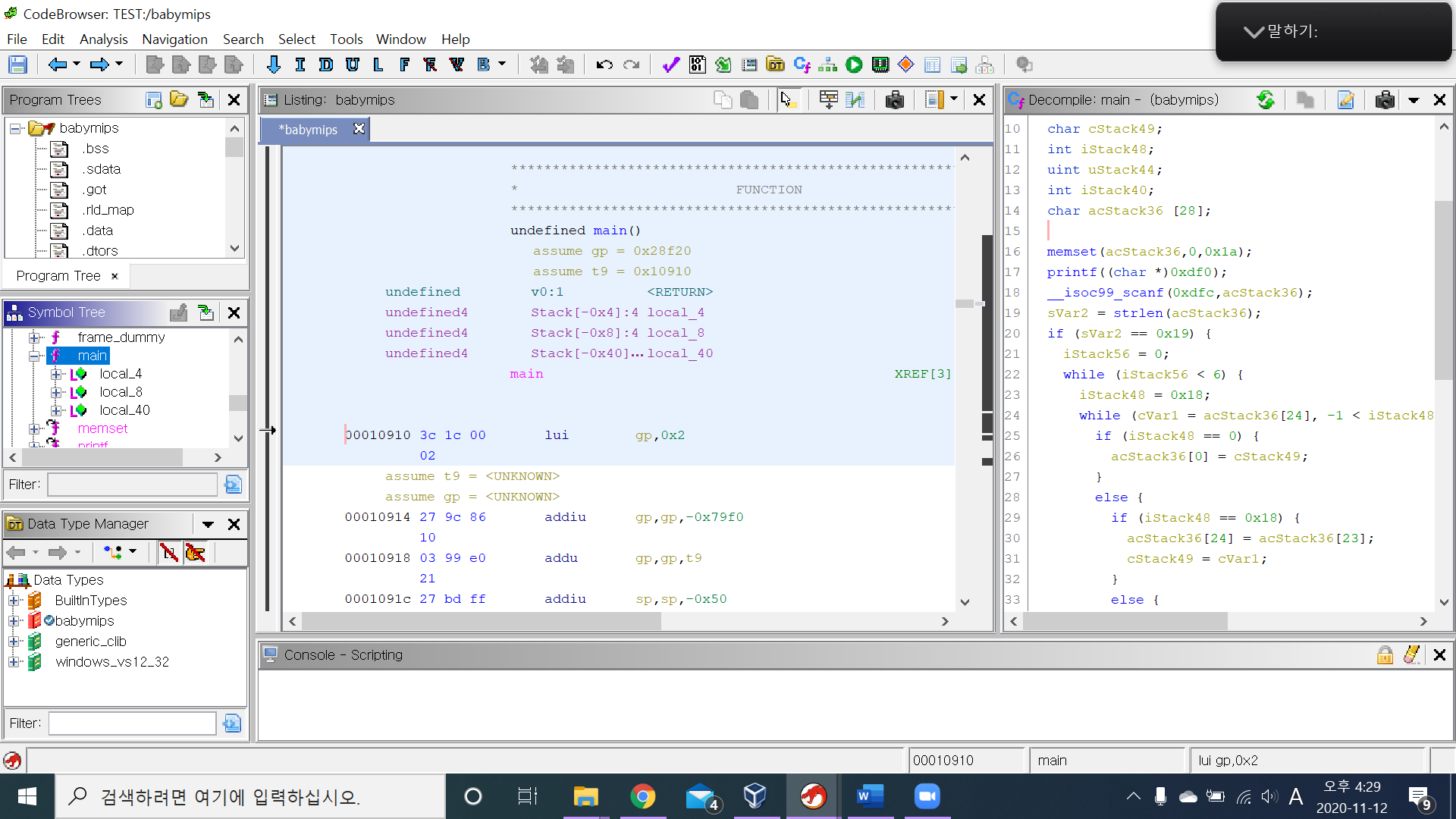
1. 과정

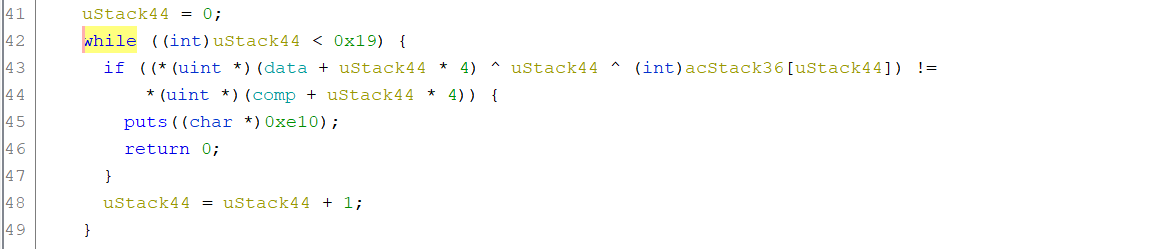
우선 babymips 문제를 기드라로 열어 메인 함수의 디컴파일된 코드를 찾았다.



아래와 같은 코드가 쭉 있다.



이 문제를 해결하는데 핵심이 되는 부분은 43번 줄의 if 조건절이다.



조건절을 만족한다면 return되고, 만족하지 않는다면 다음 인덱스로 넘어가 계속 확인하는 문이었다.

