#### gcc version

Target: x86\_64-redhat-linux

Thread model: posix

gcc version 4.8.3 20140911 (Red Hat 4.8.3-9) (GCC)

#### gdb version

GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-64.el7

This GDB was configured as "x86\_64-redhat-linux-gnu".

#### objdump version

GNU objdump version 2.23.52.0.1-30.el7 20130226

#### Ready for disassembly

**Get the assembly code file by the command in terminal**

objdump -d bomb > bomb64.s

**Get strings of bomb**

strings bomb > bomb.strings

**Open binary file by hexedit**

hexedit bomb

#### 2.Static and Dynamic Analysis bomb

use command "objdump -p bomb" to get Program Header

the bomb start at address 0x0000000000400000 (tips: look more by stack protector)

in hexedit, we can see tha bomb start at 0x00000000, so the real address must be add 0x00400000

##### phase\_1:

**Static analysis:**

open bomb64.s in vi

go to the function phase\_1 at address **0x400ee0**

in phase\_1, call function of strings\_not\_equal with two argments,

one is the input\_string by argment of phase\_1, another is a point to address **0x402400**

in hexedit, go to address (0x402400-0x00000000) **0x0002400**,

we can see the address point to a string with 0 terminate.

42 6F 72 64 65 72 20 72 65 6C 61 74 69 6F 6E 73

20 77 69 74 68 20 43 61 6E 61 64 61 20 68 61 76

65 20 6E 65 76 65 72 20 62 65 65 6E 20 62 65 74

74 65 72 2E 00 00 00 00

the string is "Border relations with Canada have never been better."

to verify the string, just run the bomb and type the string command line

the phase\_1 is passed, get it!

**Dynamic analysis:**

use gdb to debug the bomb, the command is below:

gdb bomb

(gdb) break(b) phase\_1(\*0x400ee0)

(gdb) run(r)

at here, we type a string whatever, like "test", then break at phase\_1

(gdb) x/s $edi

(gdb) x/s $esi

show the <input\_strings>: "test"

(gdb) stepi 2

after instruct mov $0x402400,$esi, we set esi with the point to address 0x402400

(gdb) x/s $esi

we can see the address 0x402400 point to a string of"Border relations with Canada have never been better.",and is not equal to the string "test", then the function strings\_not\_equal return 1

instruct "test %eax,%eax" set condition to 1, so "je 0x400ef7" not execute,then call function explode\_bomb. the explode\_bomb just puts(0x4025a3), puts(0x4025ac), and eixt(8);

0x4025a3 and 0x4025ac can be look at hexedit with address 0x0025a3, 0x0025ac, or in gdb use commnad x/s 0x4025a3(0x4025ac).

now, we use the correct string(point to 0x402400) replace of "test", phase\_1 passed

##### phase\_2:

go to the function phase\_2 at address **0x400efc**

in phase\_2, call function of read\_six\_numbers

in function read\_six\_numbers, point to address **0x4025c3** is a string "%d %d %d %d %d %d", call sscanf to get six numbers by the input\_string, and store in an array. the inpust\_string must be six numbers separate by 0x20(space in ascii), and the sixth number must greater than 0x5

back to phase\_2, at address **400f0a**, **400f0e**, with instruct "cmp $0x1,(%rsp) je 400f30", knonw that the first number must be 0x1

at address **400f30**, set %rbx to the next number, set %rbp to the end of array

from address **400f17** to **400f2c**, to test the current number is double of the last number in loop second to sixth

if all the six numbers meet the conditions, phase\_2 passed

##### phase\_3:

in phase\_3, by the address **0x4025cf**(string: "%d %d"), we know that sscanf to get two numbers by input\_string

by the instruct at address **400f60**, **400f63**, **400f6a**, **400f6f**, the second number must be 0 <= second\_num <= 7

at address 400f75, jmp \*0x402470(,%rax,8) indirect jump to an address point to 0x402470 by Scaled indexed

the address of 0x402470 is:

7C 0F 40 00 00 00 00 00 B9 0F 40 00 00 00 00 00

83 0F 40 00 00 00 00 00 8A 0F 40 00 00 00 00 00

91 0F 40 00 00 00 00 00 98 0F 40 00 00 00 00 00

9F 0F 40 00 00 00 00 00 A6 0F 40 00 00 00 00 00

**the index 0 - 7 point to 0x402470 + (0 - 7) \* 8**

index 0: 00400f7c, num = 0xcf = 207, solution is (0, 207)

index 1: 00400fb9, num = 0x137 = 311, solution is (1, 311)

index 2: 00400f83, num = 0x2c3 = 707, solution is (2, 707)

index 3: 00400f8a, num = 0x100 = 256, solution is (3, 256)

index 4: 00400f91, num = 0x185 = 389, solution is (4, 389)

index 5: 00400f98, num = 0xce = 206, solution is (5, 206)

index 6: 00400f9f, num = 0x2aa = 682, solution is (6, 682)

index 7: 00400fa6, num = 0x147 = 327, solution is (7, 327)

get it!