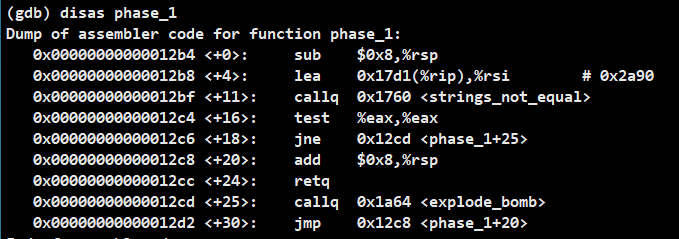


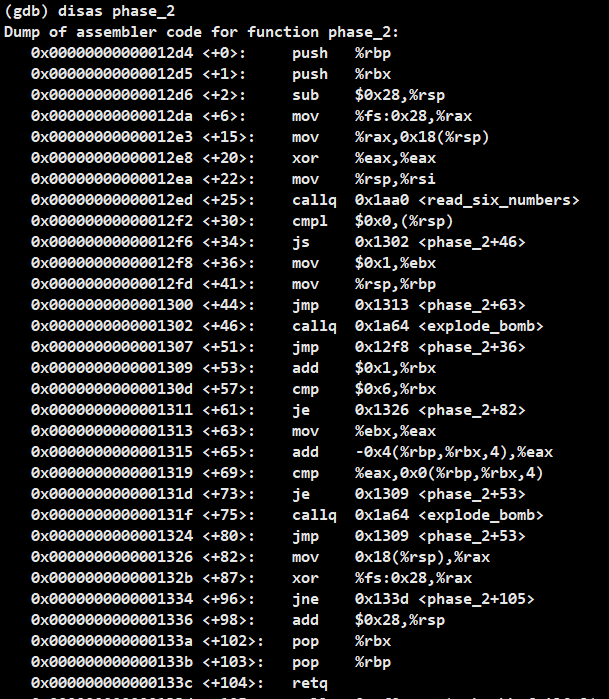
**Phase 1**

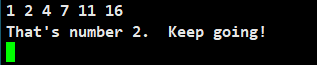
Phase 1 내부의 strings\_not\_equal 함수는 두개의 문자열을 인자로 받아 같으면 0을 반환하는 함수이기 때문에, 현재 phase\_1 함수의 %rdi 에는 내가 입력할 값이, 그리고 %rsi 에는 0x2a90이라는 주소에 저장된 문자열이 포함 되어있는데, 해당 문자열을 보면



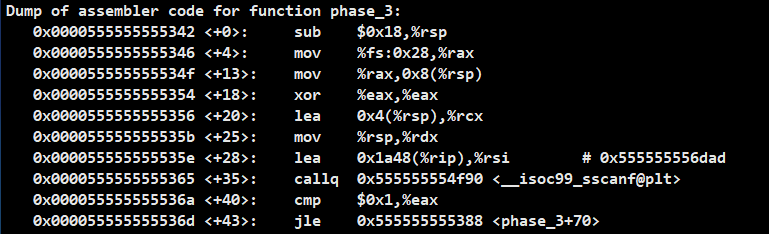
이므로 이를 입력하면 풀리게 된다.

**Phase 2**

read\_six\_numbers 함수는 정수 6개를 입력 받고, 입력 받은 개수를 return 하는데, phase\_2+30의 cmpl $0x0,(%rsp) 이후의 js 조건을 보면, %rsp가 가르키는, 즉 첫번째로 입력 받은 숫자가 음수일 경우 폭탄이 터지게 된다. 그러므로 첫번째 수는 가장 간단한 자연수인 1으로 잡고, 이후 코드들을 살펴 보게 되면 rbx 값이 1부터 6까지 증가하며 6에서 반복이 멈추고, 0x0(%rbp, %rbx, 4), 즉 다음 숫자 값은 rbx+ -0x4(%rbp, %rbx, 4), 즉 rbx의 값 + 전의 숫자 값 에 의해 결정되므로, 1에서 1을 더하고, 그 뒤에 2에서 2를 더하게 되면서 답은 이런 순열이 된다.



