Advanced Network Security

Assignment – Sheldon1

Student Name: P. Anojithan

Student Number: MS19814766

Course: M.Sc. Information Technology specialization in Cyber Security

Introduction

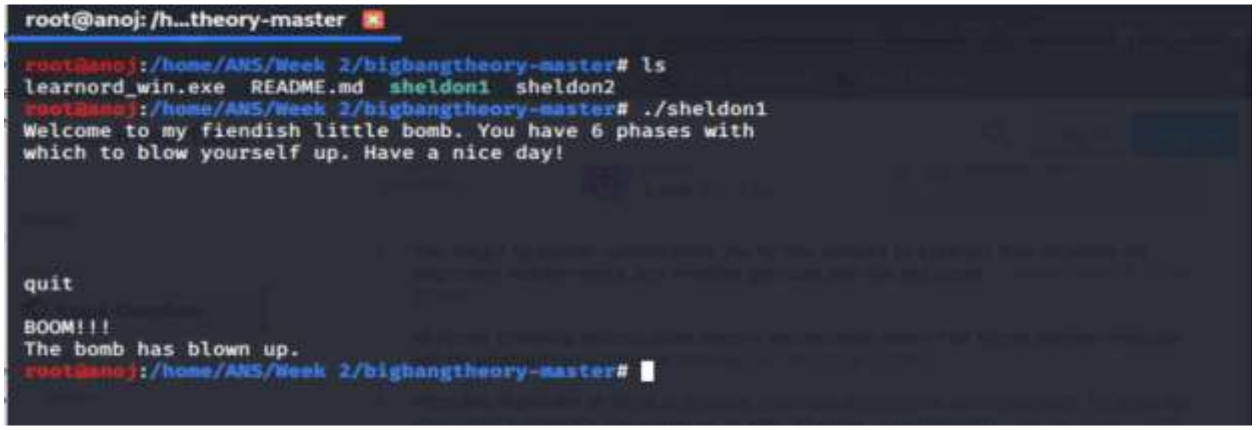
This assignment has been done to test mid-level experience in Assembly Language. Here I used Kali Linux to do the practical. I have completed this report with the screenshots I have taken during the lab.

Objective of the lab is to diffuse the bomb by code which has six phases.

Procedure

Step 1: Navigate the file and run the file.

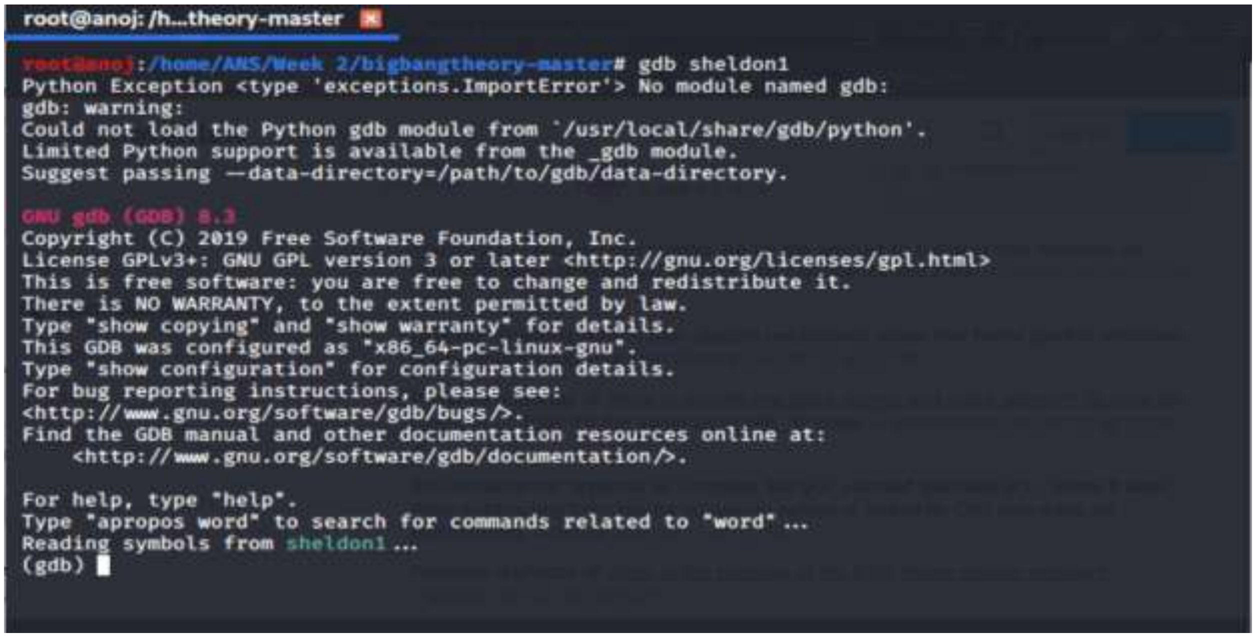
./schldon1



```
root@anoj: /h...theory-master

root@anoj:/home/ANS/Week 2/bigbangtheory-master# ls
learnord_win.exe  README.md  sheldon1  sheldon2
root@anoj:/home/ANS/Week 2/bigbangtheory-master# ./sheldon1
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!



quit

BOOM!!!
The bomb has blown up.
root@anoj:/home/ANS/Week 2/bigbangtheory-master# 
```

Step 2: Opened the file using gdb.

