Bomb001 phase_1

The first phase_1 of bomb001 is about giving a string to Dr. Evil. If the user input string store in %eax matches the string of Dr Evil stored in 0x4023d0 than the phase 1 of the bomb will be deffused, prompting us to next phase, else it will get blow up. As we don't know the string set by Dr Evil, It is our work to find the correct string to deffused the bomb. The string which I got from phase_1 bomb001 is "The moon unit will be divided into two divisions."

First, open the terminal from bomb001 folder and run the command "objdump -d bomb > bomb.s" The command will convert the bomb file from machine language into assembly language in bomb.s file to look more at the assembly code for the next phase. And it also dissamble the code and major function, reads the assembler code and displays information about one or more object files.

```
sonam@sonam-Acer-One-14-Z1-4718:~/Desktop/Fifth Semester/CS I/Assignment 1_2/Ass
ignment 1/bomb001$ objdump -d bomb > bomb.s
sonam@sonam-Acer-One-14-Z1-4718:~/Desktop/Fifth Semester/CS I/Assignment 1_2/Ass
ignment 1/bomb001$ ls
bomb bomb.c bomblab.pdf bomb.s
sonam@sonam-Acer-One-14-Z1-4718:~/Desktop/Fifth Semester/CS I/Assignment 1_2/Ass
ignment 1/bomb001$
```

chmod 777 bomb will set the permission of bomb file to read, write and execute the code by all the users.

The *gdb bomb* command will debug the assembly file (bomb.s) and helps to run a program up to certain point and stop to print out the values of certain variables. After that we have to set the break point at phase 1 in order to stop at phase_1 for debugging purpose. After setting the break-point we can run the program using "run" command.

```
gnment 1/bomb001$ chmod 777 bomb
  sonam@sonam-Acer-One-14-Z1-4718:~/Desktop/Fifth Semester/CS I/Assignment 1_2/Ass
  ignment 1/bomb001$ gdb bomb
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
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:
<a href="http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/>">http://www.gnu.org/software/gdb/bugs/</a>
Find the GDB manual and other documentation resources online at:
             <a href="http://www.gnu.org/software/gdb/documentation/">http://www.gnu.org/software/gdb/documentation/>.">http://www.gnu.org/software/gdb/documentation/>.</a>
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
 (gdb)
```

After that set the break point "break phase_1" in the phase_1 of bomb001 to stop at phase_1 for debugging purpose. It will ensure that the bomb doesn't blow up when we run the program. Also set break explode_bomb function in order to pause the execution after entering the phase_1 function. After that disassemble the phase_1 using "disas phase_1" to go inside the assembly code of the

phase_1 and then run the program using *r*.

```
Type "apropos word" to search for commands related to "word"\dots
Reading symbols from bomb...
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) b explode_bomb
Breakpoint 2 at 0x40143d
(gdb) disas
No frame selected.
(gdb) disas phase_1
Dump of assembler code for function phase_1:
   0x0000000000400e8d <+0>: sub $0x8,%rsp
   0x0000000000400e91 <+4>:
                                        $0x4023d0,%esi
                                MOV
   0x0000000000400e96 <+9>:
                                 callq
                                        0x40133e <strings_not_equal>
   0x00000000000400e9b <+14>: test %eax,%eax
  0x0000000000400e9d <+16>: je 0x400ea4 <phase_1+23>
0x0000000000400e9f <+18>: callq 0x40143d <explode_bomb>
   0x00000000000400ea4 <+23>: add $0x8,%rsp
   0x0000000000400ea8 <+27>:
                                retq
End of assembler dump.
(gdb) r
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb
Welcome to my fiendish little bomb. You have 6 phases with
```

After running the program give any rand input to test the string. In my case I have given *test string* and the answer does not match matched the input given by Dr. Evil, yet the bomb didn't explode due to setting of break point in the phase_1. Then execute the next instruction until you reached test %ea %eax.

```
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
test string
Breakpoint 1, 0x0000000000400e8d in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
     np or assembler code for function phase_1:

0x00000000000400e8d <+0>: sub $0x8,%rsp

0x0000000000400e91 <+4>: mov $0x4023d0,%esi

0x00000000000400e96 <+9>: callq 0x40133e <strings_not_equal>

0x000000000000400e9b <+14>: test %eax,%eax
                                                        test %eax,%eax
je 0x400ea4 <phase_1+23>
callq 0x40143d <explode_bomb>
add $0x8,%rsp
     0x00000000000400e9d <+16>:
     0x00000000000400e9f <+18>:
End of assembler dump.
Dump of assembler code for function phase_1:
                                                      sub $0x8,%rsp
mov $0x4023d0,%esi
     0x00000000000400e8d <+0>:
0x00000000000400e91 <+4>:
                                                      mov 30...
callq 0x40133e <se.
test %eax, %eax
je 0x400ea4 <phase_1+23>
callq 0x40143d <explode_bomb>
add $0x8,%rsp
     0x00000000000400e9b <+14>:
     0x0000000000400e9d <+16>:
       0x00000000000400ea8 <+27>:
```

After reaching the instruction line 4, inspect what is being moved from address 0x4023d0. As We know it has to be a string so we use '/s' using the command x/s 0x4023d0. Running this command, we get the string input set by Dr. Evil in the phase_1 of bmb001. *The string is "The moon will be divide into two divisions"*.

```
(gdb) ni
0x00000000000400e91 in phase 1 ()
(qdb) disas
Dump of assembler code for function phase 1:
                                        $0x8,%rsp
   0x00000000000400e8d <+0>:
                                 sub
=> 0x00000000000400e91 <+4>:
                                 mov
                                         $0x4023d0, %esi
                                 callq 0x40133e <strings_not_equal>
   0x00000000000400e96 <+9>:
   0x00000000000400e9b <+14>:
                                 test
                                        %eax,%eax
   0x00000000000400e9d <+16>:
                                        0x400ea4 <phase_1+23>
                                 je
   0x00000000000400e9f <+18>:
                                 callq 0x40143d <explode_bomb>
   0x00000000000400ea4 <+23>:
                                        $0x8,%rsp
                                 add
   0x00000000000400ea8 <+27>:
                                 reta
End of assembler dump.
(gdb) x/s 0x4023d0
                 "The moon unit will be divided into two divisions."
0x4023d0:
(gdb) ni
0x00000000000400e96 in phase 1 ()
(gdb) ni
0x00000000000400e9b in phase 1 ()
```

Once we reached the text instruction %eax%eax , we can run the $\frac{i}{r}$ command to see the information of the register. Here the eax value is 1 which will call the explode_bomb, since the string does not matched with the given string.

```
function phase_1:
sub $0x8,%rsp
Dump of assembler code for
                                         mov
callq
    0x0000000000400e91 <+4>:
                                                   $0x4023d0,%esi
   0x0000000000400e96 <+9>:
0x0000000000400e9b <+14>:
0x00000000000400e9d <+16>:
                                                           3e <strings_not_equal>
                                          test
                                                   %eax,%eax
                                                   0x400ea4 <phase_1+23>
0x40143d <explode_bomb>
                                         je
                                          callq
                            <+18>:
    0x0000000000400ea4 <+23>:
                                          add
                                                   $0x8,%rsp
                                          retq
    0x00000000000400ea8 <+27>:
End of
        assembler dump.
(gdb)
rax
rbx
                   0x1
                                             4202992
                   0x4021f0
гdх
                   0x1
rsi
rdi
                   0x4023d0
                                             4203472
                   0x402401
                                              4203521
                   0×0
                                             0×0
                                             0x7fffffffde00
                   0x7fffffffde00
rsp
                                             6305696
г8
                   0x6037a0
                                             124
                   0xffffffffffff6ed
0x7ffff7df9400
0x400c60
г10
                                              -2323
                                             140737352012800
4197472
г11
                    0x7fffffffdf00
                                              140737488346880
г14
                   0x0
                                             0
г15
                   0x0
                                             0
                   0x400e9b
                                             0x400e9b <phase_1+14>
rip
eflags
                   0x283
                                                CF SF IF
                   0x33
                                             43
ss
ds
                    0x2b
                    0×0
                                             0
```

