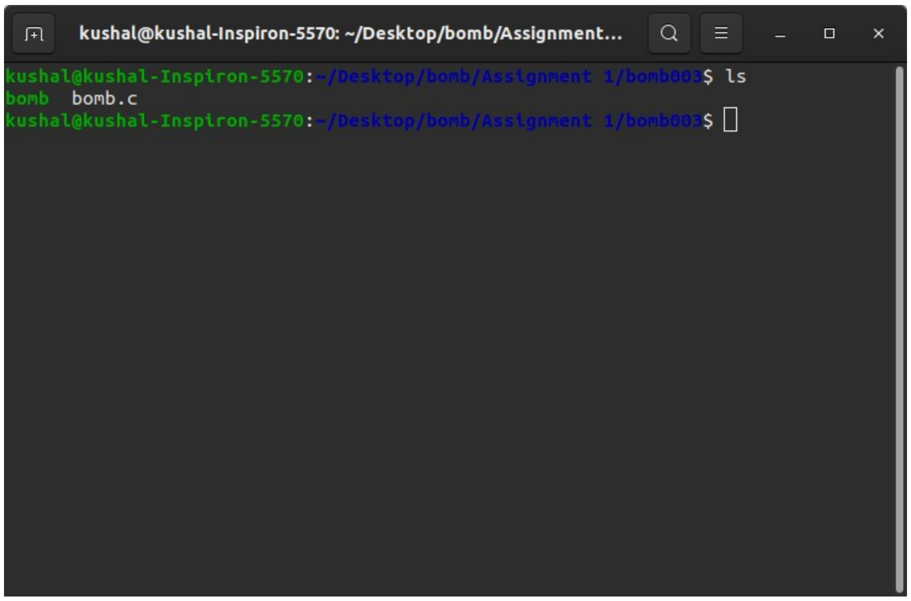
Binary Bomb(Group 3)
Phase 01
“Acquiring The String”

The first phase or the phase 1 is about giving a string to Dr. Evil. If the given string matches the string of Dr Evil than the phase 1 of the bomb will be diffused. Since we don’t know what is the string that is set by Dr Evil, Its our work to find the correct string so that the bomb will be diffused. The strings changes from bomb to bomb and the particular bomb which I am working on is bomb number 3. The string which I got from phase_1 bomb 003 is “**Border relations with Canada have never been better.**” After putting this string the phase_1 of bomb 003 will be diffused.

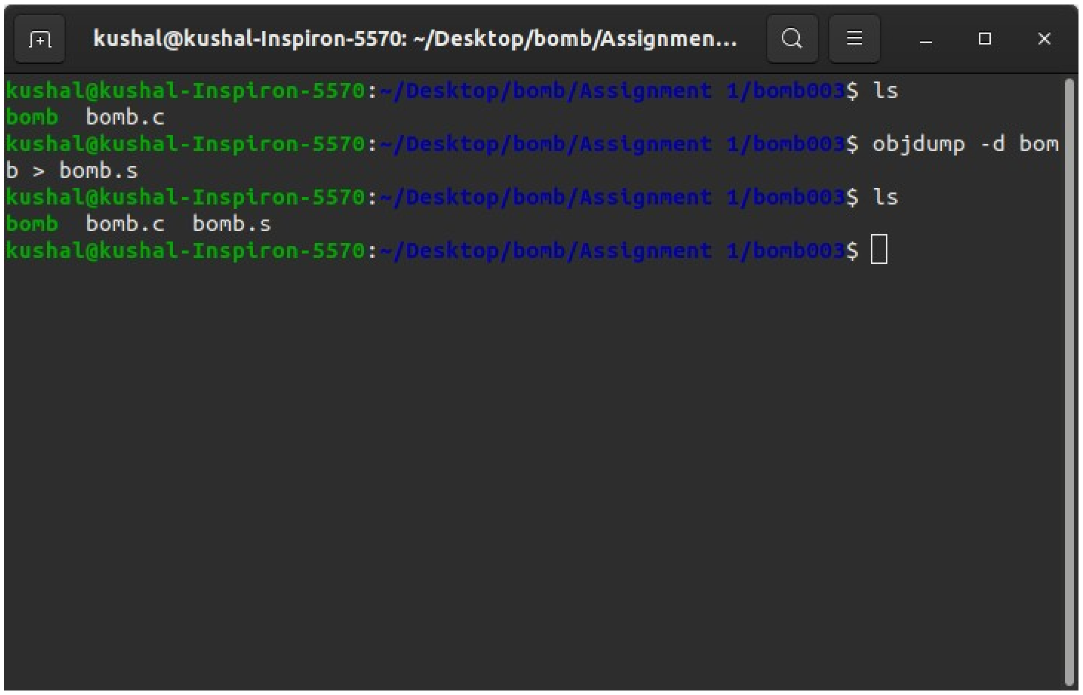
Here are the steps with which I obtained the string.

