eax, if it matches then bomb defuses else the bomb will explode. As it doesn't match the bomb has exploded as shown below. But by this time we knew the value of second integer when 1st value is 2. Thus the value for second integer is 798 when 1st integer is 2.

```
Terminal
0x0000000000400f6f <+90>: jmp 0x400f97 <phase_3+130>
0x0000000000400f71 <+92>: mov $0x270,%eax
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f76 <+97>: jmp 0x400f97 <phase_3+130>
0x0000000000400f78 <+99>: mov $0x10b,%eax
0x0000000000400f7d <+104>: jmp 0x400f97 <phase_3+130>
0x0000000000400f7f <+106>: mov $0x80,%eax
0x0000000000400f84 <+111>: jmp 0x400f97 <phase_3+130>
0x0000000000400f86 <+113>: callq 0x40142a <explode_bomb>
0x0000000000400f8b <+118>: mov $0x0,%eax
0x0000000000400f90 <+123>: jmp 0x400f97 <phase_3+130>
0x0000000000400f92 <+125>: mov $0x3f,%eax
=> 0x0000000000400f97 <+130>: cmp 0x4(%rsp),%eax
0x0000000000400f9b <+134>: je 0x400fa2 <phase_3+141>
0x0000000000400f9d <+136>: callq 0x40142a <explode_bomb>
0x0000000000400fa2 <+141>: mov 0x8(%rsp),%rax
0x0000000000400fa7 <+146>: xor %fs:0x28,%rax
0x0000000000400fb0 <+155>: je 0x400fb7 <phase_3+162>
0x0000000000400fb2 <+157>: callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fb7 <+162>: add $0x18,%rsp
0x0000000000400fbb <+166>: retq
End of assembler dump.
(gdb) i r
rax 0x31e 798
rbx 0x7fffffff3f8 140737488348152
rcx 0x0 0
```

```
Terminal
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f76 <+97>: jmp 0x400f97 <phase_3+130>
0x0000000000400f78 <+99>: mov $0x10b,%eax
0x0000000000400f7d <+104>: jmp 0x400f97 <phase_3+130>
0x0000000000400f7f <+106>: mov $0x80,%eax
0x0000000000400f84 <+111>: jmp 0x400f97 <phase_3+130>
0x0000000000400f86 <+113>: callq 0x40142a <explode_bomb>
0x0000000000400f8b <+118>: mov $0x0,%eax
0x0000000000400f90 <+123>: jmp 0x400f97 <phase_3+130>
0x0000000000400f92 <+125>: mov $0x3f,%eax
0x0000000000400f97 <+130>: cmp 0x4(%rsp),%eax
0x0000000000400f9b <+134>: je 0x400fa2 <phase_3+141>
=> 0x0000000000400f9d <+136>: callq 0x40142a <explode_bomb>
0x0000000000400fa2 <+141>: mov 0x8(%rsp),%rax
0x0000000000400fa7 <+146>: xor %fs:0x28,%rax
0x0000000000400fb0 <+155>: je 0x400fb7 <phase_3+162>
0x0000000000400fb2 <+157>: callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fb7 <+162>: add $0x18,%rsp
0x0000000000400fbb <+166>: retq
End of assembler dump.
(gdb) ni
BOOM!!!
The bomb has blown up.
[Inferior 1 (process 72635) exited with code 010]
(gdb)
```

Finally the program code was restarted and the bomb of phase 3 was defused as shown below.

```
Terminal
(gdb) run text.txt
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003/bomb text.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?
0 1 1 2 3 5
That's number 2. Keep going!
2 798

Breakpoint 1, 0x000000000400f15 in phase_3 ()
(gdb) delete
Delete all breakpoints? (y or n) y
(gdb) r text.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003/bomb text.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?
0 1 1 2 3 5
That's number 2. Keep going!
2 798
Halfway there!
```

After understanding the condition of program code, we have learned that first integer ranges from 1 to 7. Thus it should have its own respective integer as a second value. After executing the program code and providing input ranging from 1 to 7, respective second value is extracted as follows 1 & 198, 2 & 798, 3 & 665, 4 & 58, 5 & 624, 6 & 267 and 7 & 128.

For 1st integer 1:

```
Terminal
0x000000000400fbb <+166>: retq
End of assembler dump.
(gdb) i r
rax            0xc6                198
rbx            0x7fffffff3f8        140737488348152
rcx            0x0                0
rdx            0x7fffffff2e4        140737488347876
rsi            0x0                0
rdi            0x7fffffffdc90       140737488346256
rbp            0x0                0x0
rsp            0x7fffffff2e0        0x7fffffff2e0
r8             0xffffffff         4294967295
r9             0x0                0
r10            0x7ffff7f5aac0        140737353460416
r11            0x0                0
r12            0x400c60            4197472
r13            0x7fffffff3f0        140737488348144
r14            0x0                0
r15            0x0                0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x297              [ CF PF AF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                0
es             0x0                0
fs             0x0                0
```

