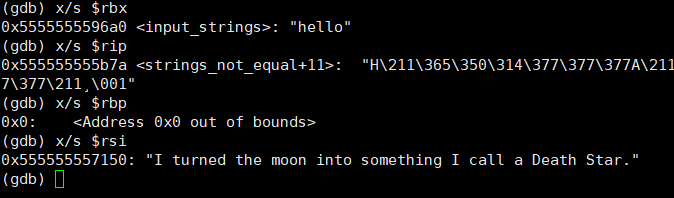
CSED 211 20200445박은하

Lab 3. Bomblab

Phase1 :

Phase1 에서 strings\_not\_equal이라는 함수를 호출하여 string을 비교한 후 같으면 통과하고 같지 않으면 폭탄이 터진다. Strings\_not\_equal에 break를 걸고 r시켜보았다.

Phase1에 hello를 넣었을 때 rbx에 저장되는 값은 hello였다. 비교되는 대상은 rsi register에 저장될 것이라고 생각했는데 x/s를 통해 rsi register에 있는 값을 string으로 변환하자 “I turned the moon into something I call a Death Star.”라는 문장이 나왔다.



phase\_1의 strings\_not\_equal에서 호출하는 string\_length함수는 input과 rsi에 담긴값을 이용해 length를 구하는 함수이다. 이는 rdi의 주소값을 증가시키면서 string에 아무것도 담겨있지 않을 때까지의 string의 length를 증가시키는 함수이다. 각각의 length가 다를시 91번째 줄로 이동하고 이때 return되는 값이 1이기 때문에 폭탄이 터진다.

strings\_not\_equal함수를 계속해서 더 살펴보면 eax=0으로 setting하고 앞에서부터 차례대로 input과 비교 string이 같은지 확인하는 과정이 있다. 마지막까지 같으면 rax를 0으로 setting하고 return한다.

phase\_2

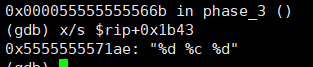
phase\_2에서는 read\_six\_numbers라는 함수를 호출한다. 그 앞에는 array의 공간을 할당하는 작업처럼 보인다. read\_six\_numbers라는 함수를 더 자세히 관찰해보면 rax에 0을 넣고 어떤 기능을 호출한다. 호출된 후에 rsi에 담긴값을 확인하면 www를 넣었을 %d가 들어와있고 rax는 0임을 확인할 수 있다. rax는 5일 때 폭탄이 터지지 않으므로, 이는 문자열을 스페이스로 분리하여 6개 들어왔을 때 폭탄이 터지지 않도록 하는 함수라는 것을 예상할 수 있었다. 즉 phase\_2에서는 6개의 정수를 받는다.

phase\_2의 단계를 따라가 보면 rsp의 시작주소에 담긴값은 0과 같아야 한다. rsp의 주소를 rbp에 저장하고 rax를 1로 초기화한다. rbp의 시작주소를 4만큼 증가한 값은 rbp의 시작주소에담긴값에 1을 더한것과 같아야 한다. 즉 2번째 정수는 1이여야 한다. phase\_2의 62번째 줄로 가면 ebx를 1만큼 증가시키고 rbp에 담긴 메모리 주소를 4만큼 증가시킨다. ebx가 6이 될 때 까지 loop를 빠져나가지 않는다는 것은 6개의 정수에 대해 어떠한 test를 한다는 뜻이다. 3번째 원소는 ebx가 2이고 이때 2와 그 전의 값1을 더한값 3이여야 한다. 4번째 원소는 ebx가 3이고 이때 3과 그 전의 값 3을 더한 6이여 한다. 같은 방식으로 하면 10, 15도 각각 5번째, 6번째 element로 구할 수 있다.

따라서 phase\_2의 결과 값은 0 1 3 6 10 15이다.

phase\_3 :

phase\_3에서 입력 받는 값의 형태가 어ᄄᅠᇂ게 되는지 찾아보기 위해 move 0x1b43(%rip) rsp를 처리한 후 0x1b43(%rip)를 x/s형태로 확인하였다. 그랬더니 %d %c %d의 형태가 나왔다. ᄄᆞ라서 정수 2개와 문자 하나를 받는다는 것을 알 수 있었다.