First thing first, I downloaded the assignment.zip folder from VLE and unzipped by extracting it. After that I went inside the bomb003 folder and opened the Terminal.



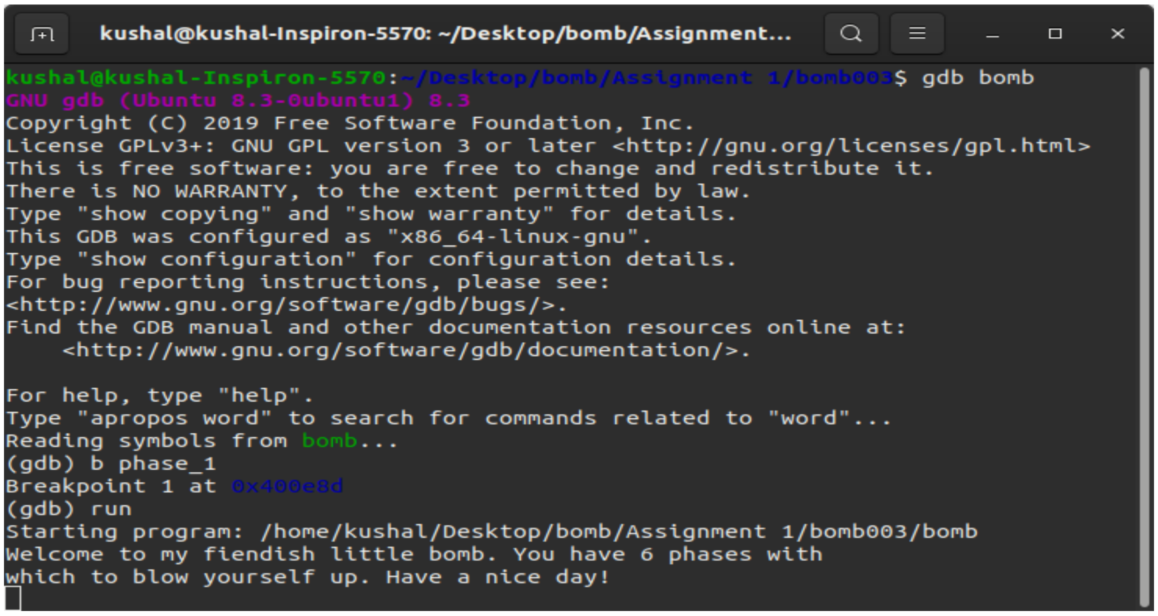
```
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ ls
bomb  bomb.c
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ 
```

After opening the terminal and wrote the “objdump -d bomb > bomb.s” command. This command will disassemble all of the code and major function of the objdump -d is it reads the assembler code and displays information about one or more object files.



