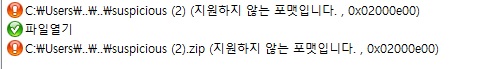
**Gongjae 양은석 Write-up**

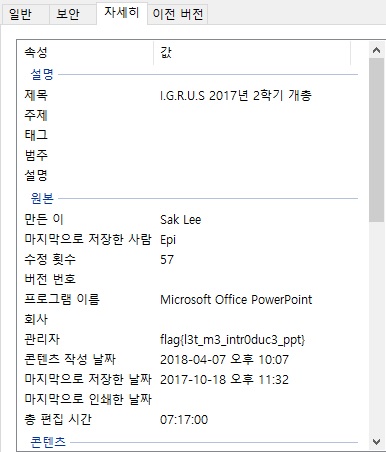
**스테가노그래피 1.**



위와 같이 파일을 압출풀기 하면 오류가 나옵니다.

오류 코드를 검색해보니 확장자를 강제로 바꿔서 오류가 생긴다고 하니, 처음에는 txt파일로 바꿨다가 스테가노그래피라서 다시 그림파일로 바꿨더니 플래그를 볼 수 있었습니다.

**MISC 1.**



Ppt 파일의 속성에 있는 자세히 탭에서 플래그를 찾을 수 있었습니다. (힌트보고 찾았습니다.)

**Misc 3.**



솔직히 이 문제는 풀이가 없습니다. 그냥 프로그램 실행하고, 마우스가 화면 밖으로 안 나가서 메뉴 찾아보다가 우연히 찾았습니다.

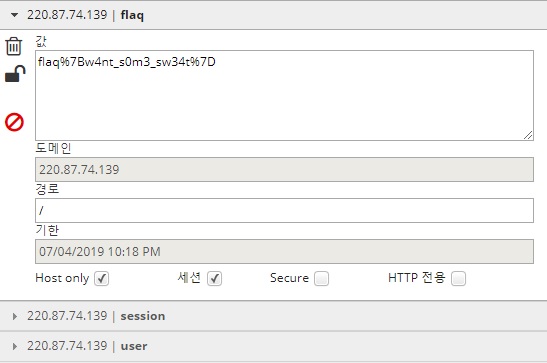
**Web 1.**



F12로 개발자 모드에서 소스를 보면 주석으로 암호가 숨어있는 게 보입니다.

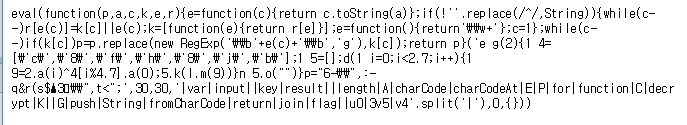
적당히 base64로 디코딩하면 플래그가 나옵니다.

**Web 2.**



처음에 쿠키 값을 보면 플래그에 g가 q로 바뀌어 있습니다. 우선 이 값을 적당히 바꿔서 넣어봤는데 오답이어서 찾아보니 user값이 guest였습니다. 이 값을 admin으로 바꿔서 새로고침을 누르니 제대로 된 플래그 값이 나왔습니다.

**Web 3.**



처음에는 그대로 해석해보려고 콘솔에도 넣어보고 했지만 못풀었습니다. 이 형식을 구글링 해보니 자바스크립트가 패킹되어있는 형태라 해서, 언패킹을 하니,

function decrypt(input) {

var key = ['P', 'A', 'C', 'K', 'A', 'G', 'E'];

var result = [];

for (var i = 0; i < input.length; i++) {

var charCode = input.charCodeAt(i) ^ key[i % key.length].charCodeAt(0);

result.push(String.fromCharCode(charCode))

}

return result.join("")

}

flag = "6-\",:-q&u0(3v5$‑3\",v4<";라는 식이 나왔습니다. 그래서 decrypt에 파라미터 값에 flag값을 넣어서 콘솔로 돌렸더니 정상적인 플래그 값이 나왔습니다.

