2020 시스템 프로그래밍

- Bomb Lab -

제출일자	2020.10.25
분 반	01
이 름	박재우
학 번	201902694

Phase 1 [결과 화면 캡처]

```
which to blow yourself up. Have a nice day!
When I get angry, Mr. Bigglesworth gets upset.

Breakpoint 1, 0x00005555555552c4 in phase_1 ()
(gdb) c
Continuing.
```

Phase 1 [진행 과정 설명]

```
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x00005555555552c4 <+0>:     sub    $0x8,%rsp
   0x00005555555552c8 <+4>:     lea    0x18b1(%rip),%rsi        # 0x555555556b80
   0x00005555555552cf <+11>:    callq  0x555555555852 <strings_not_equal>
   0x00005555555552d4 <+16>:    test   %eax,%eax
   0x00005555555552d6 <+18>:    jne    0x5555555552dd <phase_1+25>
   0x00005555555552d8 <+20>:    add    $0x8,%rsp
   0x00005555555552dc <+24>:    retq
   0x00005555555552dd <+25>:    callq  0x555555555b56 <explode_bomb>
   0x00005555555552e2 <+30>:    jmp    0x5555555552d8 <phase_1+20>
End of assembler dump.
(gdb) x/s  0x555555556b80
0x555555556b80: "When I get angry, Mr. Bigglesworth gets upset."
```

Disas로 phase_1을 연다음

의 주소를 열어보니 When I get angry, Mr. Bigglesworth gets upset. 가 나왔다.

이주소는 x/s로 rsi를 확인했을때도 알아 낼 수 있다.

밑에 strings_not_equal로 비교를 하고 있으므로

얻은 값을 입력하자 폭탄이 해체되었다.

Phase 1 [정답]

When I get angry, Mr. Bigglesworth gets upset.

Phase 2 [결과 화면 캡처]

```
Phase 1 defused. How about the next one?
0 1 1 2 3 5 8

Breakpoint 2, 0x00005555555552e4 in phase_2 ()
(gdb) c
Continuing.
```

Phase 2 [진행 과정 설명]

```
=> 0x00005555555552e4 <+0>:     push   %rbp
   0x00005555555552e5 <+1>:     push   %rbx
   0x00005555555552e6 <+2>:     sub    $0x28,%rsp
   0x00005555555552ea <+6>:     mov    %fs:0x28,%rax
   0x00005555555552f3 <+15>:    mov    %rax,0x18(%rsp)
   0x00005555555552f8 <+20>:    xor    %eax,%eax
   0x00005555555552fa <+22>:    mov    %rsp,%rsi
   0x00005555555552fd <+25>:    callq  0x555555555b92 <read_six_numbers>
   0x0000555555555302 <+30>:    cmpl   $0x0,(%rsp)
   0x0000555555555306 <+34>:    jne    0x55555555530f <phase_2+43>
   0x0000555555555308 <+36>:    cmpl   $0x1,0x4(%rsp)
   0x000055555555530d <+41>:    je     0x555555555314 <phase_2+48>
   0x000055555555530f <+43>:    callq  0x555555555b56 <explode_bomb>
   0x0000555555555314 <+48>:    mov    %rsp,%rbx
   0x0000555555555317 <+51>:    lea    0x10(%rbx),%rbp
   0x000055555555531b <+55>:    jmp    0x555555555326 <phase_2+66>
   0x000055555555531d <+57>:    add    $0x4,%rbx
   0x0000555555555321 <+61>:    cmp    %rbp,%rbx
   0x0000555555555324 <+64>:    je     0x555555555337 <phase_2+83>
   0x0000555555555326 <+66>:    mov    0x4(%rbx),%eax
   0x0000555555555329 <+69>:    add    (%rbx),%eax
   0x000055555555532b <+71>:    cmp    %eax,0x8(%rbx)
   0x000055555555532e <+74>:    je     0x55555555531d <phase_2+57>
---Type <return> to continue, or q <return> to quit---
   0x0000555555555330 <+76>:    callq  0x555555555b56 <explode_bomb>
   0x0000555555555335 <+81>:    jmp    0x55555555531d <phase_2+57>
   0x0000555555555337 <+83>:    mov    0x18(%rsp),%rax
   0x000055555555533c <+88>:    xor    %fs:0x28,%rax
   0x0000555555555345 <+97>:    jne    0x55555555534e <phase_2+106>
   0x0000555555555347 <+99>:    add    $0x28,%rsp
   0x000055555555534b <+103>:   pop    %rbx
   0x000055555555534c <+104>:   pop    %rbp
   0x000055555555534d <+105>:   retq
   0x000055555555534e <+106>:   callq  0x555555554f00 <__stack_chk_fail@plt>
```