```
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ ls
bomb  bomb.c
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ objdump -d bom
b > bomb.s
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ ls
bomb  bomb.c  bomb.s
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ 
```

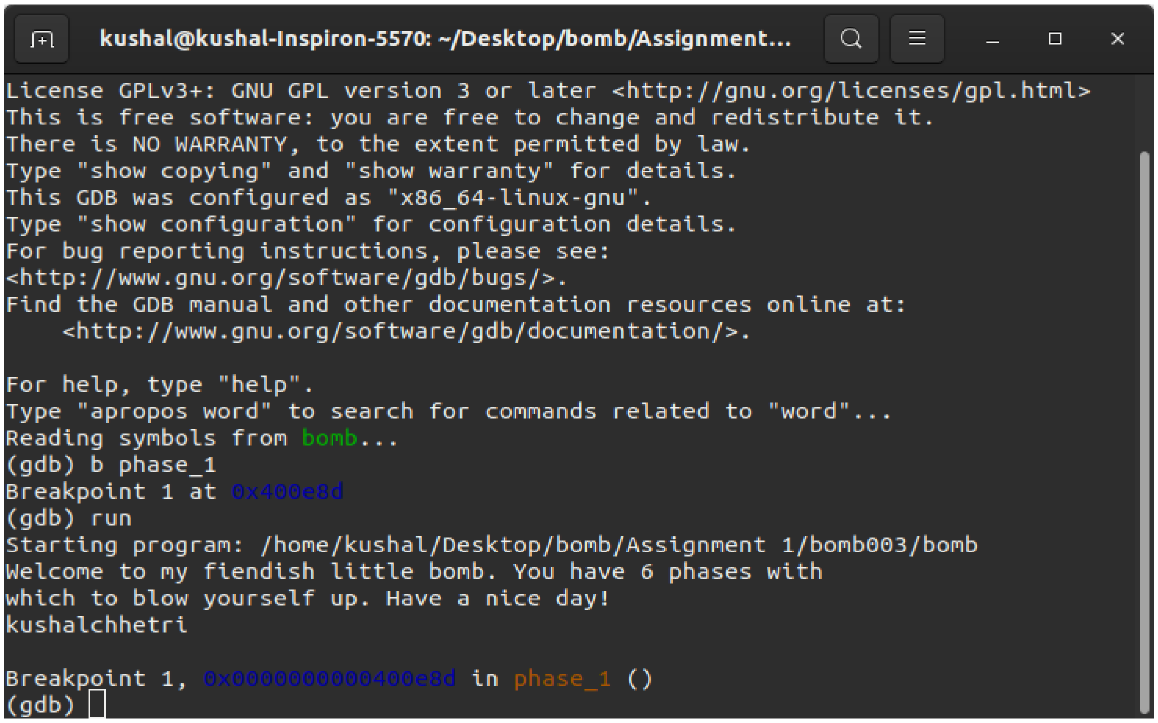
As we can see that a new file is added which has the extension “.s” which means it is the assembly file, which means all the source code inside the bomb.s will be in assembly form. The next command I used is the “gdb” command which is the debugger for C. The gdb command will help to run a program up to certain point then stop and print out the values of certain variables at that point. 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.



kushal@kushal-Inspiron-5570: ~/Desktop/bomb/Assignment...

```
kushal@kushal-Inspiron-5570:~/Desktop/bomb/Assignment 1/bomb003$ gdb bomb
GNU gdb (Ubuntu 8.3-0ubuntu1) 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-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...