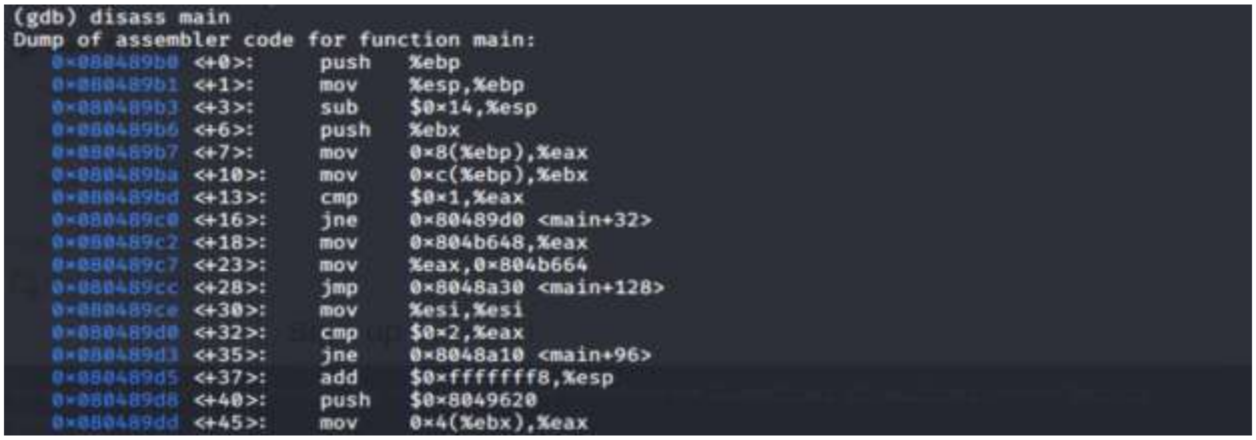
gdb sheldon1



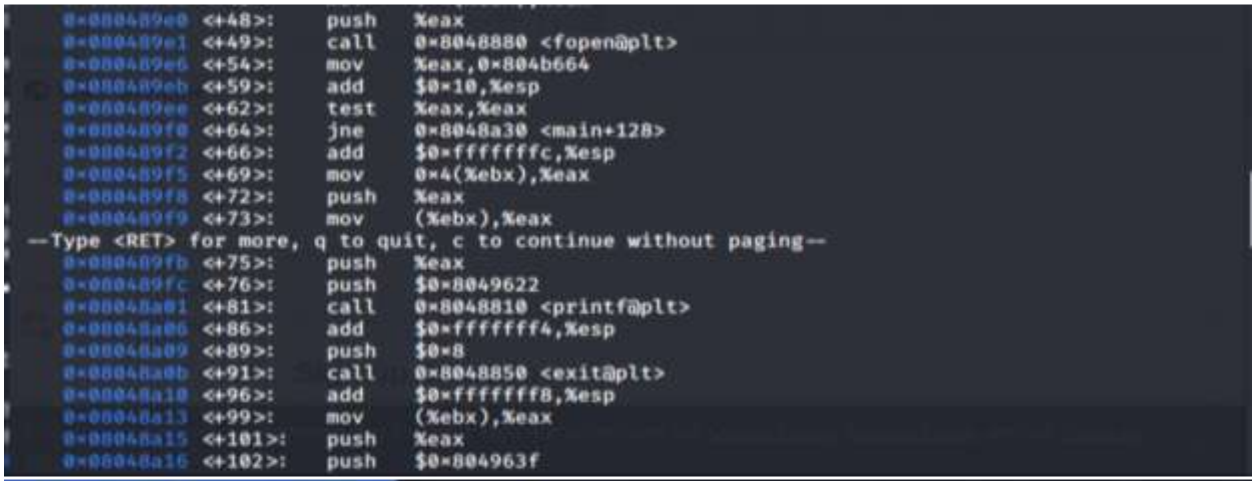
```
root@anoj: /h...theory-master

root@anoj:/home/ANS/Week 2/bigbangtheory-master# gdb sheldon1
Python Exception <type 'exceptions.ImportError'> No module named gdb:
gdb: warning:
Could not load the Python gdb module from `/usr/local/share/gdb/python'.
Limited Python support is available from the _gdb module.
Suggest passing --data-directory=/path/to/gdb/data-directory.

GNU gdb (GDB) 8.3
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sheldon1...
(gdb) 
```

Step 3: dissembled the main function.