Rsp와 0의 값을 비교

Rsp+0x4와 1 비교 즉 입력의 첫번째 숫자와 2번째 숫자는 0, 1이어야 한다.

그후 rsp의 값은 rbx에 대입되고 rbx+0x10에 있는 주소값을 rbp에 저장

rbx = rbx+0x4(주소 4byte 이동)

eax는 rbx+0x4의 값을, eax = eax+rbx즉 3번째 값은 이전값과 그 이전값을 더한다.

즉 피보나치이며 6개의 답을 입력받으므로

0 1 1 2 3 5 8 이다.

Phase 2 [정답]

0 1 1 2 3 5 8

Phase 3 [결과 화면 캡처]

```
That's number 2.  Keep going!
0 a 383

Breakpoint 3, 0x0000555555555353 in phase_3 ()
(gdb) c
Continuing.
```

Phase 3 [진행 과정 설명]

```
Dump of assembler code for function phase_3:
   0x0000555555555353 <+0>:     sub    $0x28,%rsp
   0x0000555555555357 <+4>:     mov    %fs:0x28,%rax
   0x0000555555555360 <+13>:    mov    %rax,0x18(%rsp)
   0x000055555555536S <+18>:    xor    %eax,%eax
   0x0000555555555367 <+20>:    lea    0xf(%rsp),%rcx
   0x000055555555536c <+25>:    lea    0x10(%rsp),%rdx
   0x0000555555555371 <+30>:    lea    0x14(%rsp),%r8
   0x0000555555555376 <+35>:    lea    0x1859(%rip),%rsi        # 0x555555556bd6
   0x000055555555537d <+42>:    callq  0x555555554fa0 <__isoc99_sscanf@plt>
   0x0000555555555382 <+47>:    cmp    $0x2,%eax
   0x0000555555555385 <+50>:    jle    0x5555555553a6 <phase_3+83>
   0x0000555555555387 <+52>:    cmpl   $0x7,0x10(%rsp)
   0x000055555555538c <+57>:    ja     0x555555555497 <phase_3+324>
   0x0000555555555392 <+63>:    mov    0x10(%rsp),%eax
   0x0000555555555396 <+67>:    lea    0x1853(%rip),%rdx        # 0x555555556bf0
   0x000055555555539d <+74>:    movslq (%rdx,%rax,4),%rax
   0x00005555555553a1 <+78>:    add    %rdx,%rax
   0x00005555555553a4 <+81>:    jmpq   *%rax
   0x00005555555553a6 <+83>:    callq  0x5555555555b56 <explode_bomb>
   0x00005555555553ab <+88>:    jmp    0x555555555387 <phase_3+52>
   0x00005555555553ad <+90>:    mov    $0x61,%eax
   0x00005555555553b2 <+95>:    cmpl   $0x17f,0x14(%rsp)
   0x00005555555553ba <+103>:   je     0x5555555554a1 <phase_3+334>
   0x00005555555553c0 <+109>:   callq  0x5555555555b56 <explode_bomb>
   0x00005555555553c5 <+114>:   mov    $0x61,%eax
   0x00005555555553ca <+119>:   jmpq   0x5555555554a1 <phase_3+334>
   0x00005555555553cf <+124>:   mov    $0x65,%eax
   0x00005555555553d4 <+129>:   cmpl   $0x2df,0x14(%rsp)
   0x00005555555553dc <+137>:   je     0x5555555554a1 <phase_3+334>
   0x00005555555553e2 <+143>:   callq  0x5555555555b56 <explode_bomb>
   0x00005555555553e7 <+148>:   mov    $0x65,%eax
   0x00005555555553ec <+153>:   jmpq   0x5555555554a1 <phase_3+334>
   0x00005555555553f1 <+158>:   mov    $0x66,%eax
   0x00005555555553f6 <+163>:   cmpl   $0x1c3,0x14(%rsp)
---Type <return> to continue, or q <return> to quit---
```