**Crypto 1.**

Kyv Trvjri tzgyvi, rcjf befne rj r jyzwk tzgyvi, zj fev fw kyv jzdgcvjk wfidj fw vetipgkzfe. Zk zj r jlsjkzklkzfe tzgyvi nyviv vrty cvkkvi ze kyv fizxzerc dvjjrxv (trccvu kyv gcrzekvok) zj ivgcrtvu nzky r cvkkvi tfiivjgfeuzex kf r tvikrze eldsvi fw cvkkvij lg fi ufne ze kyv rcgyrsvk. Reu yviv zj dvjjrxv wifd Alczlj Trvjri. wcrx{4cc\_i04u\_c34u\_k0\_i0d3}

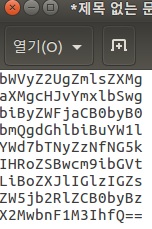
주어진 형태를 보면 마지막 wcrx{4cc\_i04u\_c34u\_k0\_i0d3}이 플래그와 형식이 같다는 것을 볼 수 있습니다. 알파벳을 9번 옮기니 정상적인 플래그 값이 나왔습니다.

**Crypto 2.**

Bnv Paokeyjm izjzmx zm s ukkbgl uw yfkxpjlqtx udxnrvwbot nwfz ss maoea s akicwa uw cfbkiqgdke Wsmyrl uqvyyja hrmwl ue nzm rvnlmxj ix i qvsowxu. Cl qy r zgzs fz hwrpudxnrvwbot mmjykclczzif. Oufx bwh, wfso{b1x3h3j3\_k1vy3l\_1k\_b00\_0ru}

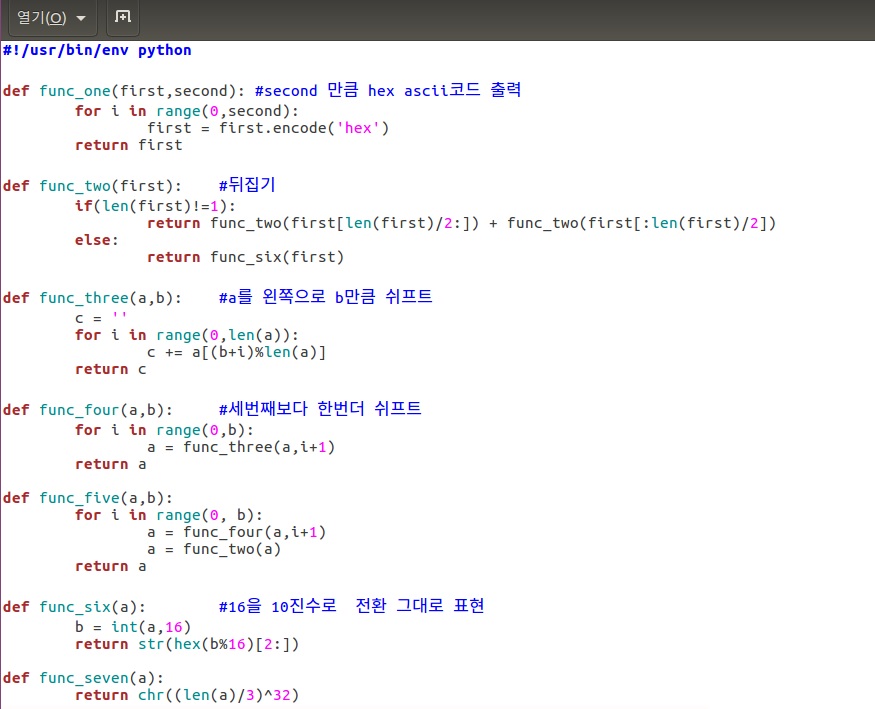
우선 key값이 IGRUS로 주어진 걸 볼 수 있습니다. 플래그의 형식을 적당히 유지하면서 키값이 필요한 암호를 찾아보니 비즈네르 암호란 걸 알 수 있었고, 복호화를 통해 플래그 값을 찾을 수 있습니다.