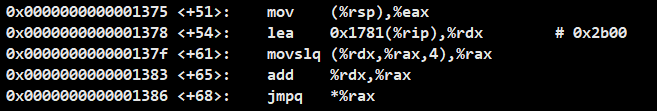
**Phase 3**

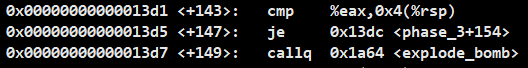
sscanf 함수를 호출하기 전에, %rsi 값으로 전달되는 0x1a48에 저장된 문자열을 보면,



%d %d임을 보아 2개의 값을 전달받는 것으로 알 수 있다. 그리고

을 통해 입력하는 첫번째 숫자 값이 7을 넘으면 폭탄이 터짐을 확인 할 수 있다.

rax에 특정 주소 값 + 첫번째 숫자 값 \* 4의 주소로 이동하는 것으로 보아, 0x2b00의 주소에는 점프 테이블 구조가 정의되어 있을 것이라고 가정하고,

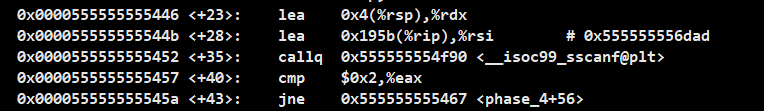


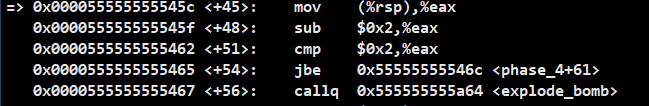
이후 코드에서 입력하는 2번째 숫자 값이 %rax의 값과 같으면 해체 되기에 첫번째 점프하는 값 +75위치의 코드를 보게 되면

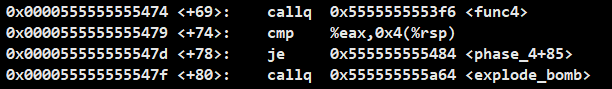


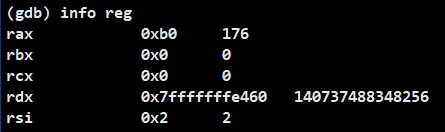
%rax에 1e0(16진수) -> 480(10진수)을 넣음을 확인하고 따라서, 1, 480을 입력했더니 성공적으로 해체되었다.

**Phase 4**

이번에도 phase 3과 마찬가지로 %d %d를 통해 숫자 2개를 입력받는 것을 확인하고,

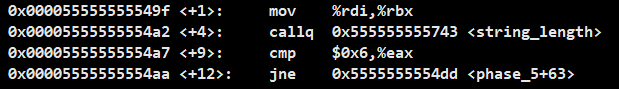
%rsp – 2 값이 2보다 클 경우 폭탄이 터지게 된다. 어떤 값이 %rsp에 들어가는지 1, 2를 넣어서 디버깅을 해보았는데, 이와 같이 2번째 숫자가 %rsp이고, 첫번째 숫자가 %rsp+4에 저장됨을 확인할 수 있었다.

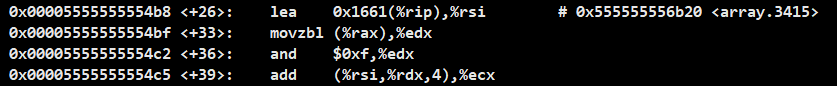
이후 +69위치를 보면 재귀적으로 짜여진 func4함수의 return 값과 0x4(%rsp), 즉 첫번째 숫자와 같으면 되므로 단순히 func4호출 바로 다음 위치까지 진행 한 뒤 %rsi, %rax의 값을 확인하게 되면

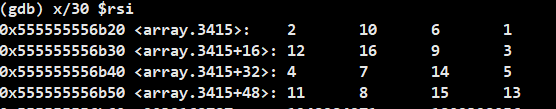


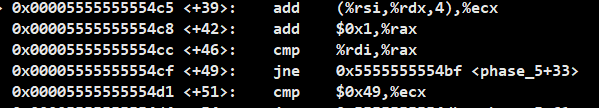
이므로 176, 2가 정답이 된다.

**Phase 5**

string\_length의 return 값, 즉 문자열의 길이가 6개가 아니면 폭탄이 터지므로 답은 6임을 확인 할 수 있다.