For 2nd integer 2 :

```
Terminal
0x0000000000400f6f <+90>: jmp 0x400f97 <phase_3+130>
0x0000000000400f71 <+92>: mov $0x270,%eax
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f76 <+97>: jmp 0x400f97 <phase_3+130>
0x0000000000400f78 <+99>: mov $0x10b,%eax
0x0000000000400f7d <+104>: jmp 0x400f97 <phase_3+130>
0x0000000000400f7f <+106>: mov $0x80,%eax
0x0000000000400f84 <+111>: jmp 0x400f97 <phase_3+130>
0x0000000000400f86 <+113>: callq 0x40142a <explode_bomb>
0x0000000000400f8b <+118>: mov $0x0,%eax
0x0000000000400f90 <+123>: jmp 0x400f97 <phase_3+130>
0x0000000000400f92 <+125>: mov $0x3f,%eax
=> 0x0000000000400f97 <+130>: cmp 0x4(%rsp),%eax
0x0000000000400f9b <+134>: je 0x400fa2 <phase_3+141>
0x0000000000400f9d <+136>: callq 0x40142a <explode_bomb>
0x0000000000400fa2 <+141>: mov 0x8(%rsp),%rax
0x0000000000400fa7 <+146>: xor %fs:0x28,%rax
0x0000000000400fb0 <+155>: je 0x400fb7 <phase_3+162>
0x0000000000400fb2 <+157>: callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fb7 <+162>: add $0x18,%rsp
0x0000000000400fbb <+166>: retq
End of assembler dump.
(gdb) i r
rax          0x31e          798
rbx          0x7fffffff3f8    140737488348152
rcx          0x0            0
```

For 3rd integer 3:

```
Terminal
End of assembler dump.
(gdb) u*0x0000000000400f97
0x0000000000400f97 in phase_3 ()
(gdb) i r
rax          0x299          665
rbx          0x7fffffff3f8    140737488348152
rcx          0x0            0
rdx          0x7fffffff2e4    140737488347876
rsi          0x0            0
rdi          0x7fffffffdc90    140737488346256
rbp          0x0            0x0
rsp          0x7fffffff2e0    0x7fffffff2e0
r8           0xffffffff      4294967295
r9           0x0            0
r10          0x7ffff7f5aac0    140737353460416
r11          0x0            0
r12          0x400c60         4197472
r13          0x7fffffff3f0    140737488348144
r14          0x0            0
r15          0x0            0
rip          0x400f97         0x400f97 <phase_3+130>
eflags      0x297          [ CF PF AF SF IF ]
cs          0x33           51
ss          0x2b           43
ds          0x0            0
es          0x0            0
```

For 4th integer 4:

```
0x0000000000400fbb <+166>:    retq
End of assembler dump.
(gdb) i r
rax            0x3a                58
rbx            0x7fffffff3f8        140737488348152
rcx            0x0                  0
rdx            0x7fffffff2e4        140737488347876
rsi            0x0                  0
rdi            0x7fffffffdc90       140737488346256
rbp            0x0                  0x0
rsp            0x7fffffff2e0        0x7fffffff2e0
r8             0xffffffff          4294967295
r9             0x0                  0
r10            0x7ffff7f5aac0       140737353460416
r11            0x0                  0
r12            0x400c60             4197472
r13            0x7fffffff3f0        140737488348144
r14            0x0                  0
r15            0x0                  0
rip            0x400f97             0x400f97 <phase_3+130>
eflags         0x293               [ CF AF SF IF ]
cs             0x33                 51
ss             0x2b                 43
ds             0x0                  0
es             0x0                  0
fs             0x0                  0
```

For 5th integer 5:

```
0x0000000000400fb7 <+162>:    add     $0x18,%rsp
0x0000000000400fbb <+166>:    retq
End of assembler dump.
(gdb) i r
rax            0x270                624
rbx            0x7fffffff3f8        140737488348152
rcx            0x0                  0
rdx            0x7fffffff2e4        140737488347876
rsi            0x0                  0
rdi            0x7fffffffdc90       140737488346256
rbp            0x0                  0x0
rsp            0x7fffffff2e0        0x7fffffff2e0
r8             0xffffffff          4294967295
r9             0x0                  0
r10            0x7ffff7f5aac0       140737353460416
r11            0x0                  0
r12            0x400c60             4197472
r13            0x7fffffff3f0        140737488348144
r14            0x0                  0
r15            0x0                  0
rip            0x400f97             0x400f97 <phase_3+130>
eflags         0x293               [ CF AF SF IF ]
cs             0x33                 51
ss             0x2b                 43
ds             0x0                  0
es             0x0                  0
```

For 6th integer 6:

```
Terminal
(gdb) u*0x000000000400f97
0x000000000400f97 in phase_3 ()
(gdb) i r
rax          0x10b          267
rbx          0x7fffffff3f8    140737488348152
rcx          0x0            0
rdx          0x7fffffff2e4    140737488347876
rsi          0x0            0
rdi          0x7fffffffdc90   140737488346256
rbp          0x0            0x0
rsp          0x7fffffff2e0    0x7fffffff2e0
r8           0xffffffff      4294967295
r9           0x0            0
r10          0x7ffff7f5aac0    140737353460416
r11          0x0            0
r12          0x400c60         4197472
r13          0x7fffffff3f0    140737488348144
r14          0x0            0
r15          0x0            0
rip          0x400f97         0x400f97 <phase_3+130>
eflags       0x297          [ CF PF AF SF IF ]
cs           0x33           51
ss           0x2b           43
ds           0x0            0
es           0x0            0
fs           0x0            0
```

For 7th integer 7:

```
Terminal
(gdb) u*0x000000000400f97
0x000000000400f97 in phase_3 ()
(gdb) i r
rax          0x80           128
rbx          0x7fffffff3f8    140737488348152
rcx          0x0            0
rdx          0x7fffffff2e4    140737488347876
rsi          0x0            0
rdi          0x7fffffffdc90   140737488346256
rbp          0x0            0x0
rsp          0x7fffffff2e0    0x7fffffff2e0
r8           0xffffffff      4294967295
r9           0x0            0
r10          0x7ffff7f5aac0    140737353460416
r11          0x0            0
r12          0x400c60         4197472
r13          0x7fffffff3f0    140737488348144
r14          0x0            0
r15          0x0            0
rip          0x400f97         0x400f97 <phase_3+130>
eflags       0x246          [ PF ZF IF ]
cs           0x33           51
ss           0x2b           43
ds           0x0            0
```