Then run the program again by giving the input string we got from the address 0x4023d0. Check the information register to see the value of eax. If the value is zero "0" then it will execute line 23 and then will return. Thus, the phase_1 of bom001 will get defused and we will be prompted to the next phase.

```
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/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.
Breakpoint 1, 0x0000000000400e8d in phase 1 ()
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x0000000000400e8d <+0>: sub $0x8,%rsp
   0x00000000000400e91 <+4>:
                                         mov
                                                  $0x4023d0,%esi
   0x00000000000400e96 <+9>: callq 0x40133e <strings_not_equal>
0x00000000000400e9b <+14>: test %eax,%eax
0x0000000000400e9d <+16>: je 0x400ea4 <phase_1+23>
0x000000000000400e9f <+18>: callq 0x40143d <explode_bomb>
   0x0000000000400ea4 <+23>: add $0x8,%rsp
    0x0000000000400ea8 <+27>:
                                         retq
End of assembler dump.
```

Since the eax value is zero, line 23 will get executed and will return the value.

```
(gdb) u* 0x0000000000400e9b
                  00400e9b in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
 0x000000000400e8d <+0>: sub $0x8,%rsp
0x000000000400e91 <+4>: mov $0x4023d0,%esi
0x0000000000400e96 <+9>: callq 0x40133e <strings_not_equal>
:> 0x0000000000400e9b <+14>: test %eax,%eax
                                                   callq 0x40133e <strings_not_e
test %eax,%eax
je 0x400ea4 <phase_1+23>
callq 0x40143d <explode_bomb>
add $0x8,%rsp
retq
    0x00000000000400e9d <+16>:
0x00000000000400e9f <+18>:
0x00000000000400ea4 <+23>:
                                                        add
retq
     0x00000000000400ea8 <+27>:
End of
           assembler dump.
(gdb) i r
rax
rax
rbx
                          0×0
                          0x4021f0
                          0x31
                                                              49
                                                              0
4203472
гdх
                          0×0
                          0x4023d0
rdi
                          0x402401
                                                              4203521
гЬр
                          0x0
0x7fffffffde00
                                                              0 \times 0
                                                              0x7fffffffde00
 sp
г8
г9
                          0x6037a0
                                                              6305696
                          0x7c
0x7f
0xffffffffffff6ed
0x7fffff7df9400
0x400c60
0x7ffffffffdf00
                                                              124
                                                              140737352012800
4197472
140737488346880
                          0 \times 0
                                                              0
r15
                          0 \times 0
                           0x400e9b
                                                               0x400e9b <phase_1+14>
                                                                 PF ZF IF
 eflags
                           0x246
                           0x33
```

Used the command *info break* to see about the information about the break point we set earlier and after that remove break point in order to defused the phase_1, else the break point will prevent the phase from defusing.

```
End of assembler dump.
(gdb) info break
Num
         Туре
                          Disp Enb Address
                                                           What
                          keep y 0x0000000000400e8d <phase 1>
         breakpoint
         breakpoint already hit 1 time
         breakpoint
                          keep y 0x000000000040143d <explode_bomb>
(gdb) delete
Delete all breakpoints? (y or n) y
(gdb) r
The program being debugged has been started already.

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

Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/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?
```

Phase 2

Set the break point in the phase_2 and explode_bomb just like in the phase_1 and run the file name.txt eg:"answer.txt " of the phase_1 we saved in. And run the program giving input string as we don't know the input format of the phase_2 answer. Eg: morning

```
Reading symbols from bomb...

(gdb) b phase_2

Breakpoint 1 at 0x400ea9

(gdb) b explode_bomb

Breakpoint 2 at 0x40143d

(gdb) r answer.txt

Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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?
morning

Breakpoint 1, 0x00000000000400ea9 in phase_2 ()

(gdb)
```

Go to disas of the phase_2.

```
disas
Dump of assembler code for function phase
  0x00000000000400ea9 <+0>:
                                  push
                                          %гьр
   0x0000000000400eaa <+1>:
                                  push
                                          %гьх
   0x00000000000400eab <+2>:
                                   sub
                                          $0x28,%rsp
                                          %fs:0x28,%rax
%rax,0x18(%rsp)
   0x00000000000400eaf <+6>:
                                  MOV
   0x00000000000400eb8 <+15>:
                                   mov
   0x00000000000400ebd <+20>:
                                   хог
                                          %eax,%eax
   0x00000000000400ebf <+22>:
                                          %rsp,%rsi
                                   mov
   0x0000000000400ec2 <+25>:
                                   callq
                                                 5f <read_six_numbers>
   0x00000000000400ec7 <+30>:
                                          $0x0,(%rsp)
                                   cmpl
   0x00000000000400ecb <+34>:
                                   ine
                                                    <phase 2+43>
                                          $0x1,0x4(%rsp)
   0x0000000000400ecd <+36>:
                                   cmpl
                                          0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x0000000000400ed2 <+41>:
                                   je
   0x00000000000400ed4 <+43>:
                                   callq
                                          %rsp,%rbx
0x10(%rsp),%rbp
   0x00000000000400ed9 <+48>:
                                   mov
   0x00000000000400edc <+51>:
                                   lea
                                          0x4(%rbx),%eax
   0x0000000000400ee1 <+56>:
                                   mov
                                          (%rbx),%eax
   0x00000000000400ee4 <+59>:
                                   add
   0x00000000000400ee6 <+61>:
                                   cmp
                                          %eax,0x8(%rbx)
   0x00000000000400ee9 <+64>:
                                   je
                                          0x400ef0 <phase
                                          0x40143d <explode bomb>
   0x0000000000400eeb <+66>:
                                   callq
   0x00000000000400ef0 <+71>:
                                          $0x4,%rbx
                                   add
   0x0000000000400ef4 <+75>:
                                   cmp
                                          %rbp,%rbx
   0x0000000000400ef7 <+78>:
                                   jne
                                                    <phase_2+56>
   0x00000000000400ef9 <+80>:
                                          0x18(%rsp),%rax
                                   mov
   0x00000000000400efe <+85>:
                                   XOL
                                          %fs:0x28,%rax
                       <+94>:
                                   je
                                          0x400f0e <phase_2+101>
                                          0x400b00 < _stack_chk_fail@plt>
   0x0000000000400f09 <+96>:
                                   callq
                       <+101>:
                                   add
                                          $0x28,%rsp
   0x0000000000400f12 <+105>:
                                          %гьх
                                   pop
                       <+106>:
                                          %гьр
                                   POP
                       <+107>:
                                   reta
End of assembler dump.
```

Then go to the line 25 using until * address to see the format of the phase_2 answer.

```
(gdb) until * 0x0000000000400ec2
     00000000400ec2 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
    )x00000000000400ea9 <+0>:
                                   push
                                           %гьр
   0x00000000000400eaa <+1>:
                                   push
                                           %гьх
                                           $0x28,%rsp
   0x00000000000400eab <+2>:
                                   sub
   0x0000000000400eaf <+6>:
                                   mov
                                          %fs:0x28,%rax
   0x00000000000400eb8 <+15>:
                                          %rax,0x18(%rsp)
                                   mov
                                          %eax,%eax
%rsp,%rsi
   0x00000000000400ebd <+20>:
                                   XOL
   0x0000000000400ebf <+22>:
                                   mov
=> 0x0000000000400ec2 <+25>:
                                   callq
                                                    <read six numbers>
   0x00000000000400ec7 <+30>:
                                           $0x0,(%rsp)
                                   cmpl
   0x0000000000400ecb <+34>:
                                           0x400ed4 <phase_2+43>
                                   ine
                                           $0x1,0x4(%rsp)
   0x00000000000400ecd <+36>:
                                   cmpl
                                          0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x00000000000400ed2 <+41>:
                                   je
   0x00000000000400ed4 <+43>:
                                   callq
                                           %rsp,%rbx
   0x0000000000400ed9 <+48>:
                                   mov
                                          0x10(%rsp),%rbp
   0x00000000000400edc <+51>:
                                   lea
   0x0000000000400ee1 <+56>:
                                           0x4(%rbx),%eax
                                   mov
                                           (%rbx),%eax
   0x00000000000400ee4 <+59>:
                                   add
                                          %eax,0x8(%rbx)
   0x0000000000400ee6 <+61>:
                                   cmp
   0x00000000000400ee9 <+64>:
                                   je
                                           0x400ef0 <phase
                                          0x40143d <explode_bomb>
   0x00000000000400eeb <+66>:
                                   callq
   0x00000000000400ef0 <+71>:
                                   add
                                           $0x4,%rbx
   0x0000000000400ef4 <+75>:
                                   cmp
                                           %rbp,%rbx
                                                ee1 <phase_2+56>
   0x00000000000400ef7 <+78>:
                                   ine
                                          0x18(%rsp),%rax
   0x00000000000400ef9 <+80>:
                                   mov
   0x00000000000400efe <+85>:
                                   XOL
                                           %fs:0x28,%rax
                                          0x400f0e <phase_2+101>
0x400b00 <__stack_chk_fail@plt>
   0x00000000000400f07 <+94>:
                                   je
   0x0000000000400f09 <+96>:
                                   callq
   0x0000000000400f0e <+101>:
                                   add
                                           $0x28,%rsp
    0x00000000000400f12 <+105>:
                                           %гьх
                                   DOD
```

