System Software Assignment #2 Bomb Lab

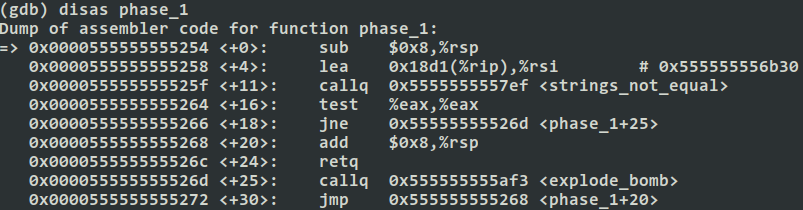
학과 : 조선해양공학과

학번 : 201329119

이름 : 김창민

1. phase\_1

skeleton file을 참고하여 phase\_1에 breakpoint를 걸어두고 “mystring”을 입력하였다.



strings\_not\_equal이라는 함수를 발견하고 phase\_1+11까지 진행한 후 레지스터를 조사하였더니 rax 레지스터에 내가 입력한 “mystring”을 가리키는 주소가 저장되어 있는것을 발견하였다.



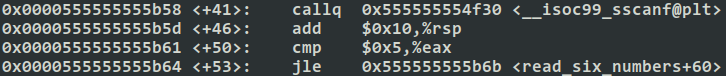
그리고 rsi로 인자를 전달하는데, 그 값을 조사해 보니 비교할 대상을 전달하는듯 하여 조사해 보았다.



다시 실행시켜 위 문자열을 입력하니 phase\_1을 통과한것을 확인할 수 있었다. 추가적으로 strings\_not\_equal은 두 문자열이 같지 않은 경우 rax에 0을 return하는것을 유추할 수 있었다.

1. phase\_2

phase\_2를 disas 해보니 read\_six\_numbers라는 함수가 있었고, 이 함수를 disas 해보았다.

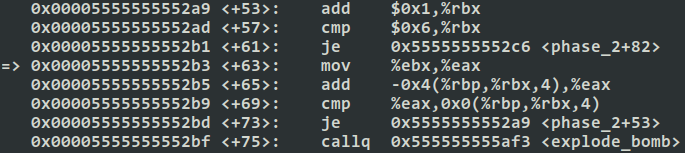


그 중 일부를 살펴보니 sscanf함수를 호출하고 리턴값인 rax가 5보다 작거나 같으면 폭탄이 터지게 되어 있었다. ssccanf함수 호출 직전까지 진행한 후 rsi 레지스터를 검사해 보니 입력형식을 담고있는 문자열을 발견할 수 있었다.

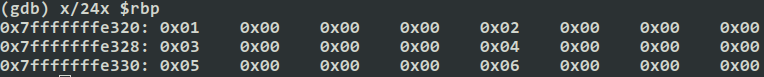


여러 가지 값을 입력해 테스트 해본 결과 sscanf함수는 위 형식과 일치하는 입력의 수를 리턴해주는것을 확인할 수 있었고, 문자열은 정수 6개로 된 입력임을 알수 있었다.

다시 정수 6개(1 2 3 4 5 6)를 입력하고 다음으로 이동한 곳에서 멈추었다.



rbp를 base register로 사용하는 것을 발견하고 메모리를 조사해 보았더니 입력한 정수 6개가 차례로 저장되어 있는것을 발견하였다.

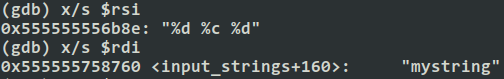


phase\_2+63에서 ebx의 값은 1에서 시작한다. ebx를 eax로 옮기고, eax에 mem(rbp+4\*(rbx-1))을 더하여 그 값을 mem(rbp+4\*rbx)와 비교하여 같은경우 rbx에 1을 더하고 rbx가 6보다 작을때까지 반복하다가 6이되면 탈출하게 된다.

즉, 1+첫 번째 정수 = 두 번째 정수, 2+두 번째 정수 = 세 번째 정수, 3+세 번째 정수 = 4번째 정수가 되는 형식으로 for문을 진행한다. 따라서, 공차가 1에서부터 시작하여 1씩 증가하는 수열을 입력하면 된다.

1. phase\_3

phase\_3을 disas하고 입력값으로 mystring을 입력한 후 sscanf의 호출직전에 breakpoint를 걸어두었다. 그리고 rsi와 rdi가 가리키는 곳을 조사했더니 다음과 같은 결과가 나왔다.



rsi는 입력문자열의 형식을 가리키는 문자열을, rdi는 내가 입력한 입력값을 가리키고 있었다. 이를 통해 유추해 볼 때 입력해야할 문자열의 형식은 "정수 문자 정수"인것을 확인할 수 있다.

따라서 위의 형식에 맞춰 다시 실행하였다. 입력을 "10 A 20”으로 하고 sscanf의 호출직후로 breakpoint를 걸었다.



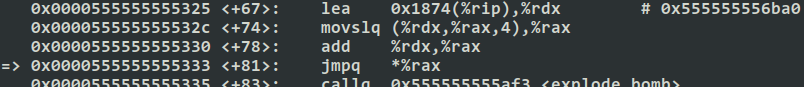
sscanf의 반환값인 rax가 3보다 크지 않으면 폭탄이 터지게 된다. 내가 입력한 문자열은 반환값 3을 반환하므로 문제가 없다.



