괄호로 말해요

태그	리버싱
상태	진행 중

잘못된 값을 입력할 경우 바로 종료.

```
{
  __asm { cpuid }
  v5 = _RBX;
  if ( (_RBX & 0x200) != 0 )
    dword_1400056FC = v18 | 2;
}
dword_140005020 = 1;
dword_140005024 = 2;
if ( (v14 & 0x100000) != 0 )
{
  dword_140005020 = 2;
  dword_140005024 = 6;
  if ( (v14 & 0x8000000) != 0 && (v14 & 0x10000000) != 0 )
  {
    __asm { xgetbv }
    v25 = _RAX;
    if ( (_RAX & 6) == 6 )
    {
      v23 = dword_140005024 | 8;
      dword_140005020 = 3;
      dword_140005024 |= 8u;
      if ( (v5 & 0x20) != 0 )
      {
```

__isa_avaliable_init

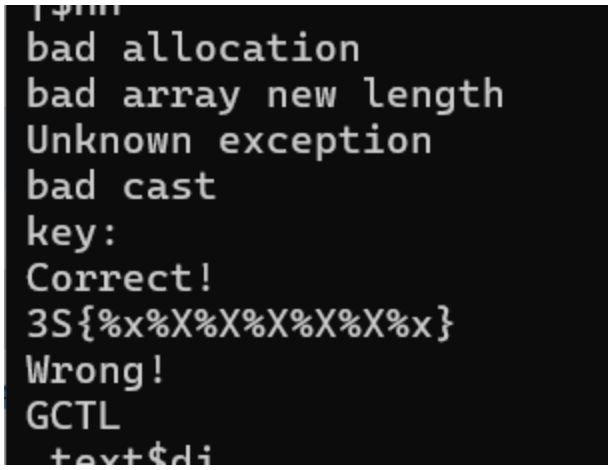
척 봐도 몹시 복잡하다.

0x가 16진수라는 뜻이고, 그 뒤는 헥사값 아스키 변환하면 되나?

기왕 exe 파일을 다운받음 김에 pe 분석기도 한번 돌려봤다.

property	value	valu
section	section[0]	secti
name	.text	.rdat
footprint > sha256	46D455196E744A5AF58E2...	178
entropy	5.892	4.16
file-ratio (94.44%)	38.89 %	41.6
raw-address (begin)	0x00000400	0x00
raw-address (end)	0x00002000	0x00
raw-size (17408 bytes)	0x00001C00 (7168 bytes)	0x00
virtual-address	0x00001000	0x00
virtual-size (17081 bytes)	0x00001AA3 (6819 bytes)	0x00

저 .text 영역에 유용한 정보가 많다고 들어본 적이 있는 것 같음.



stirngs로 문자열을 추출하자 다음과 같이 분기점이 되는 문구들과 플래그 형식이 나왔다.

```
  }
  if ( v6 == 36 && v10 == 60 && v16 == 9 && v5 == 34 )
  {
    puts("Correct!");
    sub_140001020("3S{%x%X%X%X%X%X%x}");
  }
  else
  {
    puts("Wrong!");
  }
  return 0;
```

correct! 뒤의 플래그 내용물.

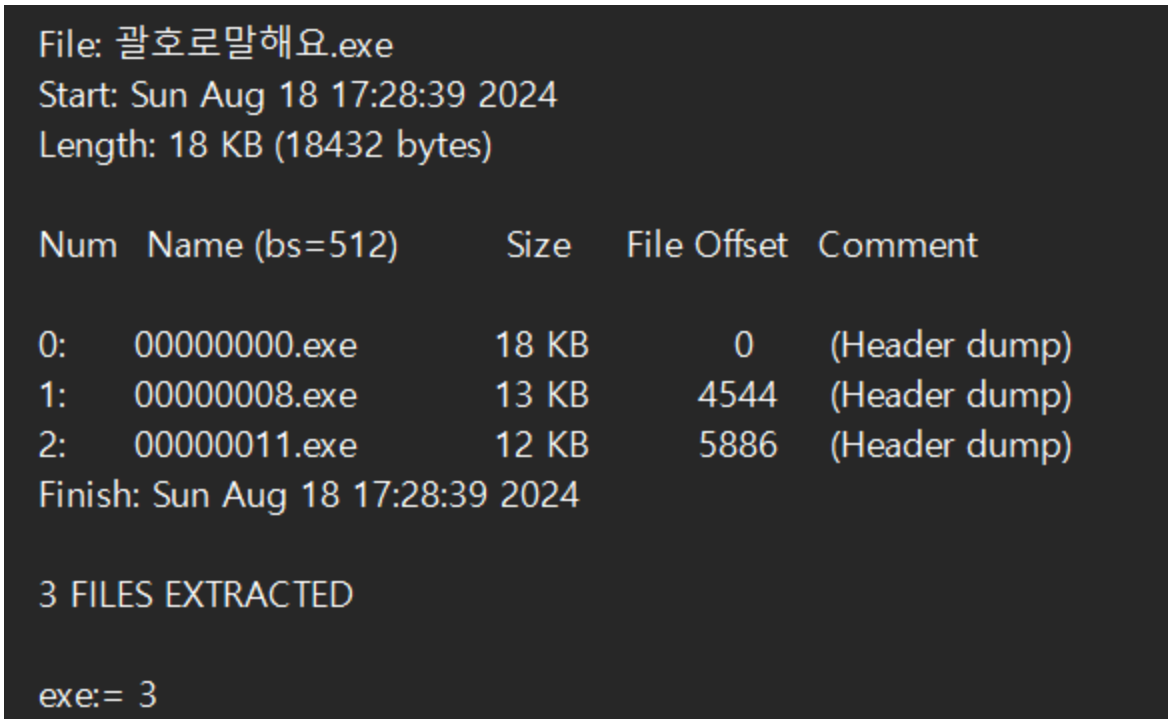
```
int sub_140001020(char *Format, ...)
{
  FILE *v2; // rbx
  unsigned __int64 *v3; // rax
  va_list va; // [rsp+58h] [rbp+10h] BYREF

  va_start(va, Format);
  v2 = _acrt_iob_func(1u);
  v3 = (unsigned __int64 *)sub_140001010();
  return _stdio_common_vfprintf(*v3, v2, Format, 0LL, va);
}
```

v3, v2, Format, 0LL(?), va...가 차례로 %x%X....를 채우나 보다.

%x는 16진수 출력문자라고 한다.

```
{
  FILE *v2; // rbx
  unsigned __int64 *v3;
  va_list va; // [rsp+58
```

파일 포인터 v2. exe파일에 내장파일이 있나보다. 그걸 추출하면 문제가 좀 쉬워질까?



첫번째 파일 제외 전부 실행불가. 첫 파일은 문제와 동일하게 작동한다.