따라서 조건절을 만족하지 않아야 배열의 모든 인덱스를 확인하게 된다.

**(data + uStack44 \* 4) ^ uStack44 ^ (int)acStack36[uStack44]) == \*(uint \*)(comp + uStack44 \* 4)**

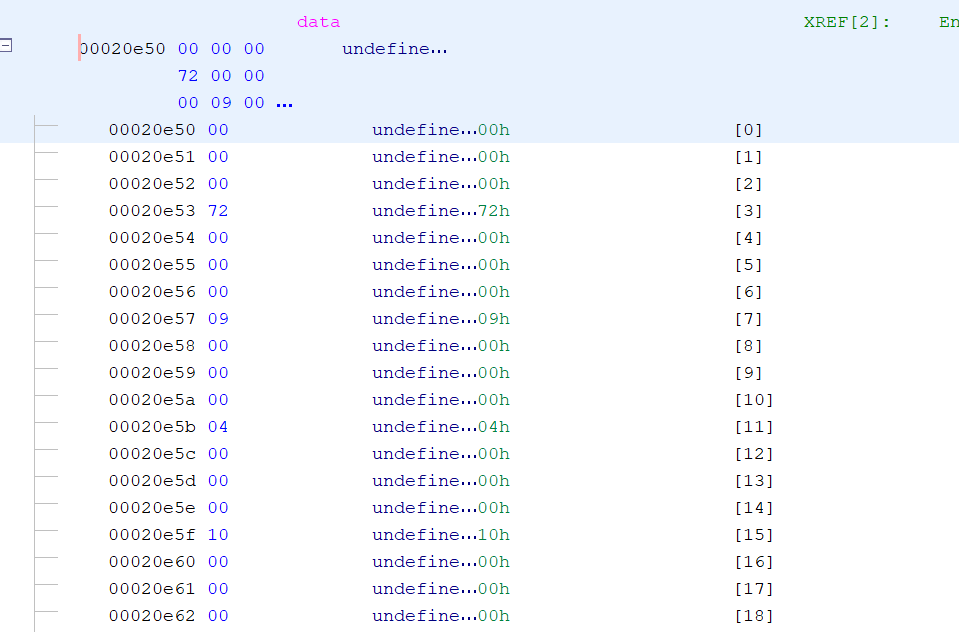
여기서 쓰이는 변수가 uStack44, data, acStack36, comp가 있는데

