Bomb Lab report (bomb104)

202155536 김주송

<phase_1>

```
(gdb) disas phase_1
Dump of assembler code for function phase_1:
=> 0x00005555555552b4 <+0>:     sub    $0x8,%rsp
   0x00005555555552b8 <+4>:     lea    0x1841(%rip),%rsi        # 0x555555556b00
   0x00005555555552bf <+11>:    callq  0x5555555557d3 <strings_not_equal>
   0x00005555555552c4 <+16>:    test   %eax,%eax
   0x00005555555552c6 <+18>:    jne    0x5555555552cd <phase_1+25>
   0x00005555555552c8 <+20>:    add    $0x8,%rsp
   0x00005555555552cc <+24>:    retq
   0x00005555555552cd <+25>:    callq  0x555555555ad7 <explode_bomb>
   0x00005555555552d2 <+30>:    jmp    0x5555555552c8 <phase_1+20>
End of assembler dump.
(gdb) x/s 0x555555556b00
0x555555556b00: "Houses will begat jobs, jobs will begat houses."
```

<strings_not_equal> 을 보면 같은 문자열을 입력해야하므로

⇨ Houses will begat jobs, jobs will begat houses.

<phase_2>

```
(gdb) x/s 0x555555556dd1
0x555555556dd1: "%d %d %d %d %d %d"
```

(gdb) disas phase_2

(gdb) disas read_six_numbers

x/s 0x555555556dd1 을 하면 6개의 정수형을 입력으로 받는다는 사실을 알 수 있다.

임의로 "1 2 3 4 5 6"을 입력해주면 1과 2는 비교 후 통과된다.

3과 4를 비교해서 같으면 점프한다. -> 세번째 인자가 4여야 한다.

1 2 4 이런식으로 더하는 값이 1씩 늘어나는 규칙을 발견할 수 있다.

⇨ 1 2 4 7 11 16

<phase_3>

```
(gdb) x/s 0x555555556ddd
0x555555556ddd: "%d %d"
```

0x555555556ddd의 값을 확인하여 2개의 정숫값을 입력받는 것을 확인할 수 있다.

```
0x000055555555536a <+40>:    cmp    $0x1,%eax
0x000055555555536d <+43>:    jle    0x55555555538c <phase_3+74>
0x000055555555536f <+45>:    cmpl   $0x7,(%rsp)
0x0000555555555373 <+49>:    ja     0x555555555408 <phase_3+198>
```

0x1보다 작거나 같으면 폭탄 -> %eax가 1보다 커야한다.

0x7보다 크면 폭탄 -> (%rsp)가 7보다 작거나 같아야한다.

Gdb를 이용하여 %rsp에 첫번째 정수가 들어가는 것을 발견할 수 있다.

```
0x000055555555539a <+88>:    mov    $0x0,%eax
0x000055555555539f <+93>:    sub    $0x204,%eax
0x00005555555553a4 <+98>:    add    $0x38c,%eax
0x00005555555553a9 <+103>:   sub    $0x6b,%eax
0x00005555555553ac <+106>:   add    $0x6b,%eax
0x00005555555553af <+109>:   sub    $0x6b,%eax
0x00005555555553b2 <+112>:   add    $0x6b,%eax
0x00005555555553b5 <+115>:   sub    $0x6b,%eax
```

%eax=0x0-0x204+0x38c-0x6b+0x6b-0x6b+0x6b-0x6b=285

⇨ 1 285

<phase_4>

```
(gdb) x/s 0x555555556ddd
0x555555556ddd: "%d %d"
```

2개의 정숫값을 입력받는 것을 확인할 수 있다.

```
0x0000555555555480 <+40>:    cmp    $0x2,%eax
0x0000555555555483 <+43>:    jne    0x55555555548b <phase_4+51>
0x0000555555555485 <+45>:    cmpl   $0xe,(%rsp)
0x0000555555555489 <+49>:    jbe    0x555555555490 <phase_4+56>
0x000055555555548b <+51>:    callq  0x555555555ad7 <explode_bomb>
0x0000555555555490 <+56>:    mov    $0xe,%edx
```

%eax값이 0x2와 같지 않다면 폭탄 -> 2와 같아야 한다.

첫번째 입력이 0xe보다 작아야 폭탄을 피해 갈 수 있다.

```
0x00005555555554a2 <+74>:    cmp    $0x5,%eax
0x00005555555554a5 <+77>:    jne    0x5555555554ae <phase_4+86>
0x00005555555554a7 <+79>:    cmpl   $0x5,0x4(%rsp)
0x00005555555554ac <+84>:    je     0x5555555554b3 <phase_4+91>
0x00005555555554ae <+86>:    callq  0x555555555ad7 <explode_bomb>
0x00005555555554b3 <+91>:    mov    0x8(%rsp),%rax
```

eax는 func4의 return 값으로부터 나온다.

두번째 입력이 5여야 한다.

```
0x000055555555542c <+19>:    cmp    %edi,%ecx
0x000055555555542e <+21>:    jg     0x55555555543e <func4+37>
0x0000555555555430 <+23>:    mov    $0x0,%eax
0x0000555555555435 <+28>:    cmp    %edi,%ecx
0x0000555555555437 <+30>:    jl     0x55555555544a <func4+49>
0x0000555555555439 <+32>:    add    $0x8,%rsp
0x000055555555543d <+36>:    retq
0x000055555555543e <+37>:    lea    -0x1(%rcx),%edx
0x0000555555555441 <+40>:    callq  0x555555555419 <func4>
0x0000555555555446 <+45>:    add    %eax,%eax
0x0000555555555448 <+47>:    jmp    0x555555555439 <func4+32>
0x000055555555544a <+49>:    lea    0x1(%rcx),%esi
```

첫번째 입력이 1-14여야 한다.

⇨ 10 5

<phase_5>

```
(gdb) x/s 0x555555556b80
0x555555556b80 <array.3417>:    "maduiersnfotvbylSo you think y
omb with ctrl-c, do you?"
(gdb) x/s 0x555555556b56
0x555555556b56: "oilers"
```

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
m	a	d	u	i	e	r	s	n	f	o	t	v	b	y	l

o	i	l	e	r	s

A	4	F	5	6	7

아스키 코드 값이 하위 4bit가 A4F567이 되는 문자를 정답으로 입력해야 한다.

A - * : J Z j

4 - $ 4 D T d t

F - / ? O _ o

5 - % 5 E U e u

6 - & 6 F V f v

7 - ' 7 G W g w

⇨ jtouvw

<phase_6>

```
(gdb) x/20wx 0x555555758230
0x555555758230 <node1>: 0x000000fe      0x00000001      0x55758240      0x00005555
0x555555758240 <node2>: 0x00000273      0x00000002      0x55758250      0x00005555
0x555555758250 <node3>: 0x000003a7      0x00000003      0x55758260      0x00005555
0x555555758260 <node4>: 0x00000041      0x00000004      0x55758270      0x00005555
0x555555758270 <node5>: 0x000000f5      0x00000005      0x55758110      0x00005555
(gdb) x/4x 0x0000555555758110
0x555555758110 <node6>: 0x000000b9      0x00000006      0x00000000      0x00000000
```

<node1>	<node2>	<node3>	<node4>	<node5>	<node6>
0x0fe	0x273	0x3a7	0x041	0x0f5	0x0b9
254	627	935	95	245	185

큰 것부터 정렬하면 321564 순이다.

7의 보수를 입력해야하므로

⇨ 4 5 6 2 1 3