1 w 3을 input으로 입력했을 때 이것이 어떻게 저장되는지 확인하기 위해 x/d $rsp+0x10, x.d $rsp+0x14, x/d $rsp+0xf를 확인했다. 각각 1, 3, w가 저장되어있었다. 따라서 rsp+0x10에는 첫 번째 숫자 rsp+0x14에는 세 번째 숫자 rsp+0xf에는 두 번째 문자가 저장됨을 알 수 있었다.

첫 번째 조건은 0x10+rsp에 저장된 값과 7을 비교한다. 7보다 크면 폭탄이 터진다. 그다음으로는 eax에 첫 번째 숫자 원소값을 옮긴다.

그리고 86번째 줄에보면 간접 점프를 한다. 이동할 주소는 첫 번째 숫자의 값에 따라 달라진다.

0)값에 0을 넣을 경우 95번째 줄로 이동

eax => 0x62

0x14+rsp=>0x103

b

259

1)값에 1을 넣을 경우 129번째 줄로 이동

이는 eax에 0x79넣고 0x14+rsp에 0x212가 같은지 판단한다. 같을시 0xf+rsp에 eax를 문자로 변환한 값이 같은지 판단한다.

y

530

2)값에 2를 넣을 경우 163번째 줄로 이동

eax=>0x77

0x14+rsp=> 0x65

w

101

3) 값에 3을 넣을 경우 194번째 줄로 이동

eax=>0x6f

0x14+rsp=>0x1b1

o

433

4) 값에 4를 넣을 경우 225번째 줄로 이동

eax=> 0x63

0x14+rsp=>0x1ae

c

430

5) 값에 5를 넣을 경우 252번째 줄로 이동

eax=> 0x6f

0x14+rsp=>0x218

o

536

6) 값에 6을 넣을 경우 279번째 줄로 이동

eax=> 0x65

0x14+rsp=>0x14e

e

334

7) 7을 넣을 경우 306번째 줄로 이동

eax=>0x6e

rsp+0x14=>0x188

n

392

따라서 가능한 모든 조합은

0 b 259

1 y 530

2 w 101

3 o 433

4 c 430

5 o 536

6 e 334

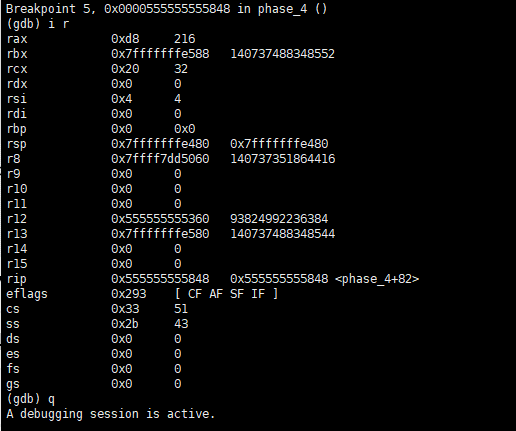
7 n 392

이다.

phase\_4:

phase\_4에서는 func4함수를 호출한다. func4로 호출할 때 argument가 되는 rsi안의 값을 x/s를 통해 확인하면 %d %d가 나온다. 이는 정수 2개를 입력으로 받는다는 것을 알 수 있다.

123 과 3을 입력했을 때 rsp에는 3이 입력되었다. 즉 rsp에는두번째 입력한 숫자가 입력되었음을 알 수 있다. 두 번째 입력한 수는 2를 뺐을 때 2보다 작거나 같아야 한다. 즉 2, 3, 4만 두 번째 정수로 가능하다. 8과 두 번째 입력수를 argument로 func4를 호출한다. func4의 return값은 첫 번째 입력값과 동일해야한다.

func4의 함수를 살펴보면 return값은 I r으로 eax를 확인함으로써 알 수 있다.

2를 넣었을 때는 108이고,

3을 넣었을 때는 162이다.

4를 넣었을 때는 216이다.

즉 108 2

162 3

216 4 가 가능하다.

func4를 함수로 짜보면

int func4(int x, int y)

{

if (y == 0)

{

return 0;

}

int bx = y;//ebx

int t = x;//ebp

//

if (y == 1)

{

return x;

}

int r12=t+func4(x, y - 1);

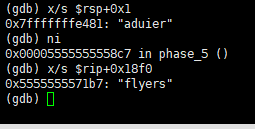
return r12+func4(t, bx - 2);

}

이러한 함수가 나온다.

phase\_5:

입력받은 string은 길이가 6개여야 한다.



abcdef를 넣었을 때 특정한 작업을 거치고 나서 보면 aduier이 argument로 넘겨지고 85번째 줄에서 argument로 넘겨지는 입력값과의 비교대상이 되는 string은 ‘flyers’이다.

특정한 변화 후에 알파벳이 각각 어떻게 변환되는지 확인한 결과는 다음과 같다.

a->a

b->d

c->u

d->i

e->e

f->r

g->s

h->n

I->f

j->o

k->t

l->v

m->b

n->u

o->l

p->m

q->a

r->d

s->u

t->i

u->e

v->r

w->s

x->n

y->f

z->o

따라서 flyers를 만들 수 있는알파벳의 조합은 ionefw이다.

phase\_6:

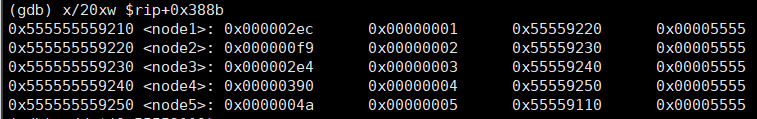
함수 read\_six\_numbers에서 6개의 숫자를 읽는다. read\_six\_numbers부터 118번 줄까지는 6개의 숫자의 배열 규칙을 나타낸다. 연속해서 같은 수가 있으면 안되고 1빼서 5보다 큰수가 있으면 안된다. 따라서 숫자들의 배열은 1부터 6까지의 숫자가 하나씩 있다.

120번째 줄부터는 수에 따라 특정한 노드값을 지닌다는 것을 알 수 있다.

169번째 줄부터는 rbx에 각 숫자에 해당하는 주소가 담겼다, 이 주소는 내림차순으로 정렬되어야 한다. 이건 두 개의 연속된 수 중 앞쪽에 있는 수가 가리키는 주소의 노드에 적힌 값이 더 커야 함을 의미한다.

debugger를 이용해 각 숫자에 따른 노드 값을 251번째 줄에서 \*(rbx)를 통해 확인할 수 있었는데 1은 748, 2는 249 3은 740 4는 912 5는 74 6은 524였다. 따라서 노드 값이 큰 순서대로 나열하면 4 1 3 6 2 5 이다.

혹은 node1의 위치를 x/20xw $rip+388b를 통해 확인할 수 있었다.

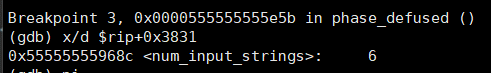


이러한 형태를 가지는데 세 번째에 적힌 값은 다음 노드의 주소를 의미했다.

따라서 node6는 0x55559110에접근하여 값을 파악할 수 있었다.

secret\_phase:

phase\_6을 완료하면 호출되는 함수이다. phase\_6일 경우에만 num\_input\_strings에 6이 들어가는 것을 확인했다.



이 때 ni를 통해 input strings에 들어가는 값을 살펴보면 162와 3으로 phase4에서 입력한 값임을 알 수 있다. 그리고 하나의 string이 더 들어왔을 때 strings\_not\_equal을 통해 비교한다. 비교하는 값을 확인하니

DrEvil이었다.

secret\_phase의 fun7에서는 rdi의 노드값이 계속해서 달라진다. 노드에 해당하는 값을 넣었을 때 출력되는 것이 7이면 secret\_phase가 해체된다. rdi값이 달라짐에 따라 해당하는 노드 안의 수를 확인해보았다.

이때 return 값이 7이라는 건 노드를 tree로 나타냈을 때 level이 3이라는 것이다.

노드가 될 수 있는값은n1=8

n21=8

n22=50

n31=6

n32=22

n33=45

n34=107

n46=47 n48=1001이다.

1001을 입력했을 때 defuse된다.