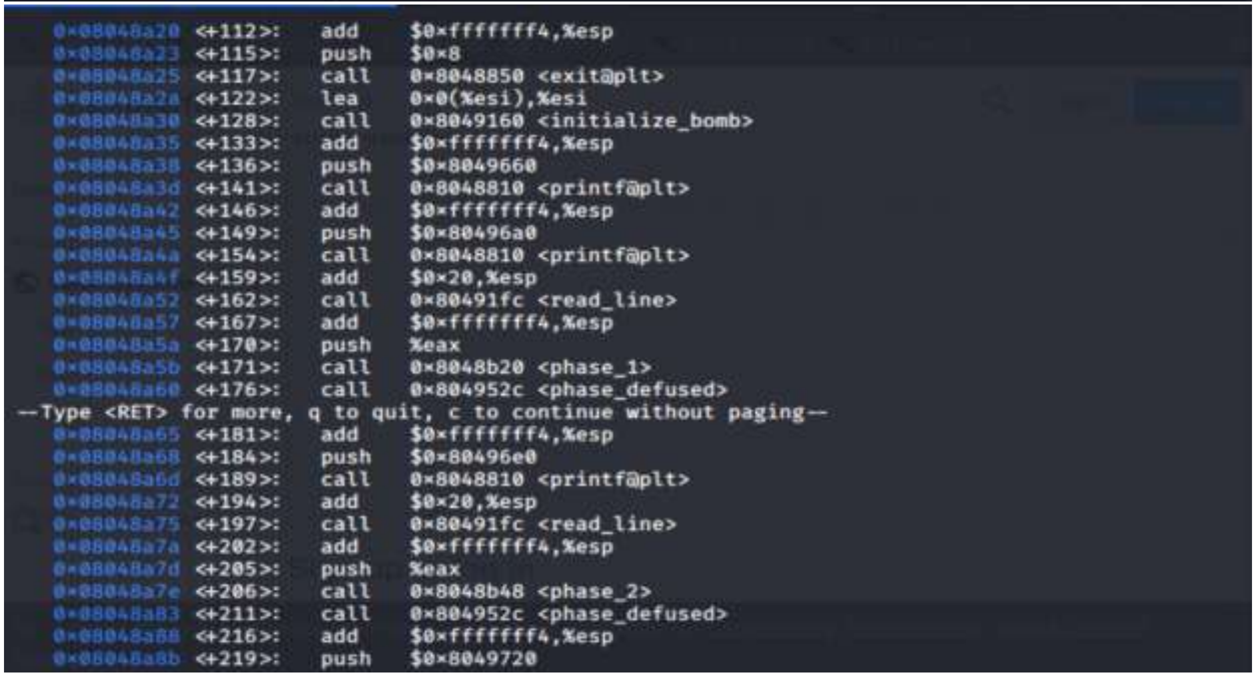
disass main



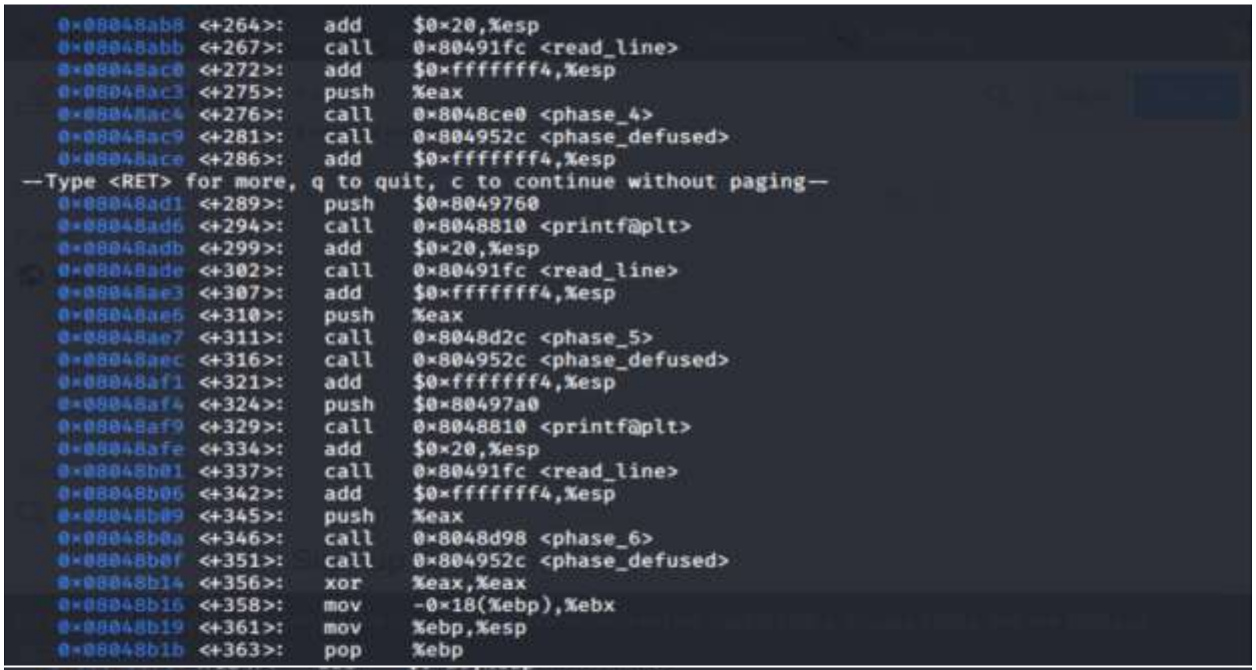
```
(gdb) disass main
Dump of assembler code for function main:
   0x080489b0 <+0>:     push   %ebp
   0x080489b1 <+1>:     mov    %esp,%ebp
   0x080489b3 <+3>:     sub    $0x14,%esp
   0x080489b6 <+6>:     push   %ebx
   0x080489b7 <+7>:     mov    0x8(%ebp),%eax
   0x080489ba <+10>:    mov    0xc(%ebp),%ebx
   0x080489bd <+13>:    cmp    $0x1,%eax
   0x080489c0 <+16>:    jne    0x80489d0 <main+32>
   0x080489c2 <+18>:    mov    0x804b648,%eax
   0x080489c7 <+23>:    mov    %eax,0x804b664
   0x080489cc <+28>:    jmp    0x8048a30 <main+128>
   0x080489ce <+30>:    mov    %esi,%esi
   0x080489d0 <+32>:    cmp    $0x2,%eax
   0x080489d3 <+35>:    jne    0x8048a10 <main+96>
   0x080489d5 <+37>:    add    $0xfffffff8,%esp
   0x080489d8 <+40>:    push   $0x8049620
   0x080489dd <+45>:    mov    0x4(%ebx),%eax
```



