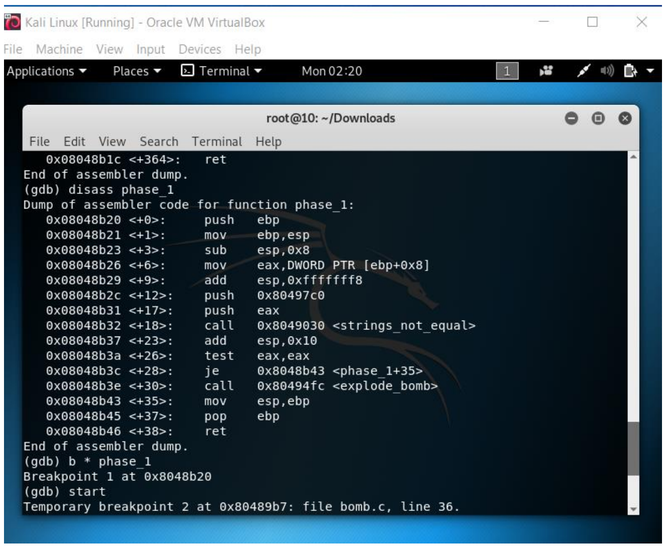
<u>Phase 1 –</u> (Public speaking is very easy.)

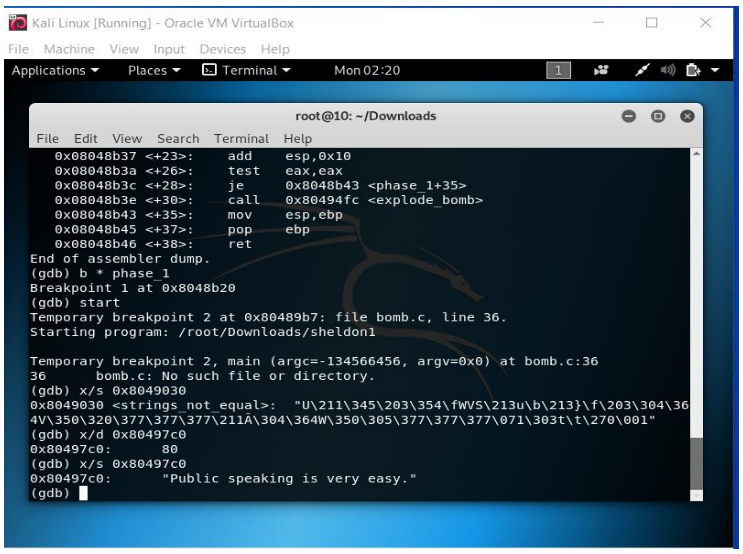
First it is essential to put a breakpoint on phase_1 function to stop the bomb from blowing up, then we can run the program and just add a random password and continue. Then disassemble the function.

Once the code has been disassembled it can be found that the string is moved to eax along with a memory address. By giving the command x/s 0x80497c0 we can access the memory address to see what's in it.

The command will then print the string "Public speaking is very easy." And that will be the password to defuse phase_1.


Kali Linux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Mon 02:20
root@10: ~/Downloads
File Edit View Search Terminal Help

```
   0x08048b1c <+364>:   ret
End of assembler dump.
(gdb) disass phase_1
Dump of assembler code for function phase_1:
   0x08048b20 <+0>:     push   ebp
   0x08048b21 <+1>:     mov    ebp,esp
   0x08048b23 <+3>:     sub    esp,0x8
   0x08048b26 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x08048b29 <+9>:     add    esp,0xfffffff8
   0x08048b2c <+12>:    push   0x80497c0
   0x08048b31 <+17>:    push   eax
   0x08048b32 <+18>:    call   0x8049030 <strings_not_equal>
   0x08048b37 <+23>:    add    esp,0x10
   0x08048b3a <+26>:    test   eax,eax
   0x08048b3c <+28>:    je     0x8048b43 <phase_1+35>
   0x08048b3e <+30>:    call   0x80494fc <explode_bomb>
   0x08048b43 <+35>:    mov    esp,ebp
   0x08048b45 <+37>:    pop    ebp
   0x08048b46 <+38>:    ret
End of assembler dump.
(gdb) b * phase_1
Breakpoint 1 at 0x8048b20
(gdb) start
Temporary breakpoint 2 at 0x80489b7: file bomb.c, line 36.
```


