화이트해커 리그 대회

학교 : 서울디지텍고등학교

이름 : 오경제

닉네임 : Gyeongje

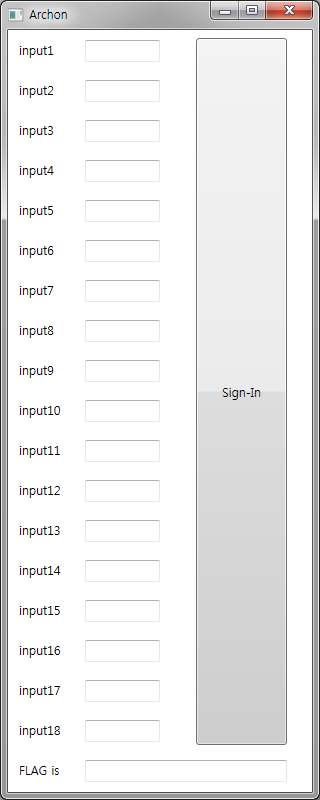
순위 : 9위

Write up 입니다.

1. Reversing 첫 번째 문제

문제 파일(Archon.exe) 은 C# 으로 만들어진 파일입니다.

그래서 저는 C# 디컴파일을 하여 코드분석을 하였습니다.



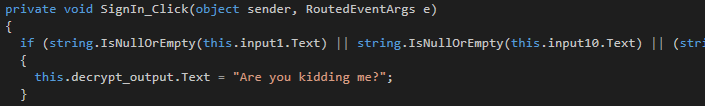
Archon 파일을 열어보면 총 18문자를 받고 Sign-In 을 하게 되는데

하나하나 문자의 조건을 통과하면 밑 text에 플래그가 나올것으로 예상했습니다.

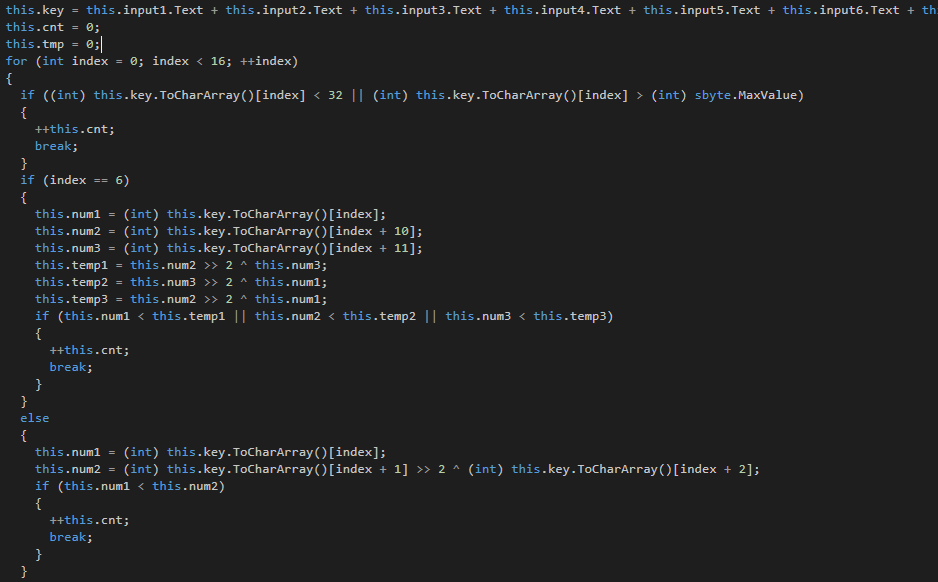
C# 디컴파일 소스코드를 보면 실행파일에 문자를 넣고 Sign\_in 을 눌렀을 때를 판단하는 함수 Sign\_In\_Click 를 보면 input 문자들의 여러 연립조건들이 보이는데 이 조건들을 하나라도 만족하지 않을시 변수 int cnt 가 1씩 증가하게 되어 실패문이 뜨게됩니다.

그러므로 cnt변수는 반드시 0이 유지되어야 합니다.

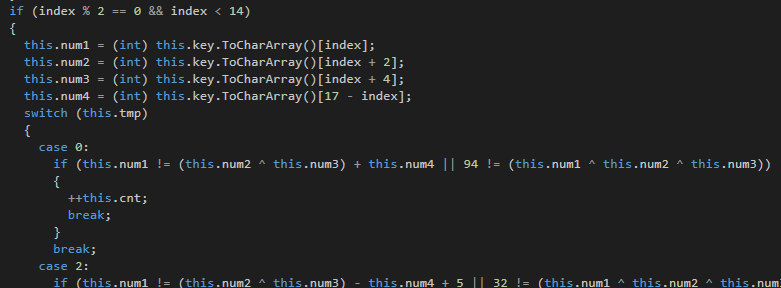
중요 코드만 뽑아서 요약하겠습니다.