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) run
Starting program: /home/kushal/Desktop/bomb/Assignment 1/bomb003/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

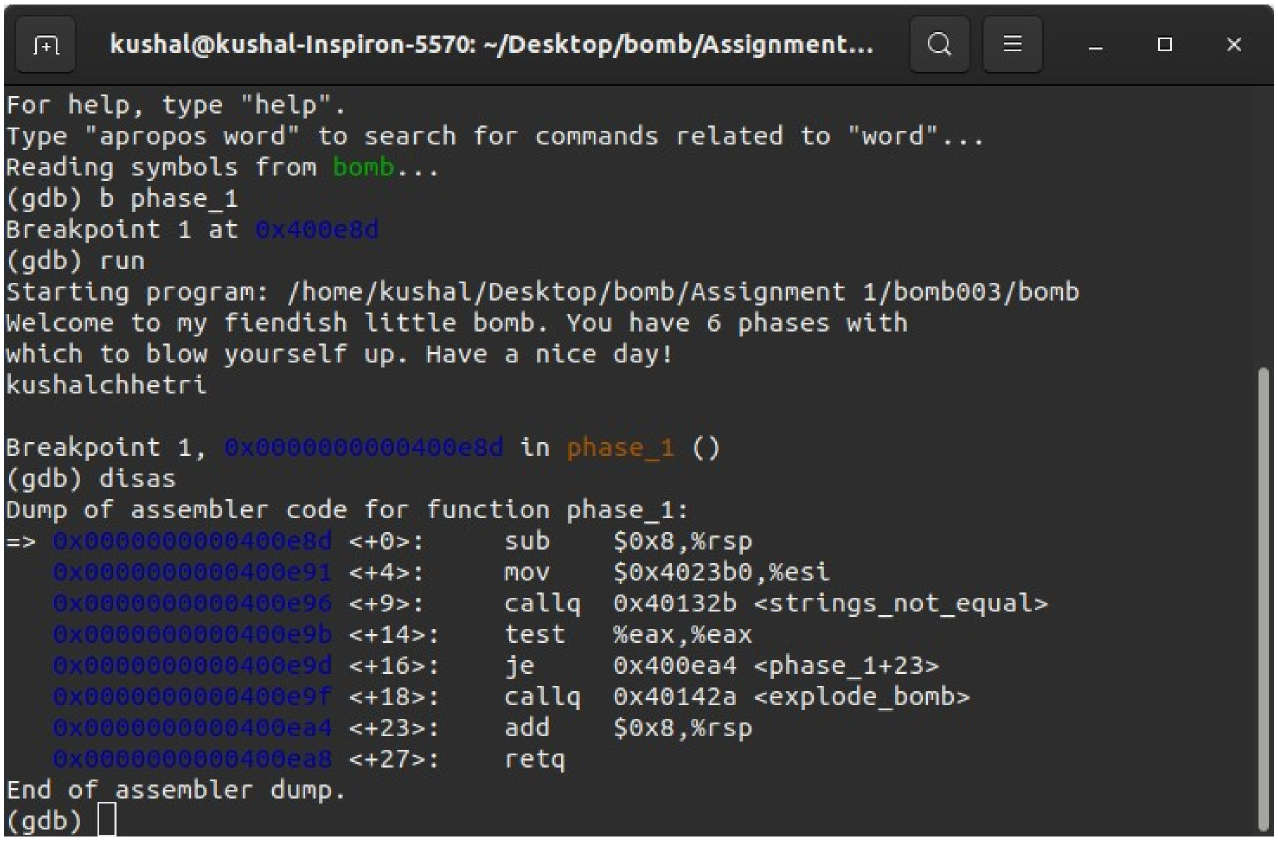
Now we can see that we have to input something. We can give a test string (which can be any string of your choice). This will not blast the bomb since we have already set the break-point. After giving the random string we can command the “disas” command to go inside the assembly code of the phase_1.



kushal@kushal-Inspiron-5570: ~/Desktop/bomb/Assignment...

```
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...