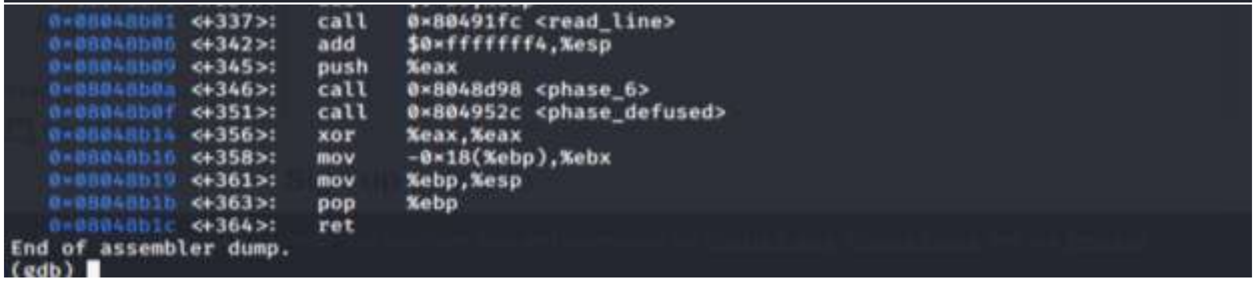
```
   0x080489e0 <+48>:    push   %eax
   0x080489e1 <+49>:    call   0x8048880 <fopen@plt>
   0x080489e6 <+54>:    mov    %eax,0x804b664
   0x080489eb <+59>:    add    $0x10,%esp
   0x080489ee <+62>:    test   %eax,%eax
   0x080489f0 <+64>:    jne    0x8048a30 <main+128>
   0x080489f2 <+66>:    add    $0xfffffffc,%esp
   0x080489f5 <+69>:    mov    0x4(%ebx),%eax
   0x080489f8 <+72>:    push   %eax
   0x080489f9 <+73>:    mov    (%ebx),%eax
--Type <RET> for more, q to quit, c to continue without paging--
   0x080489fb <+75>:    push   %eax
   0x080489fc <+76>:    push   $0x8049622
   0x08048a01 <+81>:    call   0x8048810 <printf@plt>
   0x08048a06 <+86>:    add    $0xfffffff4,%esp
   0x08048a09 <+89>:    push   $0x8
   0x08048a0b <+91>:    call   0x8048850 <exit@plt>
   0x08048a10 <+96>:    add    $0xfffffff8,%esp
   0x08048a13 <+99>:    mov    (%ebx),%eax
   0x08048a15 <+101>:   push   %eax
   0x08048a16 <+102>:   push   $0x804963f
```



```
   0x08048a20 <+112>:   add    $0xfffffff4,%esp
   0x08048a23 <+115>:   push   $0x8
   0x08048a25 <+117>:   call   0x8048850 <exit@plt>
   0x08048a2a <+122>:   lea    0x0(%esi),%esi
   0x08048a30 <+128>:   call   0x8049160 <initialize_bomb>
   0x08048a35 <+133>:   add    $0xfffffff4,%esp
   0x08048a38 <+136>:   push   $0x8049660
   0x08048a3d <+141>:   call   0x8048810 <printf@plt>
   0x08048a42 <+146>:   add    $0xfffffff4,%esp
   0x08048a45 <+149>:   push   $0x80496a0
   0x08048a4a <+154>:   call   0x8048810 <printf@plt>
   0x08048a4f <+159>:   add    $0x20,%esp
   0x08048a52 <+162>:   call   0x80491fc <read_line>
   0x08048a57 <+167>:   add    $0xfffffff4,%esp
   0x08048a5a <+170>:   push   %eax
   0x08048a5b <+171>:   call   0x8048b20 <phase_1>
   0x08048a60 <+176>:   call   0x804952c <phase_defused>
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048a65 <+181>:   add    $0xfffffff4,%esp
   0x08048a68 <+184>:   push   $0x80496e0
   0x08048a6d <+189>:   call   0x8048810 <printf@plt>
   0x08048a72 <+194>:   add    $0x20,%esp
   0x08048a75 <+197>:   call   0x80491fc <read_line>
   0x08048a7a <+202>:   add    $0xfffffff4,%esp
   0x08048a7d <+205>:   push   %eax
   0x08048a7e <+206>:   call   0x8048b48 <phase_2>
   0x08048a83 <+211>:   call   0x804952c <phase_defused>
   0x08048a88 <+216>:   add    $0xfffffff4,%esp
   0x08048a8b <+219>:   push   $0x8049720
```



```
   0x08048ab8 <+264>:   add    $0x20,%esp
   0x08048abb <+267>:   call   0x80491fc <read_line>
   0x08048ac0 <+272>:   add    $0xfffffff4,%esp
   0x08048ac3 <+275>:   push   %eax
   0x08048ac4 <+276>:   call   0x8048ce0 <phase_4>
   0x08048ac9 <+281>:   call   0x804952c <phase_defused>
   0x08048ace <+286>:   add    $0xfffffff4,%esp
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048ad1 <+289>:   push   $0x8049760
   0x08048ad6 <+294>:   call   0x8048810 <printf@plt>
   0x08048adb <+299>:   add    $0x20,%esp
   0x08048ade <+302>:   call   0x80491fc <read_line>
   0x08048ae3 <+307>:   add    $0xfffffff4,%esp
   0x08048ae6 <+310>:   push   %eax
   0x08048ae7 <+311>:   call   0x8048d2c <phase_5>
   0x08048aec <+316>:   call   0x804952c <phase_defused>
   0x08048af1 <+321>:   add    $0xfffffff4,%esp
   0x08048af4 <+324>:   push   $0x80497a0
   0x08048af9 <+329>:   call   0x8048810 <printf@plt>
   0x08048afe <+334>:   add    $0x20,%esp
   0x08048b01 <+337>:   call   0x80491fc <read_line>
   0x08048b06 <+342>:   add    $0xfffffff4,%esp
   0x08048b09 <+345>:   push   %eax
   0x08048b0a <+346>:   call   0x8048d98 <phase_6>
   0x08048b0f <+351>:   call   0x804952c <phase_defused>
   0x08048b14 <+356>:   xor    %eax,%eax
   0x08048b16 <+358>:   mov    -0x18(%ebp),%ebx
   0x08048b19 <+361>:   mov    %ebp,%esp
   0x08048b1b <+363>:   pop    %ebp
```

