patch

문제

문제 설명

flag를 그리는 루틴을 분석하고 가려진 flag를 보이게 해주세요.

Reference

GDI+ - Win32 apps | Microsoft Docs
Graphics Functions - Win32 apps | Microsoft Docs
File — x64dbg documentation

Translate

2 LEVEL 2

patch

reversing

5034 2586

문제 파일 받기

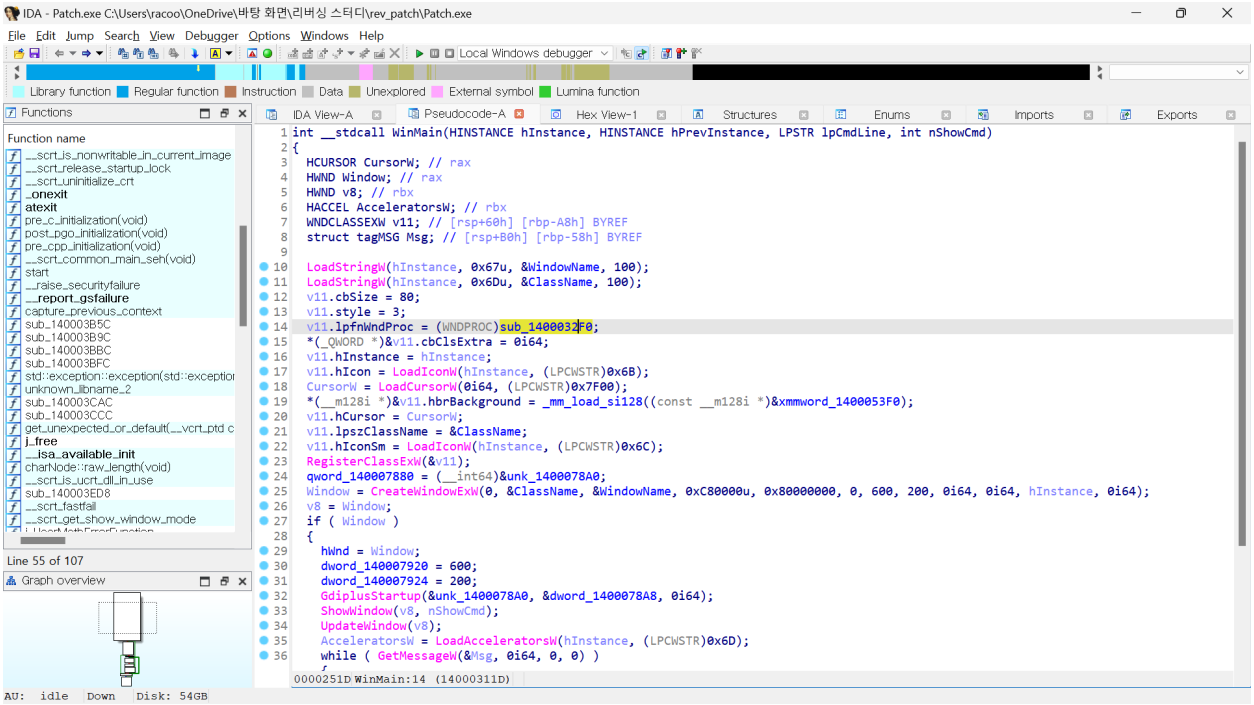
Write-up

문제 자료인 patch.exe를 열면 다음과 같다.

플래그가 가려진 상태로 있는 것을 알 수 있다.