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) run
Starting program: /home/kushal/Desktop/bomb/Assignment 1/bomb003/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
kushalchhetri

Breakpoint 1, 0x0000000000400e8d in phase_1 ()
(gdb)
```

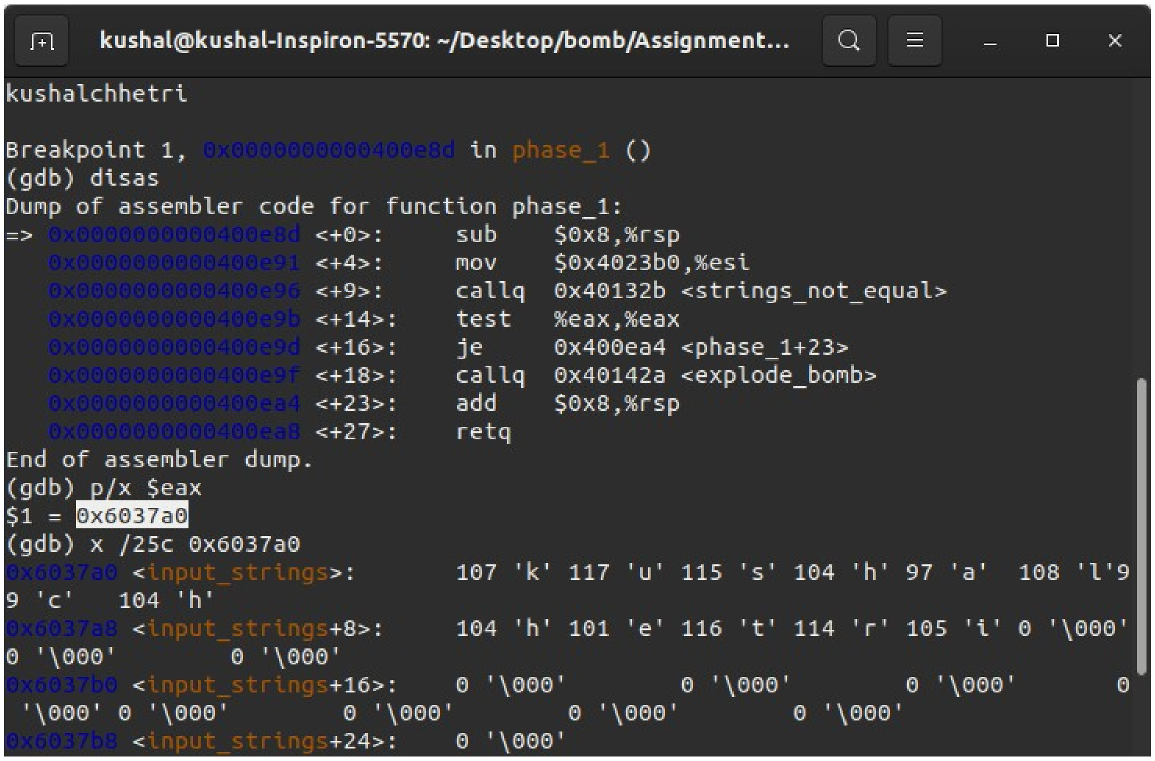
Now use the “disas” command to open the assembly file. After opening the dissembler we can see at line number that the input is string since



```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) run
Starting program: /home/kushal/Desktop/bomb/Assignment 1/bomb003/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
kushalchhetri

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)
```

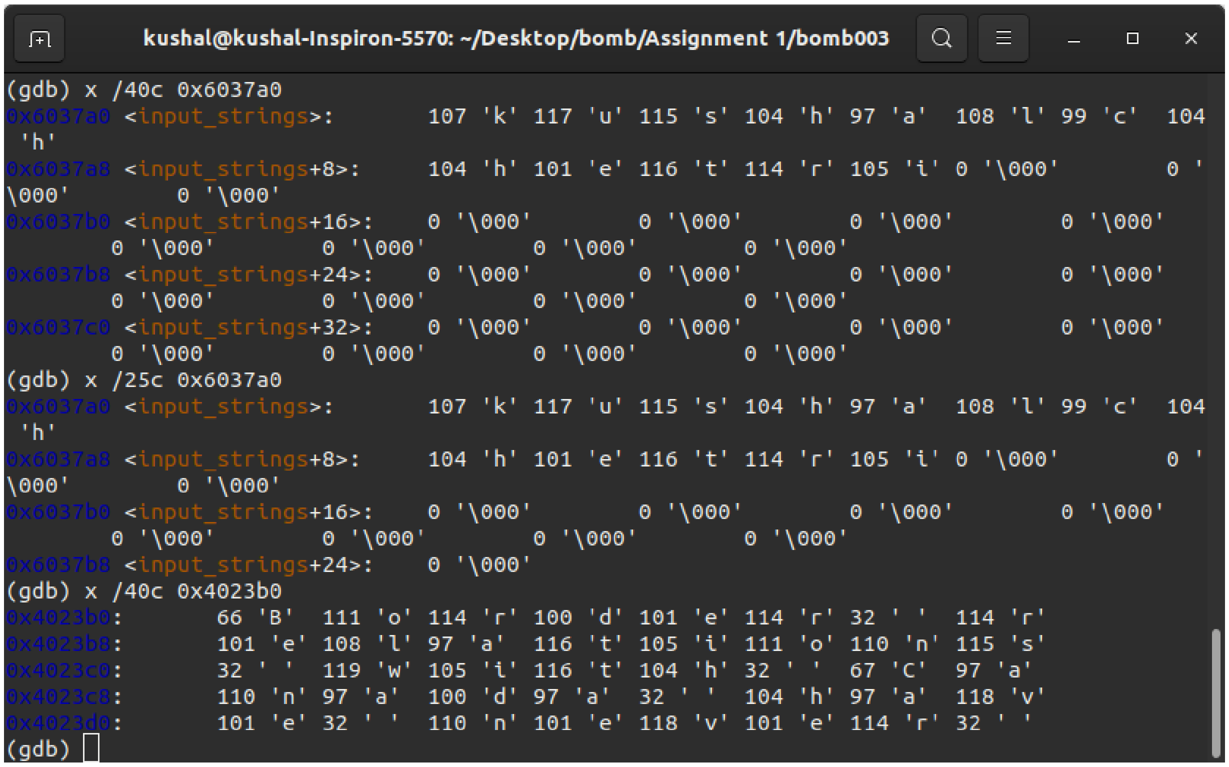
From the above code in line number <+9> we have a callq function which calls <strings_not_equal> that means we know the input is a string and not a integer. In line <+14> the assembler is testing eax register with eax register. The source eax contains the user input string (the string which we type) and the destination eax contains the real string that Dr. Evil has set. So this function compares the strings and if the strings are equal, it jumps into line <+16> which says je(jump if equal to) so if the both strings matches than it will jump to line <phase_1+23> . Else it will jump to callq and the bomb will be exploded.

