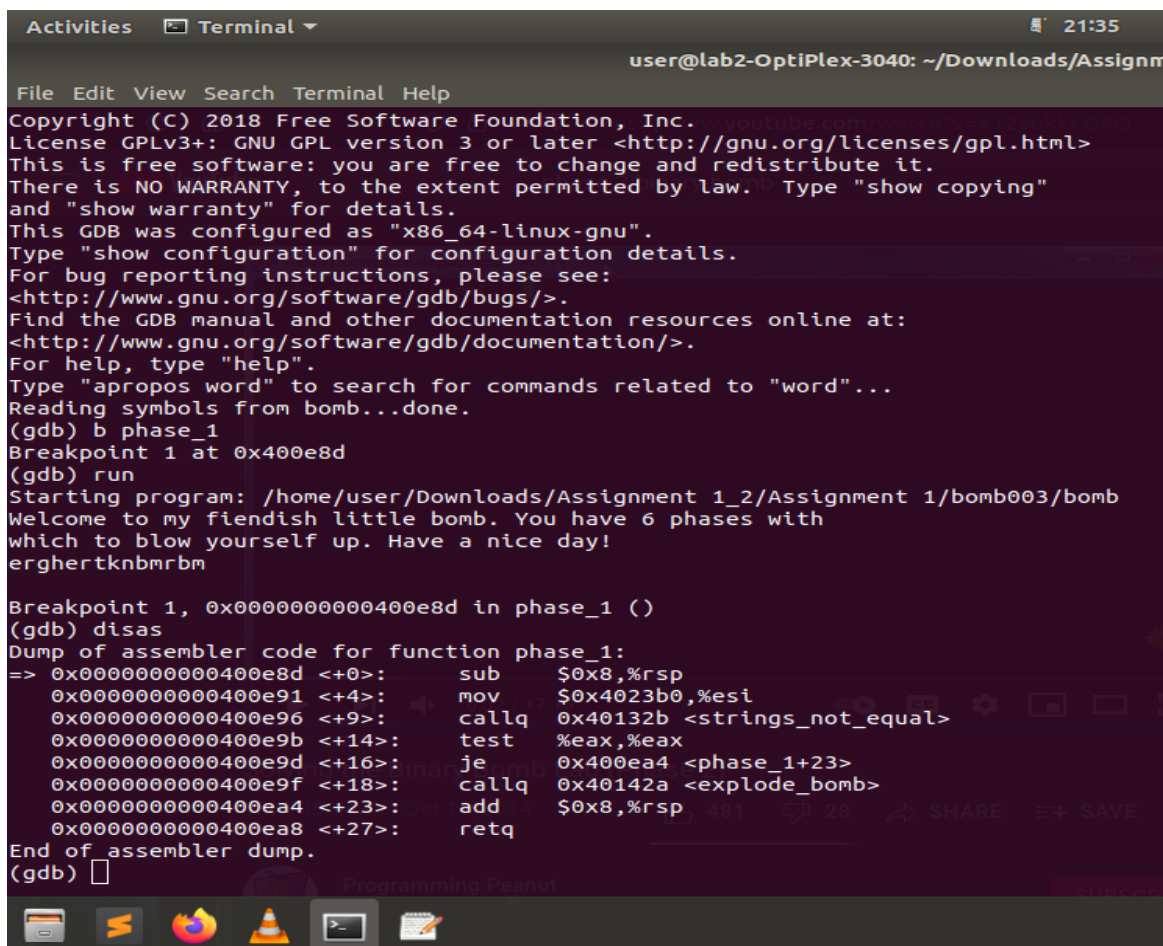


## Bomb\_3/Phase\_1

Phase 1 of Bomb 3 is all about reverse engineering process in which there is no such kind of difficulties with the assembly languages, neither to compare or any kinds of mental hoops to jump through and all the functions of phase are simple.

First of all we have to go to the **objdump** file, we can see that there is two values to get push into the stack where one is stored as a register **%eax** and the other values to be stored in a code at **\$0x80497c0** before calling the function **<strings\_not\_equal>**.



```
Activities Terminal 21:35
user@lab2-OptiPlex-3040: ~/Downloads/Assignm

File Edit View Search Terminal Help
Copyright (C) 2018 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) run
Starting program: /home/user/Downloads/Assignment 1_2/Assignment 1/bomb003/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
erghertknbmrbm

Breakpoint 1, 0x0000000000400e8d in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x0000000000400e8d <+0>:      sub    $0x8,%rsp
0x0000000000400e91 <+4>:      mov    $0x4023b0,%esi
0x0000000000400e96 <+9>:      callq 0x40132b <strings_not_equal>
0x0000000000400e9b <+14>:     test   %eax,%eax
0x0000000000400e9d <+16>:     je     0x400ea4 <phase_1+23>
0x0000000000400e9f <+18>:     callq 0x40142a <explode_bomb>
0x0000000000400ea4 <+23>:     add    $0x8,%rsp
0x0000000000400ea8 <+27>:     retq

End of assembler dump.
(gdb) □
```

Out here the user input would stored in **%eax** and would be compared with the string at **\$0x4023b0** and if it is equal or same as that of the string, the bomb will get defused and can go to the next phase.