**Crypto 3.**



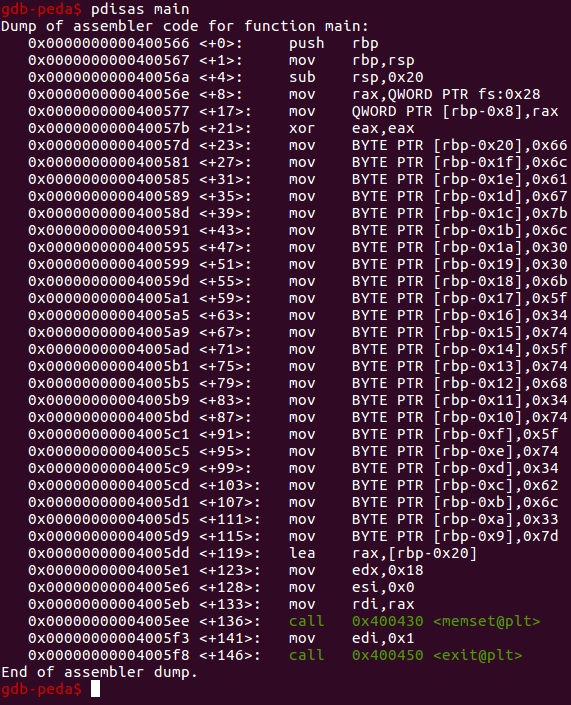
여러 개의 파일을 다 읽어보면 저렇게 일정한 길이의 암호가 잘려서 여러 개 있는 것을 볼 수 있습니다. 이 사진에서 마지막에 보면 base64의 형태가 보입니다. 이걸 보고 전부다 base64로 인코딩 되있을 거라 생각하고, 디코딩을 했더니 flag가 포함된 문장이 있었고, 문제를 클리어했습니다.

**Crypto 4.**



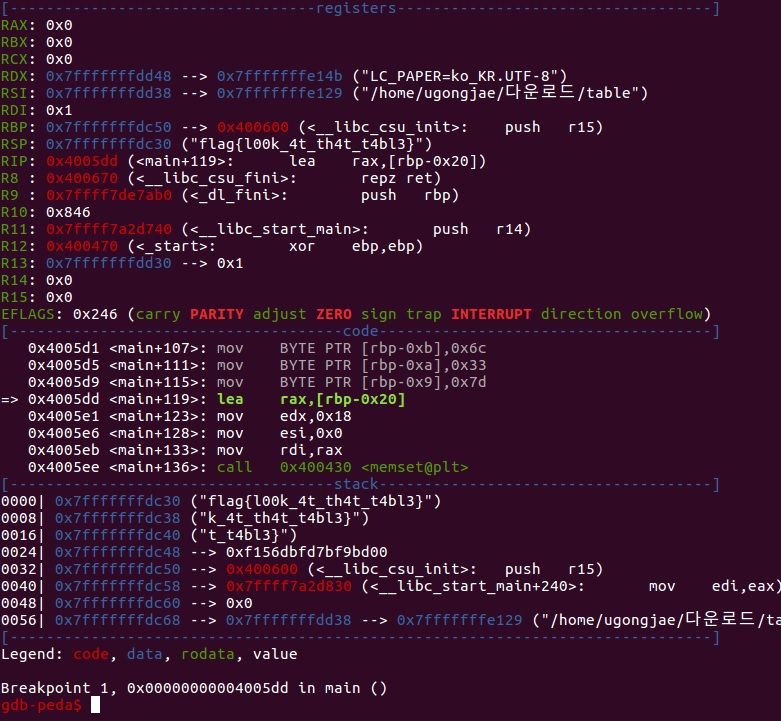
처음에 blablablablabla로 도배가 되어있는데, 보기 쉬운 이름들로 바꾸면 위와 같이 바꿀 수 있습니다. 원리를 분석해 보면 flag값을 받아와서 어떤 과정을 거쳐서 enc파일로 저장한다는 뜻입니다. 첫 번째로 Func\_one(flag,10)으로 되어있는데, 이 뜻은 flag를 hex코드로 바꾸고, 그 hex코드를 또 hex코드로 바꾸고 바꿔서 10번 반복하라는 뜻입니다. enc파일의 내용에 3이 많은 이유는 3이 hex코드로 33이라서 10번 중복되어 3이 어마어마하게 늘어납니다. Func\_two는 내용이 1글자가 될 때 까지, 반으로 자르고, 뒤집어서 각각 재귀한다는 뜻입니다. 즉 이 과정을 거치면 전체의 내용이 거꾸로 뒤집히고 각각의 숫자에 대해 func\_six를 하라는 뜻입니다. Func\_six는 hex코드를 10진수로 바꿨다가. 다시 hex코드로 바꾼다는 의미로, 결국 그대로 출력하는 코드입니다. 즉 func\_two를 종합해보면, hex코드를 거꾸로 뒤집으라는 뜻입니다. Func\_three와 func\_four는 글자를 shift하라는 의미입니다. 첫 번째 함수의 encode를 decode로 바꾸고, five함수의 순서를 바꾸고 enc값을 넣고 함수를 역순으로 실행시키면 flag값을 찾을 수 있습니다.

