Dreamhack verify Writeup

먼저 주어진 바이너리를 ida로 디컴파일했다.

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  printf("%s", off_4048);
  __isoc99_scanf("%256s", s);
  sub_165C();
  if ( (unsigned __int8)sub_160A() != 1 )
  {
    puts(off_4058);
    exit(0);
  }
  puts(off_4050);
  return 0LL;
}
```

Main 함수이다.

printf("%s", off_4048)으로 문자열 포인터 off_4048에 저장된 문자열을 출력한다.

__isoc99_scanf("%256s", s)으로 최대 256자까지 입력을 받아 변수 s에 저장한다.

호출되는 함수는 sub_165C();인데, 여기서는 어떤 작업을 하는지는 잘 모르겠다.

(unsigned __int8)sub_160A()으로 sub_160A()의 반환값이 1인지 확인한다. 반환값이 1이 아니라면 off_4058에 저장된 메시지를 출력하고 exit(0)로 프로그램을 종료한다.

프로그램이 성공적으로 진행되면 puts(off_4050);로 성공 메시지를 출력한다.

```
void sub_165C()
{
  byte_419F = 0;
}
```

sub_165C함수는 전역 변수 byte_419F를 0으로 설정하는 초기화 작업을 한다.

```
__int64 sub_160A()
{
  if ( (unsigned __int8)sub_1395() != 1 )
    return 0LL;
  if ( (unsigned __int8)sub_11D6() != 1 )
    return 0LL;
  return (unsigned __int8)sub_14B6();
}
```

Sub_160A함수는 세 개의 다른 함수 (sub_1395, sub_11D6, sub_14B6)를 순차적으로 호출하여 반환 값을 확인한다.

각각의 함수가 1을 반환해야만 최종적으로 sub_14B6()의 반환값이 반환된다.

하나라도 1을 반환하지 않으면 이 함수는 0을 반환한다.

```
_BOOL8 sub_1395()
{
  return strlen(s) == 43;
}
```

Sub_1395함수이다.

```
__int64 sub_11D6()
{
  int v1; // [rsp+8h] [rbp-18h]
  int n; // [rsp+Ch] [rbp-14h]
  int m; // [rsp+10h] [rbp-10h]
  int k; // [rsp+14h] [rbp-Ch]
  int j; // [rsp+18h] [rbp-8h]
  int i; // [rsp+1Ch] [rbp-4h]

  for ( i = 0; i <= 2; ++i )
  {
    if ( off_4060[i] != s[i] )
      return 0LL;
  }
  if ( byte_40AB != 45 )
    return 0LL;
  if ( byte_40B6 != 45 )
    return 0LL;
  if ( byte_40C1 != 45 )
    return 0LL;
  if ( byte_40CA != 125 )
    return 0LL;
  v1 = strlen(s);
  for ( j = 3; j <= 10 && j < v1; ++j )
  {
    if ( !(unsigned __int8)sub_1199((unsigned int)s[j]) )
      return 0LL;
  }
  for ( k = 12; k <= 21 && k < v1; ++k )
  {
    if ( !(unsigned __int8)sub_1199((unsigned int)s[k]) )
      return 0LL;
  }
  for ( m = 23; m <= 32 && m < v1; ++m )
  {
    if ( !(unsigned __int8)sub_1199((unsigned int)s[m]) )
      return 0LL;
  }
  for ( n = 34; n < v1 - 1; ++n )
  {
    if ( !(unsigned __int8)sub_1199((unsigned int)s[n]) )
      return 0LL;
  }
  return 1LL;
}
```

Sub_11D6함수이다.

1. 초기 조건 검증 (index 0~2)

문자열 s의 첫 3문자(s[0], s[1], s[2])가 배열 off_4060의 값과 동일해야 한다.

2. 특정 바이트 값 확인

네 개의 특정 바이트(byte_40AB, byte_40B6, byte_40C1, byte_40CA)가 각각 특정 값(45, 45, 45, 125)을 가져야 한다. 45는 아스키 코드로 '-'(하이픈), 125는 '}'를 나타낸다.

3. 문자열의 길이 확인

입력 문자열 s의 길이를 확인한다. 이후 반복문에서 사용된다.

4. 반복 조건 검증

검증 함수: (unsigned __int8)sub_1199((unsigned int)s[j])

```
_BOOL8 __fastcall sub_1199(char a1)
{
  if ( a1 <= 47 )
    return 0LL;
  if ( a1 <= 70 || a1 > 90 )
    return a1 <= 102;
  return 0LL;
}
```

sub_1199 함수는 입력된 문자(a1)가 특정 범위에 포함되는지를 검사한다. 이 함수는 문자 검증에 사용되며, 결과는 참(1) 또는 거짓(0)으로 반환된다.

문자 값이 47 이하: 입력값이 ASCII 기준으로 숫자 '0'(48)보다 작은 경우는 무조건 유효하지 않음. (예: '/'(47), '.'(46), 공백(32), 제어 문자 등.)

문자 값이 4870 또는 97102: 여기서 더 세부적으로 범위를 확인하면

- 48~57: 숫자 '0'~'9'.
- 65~70: 대문자 'A'~'F'.
- 97~102: 소문자 'a'~'f'.

문자 값이 71~90: 이 구간('G'~'Z')은 유효하지 않음.

나머지 값: 문자 값이 103 이상이거나 다른 범위에 포함되지 않으면 무효.

다시 구조적 반복 조건으로 돌아가서

Index 3~10, Index 12~21, Index 23~32 이렇게 나누어서 구간별로 문자열을 검증하고 있다.

그런데 여기서 어떻게 플래그를 얻어내야 할지는 잘 모르겠다.