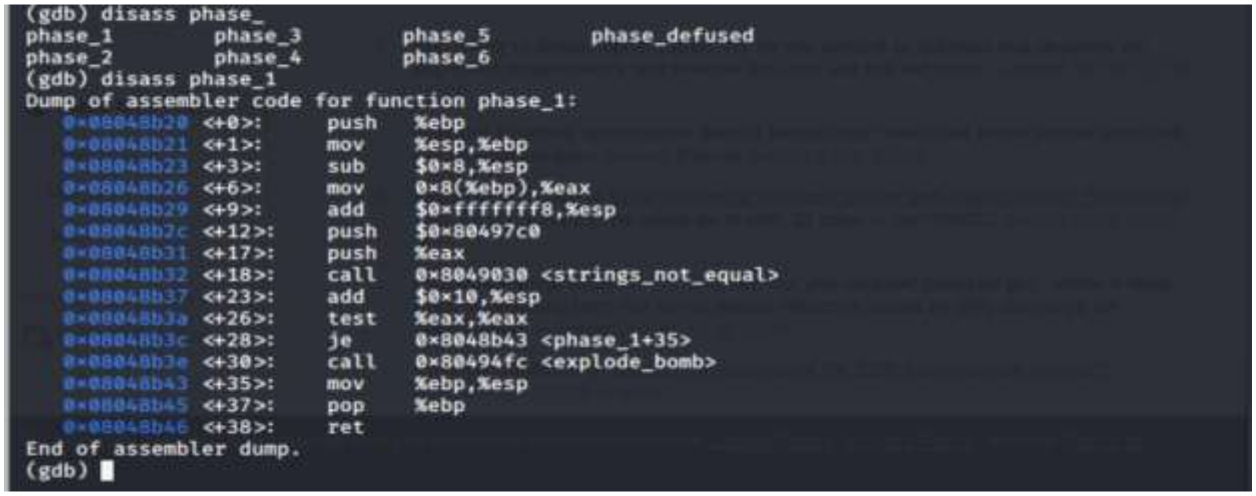


```
   0x08048b01 <+337>:   call   0x80491fc <read_line>
   0x08048b06 <+342>:   add    $0xfffffff4,%esp
   0x08048b09 <+345>:   push   %eax
   0x08048b0a <+346>:   call   0x8048d98 <phase_6>
   0x08048b0f <+351>:   call   0x804952c <phase_defused>
   0x08048b14 <+356>:   xor    %eax,%eax
   0x08048b16 <+358>:   mov    -0x18(%ebp),%ebx
   0x08048b19 <+361>:   mov    %ebp,%esp
   0x08048b1b <+363>:   pop    %ebp
   0x08048b1c <+364>:   ret
End of assembler dump.
(gdb) 
```

After disassembled the main, its visible there are 6 phases involved namely, Phase_1, Phase_2, Phase_3, Phase_4, Phase_5 and Phase_6.