After this we can check that the input we gave is stored in th $eax register. For that we have to command “**p/x $eax**” to get the address of eax register. The address was found as “0x6037a0” After that we have to command **“x/ 25c 0x6037a0”.** This will give the test string we gave ealier. So from this we know that the string we give is stored inside the eax register.



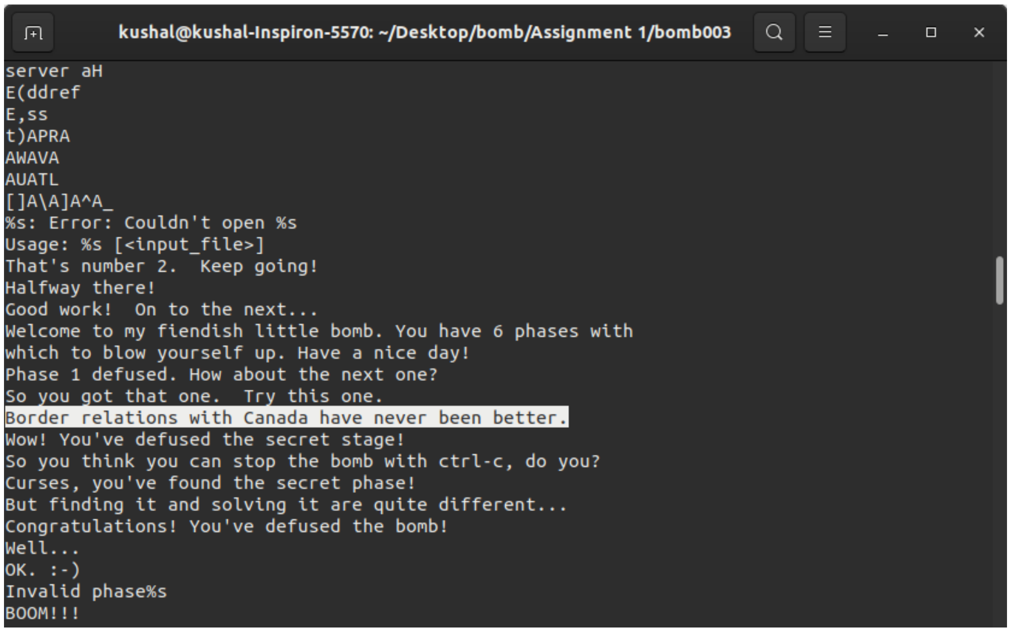
```
kushalchhetri

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) p/x $eax
$1 = 0x6037a0
(gdb) x /25c 0x6037a0
0x6037a0 <input_strings>:       107 'k' 117 'u' 115 's' 104 'h' 97 'a'  108 'l'9
9 'c'   104 'h'
0x6037a8 <input_strings+8>:     104 'h' 101 'e' 116 't' 114 'r' 105 'i' 0 '\000'
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'
```

To get the final string which Dr. Evil has set we have to give the fixed address of the eax register which we will get from the disas of phase_1 at line number <+4> since the value of esi is been moved to 0x4023b0. Copy this address and put it inside the “**x / 40c 0x4023b0**” We will getting the string which was set by Dr.Evil. Here we did 40c to get the 40 characters of the string.



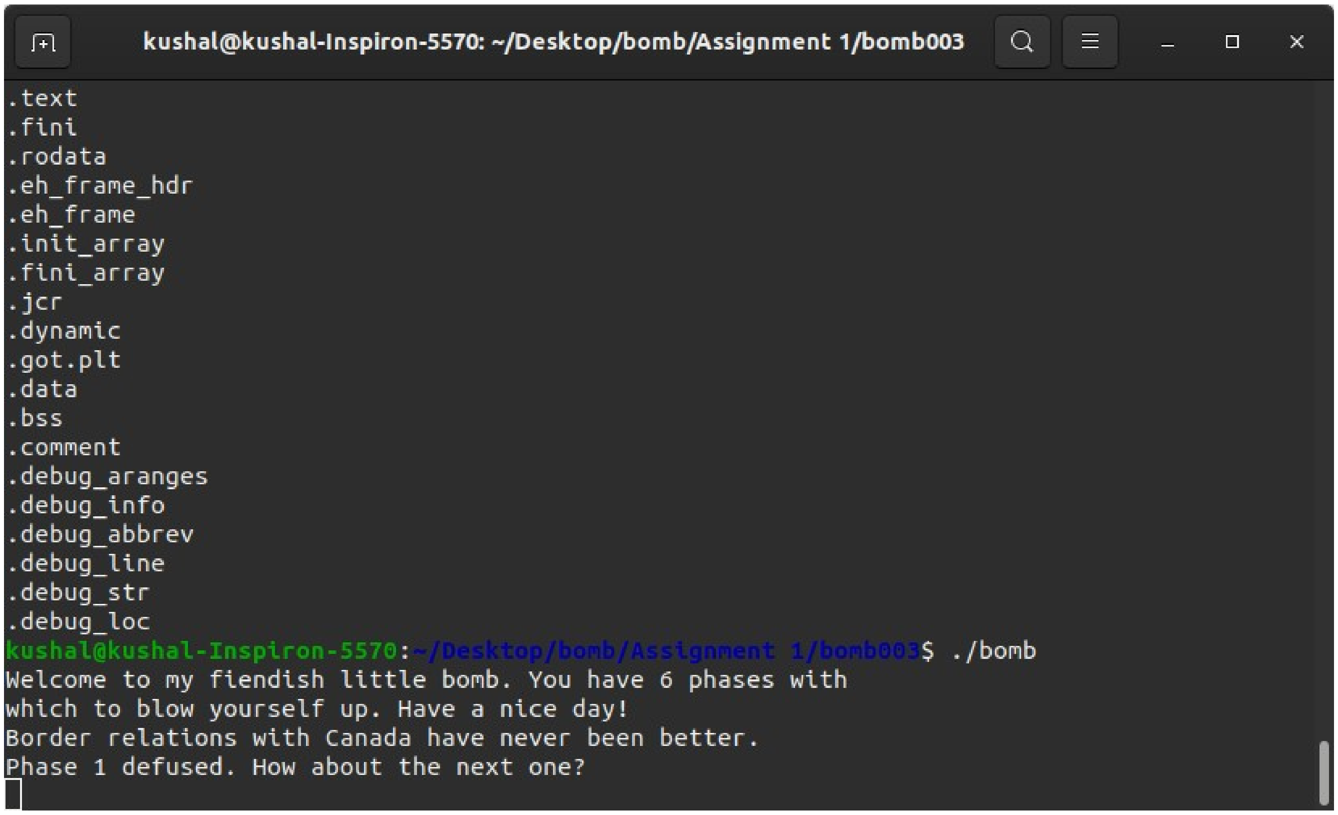
```
kushal@kushal-Inspiron-5570: ~/Desktop/bomb/Assignment 1/bomb003
(gdb) x /40c 0x6037a0
0x6037a0 <input_strings>:       107 'k' 117 'u' 115 's' 104 'h' 97 'a'  108 'l' 99 'c'  104
 'h'
0x6037a8 <input_strings+8>:     104 'h' 101 'e' 116 't' 114 'r' 105 'i' 0 '\000'        0 '
\000'       0 '\000'
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'        0 '\000'        0 '\000'        0 '\000'
        0 '\000'        0 '\000'        0 '\000'        0 '\000'
0x6037c0 <input_strings+32>:    0 '\000'        0 '\000'        0 '\000'        0 '\000'
        0 '\000'        0 '\000'        0 '\000'        0 '\000'
(gdb) x /25c 0x6037a0
0x6037a0 <input_strings>:       107 'k' 117 'u' 115 's' 104 'h' 97 'a'  108 'l' 99 'c'  104
 'h'
0x6037a8 <input_strings+8>:     104 'h' 101 'e' 116 't' 114 'r' 105 'i' 0 '\000'        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 /40c 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 'i' 111 'o' 110 'n' 115 's'
0x4023c0:       32 ' '  119 'w' 105 'i' 116 't' 104 'h' 32 ' '  67 'C'  97 'a'
0x4023c8:       110 'n' 97 'a'  100 'd' 97 'a'  32 ' '  104 'h' 97 'a'  118 'v'
0x4023d0:       101 'e' 32 ' '  110 'n' 101 'e' 118 'v' 101 'e' 114 'r' 32 ' '
(gdb) 
```

In-order to check if this string is correct or not we have do “**strings bomb**” to get all the strings of the bomb .



```
kushal@kushal-Inspiron-5570: ~/Desktop/bomb/Assignment 1/bomb003
server aH
E(ddref
E,ss
t)APRA
AWAVA
AUATL
[]A\A]A^A_
%s: Error: Couldn't open %s
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!!!
```

Finally we got the string now run the bomb file by commanding “**./bomb**” and write the string “**Border relations with Canada have never been better.**” And the bomb at phase_1 will be diffused.


kushal@kushal-Inspiron-5570: ~/Desktop/bomb/Assignment 1/bomb003

```
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.jcr
.dynamic
.got.plt
.data
.bss
.comment
.debug_aranges
.debug_info
.debug_abbrev
.debug_line
.debug_str
.debug_loc
kushal@kushal-Inspiron-5570:~/Desktop/bomb/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?
```

Phase 1 Over!!!

Haha Dr. Evil, I acquired your bomb at phase_1 !!!
Dr just wait and see how I will diffuse phase_2 and Phase_3