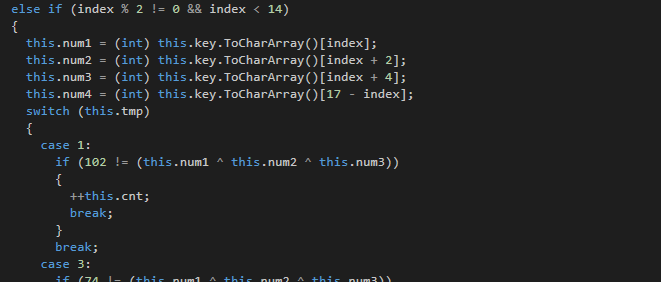


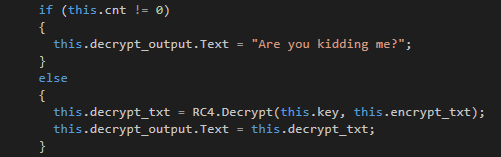
- Input1~18 text가 안비었는지 확인합니다



- 각 input 문자들을 key에 합쳐 key배열을 만든다 그 다음 이 배열을 중심으로 여러 연산을 하게 되는데 key[6] 일때만 따로 연산을 해 조건을 비교하고 있으며 나머지 부분은 else 부분으로가 쉬프트 연산과 xor연산을 통해 조건을 비교하고 있습니다







그 다음 index가 짝수, 홀수일 때를 구별하여 Switch 함수를 통해 각 case 별로 연산을 진행하고 있습니다 이 조건들은 z3을 통해 연립하여 key값을 알아낼 수 있었습니다.

그 key 값을 실행파일에 넣으면 cnt변수는 조건을 모두 만족해 0이므로 “Are you kidding me?” 실패문은 나오지 않습니다

그 후에 만족한 key 값과 상단에 있던 encrypt\_txt를 정상적인

key 값과 RC4 복호화하여 flag를 나오게 하였습니다

아래는 칼리에서 짠 위 파일 z3\_solve.py 소스입니다

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | from z3 import \*    s = Solver()  key = [BitVec("key[%d]"% i, 8) for i in range(0, 18)]    for i in range(0,16):      s.add(key[i] >= 32)      s.add(key[i] <= 127)    for i in range(0,16):      if i == 6 :          s.add(key[i] >= (key[i+10] >> 2 ^ key[i+11]))          s.add(key[i+10] >= (key[i+11] >> 2 ^ key[i]))          s.add(key[i+11] >= (key[i+10] >> 2 ^ key[i]))      else:          s.add(key[i] >= (key[i+1] >> 2 ^ key[i+2]))    s.add(key[0]==((key[2] ^ key[4]) + key[17]))  s.add((key[0] ^ key[2] ^ key[4]) == 94)  s.add((key[1] ^ key[3] ^ key[5]) == 102)  s.add(key[2] == ((key[4] ^ key[6]) - key[15] + 5))  s.add((key[2] ^ key[4] ^ key[6]) == 32)  s.add((key[3] ^ key[5] ^ key[7]) == 74)  s.add(key[4] == (((key[6] ^ key[8]) - key[13]) \* 18))  s.add((key[4] ^ key[6] ^ key[8]) == 11)  s.add((key[5] ^ key[7] ^ key[9]) == 107)  s.add(key[6] == ((key[8] ^ key[10]) + key[11] -2))  s.add((key[6] ^ key[8] ^ key[10]) == 49)  s.add((key[7] ^ key[9] ^ key[11]) == 57)  s.add(key[8] == ((key[10] ^ key[12] ^ key[9]) - 4))  s.add((key[8] ^ key[10] ^ key[12]) == 89)  s.add((key[9] ^ key[11] ^ key[13]) == 89)  s.add(key[10] == ((key[12] ^ key[14] ^ key[7]) + 19))  s.add((key[10] ^ key[12] ^ key[14]) == 114)  s.add((key[11] ^ key[13] ^ key[15]) == 41)  s.add(key[12] == ((-((key[14] ^ key[16]) & 2)) + key[5]))  s.add((key[12] ^ key[14] ^ key[16]) == 69)  s.add((key[13] ^ key[15] ^ key[17]) == 40)    s.check()  arr = s.model()    passwd = ''.join([chr(int(str(arr[key[i]]))) for i in range(0,18)])    print passwd  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |  |
| --- | --- | --- |
| 1  2 | root@Gyeongje:~/바탕화면 # python z3\_solve.py  Is\_tHa7\_tUr3\_?\_%E2 | [cs](http://colorscripter.com/info#e) |

