[DreamHack] rev-basic-3

```
__int64 sub_140001120()
{
  char v1; // [rsp+20h] [rbp-118h]

  memset(&v1, 0, 0x100ui64);
  sub_1400011B0("Input : ");
  sub_140001210("%256s", &v1);
  if ( (unsigned int)sub_140001000((__int64)&v1) )
    puts("Correct");
  else
    puts("Wrong");
  return 0i64;
}
```

메인함수다.

if문에 함수가 있어 저 함수의 결과에 따라 맞고 틀린 결과를 판단한다고 볼 수 있겠다.

함수를 확인해보면

```
signed __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;
}
```

우리가 입력한 문자열에서 24번 반복하며 문자를 하나씩 가져와서 특정한 연산을 거쳐 조건문을 통과해 0 또는 1을 반환한다.

배열에서 문자를 차례로 하나씩 뽑아와서 비교를 하는데 어떤 문자들인지 확인해보자

```
.data:0000000140003000 ; unsigned __int8 byte_140003000[32]
.data:0000000140003000 byte_140003000  db 73, 96, 103, 116, 99, 103, 66, 102, 128, 120, 2 dup(105)
.data:0000000140003000                                         ; DATA XREF: sub_140001000+28↑o
.data:0000000140003000                 db 123, 153, 109, 136, 104, 148, 159, 141, 77, 165, 157
.data:0000000140003000                 db 69, 8 dup(0)
```

다음과 같이 문자들이 들어있다. 2 dup(105)는 105가 2개 들어있다는 뜻으로 해석된다.

여기서 우리는 24개의 유의미한 숫자만 골라내면 되므로 제일 뒤의 0 8개는 빼고 나머지 값들을 신경쓴다.

조건문 속 특정 연산도 한 번 살펴보자

```
(i ^ *(unsigned __int8 *)(a1 + i)) + 2 * i )
```

입력 값에서 문자를 하나 가져와 i와 xor연산을 한다.

이후 2*i를 더해주고 비교한다.

이 연산을 역으로 해서 조건을 통과할 수 있는 문자를 추측해보는 코드를 만들어보면

```
byte = [73, 96, 103, 116, 99, 103, 66, 102, 128, 120, 105, 105, 123, 153, 109, 136, 104, 148, 159, 141, 77, 165, 157, 69]

for i in range(24):
    print("%c" %((byte[i]-(2*i))^i), end='')
```

이렇게 만들 수 있다.

값을 앞에서부터 하나씩 가져와서 2*i를 빼주고 i와 xor연산을 해준 값을 문자로 나타내서 연결해주면 된다.

```
I_am_X0_xo_Xor_eXcit1ng
```

코드를 실행하면 flag를 구할 수 있다.