Step 4: Disassembled phase_1

disass Phase_1

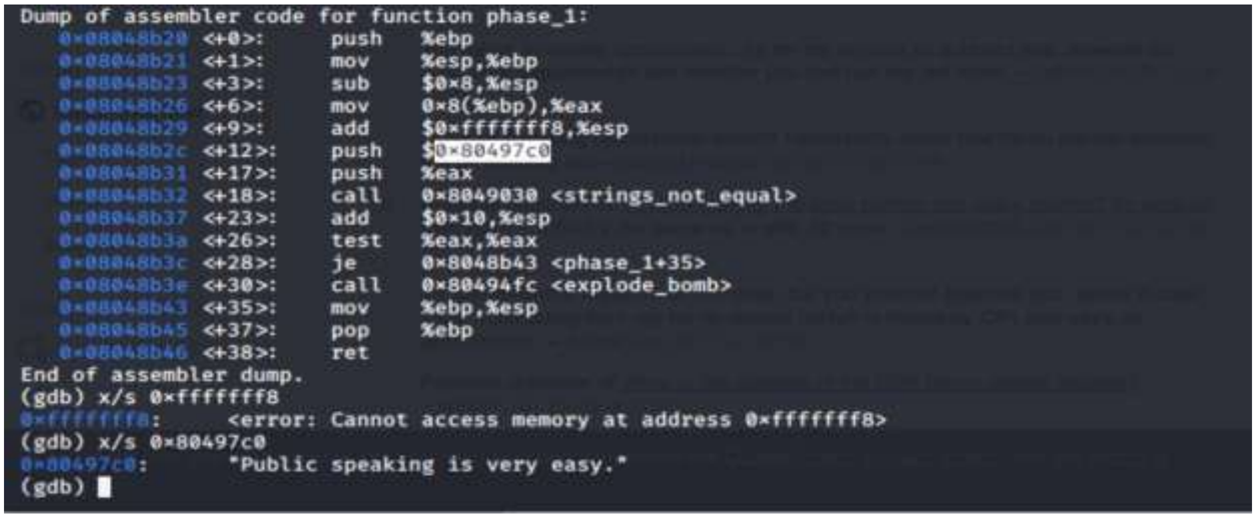


```
(gdb) disass phase_
phase_1         phase_3         phase_5         phase_defused
phase_2         phase_4         phase_6
(gdb) disass phase_1
Dump of assembler code for function phase_1:
   0x08048b20 <+0>:     push   %ebp
   0x08048b21 <+1>:     mov    %esp,%ebp
   0x08048b23 <+3>:     sub    $0x8,%esp
   0x08048b26 <+6>:     mov    0x8(%ebp),%eax
   0x08048b29 <+9>:     add    $0xfffffff8,%esp
   0x08048b2c <+12>:    push   $0x80497c0
   0x08048b31 <+17>:    push   %eax
   0x08048b32 <+18>:    call   0x8049030 <strings_not_equal>
   0x08048b37 <+23>:    add    $0x10,%esp
   0x08048b3a <+26>:    test   %eax,%eax
   0x08048b3c <+28>:    je     0x8048b43 <phase_1+35>
   0x08048b3e <+30>:    call   0x80494fc <explode_bomb>
   0x08048b43 <+35>:    mov    %ebp,%esp
   0x08048b45 <+37>:    pop    %ebp
   0x08048b46 <+38>:    ret
End of assembler dump.
(gdb) 
```

Step 5: checked for string values.

