rev-basic-3 문제 write-up

강승민

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    char v4[256]; // [rsp+20h] [rbp-118h] BYREF
4
5    memset(v4, 0, sizeof(v4));
6    sub_1400011B0("Input : ", argv, envp);
7    sub_140001210("%256s", v4);
8    if ( (unsigned int)sub_140001000(v4) )
9      puts("Correct");
10   else
11     puts("Wrong");
12   return 0;
13 }
```

해당 문제를 ida64의 hex-ray기능을 이용해서 main함수를 보면 sub_140001000(v4) 함수의 결과에 따라 출력이 달라지는걸 볼 수 있다.

```
__int64 __fastcall sub_140001000(__int64 a1)
{
  int i; // [rsp+0h] [rbp-18h]

  for ( i = 0; (unsigned __int64)i < 0x18; ++i )
  {
    if ( byte_140003000[i] != (i ^ *(unsigned __int8 *)(a1 + i)) + 2 * i )
      return 0i64;
  }
  return 1i64;
}
```

sub_140001000 함수의 내부를 보면 어떤 배열의 길이만큼 반복문을 돌면서 (i^(입력 문자))+i*2를 하여 특정 배열과 비교한다.

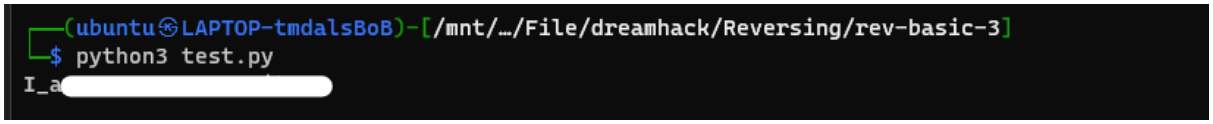
```
.data:0000000140003000 ; unsigned __int8 byte_140003000[32]
.data:0000000140003000 byte_140003000  db 49h, 60h, 67h, 74h, 63h, 67h, 42h, 66h, 80h, 78h, 2 dup(69h)
.data:0000000140003000                                         ; DATA XREF: sub_140001000+28↑o
.data:0000000140003000                 db 7Bh, 99h, 6Dh, 88h, 68h, 94h, 9Fh, 8Dh, 4Dh, 0A5h, 9Dh
.data:0000000140003000                 db 45h, 8 dup(0)
```

이를 반대로 코딩하기위해 배열을 확인하면 위 사진과 같고 xor연산의 반대는 xor연산으로 같기 때문에 ((배열-i*2)^i)를 하면 될 것으로 보인다.

```
temp = [0x49                                                                                    0x45]

for i in range(0x18):
    print(chr((temp[i]-2*i)^i),end="")
```

이를 파이썬으로 짜면 사진과 같고



Flag는 이와 같다. (flag는 일부로 가렸습니다.)