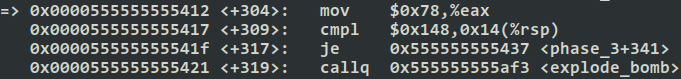
스택포인터+16의 위치에 있는 값과 7을 비교한다. 그 값이 7보다 크거나 0보다 작은경우 폭탄이 터진다.

어떤 값인지 조사하였더니 입력한 문자열 중 첫 번째 값이었다. 따라서 첫번째 값을 0~7중 하나로 정해야한다.

입력을 다시 "7 A 20”으로 하고 다음 분기 명령으로 이동하였다.



rax가 가리키는 곳의 instruction으로 이동하였다.



rsp+14에는 내가 입력한 세번째 정수 20이 저장되어 있었고, 이를 0x148 = 328과 비교하여 같지 않으면 폭탄이 터지게 되므로 세번째 정수를 328로 한다.



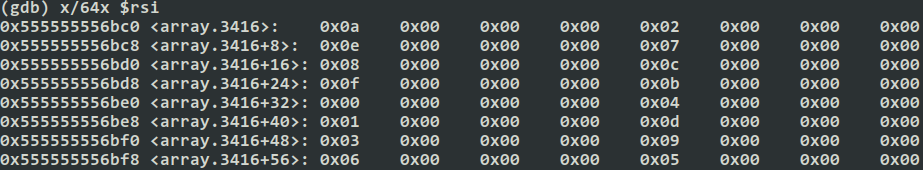
al에는 0x78이 저장되어 있었고, rsp+0xf가 가리키는 곳을 조사하였더니 내가 입력한 A를 의미하는 0x41이 저장되어 있었다. 따라서 두 번째 값은 0x78인 'x’를 입력하였다.

1. phase\_4

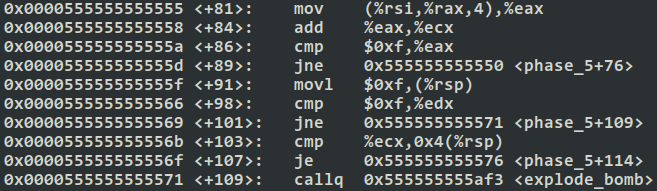
sscanf의 호출직전까지 진행하여 rsi를 조사한 후 정수 두개를 입력해야 한다는것을 발견하였다. 10과 20을 입력하였더니 20에서 2를 뺀 값과 2를 비교하여 작거나 같아야 폭탄을 회피할 수 있었다. 그래서 10과 4를 입력하였다. 다음으로 func4 함수를 만나게 되는데 func4 함수는 재귀함수이다. 함수의 결과값과 첫 번째 입력한 값을 비교하게 되는데 이때의 값이 352가 나오게 되길래 첫 번째 인자로 352를 입력하였더니 다음 단계로 넘어가게 되었다.

1. phase\_5

이번에도 정수 두개를 입력해야 하는데, rsi에서부터 시작하는 배열을 가지고 특수한 연산을 하길래 분석해 보았다.



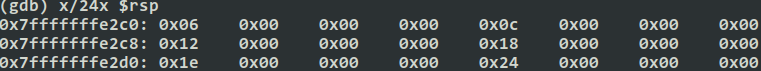
위와 같은 배열에서



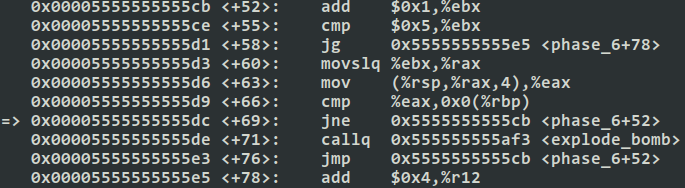
첫 번째 인자로 넣은 offset만큼 떨어진 곳에 있는 값부터 시작하여, 그 값이 다시 offset이 되고, 이를 15번만큼 반복하였을 때 값이 15가 되도록 하면서 값들을 축적한 값을 두 번째 인자로 해야 폭탄이 터지지 않는다. 따라서 5번째 입력은 5 115로 하였다.

1. phase\_6

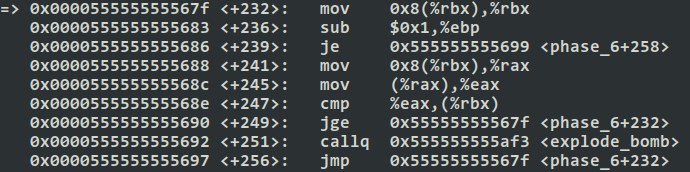
이번에는 6개의 정수를 입력해야 한다. 입력한 6개의 정수는 rsp에 아래와 같이 저장되어 있다.



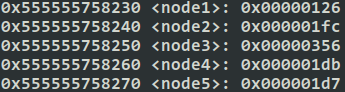
그리고 아래 부분에서 첫 번째 정수와 나머지 5개의 정수가 같으면 안된다는것을 해석하였다.



6 5 4 3 2 1을 입력하고 다음번 폭탄이 터지기 직전으로 갔다.



메모리를 하나씩 조사하는데, 1부터 6까지의 Node에 저장된 값들을 확인할 수 있었고, 내가 입력한 숫자의 노드값이 큰 순으로 숫자들을 정렬해서 입력해야 폭탄이 터지지 않는다는것을 발견하였다.



큰 순으로 정렬하면 3 - 6 - 2 - 4 - 5 - 1 이 된다.

그렇게 6개의 bomb를 모두 해체할 수 있었다.