Kali Linux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Mon 02:20
root@10: ~/Downloads
File Edit View Search Terminal Help

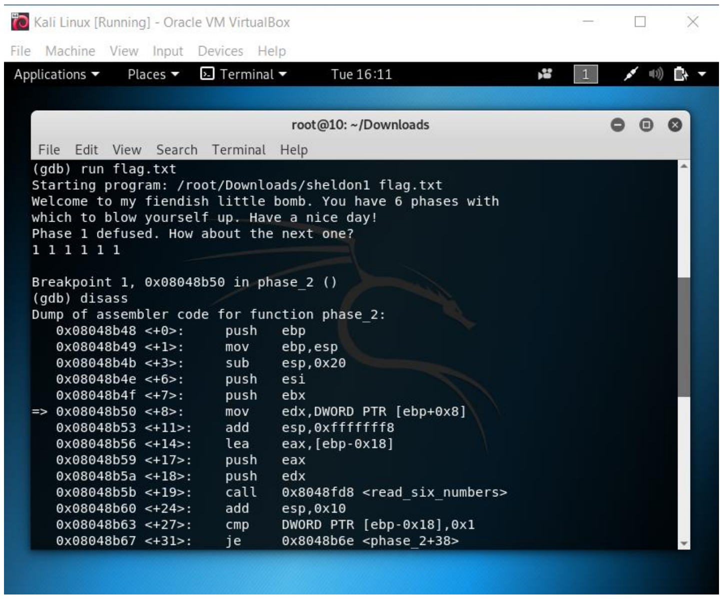
```
   0x08048b37 <+23>:    add    esp,0x10
   0x08048b3a <+26>:    test   eax,eax
   0x08048b3c <+28>:    je     0x8048b43 <phase_1+35>
   0x08048b3e <+30>:    call   0x80494fc <explode_bomb>
   0x08048b43 <+35>:    mov    esp,ebp
   0x08048b45 <+37>:    pop    ebp
   0x08048b46 <+38>:    ret
End of assembler dump.
(gdb) b * phase_1
Breakpoint 1 at 0x8048b20
(gdb) start
Temporary breakpoint 2 at 0x80489b7: file bomb.c, line 36.
Starting program: /root/Downloads/sheldon1

Temporary breakpoint 2, main (argc=-134566456, argv=0x0) at bomb.c:36
36      bomb.c: No such file or directory.
(gdb) x/s 0x8049030
0x8049030 <strings_not_equal>:  "U\211\345\203\354\fWVS\213u\b\213}\f\203\304\36
4V\350\320\377\377\377\211Ã\304\364W\350\305\377\377\377\071\303t\t\270\001"
(gdb) x/d 0x80497c0
0x80497c0:      80
(gdb) x/s 0x80497c0
0x80497c0:      "Public speaking is very easy."
(gdb) 
```

<u>Phase 2 –</u> (1 2 6 24 120 720)