## Bomb\_3/Phase\_1

In order to get the actual string first of all;

1. We have to go inside the **gdb** and set the breakpoint on **<phase\_1>** function which make sure that the bomb won't get blow up when running the program. And the command to set the break point is **b Phase\_1**. After that we have to run the program with run command.
2. After the run command, we have to enter the string which is not known but, we can enter any string as we had set the break point which will not let the bomb to blow up when entering wrong string.

```
which to blow yourself up. Have a nice day!
erghertknbrbm

Breakpoint 1, 0x00000000400e8d in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x00000000400e8d <+0>:  sub    $0x8,%rsp
0x00000000400e91 <+4>:  mov     $0x4023b0,%esi
0x00000000400e96 <+9>:  callq   0x40132b <strings_not_equal>
0x00000000400e9b <+14>:  test    %eax,%eax
0x00000000400e9d <+16>:  je      0x400ea4 <phase_1+23>
0x00000000400e9f <+18>:  callq   0x40142a <explode_bomb>
0x00000000400ea4 <+23>:  add     $0x8,%rsp
0x00000000400ea8 <+27>:  retq
End of assembler dump.
(gdb) p/x $eax
$1 = 0x6037a0
(gdb) x/25c 0x6037a0
0x6037a0 <input_strings>: 101 'e' 114 'r' 103 'g' 104 'h' 101 'e' 114 'r' 116 't' 107 'k'
0x6037a8 <input_strings+8>: 110 'n' 98 'b' 109 'm' 114 'r' 98 'b' 109 'm' 0 '\000' 0 '\000'
0x6037b0 <input_strings+16>: 0 '\000' 0 '\000' 0 '\000' 0 '\000' 0 '\000' 0 '\000' 0 '\000' 0 '\000'
0x6037b8 <input_strings+24>: 0 '\000'
(gdb) x/25c 0x40132b
0x40132b <strings_not_equal>: 65 'A' 84 'T' 85 'U' 83 'S' 72 'H' -119 '\211' -5 '\373' 72 'H'
0x401333 <strings_not_equal+8>: -119 '\211' -11 '\365' -24 '\350' -45 '\323' -1 '\377' -1 '\377' -1 '\377' 65 'A'
0x40133b <strings_not_equal+16>: -119 '\211' -60 '\304' 72 'H' -119 '\211' -17 '\357' -24 '\350' -56 '\310' -1 '\3
77'
0x401343 <strings_not_equal+24>: -1 '\377'
(gdb) x/25c 0x4023b0
0x4023b0: 66 'B' 111 'o' 114 'r' 100 'd' 101 'e' 114 'r' 32 ' ' 114 'r'
0x4023b8: 101 'e' 108 'l' 97 'a' 116 't' 105 't' 111 'o' 110 'n' 115 's'
0x4023c0: 32 ' ' 119 'w' 105 'i' 116 't' 104 'h' 32 ' ' 67 'C' 97 'a'
0x4023c8: 110 'n'
```

3. Now we can look at the values in the two memory addresses of interest, **\$eax** and **0x4023b0**, from there we need to use the **gdb** command **x/25c** where we are ask for 25 characters of memory address but, before that we have to find the address of **%eax** and for that we have to use **p/x \$eax** to find its address.
4. After finding the address of **%eax**, we have to run the command **x/25c 0x840132b** to get the inputted string.
5. The last command to run is the **x/25c 0x4023b0** which will give us the characters that follow the memory address.

## Bomb\_3/Phase\_1

- Now we got the hint, so can go to the **strings** and search for the given string and can defuse the phase one bomb.

```
Activities Terminal user@lab2-OptiPlex-3040: ~/Downloads/Assignment 1_2/Assign
File Edit View Search Terminal Help
Usage: %s [<input_file>]
That's number 2. Keep going!
Halfway there!
Good work! On to the next...
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?
So you got that one. Try this one.
Border relations with Canada have never been better.
Wow! You've defused the secret stage!
So you think you can stop the bomb with ctrl-c, do you?
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Congratulations! You've defused the bomb!
Well...
OK. :-)
Invalid phase%s
BOOM!!!
The bomb has blown up.
%d %d %d %d %d %d
Error: Premature EOF on stdin
GRADE_BOMB
Error: Input line too long
%d %d %s
DrEvil
greatwhite.ics.cs.cmu.edu
angelshark.ics.cs.cmu.edu
makoshark.ics.cs.cmu.edu
whaleshark.ics.cs.cmu.edu
Program timed out after %d seconds
Error: HTTP request failed with error %d: %s
GET /%s/submitr.pl/?userid=%s&userpwd=%s&lab=%s&result=%s&submit=submit HTTP/1.0
Error: Unable to connect to server %s
%%%02X
%s %d %[a-zA-z ]
chance@cs.cmu.edu
```

```
.debug_tline
.debug_str
.debug_loc
user@lab2-OptiPlex-3040:~/Downloads/Assignment 1_2/Assignment 1/bomb003$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
█
```

Now the bomb is defused and we can head to next phase.