Report for Assignment 3 20131218 박기훈

1. Phase_1

In phase_1, I could defuse bomb easily by using gdb. In line <+9>, strings_not_equal's role is judging the input string is same as "0x4022e0". So I examined 0x4022e0 by using x/s, then I could get the solution, "I was trying to give Tina Fey more material."

2. Phase_2

In phase_2, in line <+9>, read_six_numbers's role is just reading six numbers of input. If input size is smaller than 6, explode_bomb is executed. In line <+14>, compare first input with 1, then in line <+52>, %rbx gets the next input integer. In line <+30>, %eax = %eax * 2, then compare %rbx with %eax. That is, first input integer should be 1, and next integer should be double value of previous integer. So solution is that, "1 2 4 8 16 32".

3. Phase_3

Phase_3 can be solved very easy by using gdb. At first, before calling scanf function, 0x402336 is stored at %esi, and %eax gets 0. Then we check the 0x402336 by using x/s, which is "%d %c %d". That is, input form should be "%d %c %d", which is int, char, int. In line <+34>, check the input form is correct.

Then, input the sample solution with format "%d %c %d", like "1 a 2". I just debug line by line with this sample solution, in line <+119>, explode_bomb is executed. In line <+105>, input integer compared with 0x359, so input integer should be 0x359, which is 857.

Input again sample solution with "1 a 857", then next bomb is executed in line <+323>. Our char input is stored in 0x7(%rsp), so char input is compared with char of 0x64, because in line <+100> 0x64 is stored in %eax, and jump to <+317> directly. As seeing in ASCII code, 0x64 is "d". So solution should be "1 d 857".

4. Phase_4

At first, check the input format by using x/s 0x4024cf before calling scanf function. Then we can know that input format should be "%d %d", two integers. Second integer is stored at 0xc(%rsp) and first integer is stored at 0x8(%rsp). In line <+29>, check the input format is correct, and in line <+38>~<+44> check whether the second integer is 2 or 3 or 4, if not bomb. I just choose first second as 3. In line <+60>, second integer and 6 (which are stored at %esi, %edi) are going to func4, and return value of func4 is compared with first integer in line <+65>. So second value should be same as return value of func4.

In func4, there is two recursion, at line <+22> and line <+36>. At line <+6>, if %edi is equal or less than 0, fun4 returns 0. At line <+14>, if %edi is equal to 1, come out from func4 with return value of %esi. At line <+22> recursive func4 with %edi-1, and at line <+36> recursive func4 with %edi-2. So integrated result of all recursion, the return value of fun4 is second input integer*20. Therefore, first input integer should be second input integer*20, solution can be "40 2", "60 3", "80 4".

5. Phase_5

At line <+4>, string_length function returns the length of input string. So input string size should be 6. At line <+29>, hex value of first char of input string is stored at %ecx, and at line <+33> least significant byte value of former hex value is stored in %ecx. Then we can check each stored value from 0x402380 + 4bits, and sum of 6 stored value should be 0x2d at line <+53>. So we can set 6 inputs like below,

0x402384 : 0x0a -> 0x61 -> a
0x40238c : 0x01 -> 0x63 -> c
0x402390 : 0x0c -> 0x64 -> d
0x402394 : 0x10 -> 0x65 -> e
0x40239c : 0x03 -> 0x67 -> g *2

Solution is acdegg, which regardless of the order. (edgacg, gacdeg,... also can be solution)

6. Phase_6

At line <+15>, we can know that input format is 6 integers. At line <+31>~<+100>, which is Loop_1, 'i' increases from 1 to 6, and at line <+34>~<+46> check whether 1<='i'th element<=6, that is all input integers should be 1~6. At line <+78>~<+83>, which is Loop_2, 'j' increases from I to 6, and check whether 'j'th element is same as 'i'th element. if same, bomb. At line <+66>, in last loop, jump to <+134>

At line <+102>~<+153>, which is Loop_3, in this loop, linked address of each input integer has certain value. At line <+102>~<+111>, as the magnitude of input integer, assign the certain address and value. Then, at last loop, jump to <+155> at line <+132>.

At line <+206>~<+214>, compare the certain value of 'i'th input integer with 'i-1'th input integer, and 'i'th certain value should bigger than 'i-1'th. And each certain value of input integer is that, 6(0x3be), 5(0xf6), 4(0x31b), 3(0x242), 2(0x27b), 1(0xb8), so solution is that "6 4 2 3 5 1", such as 0x3be > 0x31b > 0x27b > 0x242 > 0xf6 > 0xb8.

7. Secret_Phase

Secret_Phase is opened only if I defuse 6 phases. As observing phase_defused, at line <+97>, it calls secret_phase. At line <+28>, we can know that input format is "%d %d %s" by using x/s 0x402519, and at line <+33>, "60 3" is stored in 0x603890, which is solution of phase_4. So if I change the solution format of phase_4 properly, I can enter into secret_phase. At line <+53>~<+63>, input char is compared with 0x402522(=DrEvil) with strings_not_equal function. So it I input the phase_4 solution as "60 3 DrEvil", I can enter into secret_phase.

In secret_phase function, at line <+24>, %rax is my input value, and strtol function convert my input value into base 10 format. At line <+30>, input value <= 1000 in base 10. At line <+49>, my input value in base 10 is stored in %esi, 0x603110(=0x24) is stored in %edi, then enter into fun7. At line <+54>, return value of fun7 should be 3 to avoid bomb.

In fun7, there are two recursion at line <+19>,<+41>. First recursion of line <+19> is executed if input is smaller than %rdi, otherwise second recursion of line <+41> is executed. So it makes some tree, first recursion goes to right down, and second recursion goes to left down. And at line <+24>,<+46>, come back from right down recursion, %eax = 2*%eax, and come back from left down recursion, %eax = %rax+%rax+1. So to get the return value of 3, we should come back from left down recursion twice. Twice left down tree value is 0x6031d0, which is 0x6b and 107 in base 10. So solution of secret_phase is 107.