x/s 0x80497c0

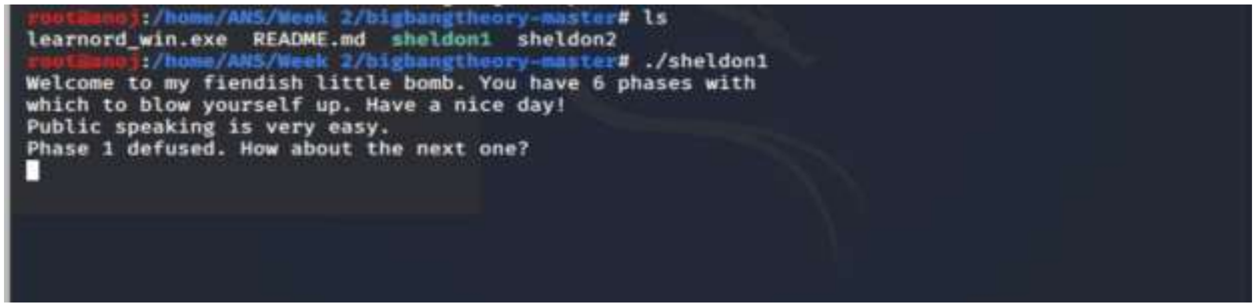
The string reviled. It was “Public speaking is very easy.”



```
Dump of assembler code for function phase_1:
   0x08048b20 <+0>:     push   %ebp
   0x08048b21 <+1>:     mov    %esp,%ebp
   0x08048b23 <+3>:     sub    $0x8,%esp
   0x08048b26 <+6>:     mov    0x8(%ebp),%eax
   0x08048b29 <+9>:     add    $0xfffffff8,%esp
   0x08048b2c <+12>:    push   $0x80497c0
   0x08048b31 <+17>:    push   %eax
   0x08048b32 <+18>:    call   0x8049030 <strings_not_equal>
   0x08048b37 <+23>:    add    $0x10,%esp
   0x08048b3a <+26>:    test   %eax,%eax
   0x08048b3c <+28>:    je     0x8048b43 <phase_1+35>
   0x08048b3e <+30>:    call   0x80494fc <explode_bomb>
   0x08048b43 <+35>:    mov    %ebp,%esp
   0x08048b45 <+37>:    pop    %ebp
   0x08048b46 <+38>:    ret
End of assembler dump.
(gdb) x/s 0xfffffff8
0xfffffff8:     <error: Cannot access memory at address 0xfffffff8>
(gdb) x/s 0x80497c0
0x80497c0:      "Public speaking is very easy."
(gdb)
```

Step 6: Phase 1 diffused

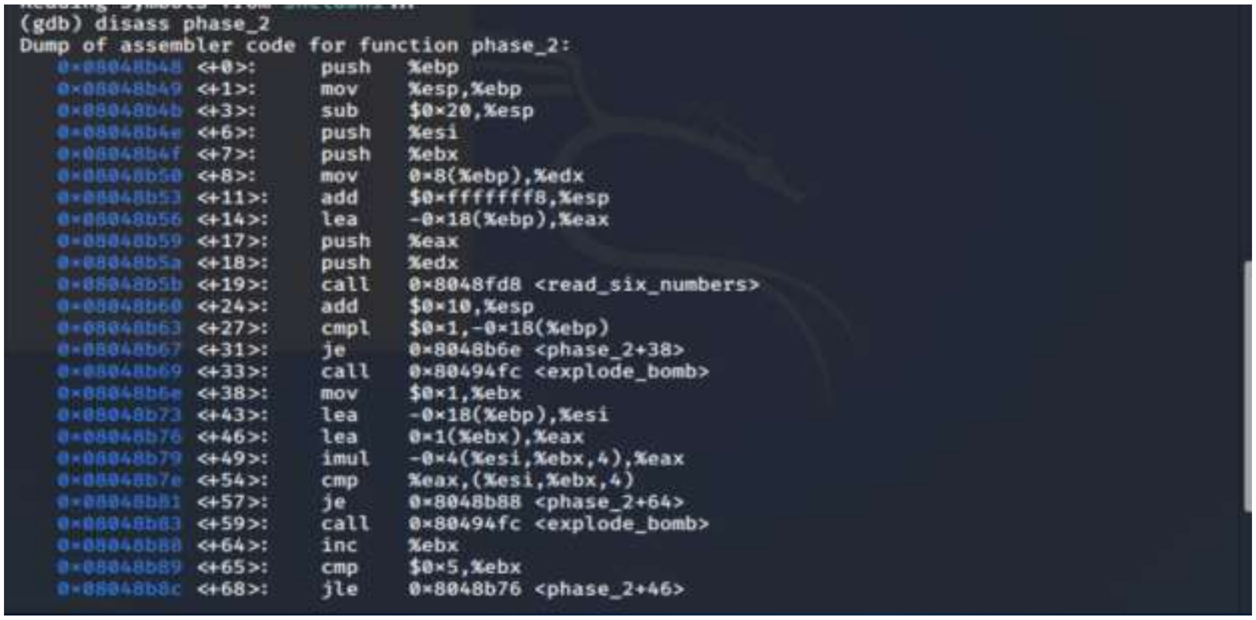
Run the file sheldon1 (./sheldon1)and pasted the text retrieved from the phase 1.