Rcx = 첫번째 값, Rdx. =두번째 값, R8 = 3번째 값이다.

#0x5555555556bd6에 %d, %c, %d 라고 써 있으므로 입력은 숫자 문자 숫자이다.

Eax가 2와 같거나 작으면 폭탄은 터진다. 또한 첫 입력이 7보다 크면 폭탄은 터진다.

첫입력이 0일경우 두번째 입력과 0x17f랑 비교하는데 0x17은 383이다

Eax와 0x61을 비교하는데 이는 97이므로 문자로 'a'이다

Phase 3 [정답]

0 a 383

Phase 4 [결과 화면 캡처]

```
Halfway there!
8 35 DrEvil

Breakpoint 4, 0x00005555555554fa in phase_4 ()
(gdb) c
Continuing.
```

Phase 4 [진행 과정 설명]

```
Dump of assembler code for function phase_4:
   0x00005555555554fa <+0>:     sub    $0x18,%rsp
   0x00005555555554fe <+4>:     mov    %fs:0x28,%rax
   0x0000555555555507 <+13>:    mov    %rax,0x8(%rsp)
   0x000055555555550c <+18>:    xor    %eax,%eax
   0x000055555555550e <+20>:    lea    0x4(%rsp),%rcx
   0x0000555555555513 <+25>:    mov    %rsp,%rdx
   0x0000555555555516 <+28>:    lea    0x1950(%rip),%rsi        # 0x555555556e6d
   0x000055555555551d <+35>:    callq  0x555555554fa0 <__isoc99_sscanf@plt>
   0x0000555555555522 <+40>:    cmp    $0x2,%eax
   0x0000555555555525 <+43>:    jne    0x55555555552d <phase_4+51>
   0x0000555555555527 <+45>:    cmpl   $0xe,(%rsp)
   0x000055555555552b <+49>:    jbe    0x555555555532 <phase_4+56>
   0x000055555555552d <+51>:    callq  0x555555555b56 <explode_bomb>
   0x0000555555555532 <+56>:    mov    $0xe,%edx
   0x0000555555555537 <+61>:    mov    $0x0,%esi
   0x000055555555553c <+66>:    mov    (%rsp),%edi
   0x000055555555553f <+69>:    callq  0x5555555554c6 <func4>
   0x0000555555555544 <+74>:    cmp    $0x23,%eax
   0x0000555555555547 <+77>:    jne    0x555555555550 <phase_4+86>
   0x0000555555555549 <+79>:    cmpl   $0x23,0x4(%rsp)
   0x000055555555554e <+84>:    je     0x555555555555 <phase_4+91>
   0x0000555555555550 <+86>:    callq  0x555555555b56 <explode_bomb>
   0x0000555555555555 <+91>:    mov    0x8(%rsp),%rax
   0x000055555555555a <+96>:    xor    %fs:0x28,%rax
   0x0000555555555563 <+105>:   jne    0x55555555556a <phase_4+112>
   0x0000555555555565 <+107>:   add    $0x18,%rsp
   0x0000555555555569 <+111>:   retq
   0x000055555555556a <+112>:   callq  0x555555554f00 <__stack_chk_fail@plt>
```

#의 주소 0x555555556e6d를 열어보면 %d, %d가 나오는데 따라서

입력은 숫자 숫자라는 것을 알 수 있다.

Jne로 eax와 2로 비교하므로 eax가 2보다 작으면 터진다.

또한 두번째 숫자가 16보다 작아도 터지고 첫번째 숫자가 14보다 커도 터진다.