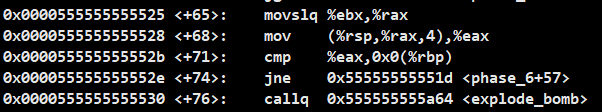
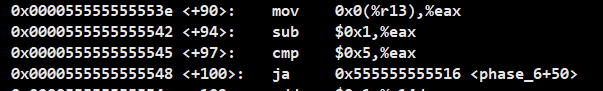
이후 +33~+36줄의 movzb1과 and명령어의 수행결과를 확인하기 위해 abcdef를 넣어서 레지스터를 본 결과, %rax가 a일때 %rdx의 값이 1, b일때 2, c일때 3... 임을 확인하였고, %rsi에 담은 array.3415의 정보 또한 확인하였다.

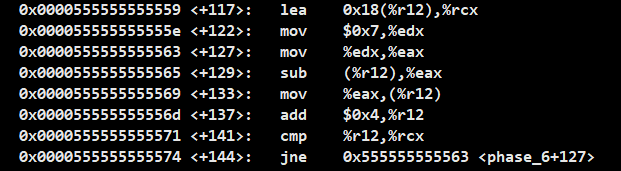


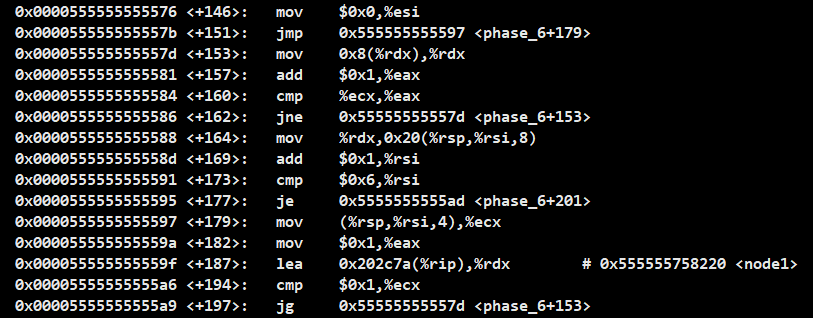
이후 %rcx에 저 배열의 index = %rdx의 값임을 확인하고 해당 인덱스의 값을 문자열이 끝날 때 지 계속 더하여 최종적으로 +51번째 줄의 0x49(16진수) -> 73(10진수)를 만들면 되므로 14[10번째 index]+14+14+14+14+3[7번째 index] => j+j+j+j+j+g을 통해 73을 만들었다. jjjjjg를 입력하니 성공적으로 통과되었다.

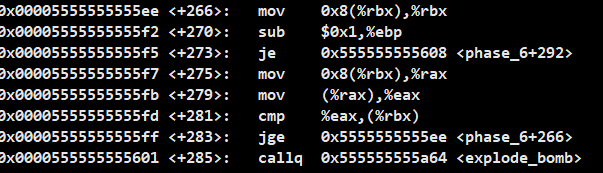
**Phase 6**

read\_six\_numbers를 통해 6개의 정수를 입력 받음을 확인하였다.

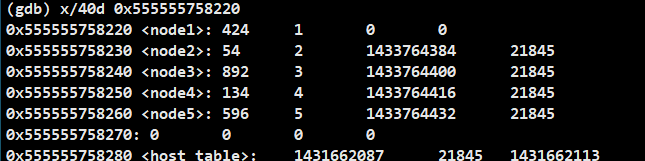
복잡한 이중 반복하는 코드였지만 ?-1 <= 5, 즉 6이하의 값으로 구성되고 각 숫자들이 서로 같지 않아야 한다는 조건을 통해 1,2,3,4,5,6으로 구성된 숫자 들이어야 함을 확인했다.

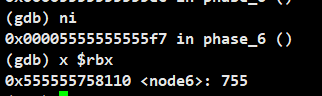
그 이후 이 반복문이 내가 입력한 숫자들을 7로 뺀 결과로 바꾼다는 것을 확인하였다.

이후 입력한 결과에 따라 node들이 메모리 상에 저장됨을 확인하고,

이전 값이 다음 값보다 크거나 같을 경우에만 폭탄이 터지지 않으므로 내림차순으로 정렬되게끔 입력해야 한다는 것을 알았다.

그러므로 각 노드의 값을 아래와 같이 확인 하였고,





(phase\_6+275줄에서 확인한 rbx = rsp+32값.)

892>755>596>424>134>54 => 3>6>5>1>4>2 이지만 중간에 내 입력 값을 7로 뺀 결과로 바꾸는 코드가 있었으므로 미리 7로 뺀 결과, 즉 4 1 2 6 3 5를 입력하면 해결하게 된다.