IDA 리버싱 툴을 활용하여 해당 .exe 파일을 뜯어봄

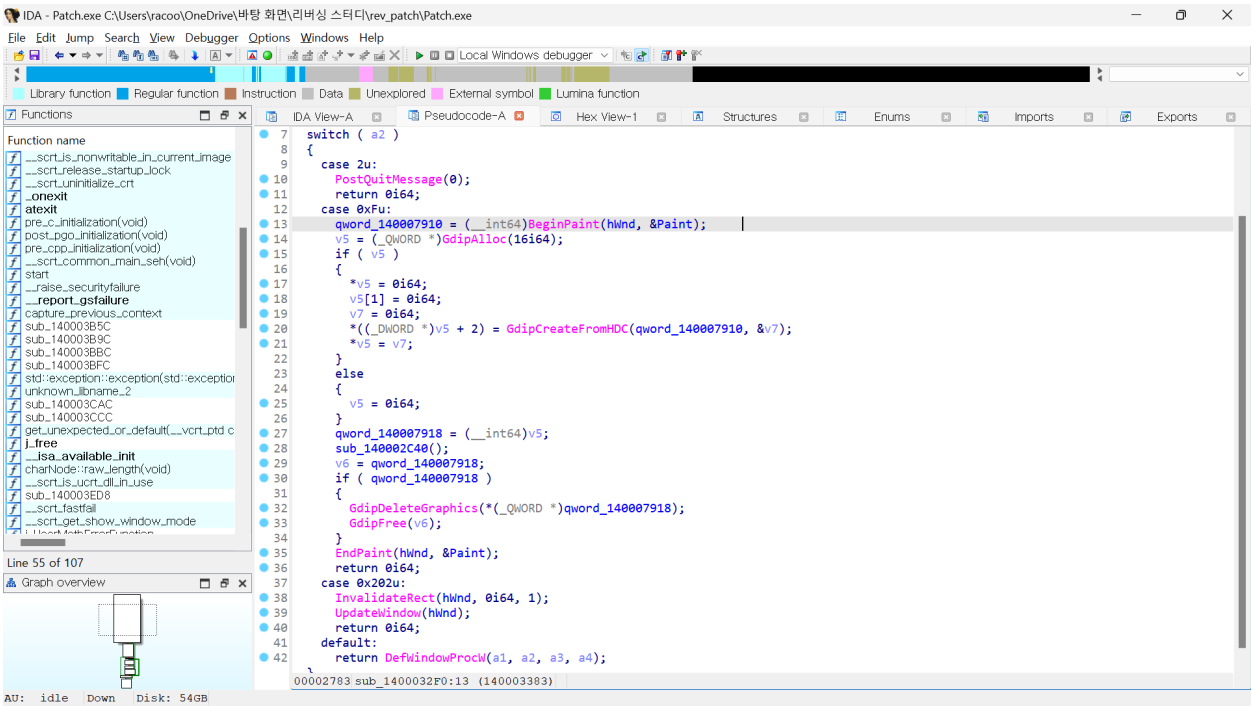
먼저 서브함수를 찾아 파악하는 것을 길로 잡았다.




```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  HCURSOR CursorW; // rax
  HWND Window; // rax
  HWND v8; // rbx
  HACCEL AcceleratorsW; // rbx
  WNDCLASSEXW v11; // [rsp+60h] [rbp-A8h] BYREF
  struct tagMSG Msg; // [rsp+B0h] [rbp-58h] BYREF

  LoadStringW(hInstance, 0x67u, &WindowName, 100);
  LoadStringW(hInstance, 0x6Du, &ClassName, 100);
  v11.cbSize = 80;
  v11.style = 3;
  v11.lpfnWndProc = (WNDPROC)sub_1400032F0;
  *(_QWORD *)&v11.cbClsExtra = 0i64;
  v11.hInstance = hInstance;
  v11.hIcon = LoadIconW(hInstance, (LPCWSTR)0x6B);
  CursorW = LoadCursorW(0i64, (LPCWSTR)0x7F00);
  *(__m128i *)&v11.hbrBackground = _mm_load_si128((const __m128i *)&xmmword_1400053F0);
  v11.hCursor = CursorW;
  v11.lpszClassName = &ClassName;
  v11.hIconSm = LoadIconW(hInstance, (LPCWSTR)0x6C);
  RegisterClassExW(&v11);
  qword_140007880 = (__int64)&unk_1400078A0;
  Window = CreateWindowExW(0, &ClassName, &WindowName, 0xC80000u, 0x80000000, 0, 600, 200, 0i64, 0i64, hInstance, 0i64);
  v8 = Window;
  if ( Window )
  {
    hWnd = Window;
    dword_140007920 = 600;
    dword_140007924 = 200;
    GdiplusStartup(&unk_1400078A0, &dword_1400078A8, 0i64);
    ShowWindow(v8, nShowCmd);
    UpdateWindow(v8);
    AcceleratorsW = LoadAcceleratorsW(hInstance, (LPCWSTR)0x6D);
    while ( GetMessageW(&Msg, 0i64, 0, 0) )
```