두번째 숫자가 35가 아니어도 터진다. 따라서 두번째 숫자는 35이다.

Phase 4 [정답]

8 35 DrEvil

Phase 5 [결과 화면 캡처]

```
So you got that one.  Try this one.
beldog

Breakpoint 5, 0x000055555555556f in phase_5 ()
(gdb) c
Continuing.
```

Phase 5 [진행 과정 설명]

```
Dump of assembler code for function phase_5:
   0x000055555555556f <+0>:     push   %rbx
   0x0000555555555570 <+1>:     sub    $0x10,%rsp
   0x0000555555555574 <+5>:     mov    %rdi,%rbx
   0x0000555555555577 <+8>:     mov    %fs:0x28,%rax
   0x0000555555555580 <+17>:    mov    %rax,0x8(%rsp)
   0x0000555555555585 <+22>:    xor    %eax,%eax
   0x0000555555555587 <+24>:    callq  0x555555555835 <string_length>
   0x000055555555558c <+29>:    cmp    $0x6,%eax
   0x000055555555558f <+32>:    jne    0x5555555555e6 <phase_5+119>
   0x0000555555555591 <+34>:    mov    $0x0,%eax
   0x0000555555555596 <+39>:    lea    0x1673(%rip),%rcx        # 0x555555556c10 <array.3418>
   0x000055555555559d <+46>:    movzbl (%rbx,%rax,1),%edx
   0x00005555555555a1 <+50>:    and    $0xf,%edx
   0x00005555555555a4 <+53>:    movzbl (%rcx,%rdx,1),%edx
   0x00005555555555a8 <+57>:    mov    %dl,0x1(%rsp,%rax,1)
   0x00005555555555ac <+61>:    add    $0x1,%rax
   0x00005555555555b0 <+65>:    cmp    $0x6,%rax
   0x00005555555555b4 <+69>:    jne    0x55555555559d <phase_5+46>
   0x00005555555555b6 <+71>:    movb   $0x0,0x7(%rsp)
   0x00005555555555bb <+76>:    lea    0x1(%rsp),%rdi
   0x00005555555555c0 <+81>:    lea    0x1618(%rip),%rsi        # 0x555555556bdf
   0x00005555555555c7 <+88>:    callq  0x555555555852 <strings_not_equal>
   0x00005555555555cc <+93>:    test   %eax,%eax
   0x00005555555555ce <+95>:    jne    0x5555555555ed <phase_5+126>
   0x00005555555555d0 <+97>:    mov    0x8(%rsp),%rax
   0x00005555555555d5 <+102>:   xor    %fs:0x28,%rax
   0x00005555555555de <+111>:   jne    0x5555555555f4 <phase_5+133>
   0x00005555555555e0 <+113>:   add    $0x10,%rsp
   0x00005555555555e4 <+117>:   pop    %rbx
   0x00005555555555e5 <+118>:   retq
   0x00005555555555e6 <+119>:   callq  0x555555555b56 <explode_bomb>
   0x00005555555555eb <+124>:   jmp    0x555555555591 <phase_5+34>
   0x00005555555555ed <+126>:   callq  0x555555555b56 <explode_bomb>
   0x00005555555555f2 <+131>:   jmp    0x5555555555d0 <phase_5+97>
   0x00005555555555f4 <+133>:   callq  0x555555554f00 <__stack_chk_fail@plt>
```

0x555555556c10의 주소를 열어보면

"maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?"가 나온다.

앞에 maduiersnfotvbyl는 총 16개이다.

두번째 #0x555555556bdf에는 devils가 있다. 이를 string_not_equal로 비교를 한다.

하지만 devils를 정답으로 입력하면 폭탄이 터진다.

Phase_5함수에 의해 입력된 단어의 하위 4bye만 남기므로

M부터 ㅣ까지 16개의 알파벳을 각각 16진수로 0부터 15까지 변환한뒤 devils의 단어를

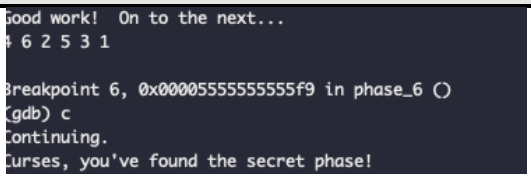
만든뒤 아스키코드표를 이용하여 남겨진 하위 4byte로 16진수로 변환한 수들을 대입하면

Beldog라는 값이 나온다.