위 key 문자열은 총 18이므로 실행파일 text form에 하나하나 넣고 Sign-In을 누르면

Flag가 나옵니다

(전에 몰랐는데 이번에 z3\_solve 소스를 짜면서 알게된 것도 있었습니다  
사실 첫번째 반복문 에서 s.add(key[i[<=127] 부분을 s.add(key[i]<128) 로 써놓고 실행시키는데 자꾸 s.check 가 unset이 되서 오류를 찾느라 고생했었는데 key[i]<=127 로 하니 정상적으로 실행이 되었습니다..  
진짜 이거 하나 때문에 코드를 한줄한줄 다보면서 오류를 고쳤는데 아스키코드 최대값이 127이라서 오류가 난거같았습니다. 너무 어이가 없었.. 하..]

**CATSEC{d0\_7oukN0VV\_Me1Omanc3\_??goO0od!!}**

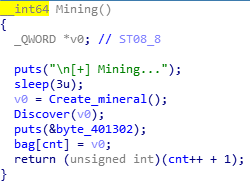
2. Pwnable 첫 번째 문제

포너블은 공부를 안해서 잘 못하는 편인데 첫 번째 문제가 꽤 많이 풀리길래 익스를 안짜도 풀 수 있는 문제인가 싶어서 접근해보았습니다. 역시 익스를 안짜도 되는 문제였습니다.

서버에 접속해서 문제를 보면 일단 메뉴 창이 뜨는데

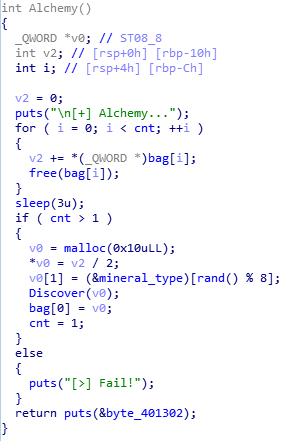
1. Mining  
2. Selling  
3. Alchmy  
4. Exit  
>>

채굴, 판매, 연금술? 총 3가지 메뉴가 있었는데 약간의 게싱과 꼼수를 이용해   
문제를 풀었습니다



먼저 채굴 함수를 보면 미네랄을 생성해서 v0 변수에 넣고 값을 출력하고 bag[cnt] 에 값을 넣고 cnt값을 ++해주고 return 해줍니다.

cnt변수가 수상해 보였습니다.



연금술 코드를 보면 if문이 하나 보이는데 리버싱을 주로 공부하는 저는 if문을 보면 어떤 상황에서 이 if문을 만족해야 하고 어떤 상황에서 이 if문을 패스해야 하는지 분석하는 버릇? 습관이 있어서 자세히 보았습니다.

채굴 부분에서 수상했던 cnt값을 연금술 함수에서 쓰는 걸로 보아 이 함수를 잘 이용해야 Flag가 나올 거라고 추측했습니다 일단은 3가지? 루트를 생각해보았는데

1. cnt = 0  
- 반복문 통과, if문 통과 Fail 출력

2. cnt = 1 (채굴을 1번 한 상태)

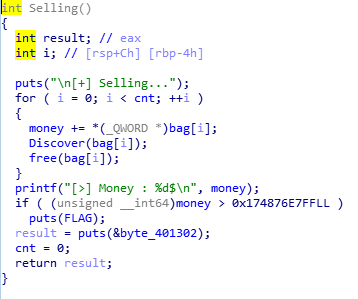
- 반복문 실행, if문 통과

3. cnt = 2 (채굴을 2번 한 상태)

- 반복문 실행, if문 실행

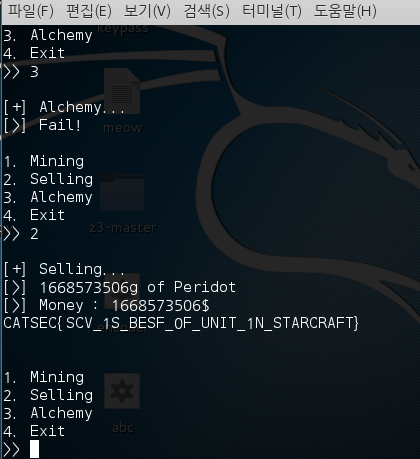
뭔가 리버싱 익숙했던 저의 손은 일단은 if문을 실행시켰기 때문에 3번을 가리켰고

1 -> 1 -> 2 -> 3 을 입력해 실행시켜봤지만 역시 Flag는 나오지 않았습니다 ㅎ..



