

Task: Assignment consists of three different binary bombs and each bomb consists of three phase. We are asked to defuse binary bomb stepping each phase, phase_1, phase_2, and phase_3 accordingly. To execute this assembly files, linux shell was used through out the assignment.

Phase 01

After saving the files to linux system, first of all, to access the folder, “chmod +x bomb” command was executed and permission to access file was granted by the system. To view the executable assembly form for the bomb file, “objdump -d bomb > bomb.s” was executed and “strings bomb” command was executed to display all the strings stored in the bomb file.

```
Terminal
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ chmod +x bomb
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ ls
bomb bomb.c bomblab.pdf
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ objdump -d bomb > bomb
.s
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ ls
bomb bomb.c bomblab.pdf bomb.s
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ strings bomb
/lib64/ld-linux-x86-64.so.2
libc.so.6
socket
fflush
strcpy
__printf_chk
exit
fopen
__isoc99_sscanf
connect
signal
puts
```

```
Terminal
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
```

On continuation, debugger gdb was used to execute the assembly programs and break point was set to ensure that the bomb will not blow up. To run the program, “run” or “r” command is used and program asks user to give input. For a trail, “Hello, how can I crack you” was given as a test string and this test string is stored in \$eax register. To check our string in \$eax, “p/x \$eax” was executed and it displays address of \$eax register. Now it becomes easy to check the string from the given address. “x /25c 0x6037a0” was executed to display the string.

```
ghishing@Boom:~/Desktop/GCIT/semesterV/Pema Wangmo/Assignment 1_2/Assignment 1/bomb003$ gdb bomb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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) r
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/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!
Hello, how can I crack you

Breakpoint 1, 0x000000000400e8d in phase_1 ()
(gdb)
```

```
(gdb) p/x $eax
$1 = 0x6037a0
(gdb) x /25c 0x6037a0
0x6037a0 <input_strings>:      72 'H' 101 'e' 108 'l' 108 'l' 111 'o' 44 ',' 32 ' ' 104 'h'
0x6037a8 <input_strings+8>:    111 'o' 119 'w' 32 ' ' 99 'c' 97 'a' 110 'n' 32 ' ' 73 'I'
0x6037b0 <input_strings+16>:   32 ' ' 99 'c' 114 'r' 97 'a' 99 'c' 107 'k' 32 ' ' 121 'y'
0x6037b8 <input_strings+24>:  111 'o'
(gdb)
```

Understanding the programs

```
Terminal
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x000000000400e8d <+0>:      sub    $0x8,%rsp
    0x000000000400e91 <+4>:      mov    $0x4023b0,%esi
    0x000000000400e96 <+9>:      callq 0x40132b <strings_not_equal>
    0x000000000400e9b <+14>:     test   %eax,%eax
    0x000000000400e9d <+16>:     je     0x400ea4 <phase_1+23>
    0x000000000400e9f <+18>:     callq 0x40142a <explode_bomb>
--Type <RET> for more, q to quit, c to continue without paging--
```

We have used “disas” to figure out the function in the program for the phase one. As soon as command was executed it displays the program code in assembly language. From this code we can understand function is being called `<strings_not_equal>` , some string value is stored in `$0x4023b0` address(which the address of `%eax`). From this we can understand that answer for the given program is in string. To view the value stored in `$0x4023b0` we use “x/25c 0x4023b0” to display the value till 25 characters. It displays the strings which is initially stored in the program in `$eax` register and the string was “Border relations with Canada have never been better.”

```
Terminal
(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 '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'
(gdb)
```

From the code we can understand that input value given by user is stored in `%eax` register and it compares to the initial value stored in address `$0x4023b0`. If it matches, program will take user to another phase but if it fails then, it displays the message “Bomb blown up”. To figure out the program logic, program was run and string value “Border relations with Canada have never been better.” was provided.

```
Terminal
(gdb) r
Starting program: /home/ghishing/Desktop/GCIT/semesterV/Pema Wangmo/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?
```

The user input and the string stored in the program register (%eax) matched and the bomb defused successfully.