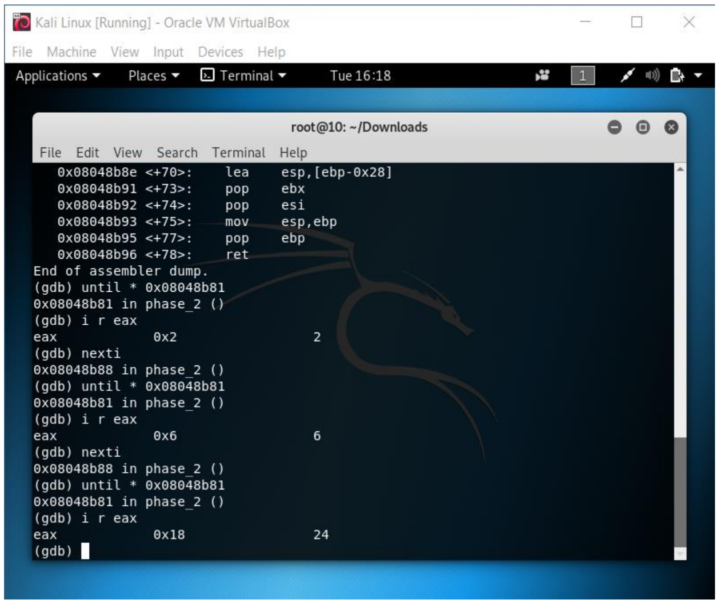
Upon disassembling phase_2 it can be seen that there is a function called "read-six_numbers" which means that the password to defuse phase_2 contains 6 numbers. Then we can identify that from +46 till ebx equals to 5. And there is also a cmp statement which makes the bomb go off by calling 'explode_bomb".

It can be seen that the first integer is being compared to 1, then jumoing into line +38. By using 'until' command to find what the second integer is being compared against. As it is being compared against eax, we can find that it holds the integer '2' which should be the second integer that the password requires.

This can be examined furthur by using the "nexti" command and get the value inside for every iteration which will be 1, 2, 6, 24, 120, and 720 respectively.



```
(gdb) run flag.txt
Starting program: /root/Downloads/sheldon1 flag.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?
1 1 1 1 1 1

Breakpoint 1, 0x08048b50 in phase_2 ()
(gdb) disass
Dump of assembler code for function phase_2:
   0x08048b48 <+0>:     push   ebp
   0x08048b49 <+1>:     mov    ebp,esp
   0x08048b4b <+3>:     sub    esp,0x20
   0x08048b4e <+6>:     push   esi
   0x08048b4f <+7>:     push   ebx
=> 0x08048b50 <+8>:     mov    edx,DWORD PTR [ebp+0x8]
   0x08048b53 <+11>:    add    esp,0xfffffff8
   0x08048b56 <+14>:    lea    eax,[ebp-0x18]
   0x08048b59 <+17>:    push   eax
   0x08048b5a <+18>:    push   edx
   0x08048b5b <+19>:    call   0x8048fd8 <read_six_numbers>
   0x08048b60 <+24>:    add    esp,0x10
   0x08048b63 <+27>:    cmp    DWORD PTR [ebp-0x18],0x1
   0x08048b67 <+31>:    je     0x8048b6e <phase_2+38>
```



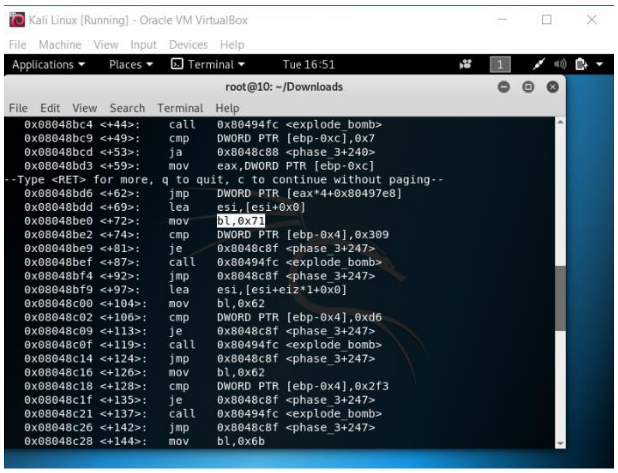
```
   0x08048b8e <+70>:    lea    esp,[ebp-0x28]
   0x08048b91 <+73>:    pop    ebx
   0x08048b92 <+74>:    pop    esi
   0x08048b93 <+75>:    mov    esp,ebp
   0x08048b95 <+77>:    pop    ebp
   0x08048b96 <+78>:    ret
End of assembler dump.
(gdb) until * 0x08048b81
0x08048b81 in phase_2 ()
(gdb) i r eax
eax            0x2                 2
(gdb) nexti
0x08048b88 in phase_2 ()
(gdb) until * 0x08048b81
0x08048b81 in phase_2 ()
(gdb) i r eax
eax            0x6                 6
(gdb) nexti
0x08048b88 in phase_2 ()
(gdb) until * 0x08048b81
0x08048b81 in phase_2 ()
(gdb) i r eax
eax            0x18                24
(gdb)
```