Money를 최대한 많이 올려야 되기 때문에 채광 노가다를 하면 되지 않을까 싶었지만 미네랄을 모으는 bag배열이 \*void[20] 이었습니다. 20번 이상하면 배열 오류가 날 거 같기도 하고 0x1748866E7FF 값을 20번의 채굴으로 모을 수 있을 거 같지도 않았습니다

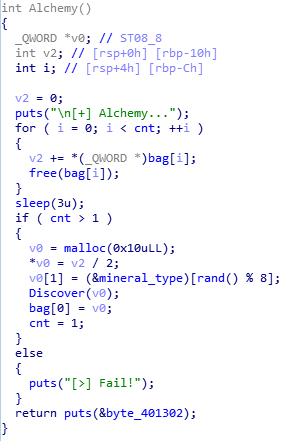
그러다가 자세히 생각은 안나지만 잘못 눌러서 연금술을 2번 입력했었던 적이 있었는데

 !? Flag가 나왔습니다

~~Flag가 나와서 웬 떡이냐 하고 인증한 후 1초의 고민도 없이 바로 다른 문제로 갈아탔습니다. 그래서 사실 Flag가 왜 나왔는지 분석하는 건 지금 write up을 쓸 때..~~

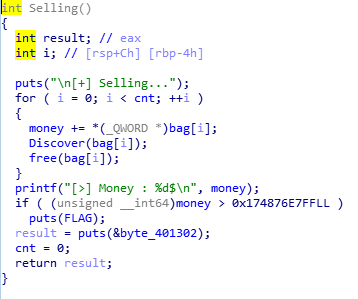
Flag가 왜 나왔는지 생각을 해보았습니다 대회 당시에 채굴을 2번이상 하고 연금술을 2번 입력했던 것 같아서 채굴을 2번 했다고 가정했습니다

채굴 -> 채굴 -> 연금술 -> 연금술 -> 판매 -> Flag 출력입니다.



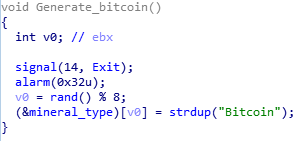
일단은 2번 채굴을 하였기 때문에 cnt 는 2고 bag[0], bag[1] 는 미네랄이 있는 상태입니다. 그렇담 첫 번째 반복문에서 v2 = bag[0]+bag[1] 이 되고 할당메모리가 free 됩니다.

그 다음 cnt가 2므로 if 조건을 만족하기 때문에 들어가서 bag[0]에 값 입력해주고 cnt = 1이 되게 해줍니다 이제 남은 연금술 한번을 더 입력해주면 cnt 가 1인 상태로   
v2 += bag[0] 를 해주고 free 된 다음에 Fail이 출력됩니다.



이제 남은 판매를 입력해주게 되면 bag 배열에 있던 값이 money에 옮겨지게 되는데 연금술에서 cnt은 1로 지정을 해줬기 때문에 bag[0] 값이 money로 들어가게 됩니다.

정상적인 미네랄 값이 들어갔다면 Flag가 나올리 없지만 뭔가 이상한 값이 bag[0] 에 들어있었던 것 같습니다 위 연금술 코드를 보면 if문속 포인터 v0 변수에 malloc 할당을 해주는데 그 다음 &mineral\_type? 코드가 있는걸 확인할 수 있습니다.



메인 함수 내 Init함수 안에 있는 Generate\_bitcoin() 함수 모습입니다.

아마 밑에 있는 strdup함수 때문 인거 같아 어떤 역할을 하는지 찾아보았는데

복사할 문자열 크기에 맞는 메모리를 확보한 후 문자열을 복사한 후, 확보한 메모리의 포인터를 반환해 줍니다.

라 설명되어 있었습니다. 일단은 Flag가 나오는 원인이 strdup 때문인거 같은데.. 잘 모르겠네요 write up 나오면 참고하면서 다시 분석해봐야 할 것 같습니다.

**CATSEC{SCV\_1S\_BESF\_0F\_UNIT\_1N\_STARCRAFT}**

마지막으로 대회 수고하셨습니다!