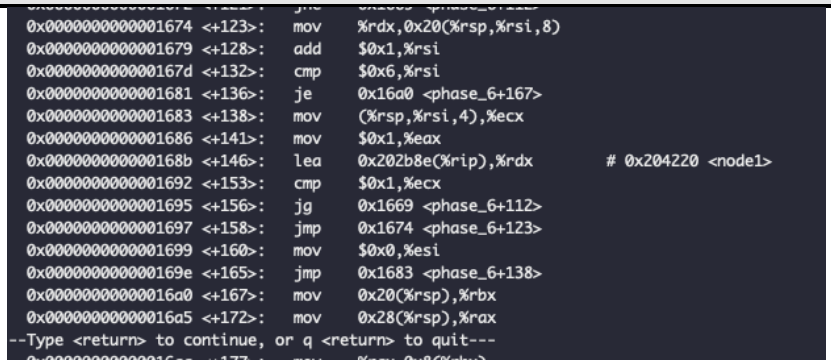
Phase 5 [정답]

beldog

Phase 6 [결과 화면 캡처]



Phase 6 [진행 과정 설명]



0x204220을 x/s로 열면 node1을 발견할 수 있다.

x/24s로 node5까지 확인 할 수 있다.

Phase_6함수는 read_six_numbers로 총 6개의 수를 입력받는다는 것을 알 수 있는데

따라서 node6까지 있을 수 있다는 것을 유추할 수 있다.

x/24x로 각 noded의 주소를 살펴보면 값과 다음 노드의 주소가 들어있는데 이를 통해

node6의 주소와 값들의 순서를 알 수 있다.

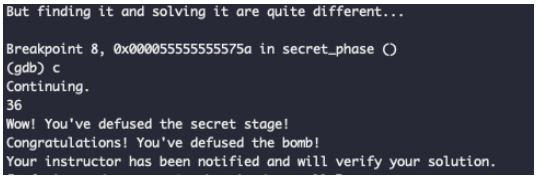
1 : 81 2: 524 3 : 187 4: 718 5 : 473 6 : 648
따라서 총 6개의 Node를 확인할 수 있고 값들을 크기 순서대로 나타내면

4 6 2 5 3 1 이다.

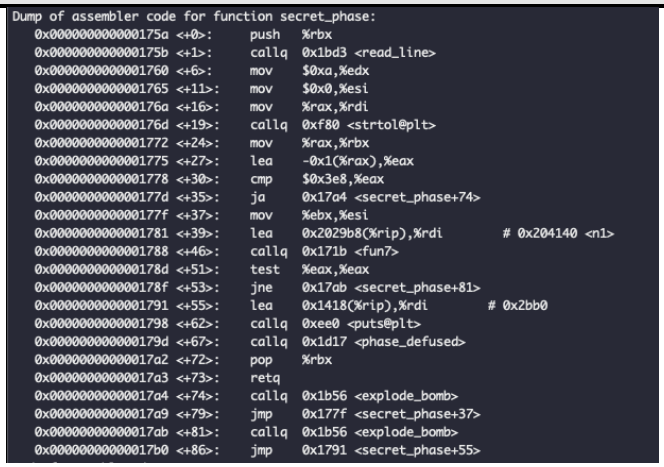
Phase 6 [정답]

4 6 2 5 3 1

S_Phase [결과 화면 캡처]



S_Phase [진행 과정 설명]



Strol로 입력받은 값을 10진수로 변경하는데 이 값이 999를 넘으면 폭탄이 터진다.

Strtol을 이용함으로 입력되는 값은 1개라고 유추할 수 있다.

x/d로 0x204140를 확인하면 36이 나온다.

36을 입력하니 해제되었다.

S_Phase [정답]