C코드로 변환하여 `sub_1400032F0` 코드를 찾아 뜯어보았다.




```
  switch ( a2 )
  {
    case 2u:
      PostQuitMessage(0);
      return 0i64;
    case 0xFu:
      qword_140007910 = (__int64)BeginPaint(hWnd, &Paint);
      v5 = (_QWORD *)GdipAlloc(16i64);
      if ( v5 )
      {
        *v5 = 0i64;
        v5[1] = 0i64;
        v7 = 0i64;
        *((_DWORD *)v5 + 2) = GdipCreateFromHDC(qword_140007910, &v7);
        *v5 = v7;
      }
      else
      {
        v5 = 0i64;
      }
      qword_140007918 = (__int64)v5;
      sub_140002C40();
      v6 = qword_140007918;
      if ( qword_140007918 )
      {
        GdipDeleteGraphics(*(_QWORD *)qword_140007918);
        GdipFree(v6);
      }
      EndPaint(hWnd, &Paint);
      return 0i64;
    case 0x202u:
      InvalidateRect(hWnd, 0i64, 1);
      UpdateWindow(hWnd);
      return 0i64;
    default:
      return DefWindowProcW(a1, a2, a3, a4);
  }
```

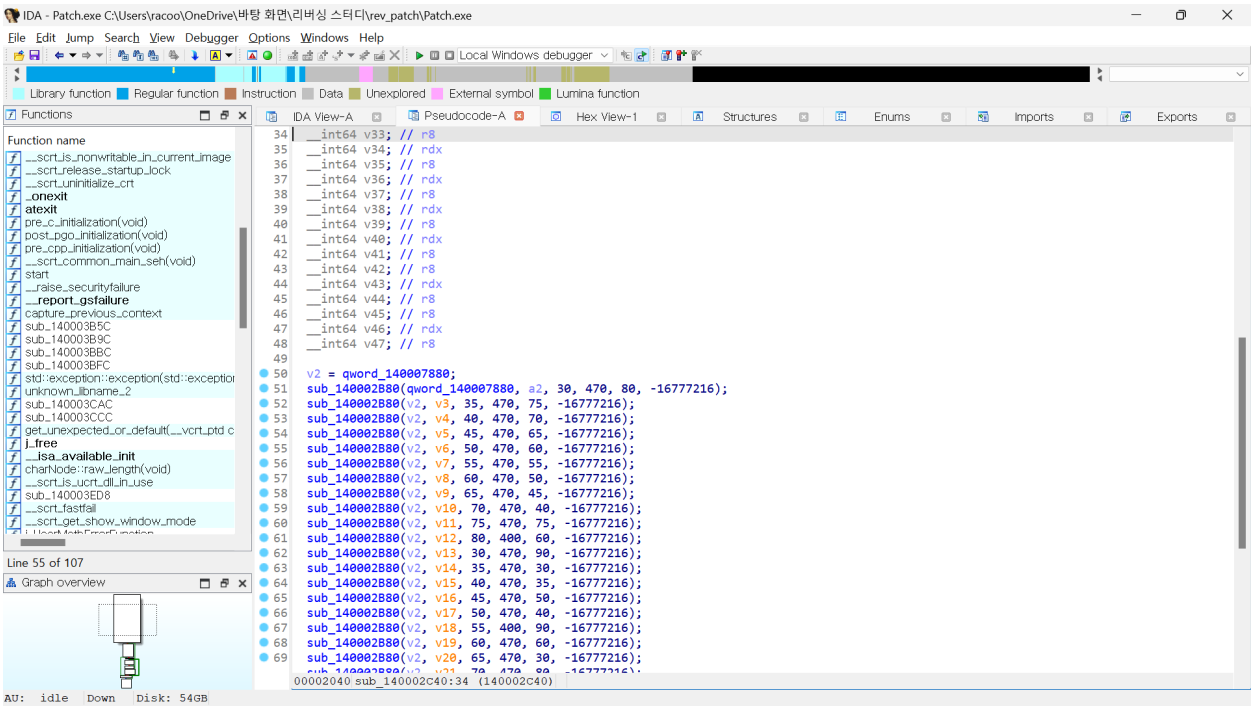
`sub_1400032F0`는 스위치 문으로 구성되어있었다.