**Reversing 1.**



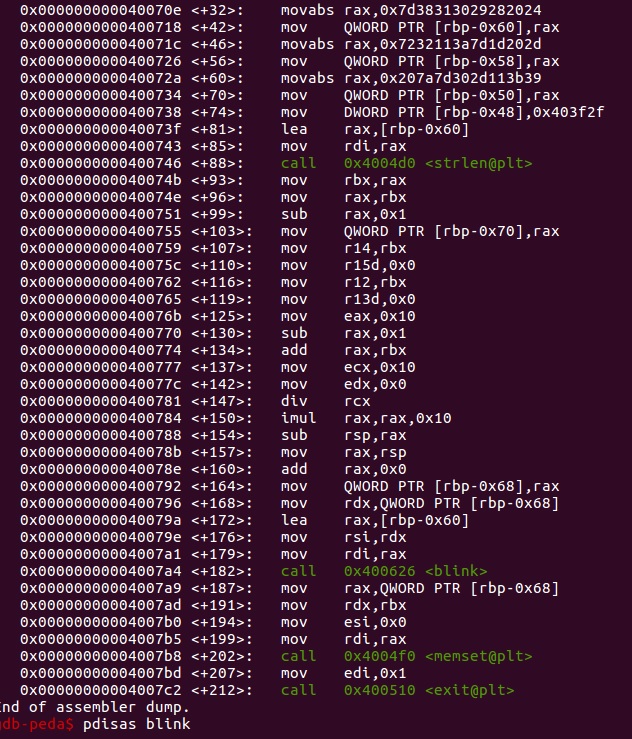
파일을 gdb로 뜯어보면 main+23부터 main+115까지 뭔가 값을 넣는 것을 볼 수 있습니다.