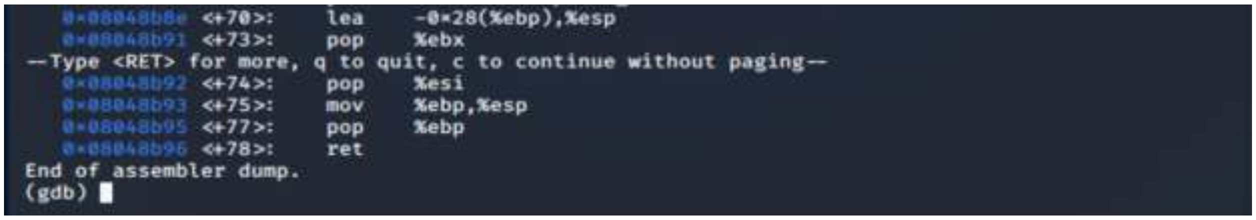
```
root@manoj:/home/ANS/Week 2/bigbangtheory-master# ls
learnord_win.exe  README.md  sheldon1  sheldon2
root@manoj:/home/ANS/Week 2/bigbangtheory-master# ./sheldon1
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
```

Step 7: disassembled phase_2

disass Phase_2



```
(gdb) disass phase_2
Dump of assembler code for function phase_2:
   0x08048b48 <+0>:     push   %ebp
   0x08048b49 <+1>:     mov    %esp,%ebp
   0x08048b4b <+3>:     sub    $0x20,%esp
   0x08048b4e <+6>:     push   %esi
   0x08048b4f <+7>:     push   %ebx
   0x08048b50 <+8>:     mov    0x8(%ebp),%edx
   0x08048b53 <+11>:    add    $0xfffffff8,%esp
   0x08048b56 <+14>:    lea    -0x18(%ebp),%eax
   0x08048b59 <+17>:    push   %eax
   0x08048b5a <+18>:    push   %edx
   0x08048b5b <+19>:    call   0x8048fd8 <read_six_numbers>
   0x08048b60 <+24>:    add    $0x10,%esp
   0x08048b63 <+27>:    cmpl   $0x1,-0x18(%ebp)
   0x08048b67 <+31>:    je     0x8048b6e <phase_2+38>
   0x08048b69 <+33>:    call   0x80494fc <explode_bomb>
   0x08048b6e <+38>:    mov    $0x1,%ebx
   0x08048b73 <+43>:    lea    -0x18(%ebp),%esi
   0x08048b76 <+46>:    lea    0x1(%ebx),%eax
   0x08048b79 <+49>:    imul   -0x4(%esi,%ebx,4),%eax
   0x08048b7e <+54>:    cmp    %eax,(%esi,%ebx,4)
   0x08048b81 <+57>:    je     0x8048b88 <phase_2+64>
   0x08048b83 <+59>:    call   0x80494fc <explode_bomb>
   0x08048b88 <+64>:    inc    %ebx
   0x08048b89 <+65>:    cmp    $0x5,%ebx
   0x08048b8c <+68>:    jle    0x8048b76 <phase_2+46>
```



```
   0x08048b8e <+70>:    lea    -0x28(%ebp),%esp
   0x08048b91 <+73>:    pop    %ebx
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048b92 <+74>:    pop    %esi
   0x08048b93 <+75>:    mov    %ebp,%esp
   0x08048b95 <+77>:    pop    %ebp
   0x08048b96 <+78>:    ret
End of assembler dump.
(gdb) 
```

Here the functions expecting a 6 numbers which runs in a loop, where the condition is i=0, i<=5 and i++

eax = ebx + 1

eax = [esi + ebx*4 - 4]

for each iteration the calculated values were,

0: 1

1: 2

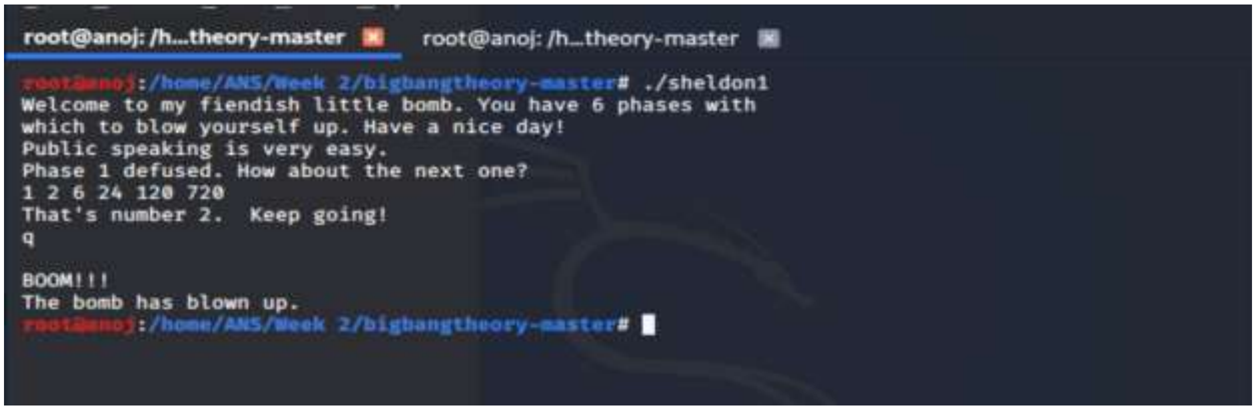
2: 6

3: 24

4:120

5: 720

Step 8: Checked for the numbers and it worked.



```
root@anoj:/home/ANS/Week 2/bigbangtheory-master# ./sheldon1
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
1 2 6 24 120 720
That's number 2.  Keep going!
q

BOOM!!!
The bomb has blown up.
root@anoj:/home/ANS/Week 2/bigbangtheory-master# 
```

Step 9: