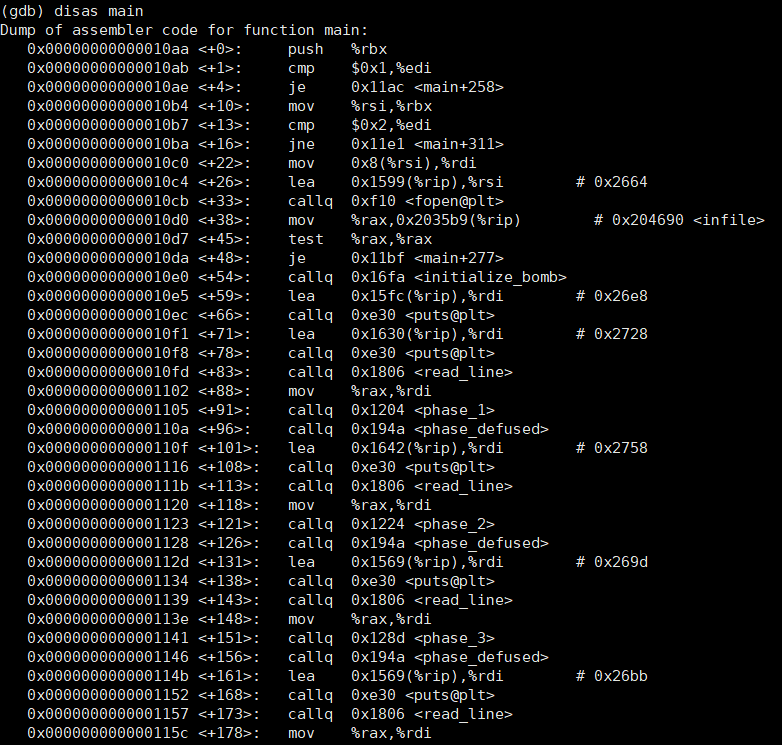
CSED211 LAB3 report

20210054 정하우

Bomb lab을 하기 위해 gdb로 디버깅을 시작한다.

Disas(disassemble)명령어를 통해 maind을 먼저 본다.



R(run)명령어를 이용해서 프로그램을 실행시킨다. 그리고 아무 단어나 치면 폭탄이 터지는 것을 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<phase\_1>

그래서 phase\_1에 breakpoint를 두고 그 밑으로는 실행이 안되도록 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

hi라는 단어를 쳐도 폭탄이 터지지 않고 phase\_1에 멈춰있는 것을 볼 수 있다.

화살표를 보면 프로그램이 어디에 위치해 있는지 볼 수 있다.

<+11>에서 strings\_not\_equal 함수를 부른다.

<+16>에서 함수 return값이 들어갈 %eax 레지스터를 test하고

<+18>에서 결과가 같지 않으면

<+25>로 jump한다. (<phase\_1+25>)이 부분에서 explode\_bomb 함수가 실행되므로 이 함수가 작동되지 않도록 하는게 목표이다.

<+18>에서 두 레지스터의 값을 비교하는 방법은 다음과 같다.

Jne 뜻은 jump if not equal이란 뜻이다. ZF=0일 때 실행되는데,

<+16> test에서 and 연산을 한다. 즉 같으면 1 다르면 0이 return된다. 결국 문자열이 같아야 explode\_bomb가 실행되지 않음을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Disas strings\_not\_equal 를 작동시킨 화면이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Strings\_not\_equal 함수에도 breakpoint를 두고 c(continue) 명령어를 이용해 함수를 디버깅한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Rdi 와 rsi 인자에 함수가 들어있는데, I r 명령어를 통해 정보를 볼 수 있다. 여기서 rdi와 rsi의 정보를 보면

텍스트이(가) 표시된 사진

자동 생성된 설명

rdi에는 처음에 입력한 문자열이, rsi에는 처음보는 문자열이 담겨있다는 것을 알 수 있다.

이제 우리는 phase\_1 입력값으로

“We have to stand with our North Korean allies.”

를 넣어야 한다는 사실을 알고있다.

D 명령어를 이용해 모든 breakpoint를 삭제하고 r명령어를 통해 프로그램을 실행시킨다.

우리가 알고 있는 정답을 입력값으로 넣으면

텍스트이(가) 표시된 사진

자동 생성된 설명

Phase\_1 이 잘 해결 됨을 볼 수 있다.

<phase\_2>

이제 phase\_2에 breakpoint를 걸로 프로그램을 실행시켜보자. 할때마다 breakpoint 하지말고 나중에는 시작부터 모든 phase에 breakpoint를 넣어줘도 될 듯 하다. B phase\_2는 생략하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Read\_six\_numbers에 breakpoint를 건다. 그리고 disas read\_six\_numbers를 하자

텍스트이(가) 표시된 사진

자동 생성된 설명<+36>을 보면 return값을 넣는 %eax에 0(0x0)을 넣는걸 볼 수 있다.

<+50>을 보면 %eax와 5(0x5)를 비교한다.

<+53>에서 만약 5보다 작거나 같으면(jle작거나 같을 때(ZF=1 or SF!=OF) 작동되는 구문이다.)