uStack44: 인덱스 값

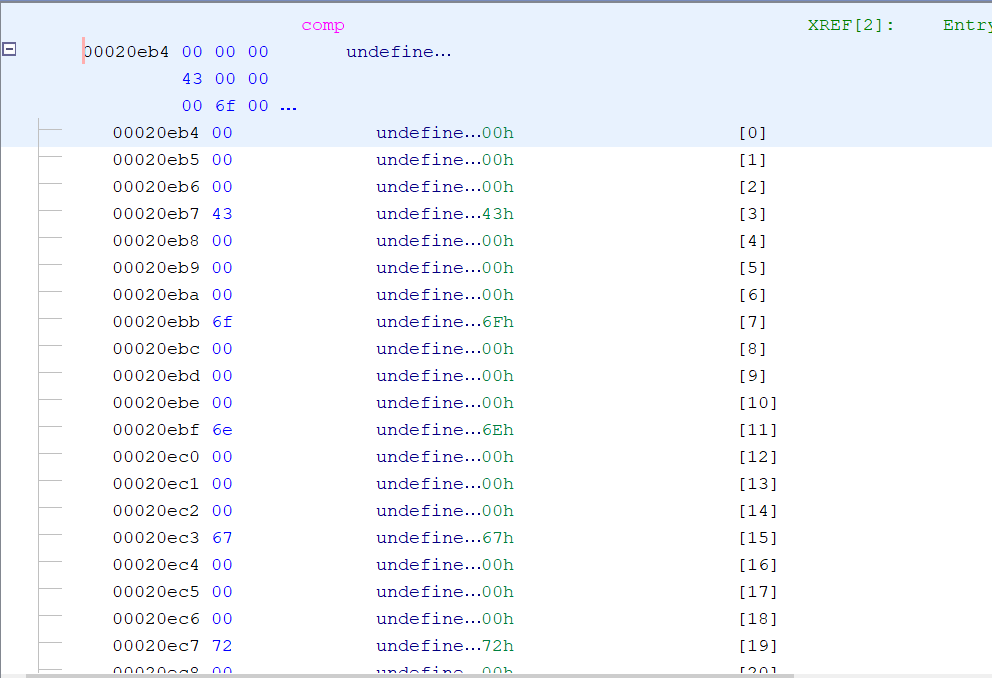
data: 정해진 배열 값

comp: 정해진 배열 값

acStack36: 위 코드를 보아(scanf) 입력 받는 값으로 보인다.



data = {0x72, 0x09, 0x04, 0x10, 0x49, 0x19, 0x3a, 0x1c, 0x34, 0x3b, 0x26, 0x2a, 0x15, 0x30, 0x52, 0x1b, 0x0d, 0x0c, 0x12, 0x38, 0x49, 0x04, 0x02, 0x69, 0x4b}



comp = {0x43, 0x6f, 0x6e, 0x67, 0x72, 0x61, 0x74, 0x7a, 0x5f, 0x59, 0x6f, 0x75, 0x5f, 0x46, 0x6f, 0x75, 0x6e, 0x64, 0x5f, 0x46, 0x6c, 0x61, 0x67, 0x21, 0x21}

data와 comp는 클릭해보면 해당 배열 안의 값들을 볼 수 있는데

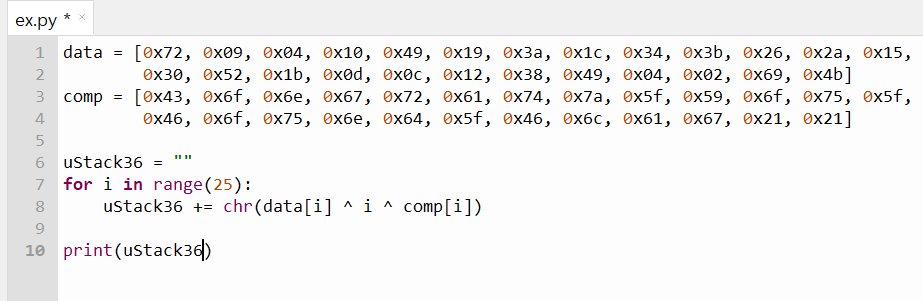
4 인덱스만큼 차이나는 곳에 유의미한 값이 들어있는 것을 확인할 수 있다.

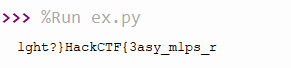
쓸모 있는 값들을 따로 추출해 적어 두고 싶을 다시 한번 정리해보았다.

data XOR uStack44 XOR acStack36[uStack44] == comp

* acStack36[uStack44] == data ^ uStack44(인덱스 값) ^ comp

이제 이 공식을 파이썬 코드로 짜서 실행해보겠다.





값의 순서가 좀 이상한데 제대로 맞춰주면 플래그를 구할 수 있다.

2. 결과

HackCTF{3asy\_mlps\_r1ght?}