Phase 3 – (0 q 777)

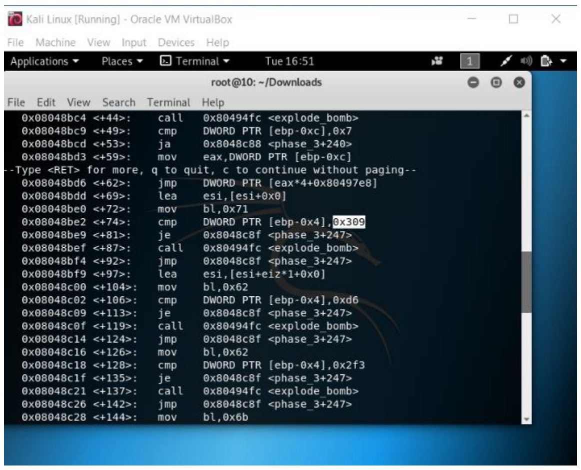
Once phase_3 is disassembled it can be seen that, there is a memory address connected with the scanf function. By using x/s command it can be seen that the password for the next level will be an integer, character and another integer respectively.

It can be seen that there are cases where inputs between 0 to 7 are accepted. In a case where 1 is given as the first integer, at line "+73" and "+72" the second integer is being compared against 0x390 and moving 0x71 to variable bl. 0x309 is 777 and 0x71 is 113 which is also letter q in ASCII.

Therefore, the password to defuse phase_3 will be 0 q 777.



```
   0x08048bc4 <+44>:    call   0x80494fc <explode_bomb>
   0x08048bc9 <+49>:    cmp    DWORD PTR [ebp-0xc],0x7
   0x08048bcd <+53>:    ja     0x8048c88 <phase_3+240>
   0x08048bd3 <+59>:    mov    eax,DWORD PTR [ebp-0xc]
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048bd6 <+62>:    jmp    DWORD PTR [eax*4+0x80497e8]
   0x08048bdd <+69>:    lea    esi,[esi+0x0]
   0x08048be0 <+72>:    mov    bl,0x71
   0x08048be2 <+74>:    cmp    DWORD PTR [ebp-0x4],0x309
   0x08048be9 <+81>:    je     0x8048c8f <phase_3+247>
   0x08048bef <+87>:    call   0x80494fc <explode_bomb>
   0x08048bf4 <+92>:    jmp    0x8048c8f <phase_3+247>
   0x08048bf9 <+97>:    lea    esi,[esi+eiz*1+0x0]
   0x08048c00 <+104>:   mov    bl,0x62
   0x08048c02 <+106>:   cmp    DWORD PTR [ebp-0x4],0xd6
   0x08048c09 <+113>:   je     0x8048c8f <phase_3+247>
   0x08048c0f <+119>:   call   0x80494fc <explode_bomb>
   0x08048c14 <+124>:   jmp    0x8048c8f <phase_3+247>
   0x08048c16 <+126>:   mov    bl,0x62
   0x08048c18 <+128>:   cmp    DWORD PTR [ebp-0x4],0x2f3
   0x08048c1f <+135>:   je     0x8048c8f <phase_3+247>
   0x08048c21 <+137>:   call   0x80494fc <explode_bomb>
   0x08048c26 <+142>:   jmp    0x8048c8f <phase_3+247>
   0x08048c28 <+144>:   mov    bl,0x6b
```

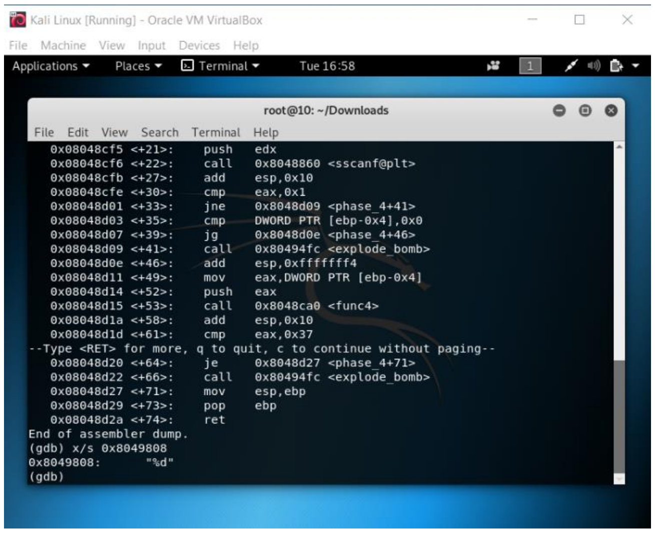


```
   0x08048bc4 <+44>:    call   0x80494fc <explode_bomb>
   0x08048bc9 <+49>:    cmp    DWORD PTR [ebp-0xc],0x7
   0x08048bcd <+53>:    ja     0x8048c88 <phase_3+240>
   0x08048bd3 <+59>:    mov    eax,DWORD PTR [ebp-0xc]
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048bd6 <+62>:    jmp    DWORD PTR [eax*4+0x80497e8]
   0x08048bdd <+69>:    lea    esi,[esi+0x0]
   0x08048be0 <+72>:    mov    bl,0x71
   0x08048be2 <+74>:    cmp    DWORD PTR [ebp-0x4],0x309
   0x08048be9 <+81>:    je     0x8048c8f <phase_3+247>
   0x08048bef <+87>:    call   0x80494fc <explode_bomb>
   0x08048bf4 <+92>:    jmp    0x8048c8f <phase_3+247>
   0x08048bf9 <+97>:    lea    esi,[esi+eiz*1+0x0]
   0x08048c00 <+104>:   mov    bl,0x62
   0x08048c02 <+106>:   cmp    DWORD PTR [ebp-0x4],0xd6
   0x08048c09 <+113>:   je     0x8048c8f <phase_3+247>
   0x08048c0f <+119>:   call   0x80494fc <explode_bomb>
   0x08048c14 <+124>:   jmp    0x8048c8f <phase_3+247>
   0x08048c16 <+126>:   mov    bl,0x62
   0x08048c18 <+128>:   cmp    DWORD PTR [ebp-0x4],0x2f3
   0x08048c1f <+135>:   je     0x8048c8f <phase_3+247>
   0x08048c21 <+137>:   call   0x80494fc <explode_bomb>
   0x08048c26 <+142>:   jmp    0x8048c8f <phase_3+247>
   0x08048c28 <+144>:   mov    bl,0x6b
```

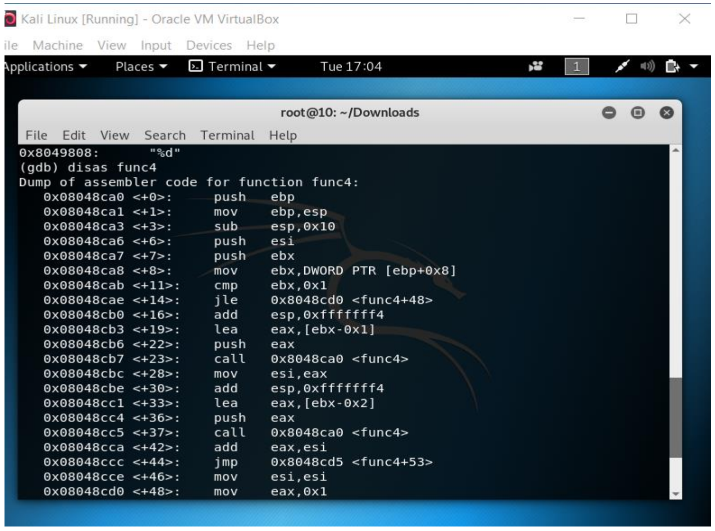
Phase_4

Upon disassembling phase_4, it can be seen that there is a memory address pushed into scanf. With the use of x/s command the memory address can be examined, and it seems that the password should be an integer.