Enter si command (step into) to check the input format. And x/s 0x4025c3 to see how many integer are there in the phase_2. So from this we get a hint that the phase_2 contains six integer as an answer to defused the bomb.

```
(gdb) si
(gdb) disas
Dump of assembler code for function read six numbers:
=> 0x0000000000040145f <+0>:
                                sub
                                       $0x8,%rsp
   0x00000000000401463 <+4>:
                                mov
                                       %rsi,%rdx
   0x00000000000401466 <+7>:
                                       0x4(%rsi),%rcx
                                lea
  0x0000000000040146a <+11>:
                                lea
                                       0x14(%rsi),%rax
  0x0000000000040146e <+15>:
                                       %гах
                                push
   0x000000000040146f <+16>:
                                lea
                                       0x10(%rsi),%rax
   0x00000000000401473 <+20>:
                                push
                                       %гах
   0x00000000000401474 <+21>:
                                lea
                                       0xc(%rsi),%r9
   0x00000000000401478 <+25>:
                                lea
                                       0x8(%rsi),%r8
   0x000000000040147c <+29>:
                                mov
                                       $0x4025c3,%esi
  0x00000000000401481 <+34>:
                                       $0x0, %eax
                                mov
  0x00000000000401486 <+39>:
                                callq 0x400bb0 < isoc99 sscanf@plt>
  0x0000000000040148b <+44>:
                                add
                                       $0x10,%rsp
  0x0000000000040148f <+48>:
                                       $0x5, %eax
                                CMP
                                       0x401499 <read six numbers+58>
  0x0000000000401492 <+51>:
                                jg
                                       0x40143d <explode_bomb>
                                callq
  0x00000000000401494 <+53>:
  0x00000000000401499 <+58>:
                                add
                                       $0x8,%rsp
   0x0000000000040149d <+62>:
                                retq
End of assembler dump.
(gdb) x/s 0x4025c3
x4025c3:
                "%d %d %d %d %d"
```

After knowing the answer format, give any six random integer number as a input. Then to to disas assembly language of the file.

```
(gdb) r answer.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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 2 3 4 5
 Breakpoint 1, 0x0000000000400ea9 in phase_2 ()
 Dump of assembler code for function phase_2:
                                                                  $0x28,%rsp
%fs:0x28,%rax
                                                        sub
      0x00000000000400eab <+2>:
     0x00000000000400eaf <+6>:
                                                       mov
                                                                   %rax,0x18(%rsp)
%eax,%eax
      0x00000000000400ebd <+20>:
     0x0000000000400ebf <+22>:
0x00000000000400ec2 <+25>:
0x00000000000400ec7 <+30>:
                                                       mov
                                                                  %rsp,%rsi
                                                       callq
                                                       cmpl $0x0,(%rsp)
      0x00000000000400ecb <+34>:
0x00000000000400ecd <+36>:
0x000000000000400ed2 <+41>:
                                                       callq 0x40143d <explode_bomb>
      0x00000000000400ed4 <+43>:
                                                                  %rsp,%rbx
0x10(%rsp),%rbp
0x4(%rbx),%eax
      0x00000000000400ed9 <+48>:
                                                       mov
lea
      0x00000000000400edc <+51>:
      0x00000000000400ee1 <+56>:
     0x00000000000400ee4 <+59>:
0x00000000000400ee6 <+61>:
                                                       add
                                                                   (%rbx),%eax
                                                                   %eax,0x8(%rbx)
                                                       CMP
                                                        callq
      0x0000000000400eeb <+66>:
                                                                  $0x4,%rbx
%rbp,%rbx
      0x00000000000400ef0 <+71>:
                                                       add
                                                       cmp
                                                               0x400ee1 <phase_2+56>
0x18(%rsp),%rax
%fs:0x28,%rax
     0x00000000000400ef7 <+78>:
0x00000000000400ef9 <+80>:
0x00000000000400efe <+85>:
                                                       mov
                                                       хог
```

```
<read six numbers>
                                          $0x0,(%rsp)
0x400ed4 <phase_2+43>
   cmpl
                                   je
callq
mov
lea
mov
                                          0x40143d <expto
%rsp,%rbx
0x10(%rsp),%rbp
0x4(%rbx),%eax
(%rbx),%eax
%eax,0x8(%rbx)
                                   add
                                   cmp
je
callq
add
                                                ef0 <phase_2+7
43d <explode_b
    0x00000000000400ee9 <+64>:
   $0x4,%rbx
%rbp,%rbx
                                   cmp
jne
   0x18(%rsp),%rax
%fs:0x28,%rax
                                   mov
                                   хог
                                   je
callq
                                           0x400f0e <phase_2+101>
0x400b00 <__stack_chk_fail@pl</pre>
                                           $0x28,%rsp
                                   add
                                          %гьр
```

Here, in info register the value of rsp is 0 which is equal to the value of 0, so It will execute line 36and we can say that the first integer we gave is correct.

```
End of assembler dump.
(gdb) x/d $rsp
0x7fffffffddc0: 0
(gdb)
```

Execute the compared function. Compared the value of 1 with the rsp value plus 4. If both the value is equal, it will execute line 48 or else bomb will get exploded.

```
(qdb) u* 0x0000000000400ecd
0x00000000000400ecd in phase 2 ()
(gdb) disas
Dump of assembler code for function phase_2:
                                 push
  0x0000000000400ea9 <+0>:
                                        %гьр
  0x00000000000400eaa <+1>:
                                 push
                                        %rbx
  0x00000000000400eab <+2>:
                                 sub
                                        $0x28,%rsp
  0x0000000000400eaf <+6>:
                                        %fs:0x28,%rax
                                 MOV
   0x00000000000400eb8 <+15>:
                                        %rax,0x18(%rsp)
                                 mov
   0x00000000000400ebd <+20>:
                                 XOL
                                        %eax,%eax
  0x00000000000400ebf <+22>:
                                 mov
                                        %rsp,%rsi
  0x00000000000400ec2 <+25>:
                                 callq
                                        0x40145f <read six numbers>
  0x0000000000400ec7 <+30>:
                                        $0x0,(%rsp)
                                 cmpl
  0x00000000000400ecb <+34>:
                                        0x400ed4 <phase 2+43>
                                 jne
                                        $0x1,0x4(%rsp)
=> 0x0000000000400ecd <+36>:
                                 cmpl
   0x0000000000400ed2 <+41>:
                                 je
                                        0x400ed9 <phase_2+48>
                                        0x40143d <explode bomb>
   0x00000000000400ed4 <+43>:
                                 callq
  0x00000000000400ed9 <+48>:
                                 mov
                                        %rsp,%rbx
  0x00000000000400edc <+51>:
                                        0x10(%rsp),%rbp
                                 lea
  0x0000000000400ee1 <+56>:
                                 mov
                                        0x4(%rbx),%eax
  0x00000000000400ee4 <+59>:
                                 add
                                        (%rbx),%eax
   0x0000000000400ee6 <+61>:
                                        %eax,0x8(%rbx)
                                 cmp
```

The value value of rsp plus 4 is 1. since the value are equal it will execute line 48. Here, the second input integer value is correct.

Go diretly util* address of the line 61 where there is a compared function. If the value of eax and rbs plus 8 is equal, it will execute line 71, else bomb will get exploded.

```
0x00000000000400ee6
     00000000400ee6 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
   0x0000000000400ea9 <+0>:
                                    push
                                            %гБр
   0x0000000000400eaa <+1>:
                                    push
                                            %гЬх
                                           $0x28,%rsp
   0x0000000000400eab <+2>:
                                    sub
   0x00000000000400eaf <+6>:
                                   mov
                                            %fs:0x28,%rax
                                            %rax,0x18(%rsp)
   0x0000000000400eb8 <+15>:
                                    mov
   0x0000000000400ebd <+20>:
                                   хог
                                            %eax,%eax
                                            %rsp,%rsi
   0x0000000000400ebf <+22>:
                                   mov
                                    callq
   0x0000000000400ec2 <+25>:
                                                     <read_six_numbers>
   0x0000000000400ec7 <+30>:
                                    cmpl
                                            $0x0,(%rsp)
   0x0000000000400ecb <+34>:
                                            0x400ed4 <phase 2+43>
                                    jne
   0x0000000000400ecd <+36>:
                                    cmpl
                                            $0x1,0x4(%rsp)
                                           0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x0000000000400ed2 <+41>:
                                    je
   0x00000000000400ed4 <+43>:
                                    callq
                                           %rsp,%rbx
0x10(%rsp),%rbp
   0x00000000000400ed9 <+48>:
                                    mov
   0x0000000000400edc <+51>:
                                    lea
   0x0000000000400ee1 <+56>:
                                   mov
                                            0x4(%rbx),%eax
                                           (%rbx),%eax
   0x0000000000400ee4 <+59>:
                                    add
=> 0x0000000000400ee6 <+61>:
                                    cmp
                                           %eax,0x8(%rbx)
                                           0x400ef0 <phase_2+71>
0x40143d <explode_bomb>
   0x0000000000400ee9 <+64>:
                                    je
   0x0000000000400eeb <+66>:
                                    callq
   0x0000000000400ef0 <+71>:
                                    add
                                            $0x4,%rbx
                                           %rbp,%rbx
   0x0000000000400ef4 <+75>:
                                    CMP
   0x0000000000400ef7 <+78>:
                                    jne
                                                     <phase 2+56>
   0x0000000000400ef9 <+80>:
                                            0x18(%rsp),%rax
                                    mov
   0x0000000000400efe <+85>:
                                            %fs:0x28,%rax
                                    XOL
                                           0x400f0e <phase_2+101>
0x400b00 <__stack_chk_fail@plt>
                        <+94>:
                                    je
                                    callq
   0x0000000000400f09 <+96>:
   0x0000000000400f0e <+101>:
0x00000000000400f12 <+105>:
                                            $0x28,%rsp
                                    add
                                            %гЬх
                                    pop
   0x0000000000400f13 <+106>:
                                            %гьр
                                    pop
```

Since the value of eax is 1 and the value of rbx plus 8 is 2 which is not equal, the bomb will get exploded. Thus, we know that the third input string should be 1. So we will run the program again by giving the third integer as 1.

```
гЬх
                  0x7ffffffddc0
                                          140737488346560
гсх
                  0×0
                                          0
                  0x7fffffffddd4
rdх
                                          140737488346580
rsi
                  0x0
                                          0
                  0x7fffffffd750
rdi
                                          140737488344912
                  0x7ffffffddd0
                                          0x7fffffffddd0
0x7fffffffddc0
гЬр
                  0x7fffffffddc0
0xffffffff
гsр
r8
                                          4294967295
г9
                  0x0
                  0x7ffff7f60ac0
                                          140737353484992
r10
r11
                  0x0
                                          0
г12
                  0x400c60
                                          4197472
                  0x7fffffffdef0
                                          140737488346864
                  0×0
                                          0
г14
                  0×0
                                          0
 ip
flags
                  0х400ееб
                                          0x400ee6 <phase 2+61>
ef
                                          [ IF
51
                  0x202
                                               - 1
                  0x33
cs
SS
                  0x2b
                                          43
ds
                  0 \times 0
                                          0
                  0×0
                                          0
fs
                  0×0
                                          0
                  0×0
             0x7ffffffddc8
(gdb) x/d
```

After getting the correct answer, execute the line 75 using util* address of line 75. In this, if the value of rbp and rbx is not eqal it will excute line 56 and if it is equal It will execute next instructions. As the value of rbp and rbx is is not it will execute line 56.

```
=> 0x0000000000400ef4 <+75>:
                                         %rbp,%rbx
                                 CMP
   0x00000000000400ef7 <+78>:
                                         0x400ee1 <phase 2+56>
                                 ine
   0x00000000000400ef9 <+80>:
                                 mov
                                         0x18(%rsp),%rax
   0x0000000000400efe <+85>:
                                 хог
                                        %fs:0x28,%rax
   0x0000000000400f07 <+94>:
                                         0x400f0e <phase_2+101>
                                 je
   0x00000000000400f09 <+96>:
                                        0x400b00 < stack_chk_fail@plt>
                                 callq
                                 add
   0x00000000000400f0e <+101>:
                                         $0x28,%rsp
   0x00000000000400f12 <+105>:
                                 pop
                                         %гЬх
   0x0000000000400f13 <+106>:
                                        %rbp
                                 pop
   0x0000000000400f14 <+107>:
                                 retq
End of assembler dump.
(gdb) x/d $rbp
 x7fffffffddd0: 4
(gdb) x/d $rbx
 x7fffffffddc4: 1
```

Again, here the value of eax is 2 and 8 plus rbx is 3 which not equal. Thus, the bomb will get exploded. So we know that the fourth value should be 2.

Run the program again by entering the fourth value 2. Then the line 71 will executed .So the phase_2 is all about looping again and again unless we get the same value set by the Dr.Evil.

```
Execute the
             mbler dump.
                                                                       line 74
  Rhythmbox
                                                                       where there
гах
                  0x2
                                                                       is compared
гЬх
                  0x7fffffffddc4
                                          140737488346564
                                                                       function.
\Gamma CX
                  0x0
                                                                       Go to u*
гdх
                  0x7fffffffddd4
                                          140737488346580
                                                                       address of
rsi
                                                                       line 75 and
rdi
                  0x7fffffffd750
                                          140737488344912
                                                                       If rbp and
                  0x7fffffffddd0
                                          0x7fffffffddd0
rbp
                                                                       rbx value is
                  0x7fffffffddc0
                                          0x7fffffffddc0
rsp
                                                                       not equal it
                  0xffffffff
г8
                                          4294967295
                                                                       will again
г9
                  0x0
                                                                       execute line
г10
                  0x7fffff7f60ac0
                                          140737353484992
                                                                       56 or if the
г11
                  0x0
                                                                       value is
г12
                  0x400c60
                                          4197472
                                                                       equal it will
                  0x7fffffffdef0
г13
                                          140737488346864
                                                                       execute the
г14
                  0x0
                                                                       next
г15
                  0x0
                                                                       instruction.
rip
                                          0x400ee6 <phase_2+61>
                  0x400ee6
eflags
                  0x202
                                          [ IF ]
cs
                  0x33
                                          51
                                          43
SS
                  0x2b
ds
                  0x0
                                          0
                                          0
es
                  0x0
fs
                                          0
                  0x0
                  0x0
                                          0
             0x7fffffffddcc
(gdb) x/d
```