서브함수이기에 어떤 동작을 하는 코드가 있을 것으로 보고 분석했다.

`BeginPaint`와 `EndPaint`라는 함수가 있다는 것을 알아냈다

이 둘은 api 함수로 그래픽 처리를 위해 사용된다는 것을 알았다.

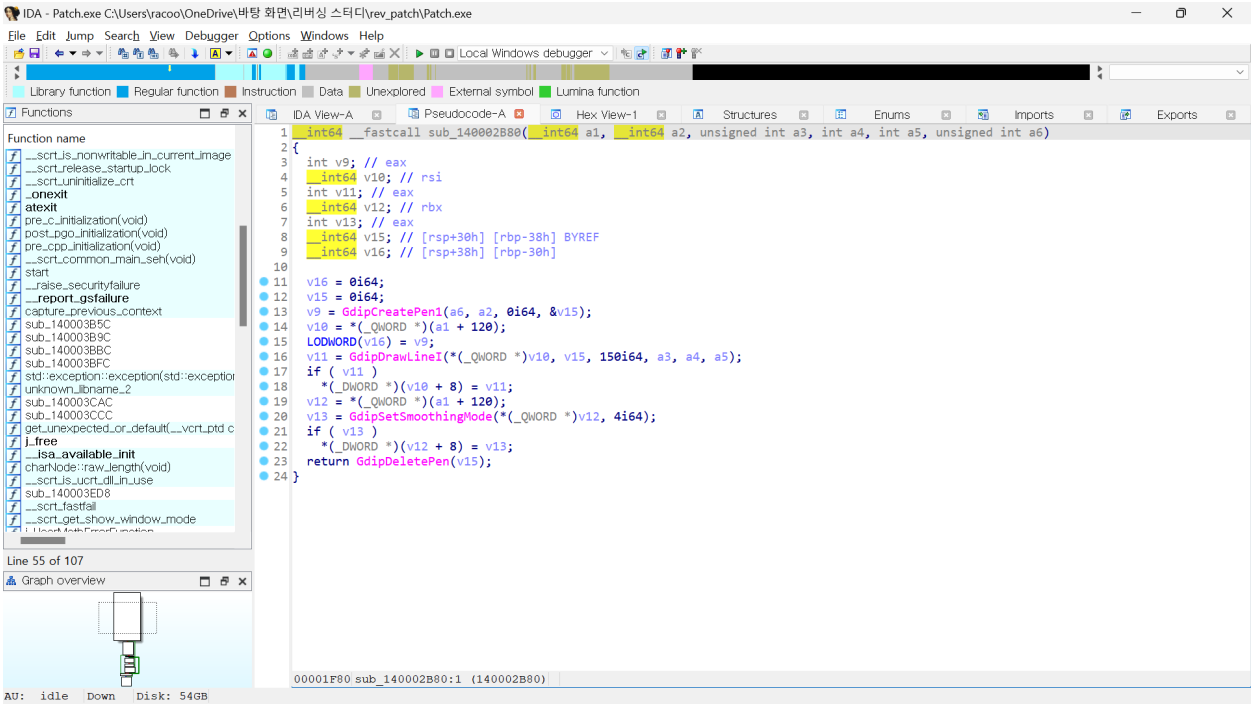
여기에 할당된 `sub_140002C40` 코드를 뜯어봤다.



