

Phase 3

In phase 3 we have to enter 2 integer in order to proceed to next phase. How we know it should be 2 integer is because of this comment vi bomb.s where we can see the assembly code.

```
362 0000000000400f0d <phase_3>:
363 400f0d: 48 83 ec 18      sub    $0x18,%rsp
364 400f11: 64 48 8b 04 25 28 00 mov    %fs:0x28,%rax
365 400f18: 00 00
366 400f1a: 48 89 44 24 08    mov    %rax,0x8(%rsp)
367 400f1f: 31 c0            xor    %eax,%eax
368 400f21: 48 8d 4c 24 04    lea    0x4(%rsp),%rcx
369 400f26: 48 89 e2          mov    %rsp,%rdx
370 400f29: be b7 25 40 00    mov    $0x4025b7,%esi
371 400f2e: e8 7d fc ff ff    callq 400bb0 <__isoc99_sscanf@plt>
372 400f33: 83 f8 01          cmp    $0x1,%eax
373 400f36: 7f 05            jg     400f3d <phase_3+0x30>
374 400f38: e8 fb 04 00 00    callq 401438 <explode_bomb>
375 400f3d: 83 3c 24 07       cmpl   $0x7,(%rsp)
376 400f41: 77 5b            ja     400f9e <phase_3+0x91>
377 400f43: 8b 04 24          mov    (%rsp),%eax
378 400f46: ff 24 c5 00 24 40 00 jmpq   *0x402400(,%rax,8)
379 400f4d: b8 80 03 00 00    mov    $0x380,%eax
380 400f52: eb 05            jmp    400f59 <phase_3+0x4c>
381 400f54: b8 00 00 00 00    mov    $0x0,%eax
382 400f59: 2d e4 03 00 00    sub    $0x3e4,%eax
383 400f5e: eb 05            jmp    400f65 <phase_3+0x58>
384 400f60: b8 00 00 00 00    mov    $0x0,%eax
385 400f65: 05 1e 02 00 00    add    $0x21e,%eax
386 400f6a: eb 05            jmp    400f71 <phase_3+0x64>
387 400f6c: b8 00 00 00 00    mov    $0x0,%eax
388 400f71: 83 e8 34          sub    $0x34,%eax
389 400f74: eb 05            jmp    400f7b <phase_3+0x6e>
390 400f76: b8 00 00 00 00    mov    $0x0,%eax
391 400f7b: 83 c0 34          add    $0x34,%eax
392 400f7e: eb 05            jmp    400f85 <phase_3+0x78>
```

Step 1:

First we have to enter gdb bomb command in order to enter the assembly code and we have to set break point for phase 3 so that bomb donot defuse when we enter the value for phase 3 . so I have enter 4 and 3 as a input for phase 3.

```
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 0x400f0d
(gdb) run
Starting program: /home/kezan/Downloads/Assignment 1_2/Assignment 1/bomb002/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
4 3

Breakpoint 1, 0x0000000000400f0d in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
```

Step 2:

After completion of above step I have used `disas` command to enter into phase 3 assembly code and I have shifted the execution of code using `u*` address of place where we want to shift and run the program.

```
Dump of assembler code for function phase_3:
0x0000000000400f0d <+0>:      sub    $0x18,%rsp
0x0000000000400f11 <+4>:      mov    %fs:0x28,%rax
0x0000000000400f1a <+13>:     mov    %rax,0x8(%rsp)
0x0000000000400f1f <+18>:     xor    %eax,%eax
0x0000000000400f21 <+20>:     lea    0x4(%rsp),%rcx
0x0000000000400f26 <+25>:     mov    %rsp,%rdx
0x0000000000400f29 <+28>:     mov    $0x4025b7,%esi
0x0000000000400f2e <+33>:     callq 0x400bb0 <__isoc99_sscanf@plt>
=> 0x0000000000400f33 <+38>:     cmp    $0x1,%eax
0x0000000000400f36 <+41>:     jg     0x400f3d <phase_3+48>
0x0000000000400f38 <+43>:     callq 0x401438 <explode_bomb>
0x0000000000400f3d <+48>:     cmpl   $0x7,(%rsp)
0x0000000000400f41 <+52>:     ja     0x400f9e <phase_3+145>
0x0000000000400f43 <+54>:     mov    (%rsp),%eax
0x0000000000400f46 <+57>:     jmpq   *0x402400(,%rax,8)
0x0000000000400f4d <+64>:     mov    $0x380,%eax
0x0000000000400f52 <+69>:     jmp    0x400f59 <phase_3+76>
0x0000000000400f54 <+71>:     mov    $0x0,%eax
0x0000000000400f59 <+76>:     sub    $0x3e4,%eax
0x0000000000400f5e <+81>:     jmp    0x400f65 <phase_3+88>
0x0000000000400f60 <+83>:     mov    $0x0,%eax
0x0000000000400f65 <+88>:     add    $0x21e,%eax
0x0000000000400f6a <+93>:     jmp    0x400f71 <phase_3+100>
0x0000000000400f6c <+95>:     mov    $0x0,%eax
0x0000000000400f71 <+100>:    sub    $0x34,%eax
0x0000000000400f74 <+103>:    jmp    0x400f7b <phase_3+110>
0x0000000000400f76 <+105>:    mov    $0x0,%eax
0x0000000000400f7b <+110>:    add    $0x34,%eax
0x0000000000400f7e <+113>:    jmp    0x400f85 <phase_3+120>
0x0000000000400f80 <+115>:    mov    $0x0,%eax
0x0000000000400f85 <+120>:    sub    $0x34,%eax
0x0000000000400f88 <+123>:    jmp    0x400f8f <phase_3+130>
```

Step 3:

I have used `ni` and `disas` command to execute the program line by line till it reach to compare the `rsp` and `eax` where we compare `rsp` and `eax` to jump to `phase_3+172` line.

```
0x0000000000400f88 <+123>:    jmp    0x400f8f <phase_3+130>
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f8a <+125>:    mov    $0x0,%eax
0x0000000000400f8f <+130>:    add    $0x34,%eax
0x0000000000400f92 <+133>:    jmp    0x400f99 <phase_3+140>
0x0000000000400f94 <+135>:    mov    $0x0,%eax
0x0000000000400f99 <+140>:    sub    $0x34,%eax
0x0000000000400f9c <+143>:    jmp    0x400fa8 <phase_3+155>
0x0000000000400f9e <+145>:    callq 0x401438 <explode_bomb>
0x0000000000400fa3 <+150>:    mov    $0x0,%eax
0x0000000000400fa8 <+155>:    cmpl   $0x5,(%rsp)
=> 0x0000000000400fac <+159>:    jg     0x400fb4 <phase_3+167>
0x0000000000400fae <+161>:    cmp    0x4(%rsp),%eax
0x0000000000400fb2 <+165>:    je     0x400fb9 <phase_3+172>
0x0000000000400fb4 <+167>:    callq 0x401438 <explode_bomb>
0x0000000000400fb9 <+172>:    mov    0x8(%rsp),%rax
0x0000000000400fbe <+177>:    xor    %fs:0x28,%rax
0x0000000000400fc7 <+186>:    je     0x400fce <phase_3+193>
0x0000000000400fc9 <+188>:    callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fce <+193>:    add    $0x18,%rsp
```

Step 4:

After finding the value for rsp we get 4 and for eax we get 0 so it will not jump to phase_3+172 line instead will cause the bomb to explode so we get hint that key for phase 3 will be 4 and 0.

```
0x00000000400f76 <+105>: mov     $0x0,%eax
0x00000000400f7b <+110>: add     $0x34,%eax
0x00000000400f7e <+113>: jmp     0x400f85 <phase_3+120>
0x00000000400f80 <+115>: mov     $0x0,%eax
0x00000000400f85 <+120>: sub     $0x34,%eax
0x00000000400f88 <+123>: jmp     0x400f8f <phase_3+130>
--Type <RET> for more, q to quit, c to continue without paging--
0x00000000400f8a <+125>: mov     $0x0,%eax
0x00000000400f8f <+130>: add     $0x34,%eax
0x00000000400f92 <+133>: jmp     0x400f99 <phase_3+140>
0x00000000400f94 <+135>: mov     $0x0,%eax
0x00000000400f99 <+140>: sub     $0x34,%eax
0x00000000400f9c <+143>: jmp     0x400fa8 <phase_3+155>
0x00000000400f9e <+145>: callq   0x401438 <explode_bomb>
0x00000000400fa3 <+150>: mov     $0x0,%eax
0x00000000400fa8 <+155>: cmpl    $0x5,(%rsp)
0x00000000400fac <+159>: jg       0x400fb4 <phase_3+167>
0x00000000400fae <+161>: cmp     0x4(%rsp),%eax
0x00000000400fb2 <+165>: je       0x400fb9 <phase_3+172>
=> 0x00000000400fb4 <+167>: callq   0x401438 <explode_bomb>
0x00000000400fb9 <+172>: mov     0x8(%rsp),%rax
0x00000000400fbe <+177>: xor     %fs:0x28,%rax
0x00000000400fc7 <+186>: je       0x400fce <phase_3+193>
0x00000000400fc9 <+188>: callq   0x400b00 <__stack_chk_fail@plt>
0x00000000400fce <+193>: add     $0x18,%rsp
0x00000000400fd2 <+197>: retq

End of assembler dump.
(gdb) nt

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 5344) exited with code 010]
(gdb)
```

Step 5:

From above step we are able to get hint of answer that could be our key for the phase 3 because we have got 4 and 0 so we can guess that it will be addition of same number till 2 integer. So we get the key for the phase 3 is **4 0** so when we put key for phase 3 we are able to defuse the bomb.

```
Starting program: /home/kezung/Downloads/Assignment 1_2/Assignment 1/bomb002/bom
b
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
4 0
Halfway there!
█
```