)x7fffffffddcc: 3

```
0x0000000000400ef4
(gdb) u*
          0000000400ef4 in phase_2
(gdb) disas
(gdb) disas

Dump of assembler code for function phase_2
0x00000000000400ea9 <+0>: push %rbp
0x00000000000400eaa <+1>: push %rbx
0x00000000000400eab <+2>: sub $0x20
0x0000000000400eab <+6>: mov %fs:0
                                                                                  $0x28,%rsp
%fs:0x28,%rax
%rax,0x18(%rsp)
      0x0000000000400ebd <+20>:
                                                                   хог
                                                                   mov
callq
cmpl
      0x00000000000400ebf <+22>:
0x0000000000400ec2 <+25>:
                                                                                 %rsp,%rsi
      0x00000000000400ec7 <+30>:
0x00000000000400ecb <+34>:
0x00000000000400ecd <+36>:
                                                                                  $0x0,(%rsp)
                                                                   jne
                                                                   cmpl
                                                                                  $0x1,0x4(%rsp)
      0x00000000000400ed2 <+41>:
0x00000000000400ed4 <+43>:
                                                                   je
                                                                   callq
      0x0000000000400ed9 <+48>:
0x00000000000400edc <+51>:
0x00000000000400ee1 <+56>:
0x00000000000400ee4 <+59>:
0x00000000000400ee6 <+61>:
                                                                                 %rsp,%rbx
0x10(%rsp),%rbp
0x4(%rbx),%eax
(%rbx),%eax
%eax,0x8(%rbx)
                                                                   mov
lea
                                                                   add
                                                                   cmp
je
callq
                                                                                       100ef0 <phase_2+71>
10143d <explode_bomb>
       0x00000000000400ee9 <+64>:
      0x00000000000400eeb <+66>:
0x00000000000400ef0 <+71>:
0x00000000000400ef4 <+75>:
                                                                                 $0x4,%rbx
%rbp,%rbx
                                                                   add
                                                                                 0X400ee1 <phase_2+56>
0X18(%rsp),%rax
%fs:0X28,%rax
0X400f0e <ph
                                                                   CMP
      0x00000000000400ef9 <+80>:
0x0000000000400efe <+85>:
                                                                   mov
                                                                   хог
                                                                                        ..\^26,8|84
@0f0e <phase_2+101>
@0b00 <__stack_chk_fail@plt>
                                             <+94>:
                                                                   je
                                                                   callq
      0x0000000000400f09 <+96>:
      0x00000000000400f0e <+101>:
0x00000000000400f12 <+105>:
                                                                   add
                                                                                 $0x28,%rsp
                                                                   pop
      0x0000000000400f13 <+106>:
                                                                   pop
                                                                                  %гьр
       0x0000000000400f14 <+107>:
                                                                   retq
```

As the value of rbp is 4 and that of rbx is 1 in information register which is not equal, so it will execute line 56.

```
(gdb)
rax
rbx
                0x7ffffffddc4
                                       140737488346564
гсх
                0×0
                0x7fffffffddd4
                                       140737488346580
rdx
rsi
                0×0
rdi
                0x7fffffffd750
                                      140737488344912
                0x7fffffffddd0
                                       0x7ffffffddd0
гЬр
                0x7ffffffddc0
                                       0x7ffffffddc0
rsp
г8
                0xffffffff
                                      4294967295
г9
                0×0
                                      0
r10
                0x7ffff7f60ac0
                                       140737353484992
                0×0
г11
                0x400c60
г12
г13
                0x7fffffffdef0
                                       140737488346864
г14
                0×0
г15
                0×0
                0x400ef4
                                      0x400ef4 <phase_2+75>
eflags
                0x202
                                        IF ]
                0x33
                0x2b
                                      43
ds
                0×0
                                      0
es
                0 \times 0
fs
                0×0
                0×0
gs
(gdb) x/d $rbp
(gdb) x/d $rbx
```

Now go to u* address of line 61. If the value of eax and 8 plus rbx is equal, it will execute line 71 or else the bomb will get exploded.

```
0x0000000000400ee6
   00000000000400ee6 in phase 2 ()
(gdb) disas
Dump of assembler code for function phase_2:
   0x00000000000400ea9 <+0>:
                                       push
                                               %гЬр
   0x0000000000400eaa <+1>:
                                       push
                                                %rbx
                                                $0x28,%rsp
   0x0000000000400eab <+2>:
                                       sub
   0x0000000000400eaf <+6>:
0x00000000000400eb8 <+15>:
                                               %fs:0x28,%rax
%rax,0x18(%rsp)
                                       mov
                                       mov
   0x00000000000400ebd <+20>:
                                                %eax,%eax
                                       хог
   0x0000000000400ebf <+22>:
                                       mov
                                                %rsp,%rsi
   0x00000000000400ec2 <+25>:
                                       callq
                                                          <read six numbers>
   0x00000000000400ec7 <+30>:
0x00000000000400ecb <+34>:
                                                $0x0,(%rsp)
                                       cmpl
                                       jne
                                                0x400ed4 <phase
   0x0000000000400ecd <+36>:
                                                $0x1,0x4(%rsp)
                                       cmpl
   0x00000000000400ed2 <+41>:
0x00000000000400ed4 <+43>:
                                                0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
                                       je
                                       callq
                                                %rsp,%rbx
0x10(%rsp),%rbp
   0x00000000000400ed9 <+48>:
                                       mov
   0x00000000000400edc <+51>:
                                       lea
   0x00000000000400ee1 <+56>:
                                                0x4(%rbx),%eax
                                       mov
   0x00000000000400ee4 <+59>:
                                       add
                                                (%rbx),%eax
                                                %eax,0x8(%rbx)
=> 0x0000000000400ee6 <+61>:
                                       cmp
   0x00000000000400ee9 <+64>:
                                               0x400ef0 <phase_2+71>
0x40143d <explode_bomb>
                                       je
   0x0000000000400eeb <+66>:
                                       callq
                                                $0x4,%rbx
   0x0000000000400ef0 <+71>:
                                       add
                                                %rbp,%rbx
   0x00000000000400ef4 <+75>:
                                       CMD
   0x00000000000400ef7 <+78>:
                                       jne
                                                0x400ee1 <phase_2+56>
                                                0x18(%rsp),%rax
%fs:0x28,%rax
   0x00000000000400ef9 <+80>:
                                       mov
   0x0000000000400efe <+85>:
                                       хог
                                                0x400f0e <phase_2+101>
0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f07 <+94>:
                                       je
   0x0000000000400f09 <+96>:
                                       callq
   0x0000000000400f0e <+101>:
                                       add
                                                $0x28,%rsp
   0x0000000000400f12 <+105>:
0x0000000000400f13 <+106>:
                                                %rbx
                                       pop
                                       pop
                                                %гьр
    0x0000000000400f14 <+107>:
                                       retq
End of assembler dump.
```

The value of eax is 2 and the 8 plus rbx is 2 which is equal, so it will execute the line 71.

```
гах
                 0x2
гЬх
                                        140737488346564
                 0x7fffffffddc4
гсх
                 0×0
                                        0
                 0x7fffffffddd4
гdх
                                        140737488346580
rsi
                 0x0
                                        0
rdi
                 0x7fffffffd750
                                        140737488344912
гЬр
                 0x7fffffffddd0
                                        0x7ffffffddd0
гѕр
                 0x7fffffffddc0
                                        0x7fffffffddc0
                 0xffffffff
                                        4294967295
г8
г9
                 0x0
                                        0
                 0x7ffff7f60ac0
r10
                                        140737353484992
г11
                0x0
г12
                 0x400c60
                                        4197472
г13
                 0x7fffffffdef0
                                        140737488346864
г14
                0x0
r15
                 0 \times 0
                                        0
rip
                 0х400ееб
                                        0x400ee6 <phase_2+61>
eflags
                 0x202
                                        [ IF ]
                                        51
cs
                 0x33
SS
                 0x2b
                                        43
ds
                 0x0
                                        0
es
                 0x0
                                        0
fs
                0 \times 0
                                        0
                                        0
                0x0
            0x7fffffffddcc
(gdb) x/d
```

Go to u* address of line 75. If the value of rbp and rbx is equal it will proceed to execute next line but it the value is not equal it will loop again to execute line 56.

```
(gdb) disas
Dump of assembler code for function phase
                                 <+0>:
                                                push
                                                           %гЬр
                                                push
                                                           $0x28,%rsp
%fs:0x28,%rax
%rax,0x18(%rsp)
    0x00000000000400eab <+2>:
0x00000000000400eaf <+6>:
                                                sub
                                                mov
     0x00000000000400eb8 <+15>:
                                                mov
                                                хог
                                                           %eax,%eax
%rsp,%rsi
    0x00000000000400ebd <+20>:
     0x0000000000400ebf <+22>:
                                                mov
                                                callq
cmpl
     0x00000000000400ec2 <+25>:
0x00000000000400ec7 <+30>:
                                                                        <read six numbers>
                                                           $0x0,(%rsp)
     0x00000000000400ecb <+34>:
                                                jne
     0x00000000000400ecd <+36>:
0x000000000000400ed2 <+41>:
                                                cmpl
                                                           $0x1,0x4(%rsp)
                                                je
callq
     0x00000000000400ed4 <+43>:
0x00000000000400ed9 <+48>:
0x00000000000400edc <+51>:
                                                           0x40143d <explode_bomb>
                                                          %rsp,%rbx
0x10(%rsp),%rbp
0x4(%rbx),%eax
(%rbx),%eax
                                                mov
lea
     0x00000000000400ee1 <+56>:
0x00000000000400ee4 <+59>:
                                                mov
                                                add
                                                           %eax,0x8(%rbx)
      x0000000000400ee6 <+61>:
                                                cmp
                                                je
callq
                                                                     0 <phase_2+71>
d <explode bomb>
     0x00000000000400ee9 <+64>:
     0x0000000000400eeb <+66>:
                                                          $0x4,%rbx
%rbp,%rbx
    0x00000000000400ef0 <+71>:
0x00000000000400ef4 <+75>:
                                                add
                                                CMP
                                                jne
                                                          0x18(%rsp),%rax
%fs:0x28,%rax
                                 <+80>:
                                                mov
     0x00000000000400efe <+85>:
                                                хог
                                                                       0x00000000000400f07 <+94>:
0x00000000000400f09 <+96>:
                                                je
callq
                                                add
                                                           $0x28,%rsp
     0x00000000000400f12 <+105>:
                                                pop
                                                           %rbx
     0x00000000000400f13 <+106>:
                                                           %гЬр
                                                pop
       0000000000400f14 <+107>:
End of assembler dump.
(gdb) x/d $rbp
0x7fffffffddd0: 4
(gdb) x/d $rbx
```

Go to u* address of line 61 where there is compared function. If the eax and eax plus 8 value is equal it will excute line 71 else bomb will get exploded.

```
(gdb) u* 0x0000000000400ee6
     00000000400ee6 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
   0x00000000000400ea9 <+0>:
                                          %гБр
                                 push
   0x0000000000400eaa <+1>:
                                          %гьх
                                  push
   0x00000000000400eab <+2>:
                                  sub
                                          $0x28,%rsp
   0x0000000000400eaf <+6>:
0x0000000000400eb8 <+15>:
                                          %fs:0x28,%rax
                                  mov
                                          %rax,0x18(%rsp)
                                  mov
   0x0000000000400ebd <+20>:
                                          %eax,%eax
                                  хог
   0x0000000000400ebf <+22>:
                                          %rsp,%rsi
                                  mov
   0x0000000000400ec2 <+25>:
                                  callq
                                                    <read six numbers>
   0x0000000000400ec7 <+30>:
                                          $0x0,(%rsp)
                                  cmpl
   0x0000000000400ecb <+34>:
                                   jne
                                          0x400ed4 <phase 2+43>
   0x0000000000400ecd <+36>:
                                   cmpl
                                          $0x1,0x4(%rsp)
                                          0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x0000000000400ed2 <+41>:
                                   je
                                  callq
   0x0000000000400ed4 <+43>:
                                          %rsp,%rbx
0x10(%rsp),%rbp
   0x00000000000400ed9 <+48>:
                                  mov
   0x0000000000400edc <+51>:
                                   lea
                                          0x4(%rbx),%eax
   0x0000000000400ee1 <+56>:
                                  mov
                                          (%rbx),%eax
   0x0000000000400ee4 <+59>:
                                   add
=> 0x00000000000400ee6 <+61>:
                                  cmp
                                          %eax,0x8(%rbx)
   0x00000000000400ee9 <+64>:
                                   je
                                          0x400ef0 <phase_2+71>
                                  callq
                                          0x40143d <explode bomb>
   0x0000000000400eeb <+66>:
   0x00000000000400ef0 <+71>:
                                          $0x4,%rbx
                                  add
   0x0000000000400ef4 <+75>:
                                   cmp
                                          %rbp,%rbx
                                          0x400ee1 <phase_2+56>
   0x0000000000400ef7 <+78>:
                                   ine
   0x00000000000400ef9 <+80>:
                                          0x18(%rsp),%rax
                                  mov
   0x0000000000400efe <+85>:
                                          %fs:0x28,%rax
                                   хог
   0x00000000000400f07 <+94>:
                                   ie
```

Since the value of eax is 3 and that of eax plus 8 in information register is 4 which is not equal leading to bomb exploitation. Thus, we know that the fourth value should be 3.

```
ssembler dump.
(gdb) i r
гах
                 0x3
                 0x7fffffffddc8
гЬх
                                         140737488346568
гсх
                 0 \times 0
                 0x7fffffffddd4
                                         140737488346580
rdх
rsi
                 0 \times 0
                                         0
rdi
                 0x7fffffffd750
                                         140737488344912
                                         0x7fffffffddd0
0x7fffffffddc0
                 0x7fffffffddd0
гЬр
                 0x7ffffffddc0
гsр
                 0xffffffff
                                         4294967295
г8
г9
                 0 \times 0
                 0x7ffff7f60ac0
                                         140737353484992
г10
г11
                 0x0
                 0x400c60
                                         4197472
г13
                 0x7fffffffdef0
                                         140737488346864
                 0x0
г14
г15
                 0x0
гір
                 0x400ee6
                                         0x400ee6 <phase_2+61>
                                         [ PF IF ]
eflags
                 0x206
cs
                 0x33
SS
                 0x2b
                                         43
ds
                 0×0
es
                 0x0
fs
                 0×0
                 0×0
gs
(gdb)
```

```
(gdb) x/d 0x7fffffffddd0
0x7fffffffddd0: 4
(gdb)
```

Run the program again giving the 3 as fourth input value. Then go to u* 0x0000000000000400ef4 the address of line 75. if the value of rbp and rbx is not equal, it will loop to execute line 56 or else it will proceed with next instruction if the value is equal.

```
0x00000000000400ef4
(gdb) u*
             0000400ef4 in phase
(gdb) disas

Dump of assembler code for function phase_2
0×00000000000400ea9 <+0>: push %rbp
0×00000000000400ea8 <+1>: push %rbx
50×2
                                                             $0x28,%rsp
%fs:0x28,%rax
%rax,0x18(%rsp)
    0x0000000000400eab <+2>:
0x0000000000400eaf <+6>:
0x00000000000400eb8 <+15>:
                                                  sub
                                                  mov
                                                  mov
    0x00000000000400ebd <+20>:
0x00000000000400ebf <+22>:
                                                             %eax,%eax
%rsp,%rsi
                                                  хог
                                                  mov
callq
    0x00000000000400ec2 <+25>:
    0x00000000000400ec7 <+30>:
                                                  cmpl
                                                             $0x0,(%rsp)
                                                             0x0000000000400ecb <+34>:
                                                  jne
cmpl
    0x00000000000400ecd <+36>:
     0x0000000000400ed2 <+41>:
    0x00000000000400ed4 <+43>:
0x00000000000400ed9 <+48>:
                                                  callq
                                                                           <explode_bomb>
                                                             %rsp,%rbx
0x10(%rsp),%rbp
0x4(%rbx),%eax
(%rbx),%eax
%eax,0x8(%rbx)
                                                  mov
     0x0000000000400edc <+51>:
                                                  lea
    0x0000000000400ee1 <+56>:
0x0000000000400ee4 <+59>:
                                                  mov
                                                  add
     0x0000000000400ee6 <+61>:
                                                  cmp
                                                  je
callq
add
    0x00000000000400ee9 <+64>:
                                                                                      2+71>
                                                                           <explode bomb>
    0x0000000000400eeb <+66>:
    0x0000000000400ef0 <+71>:
                                                             $0x4,%rbx
%rbp,%rbx
    0x00000000000400ef4 <+75>:
                                                  cmp
                                                             0x400ee1 chase_2+56>
0x18(%rsp),%rax
%fs:0x28,%rax
0x400f0e cphase_2+101>
0x400b00 <__stack_chk_fail@plt>
    0x0000000000400ef7 <+78>:
0x0000000000400ef9 <+80>:
                                                  jne
mov
     0x0000000000400efe <+85>:
                                                  хог
    0x0000000000400f07 <+94>:
0x0000000000400f09 <+96>:
                                                  je
callq
                                                  add
                                                             $0x28,%rsp
                                                             %гьх
     0x0000000000400f12 <+105>:
                                                  рор
                                                             %гьь
                                  <+106>:
                                                  pop
                                  <+107>:
                                                  retq
```

As the value of rbp is 3 where that of rbx is 1 which is not equal, so it will execute the line 56 again.

So go u* address of line 61. If the eax and 8 plus rbx value is equal it will execute line 71 or the bomb will get exploded.

```
(gdb) x/d $rbp
 x7fffffffddd0: 3
(gdb) x/d $rbx
0x7fffffffddc4: 1
(gdb) u* 0x0000000000400ee6
 (gdb)
  (0000000000400ee6 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
                                  push
   0x00000000000400ea9 <+0>:
                                          %гьр
                                          %гьх
   0x0000000000400eaa <+1>:
                                  push
   0x00000000000400eab <+2>:
                                          $0x28,%rsp
                                  sub
   0x00000000000400eaf <+6>:
0x00000000000400eb8 <+15>:
                                  mov
                                          %fs:0x28,%rax
                                          %rax,0x18(%rsp)
                                  mov
   0x0000000000400ebd <+20>:
                                  XOL
                                          %eax,%eax
   0x0000000000400ebf <+22>:
                                  mov
                                          %rsp,%rsi
   0x0000000000400ec2 <+25>:
                                  callq
                                          0x40145f <read six numbers>
   0x0000000000400ec7 <+30>:
                                          $0x0,(%rsp)
                                  cmpl
   0x00000000000400ecb <+34>:
                                  jne
                                          0x400ed4 <phase_2+43>
                                          $0x1,0x4(%rsp)
                                   cmpl
   0x0000000000400ecd <+36>:
                                  je
                                          0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x0000000000400ed2 <+41>:
   0x0000000000400ed4 <+43>:
                                   callq
                                          %rsp,%rbx
0x10(%rsp),%rbp
   0x00000000000400ed9 <+48>:
                                   mov
   0x0000000000400edc <+51>:
                                   lea
   0x0000000000400ee1 <+56>:
                                          0x4(%rbx),%eax
                                   mov
   0x00000000000400ee4 <+59>:
                                   add
                                          (%rbx),%eax
=> 0x00000000000400ee6 <+61>:
                                          %eax,0x8(%rbx)
                                   CMP
                                   je     0x400ef0 <phase_2+71>
callq     0x40143d <explode_bomb>
   0x00000000000400ee9 <+64>:
   0x00000000000400eeb <+66>:
   0x0000000000400ef0 <+71>:
                                          $0x4,%rbx
                                   add
```

Go to u* address of line 75 where there is a compared function. If the value of rbp and rbx is equal it will proceed with next instruction but it is not equal it will loop again to execute line 56. The value of rbp is 3 where as bx is 2 which is not equal, so it will execute line 56.

```
0x00000000000400ee6 <+61>:
                                 CMD
                                         %eax,0x8(%rbx)
 Ubuntu Software 400ee9 <+64>:
                                         0x400ef0 <phase 2+71>
                                 je
               400eeb <+66>:
                                 callq
                                        0x40143d <explode bomb>
                                         $0x4,%rbx
   0x00000000000400ef0 <+71>:
                                 add
=> 0x0000000000400ef4 <+75>:
                                         %rbp,%rbx
                                 CMP
   0x00000000000400ef7 <+78>:
                                 jne
                                         0x400ee1 <phase 2+56>
   0x00000000000400ef9 <+80>:
                                         0x18(%rsp),%rax
                                 MOV
   0x00000000000400efe <+85>:
                                 XOL
                                         %fs:0x28,%rax
   0x00000000000400f07 <+94>:
                                 je
                                         0x400f0e <phase 2+101>
   0x00000000000400f09 <+96>:
                                 callq
                                        0x400b00 < stack chk fail@plt>
   0x00000000000400f0e <+101>:
                                 add
                                         $0x28,%rsp
   0x00000000000400f12 <+105>:
                                         %rbx
                                 pop
   0x00000000000400f13 <+106>:
                                 pop
                                         %rbp
   0x00000000000400f14 <+107>:
                                 retq
End of assembler dump.
(gdb) x/d $rbp
 x7fffffffddd0: 3
(gdb) x/d $rbx
x7fffffffddcc: 2
```

u* address of line 61 and then go the disas of the line 60. Line 71 will be executed if eax value and rbx plus 8 value is equal. Otherwise, bomb will explode.

```
0x0000000000400ee6
     00000000400ee6 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
   0x00000000000400ea9 <+0>:
0x00000000000400eaa <+1>:
                                               %гБр
                                      push
                                      push
                                               %гьх
   0x0000000000400eab <+2>:
                                      sub
                                               $0x28,%rsp
   0x0000000000400eaf <+6>:
0x0000000000400eb8 <+15>:
                                               %fs:0x28,%rax
%rax,0x18(%rsp)
                                      MOV
                                      mov
   0x0000000000400ebd <+20>:
                                      хог
                                               %eax,%eax
   0x0000000000400ebf <+22>:
0x0000000000400ec2 <+25>:
                                               %rsp,%rsi
                                      MOV
                                      callq
                                                          <read_six_numbers>
   0x0000000000400ec7 <+30>:
                                      cmpl
                                               $0x0,(%rsp)
                                               0x400ed4 <phase 2+43>
   0x00000000000400ecb <+34>:
                                       jne
   0x00000000000400ecd <+36>:
                                       cmpl
                                               $0x1,0x4(%rsp)
                                               0x400ed9 <phase_2+48>
0x40143d <explode_bomb>
   0x0000000000400ed2 <+41>:
                                      je
                                      callq
   0x0000000000400ed4 <+43>:
                                               %rsp,%rbx
0x10(%rsp),%rbp
   0x0000000000400ed9 <+48>:
                                      mov
   0x0000000000400edc <+51>:
                                      lea
                                               0x4(%rbx),%eax
                          <+56>:
                                      mov
                          <+59>:
                                       add
                                               (%rbx),%eax
                                               %eax,0x8(%rbx)
=> 0x00000000000400ee6 <+61>:
                                      CMP
                                               0x400ef0 <phase_2+71>
0x40143d <explode_bomb>
                          <+64>:
                                       je
                                      callq
   0x0000000000400eeb <+66>:
   0x0000000000400ef0 <+71>:
                                      add
                                               $0x4,%rbx
   0x0000000000400ef4 <+75>:
                                               %rbp,%rbx
                                      cmp
                          <+78>:
                                       jne
                                               0x400ee1 <phase_2+56>
   0x0000000000400ef9 <+80>:
                                      mov
                                               0x18(%rsp),%rax
```

The eax value is 5 as well as rbx plus 8, line 71 will be executed.

```
гах
                   0x5
гЬх
                   0x7ffffffddcc
                                            140737488346572
гсх
                   0 \times 0
                                            Θ
гdх
                   0x7ffffffddd4
                                            140737488346580
rsi
                   0 \times 0
                                            0
rdi
                   0x7fffffffd750
                                            140737488344912
                                            0x7fffffffddd0
0x7fffffffddc0
                   0x7ffffffddd0
гЬр
гѕр
                   0x7ffffffddc0
                   0xffffffff
                                            4294967295
г9
                   0×0
                   0x7ffff7f60ac0
                                            140737353484992
г10
г11
                   0×0
                                            0
                   0x400c60
г12
                   0x7fffffffdef0
                                            140737488346864
г13
г14
                   0×0
г15
                   0×0
                                            0
                                            0x400ee6 <phase_2+61>
                   0x400ee6
rip
eflags
                                            [ PF IF
51
                                                     3
                   0x206
                   0x33
CS
                   0x2b
SS
                                            43
ds
                                            0
                   0 \times 0
es
fs
                   0 \times 0
                                            0
                   0 \times 0
                                            0
                  0 \times 0
(gdb) x/d 0x7ffffffddd4
   7fffffffddd4: 5
```

The line 75 which have compared function will be executed. The value of rbp and rbx is equal, so sixth value we gave is right.

```
0x00000000000400ee9 <+64>:
                                         0x400ef0 <phase 2+71>
   0x00000000000400eeb <+66>:
                                 callq
                                        0x40143d <explode bomb>
   0x0000000000400ef0 <+71>:
                                 add
                                         $0x4,%rbx
=> 0x0000000000400ef4 <+75>:
                                         %rbp,%rbx
                                 cmp
   0x00000000000400ef7 <+78>:
                                 jne
                                         0x400ee1 <phase 2+56>
   0x00000000000400ef9 <+80>:
                                 mov
                                         0x18(%rsp),%rax
   0x0000000000400efe <+85>:
                                 хог
                                         %fs:0x28,%rax
   0x00000000000400f07 <+94>:
                                 je
                                         0x400f0e <phase 2+101>
                                        0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f09 <+96>:
                                 callq
   0x00000000000400f0e <+101>:
                                 add
                                         $0x28,%rsp
   0x00000000000400f12 <+105>:
                                 pop
                                         %rbx
   0x00000000000400f13 <+106>:
                                         %гьр
                                 pop
   0x00000000000400f14 <+107>:
                                 reta
End of assembler dump.
(gdb) x/d $rbp
0x7fffffffddd0: 3
(gdb) x/d $rbx
)x7fffffffddd0: 3
```

Finally the phase_2 will get defused when we enter the Fibonacci series we got from the above.

```
(gdb) break into
Function "info" not defined.
 Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) info break
          Type
                             Disp Enb Address
                                                                What
          breakpoint
                                                  000000400ea9 <phase_2>
                            keep y 0x6
          breakpoint already hit 1 time
                             keep y 0x0000000000040143d <explode_bomb>
          breakpoint
(gdb) delete
 Delete all breakpoints? (y or n) y
(gdb) r answer.txt
The program being debugged has been started already.

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

Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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!
```

Phase_3

Set the break point at phase_3 as well as explode_bomb. Then go the disassemble file of the phase_3.

Run the program with phase_1 answer saved. Give the random string as the input a we are not aware of the input format of the phase_3.

```
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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!
heyy

Breakpoint 3, 0x0000000000400f15 in phase_3:

> 0x0000000000400f15 <-0>: sub 50x18, %rsp
0x0000000000400f15 <-0>: sub 50x18, %rsp
0x0000000000400f15 <-0>: sub 50x18, %rsp
0x0000000000400f10 <-4>: mov %rs.0x28, %rax
0x0000000000400f22 <-13>: mov %rax, 0x8(%rsp)
0x0000000000400f22 <-13>: mov %rax, 0x8(%rsp)
0x0000000000400f22 <-12>: lea 0x4(%rsp), %rcx
0x0000000000400f29 <-2>: lea 0x4(%rsp), %rcx
0x0000000000400f20 <-2>: mov %rsp, %rdx
0x0000000000400f30 <-333: callq
0x0000000000400f30 <-333: callq
0x0000000000400f30 <-333: callq
0x00000000000400f30 <-333: callq
0x0000000000400f40 <-4>: scops 3+48>
0x0000000000400f40 <-4>: scops 3+48>
0x0000000000400f40 <-4>: scops 3+48>
0x0000000000400f40 <-4>: scops 3+48>
0x0000000000400f40 <-5>: scops 30x1, %eax
0x0000000000400f50 <-6>: mov 0x400f51 <-6>: mov 0x400f61 <-6>: mov 0x400f61
```

Then run x/s 0x4025cf to see the answer input format contained by the phase_3. From this command we came to know that the answer format of phase_3 contains two integer value.

```
(gdb) x/s 0x4025cf: "%d %d"
(gdb)
0x4025d5: "Error: Premature EOF on stdin"
(gdb) r answer.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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 9
```

After getting hint about the answer format of the phase_3, we can run the program and give any two integer value.

```
(gdb) r answer.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sonam/Desktop/Fifth Semester/CS I/Assignment 1_2/Assignment 1/bomb001/bomb answer.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 9
```

Then go directly the function having compared value by u* address of the line 38. If the value is eax is greater than 1, then line 145 will get executed, otherwise line 48 will get executed.

The value of eax is 2 which is greater than 1, so line 48 will get executed.

Go to line 48 by unil* 0x000000000000400f45. The value 7 will be compared with the value of rsp. Incase rsp value is greater than 7 then, line 14 5 will get executed.

```
(gdb) disas
Dump of assembler code for function phase_3:
   0x00000000000400f15 <+0>:
0x00000000000400f19 <+4>:
                                           $0x18,%rsp
                                   sub
                                   mov
                                           %fs:0x28,%rax
   0x00000000000400f22 <+13>:
                                           %rax,0x8(%rsp)
                                   mov
   0x00000000000400f27 <+18>:
                                   хог
                                           %eax,%eax
   0x00000000000400f29 <+20>:
0x00000000000400f2e <+25>:
                                           0x4(%rsp),%rcx
                                   lea
                                           %rsp,%rdx
                                   MOV
   0x00000000000400f31 <+28>:
                                           $0x4025cf, %esi
                                   mov
                                   callq
   0x00000000000400f36 <+33>:
                                           0x400bb0 <__isoc99_sscanf@plt>
   0x00000000000400f3b <+38>:
                                   CMP
                                           $0x1,%eax
   0x00000000000400f3e <+41>:
                                           0x400f45 <phase_3+48>
                                   jg 0x400f45 <phase_3+48>
callq 0x40143d <explode_bomb>
   0x00000000000400f40 <+43>:
                                   cmpl
                                           $0x7,(%rsp)
=> 0x00000000000400f45 <+48>:
   0x00000000000400f49 <+52>:
                                                  a6 <phase_3+145>
                                   ja
   0x00000000000400f4b <+54>:
                                           (%rsp),%eax
                                   MOV
   0x00000000000400f4e <+57>:
                                           *0x402440(,%rax,8)
                                   jmpq
   0x00000000000400f55 <+64>:
                                           $0x134,%eax
                                   MOV
   0x00000000000400f5a <+69>:
                                   jmp
                                                  61 <phase_3+76>
                                           $θxθ,%eax
   0x00000000000400f5c <+71>:
                                   mov
   0x0000000000400f61 <+76>:
                                   sub
                                           $0x85,%eax
   0x00000000000400f66 <+81>:
                                           0x400f6d <phase_3+88>
                                   jmp
   0x00000000000400f68 <+83>:
                                   mov
                                           $0x0,%eax
                                   add
   0x00000000000400f6d <+88>:
                                           $0x201,%eax
```

As the rsp value is 2 which Is lesser than 7, it will execute next instruction or the line 155. Go to u* address of line 155..

The compared function will compare the 5 with that of value for rsp. If rsp is greater than the 7, line 167 will executed. Since rsp value is 2 which is less than 7, so line 16 1will get executed. Go to u* 0x00000000000000066.

```
0x00000000000400fab <+150>:
                                               $0x0,%eax
                                      mov
   0x00000000000400fb0 <+155>:
                                      cmpl
                                               $0x5,(%rsp)
   0x00000000000400fb4 <+159>:
                                                      fbc <phase 3+167>
                                      jg
                                               0x4(%rsp),%eax
   0x00000000000400fb6 <+161>:
                                      CMP
                                      je     0x400fc1 <phase_3+172>
callq     0x40143d <explode_bomb>
   0x00000000000400fba <+165>:
   0x00000000000400fbc <+167>:
   0x00000000000400fc1 <+172>:
                                               0x8(%rsp),%rax
                                      MOV
   0x00000000000400fc6 <+177>:
                                               %fs:0x28,%rax
                                      хог
   0x00000000000400fcf <+186>:
0x00000000000400fd1 <+188>:
                                               0x400fd6 <phase_3+193>
0x400b00 <__stack_chk_fail@plt>
                                      je
                                      callq
                                               $0x18,%rsp
   0x0000000000400fd6 <+193>:
                                      add
   0x00000000000400fda <+197>:
                                      retq
--Type <RET> for more, q to quit, c to continue without paging--c
End of assembler dump.
(gdb) x/d $rsp
            0x0000000000400fb6
```

The compared function will again compare the 4 plus rsp value and eax value. If the value is equal, line 172 will get executed, otherwise, bomb will get exploded. As the value of 4 plus rsp is 0 where as eax value is 409 which is not equal. So in order to have equal value, we can substitute value 409 as eax value to balanced the value of 4 plus rsp.

After getting the second integer value, we can run the program and before defusing phase_3, delete the break point, otherwise bomb will not get defused. Thus phase_3 of bomb001 is also defused.