그 다음에 breakpoint를 걸고 실행을 해보면,

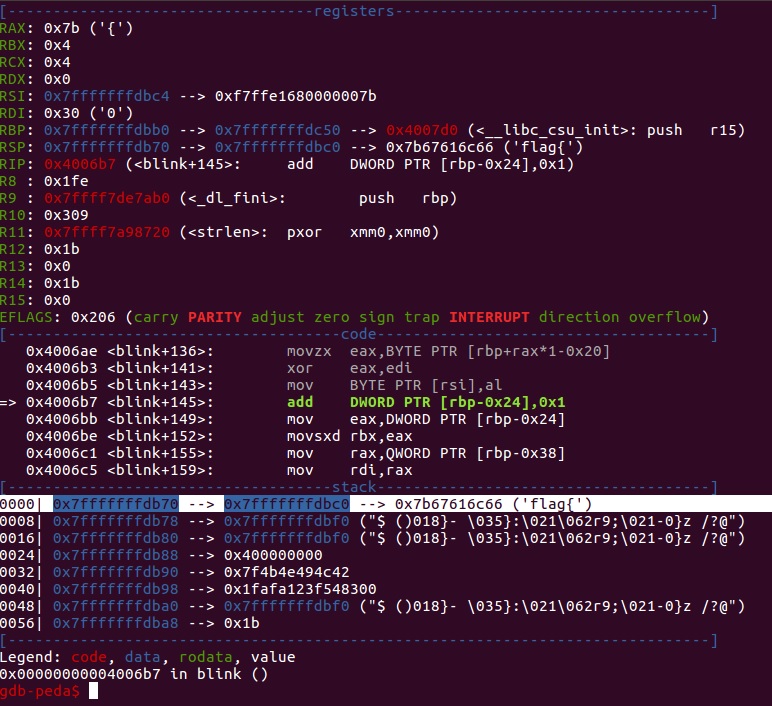


스택에 플래그 값이 들어가 있는 모습을 볼 수 있습니다.

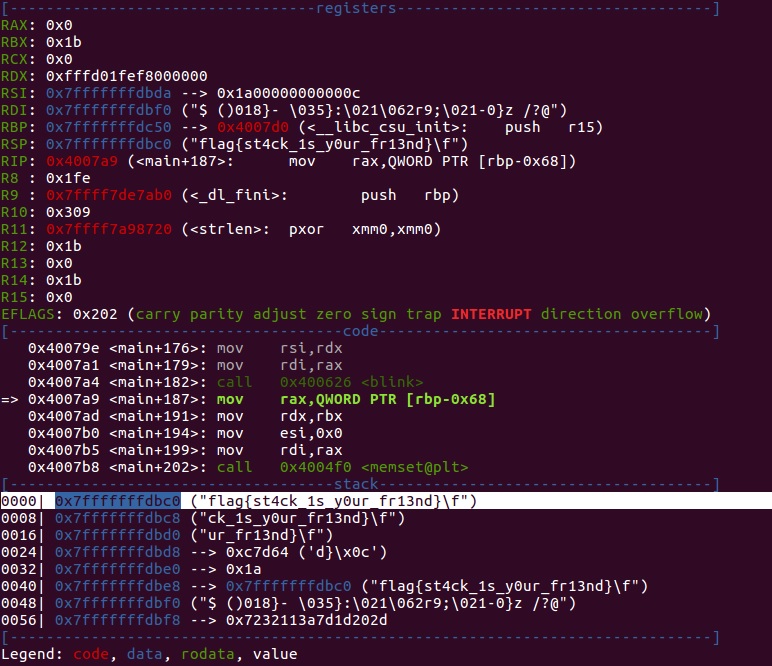
**Reversing 2.**



위와 같이 main을 뜯어볼 수 있고, blink 함수 역시 뜯어 볼 수 있습니다. 일단 복잡해서 적당히 breakpoint를 잡고 돌려보았습니다.

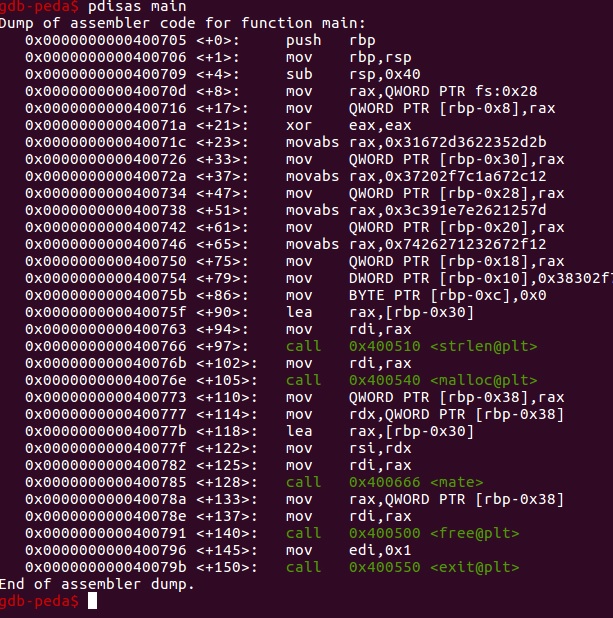


어떤 과정을 거쳐 스택에 flag값이 한글자씩 쌓여가는 것을 볼 수 있습니다. 과정을 뜯어보진 않았고, 플래그 값만 확인하면 되므로, 함수가 끝나서 반환하는 지점에 breakpoint를 잡고 다시 실행을 시키니