<+60>에 있는 <explode\_bomb> 함수가 실행된다. 즉 입력값이 5개 이하면 폭탄이 터진다는 뜻이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Add 구문을 보면 eax+eax 연산을 한다. 이 뜻은 전 숫자의 두배를 한 6개의 숫자들을 넣으면 된다는 뜻이다.





1 2 4 8 16 32

를 넣으니 phase\_2도 통과했음을 알 수 있다.

<phase\_3>

Phase\_3는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Phase\_2와 같이 <+28>부분에 lea 가 있으므로 입력값의 형태가 저장되어있다고 보고, ni 명령어를 이용해서 <+28>까지 가서 보니

텍스트이(가) 표시된 사진

자동 생성된 설명

정수 5개를 입력 받음을 알 수 있었다.

텍스트, 명판이(가) 표시된 사진

자동 생성된 설명

이 부분을 보면 <+68> jmpq \*%rax 가 있다. 이 말은 첫 정수값 만큼 주소를 이동하라는 뜻인데, 결국 1이면 <+77>로, 2면 <+84>로 가라는 뜻이다. 여기서 eax에 값을 넣어 주는데, 최종적으로 가는 주소인<+143>에서 두번째 정수(0x4(%rsp))값과 eax값을 비교하는 것을 보아 위의 과정에서 eax에 넣어진 숫자들을 비교하는 것이다. 따라서 (첫번째 숫자, 두번째 숫자) 순서쌍의 경우의 수는

(1,331)

(2,190)

(3,997)

(4,843)

(5,629)

(6,68)

(7,692)

가 된다. 뒤의 세 숫자의 대한 조건은 없으므로 아무 숫자나 넣어도 될 듯하다.

텍스트이(가) 표시된 사진

자동 생성된 설명

잘 작동함을 볼 수 있다.

<phase\_4>

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 논리로 입력값의 개수를 찾을 수 있다. 6개를 입력해야한다. 아마 모든 문제가 6개인걸 암시하는 것 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ni로 함수를 작동시키고, rsp에 내가 입력한 첫 값이 들어감을 확인한다.( 1 1 1 1 1 1입력함)

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 첫 값이 16이하여야 rsp-0xe 가 1보다 작거나 그 이하가 돼서 jbe를 만족해 <+51>에 있는 <explode\_bomb>를 거치지 않고 바로<+56>으로 점프할 수 있다.(1<=16이여서 통과함)

이때 재귀를 돌면 처음에 ecx에 7이 들어간다. 그 밑의 줄이 돌지 않기 위해서는 7보다 작거타 같은 수가 첫 수로 나와야 한다. 계속 재귀를 돌다가 어느 순간에는 첫 입력값과 ecx가 2로 같아지면서 func4가 종료되고, eax는 4보다 작거나 같은 값이 되어서 폭탄이 터지지 않게 된다. 따라서 첫 값은 2가 되어야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 $rsp+0x4 즉 두번째 입력값과 비교했을 때 4와 동일하지 않으면 점프를 하지 않아 밑에 폭탄 함수로 가게 되어 폭탄이 터진다. 따라서 두번째 입력값은 4가 되어야 한다.

나머지 4개의 값들에 대한 제약은 없으므로 아무 값이나 입력하면 된다.

따라서 답은 2 4 1 1 1 1 이 된다.

<phase\_5>



String\_length 함수를 통해 내가 입력한 문자열의 길이를 리턴받고, 리턴값이 저장되어 있는 eax값과 6과 비교해서

텍스트이(가) 표시된 사진

자동 생성된 설명

6보다 크면 폭탄이 터진다. 즉 문자열크기는 6이하여야함을 알 수 있다.

그리고 각 array에 저장된 값들을 다 더해서 0x29가 나와야한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

배열은 다음과 같고, 7\*5+6\*1=41=0x29가 나와서 2번째 array값 1개, 9번째 array값 5개를 받으면 폭탄이 해체된다고 생각해서 답을

299999

로 입력했더니 폭탄이 해체되었음을 알 수 있었다.

<phase\_6>

텍스트이(가) 표시된 사진

자동 생성된 설명

Node1의 주소를 보자

각 node들의 값들을 볼 수 있다. 그런데 6번째 node가 없으므로 찾아가야한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1 331->14b

2 315->13b

3 398->18e

4 930->3a2

5 499->1f3

eax에는 첫 숫자가, 내림차순 기준으로 그 다음 값은 rbx에저장되어 있다. 이 순으로 ni명령어를 통해 보면 6번째 수도 얻을 수 있다고 예측했다. 일단 알고있는 정보를 다 내림차순으로 나열하고 마지막에 6을 넣었다.

4 5 3 1 2 6

텍스트이(가) 표시된 사진

자동 생성된 설명

Ni를 통해 계속 입력받아보면 어느 순간 우리가 보지 못한 값을 찾을 수 있다. 그 값은 0x1be=446이고 우리는 숫자들을 내림차순으로 정렬하여 입력하면 폭탄이 해체됨을 알 수 있다.

6 1be->446

4 5 6 3 1 2

텍스트이(가) 표시된 사진

자동 생성된 설명

최종적으로 폭탄이 해체된 모습이다.