The integer is then checked if it is greater than 0 not, and then the integer is moved to eax. And eax is being compared to 0x37, which is 55 in decimal. After checking "func3" it can be seen that it is an implementation of the Fibonacci sequence, hence in order for 55 to return it should be the Fibonacci number 10 and because if 0 or 1 was given as input it will return 1. Therefore, password should be 10 -1 = 9.


Kali Linux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Tue 16:58
root@10: ~/Downloads
File Edit View Search Terminal Help

```
   0x08048cf5 <+21>:    push   edx
   0x08048cf6 <+22>:    call   0x8048860 <sscanf@plt>
   0x08048cfb <+27>:    add    esp,0x10
   0x08048cfe <+30>:    cmp    eax,0x1
   0x08048d01 <+33>:    jne    0x8048d09 <phase_4+41>
   0x08048d03 <+35>:    cmp    DWORD PTR [ebp-0x4],0x0
   0x08048d07 <+39>:    jg     0x8048d0e <phase_4+46>
   0x08048d09 <+41>:    call   0x80494fc <explode_bomb>
   0x08048d0e <+46>:    add    esp,0xfffffff4
   0x08048d11 <+49>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08048d14 <+52>:    push   eax
   0x08048d15 <+53>:    call   0x8048ca0 <func4>
   0x08048d1a <+58>:    add    esp,0x10
   0x08048d1d <+61>:    cmp    eax,0x37
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048d20 <+64>:    je     0x8048d27 <phase_4+71>
   0x08048d22 <+66>:    call   0x80494fc <explode_bomb>
   0x08048d27 <+71>:    mov    esp,ebp
   0x08048d29 <+73>:    pop    ebp
   0x08048d2a <+74>:    ret
End of assembler dump.
(gdb) x/s 0x8049808
0x8049808:      "%d"
(gdb)
```


Kali Linux [Running] - Oracle VM VirtualBox
ile Machine View Input Devices Help
Applications Places Terminal Tue 17:04
root@10: ~/Downloads
File Edit View Search Terminal Help

```
0x8049808:      "%d"
(gdb) disas func4
Dump of assembler code for function func4:
   0x08048ca0 <+0>:     push   ebp
   0x08048ca1 <+1>:     mov    ebp,esp
   0x08048ca3 <+3>:     sub    esp,0x10
   0x08048ca6 <+6>:     push   esi
   0x08048ca7 <+7>:     push   ebx
   0x08048ca8 <+8>:     mov    ebx,DWORD PTR [ebp+0x8]
   0x08048cab <+11>:    cmp    ebx,0x1
   0x08048cae <+14>:    jle    0x8048cd0 <func4+48>
   0x08048cb0 <+16>:    add    esp,0xfffffff4
   0x08048cb3 <+19>:    lea    eax,[ebx-0x1]
   0x08048cb6 <+22>:    push   eax
   0x08048cb7 <+23>:    call   0x8048ca0 <func4>
   0x08048cbc <+28>:    mov    esi,eax
   0x08048cbe <+30>:    add    esp,0xfffffff4
   0x08048cc1 <+33>:    lea    eax,[ebx-0x2]
   0x08048cc4 <+36>:    push   eax
   0x08048cc5 <+37>:    call   0x8048ca0 <func4>
   0x08048cca <+42>:    add    eax,esi
   0x08048ccc <+44>:    jmp    0x8048cd5 <func4+53>
   0x08048cce <+46>:    mov    esi,esi
   0x08048cd0 <+48>:    mov    eax,0x1
```