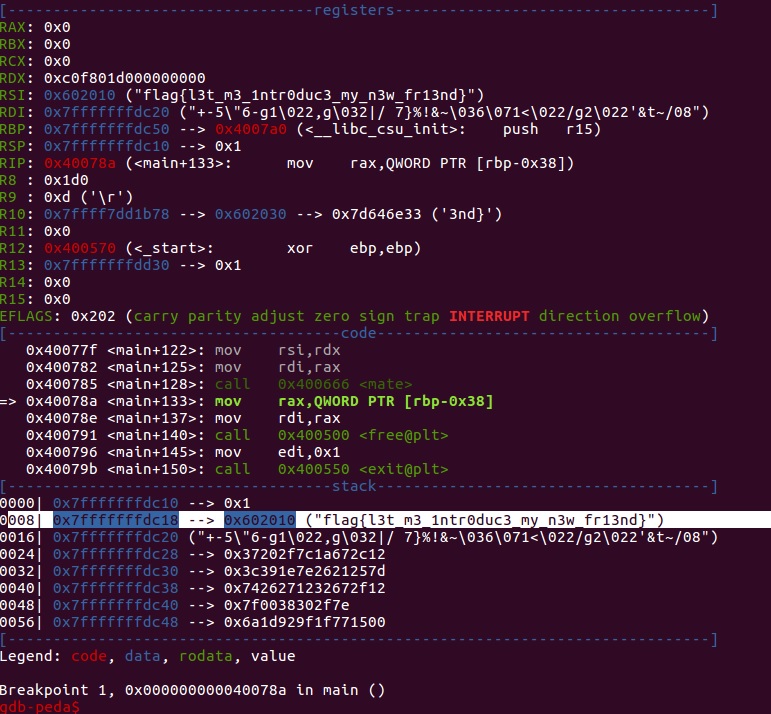


다음과 같이 플래그를 확인할 수 있었습니다.

**Reversing 3.**



우선 파일을 뜯어보면 mate함수가 들어있는 것을 볼 수 있습니다. 이전 문제가 그 함수다음에 breakpoint를 잡고 실행하면 바로 flag값을 볼 수 있었는데 여기서도 혹시 몰라서 한번 해봤습니다.



한번에 flag값을 확인할 수 있었습니다. 이후로 더 연구해 보진 않았습니다.

**Reversing 4.**

이 문제는 제대로 접근한 건지는 모르겠는데 리버싱보단 crypto에 가깝다고 생각했습니다. 우선 pyc파일로 있어서 소스코드를 확인할 수 없었습니다. 이것을 리버싱해서 접근해야 하는 문제인 듯 한데 디컴파일 툴로 py로 변경했습니다.



Key를 해석해 주는 keygen, 암호화 하는 encrypt, 그리고 main함수를 확인할 수 있는데, keygen(key[:-1])은 key값의 마지막에 있는 개행을 빼면서 키를 변경해주는 역할입니다. 이를 실행해보면 key값은 MASTERKEY라는 값으로 변경되는 모습을 확인 할 수 있습니다. 이 키 값과 flag값 한 글자씩 xor연산을 통해 enc값을 만드는 모습을 볼 수 있습니다. xor연산을 다시 하면 복호화 할 수 있는 원리를 이용해서 다시 같은 키 값을 연산시켜 줍니다.



다음과 같이 encode는 decode로 바꿔주고 역순으로 연산을 하면 flag값이 나옵니다.

그런데 여기서 어떤 문제인지, flag 값이 약간 다른 값이 나옵니다. 그걸 암호화 시켜주면 원래 주어진 enc값과 살짝 다른 값이 나옵니다. 그래서 그 값을 다시 복호화 시켜주면 정상적인 flag값을 찾을 수 있었습니다. (이 부분은 잘 모르겠습니다.)