누가봐도 수상한 함수 더미들을 볼 수 있고, 동일한 서브함수로 이루어져 있기에

이 서브함수인 `sub_140002B80` 코드도 뜯어봤다.

먼저 `sub_140002C40`코드는 `sub_140002B80` 코드에 인자 6개를 담아 출력하는 구조임을 알 수 있다.

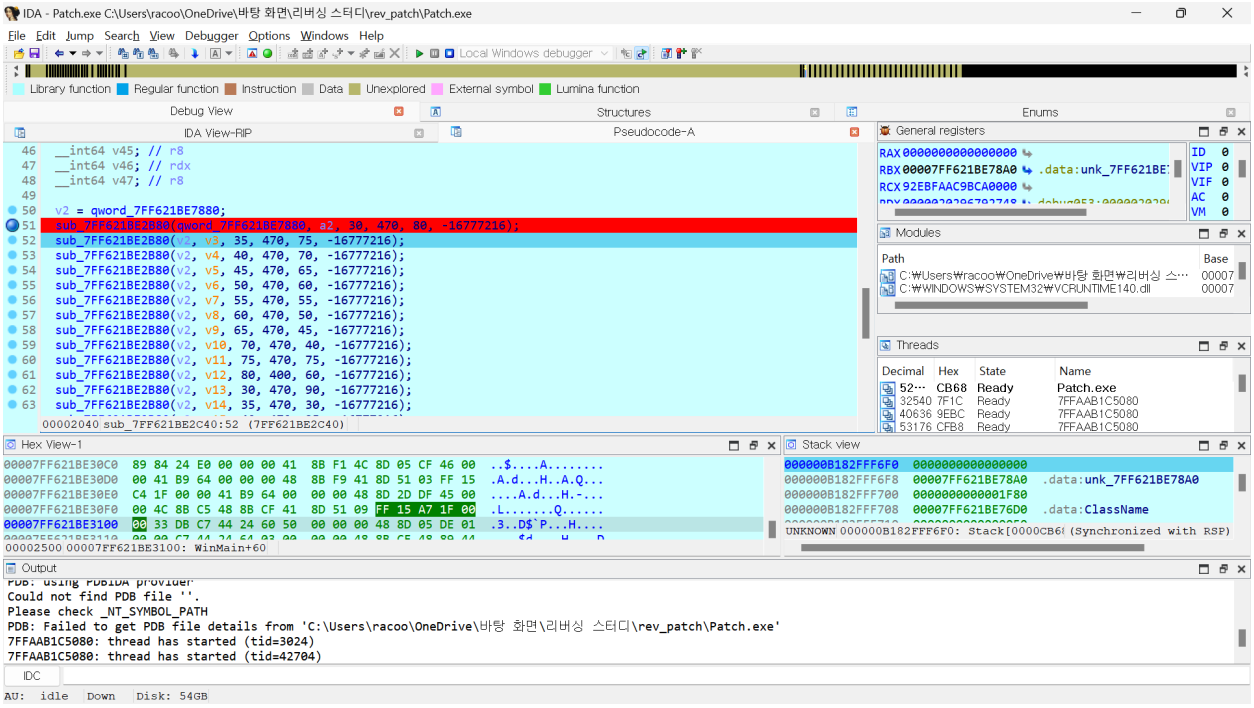


`sub_140002B80` 코드를 뜯어보니 이는 확실해졌다.

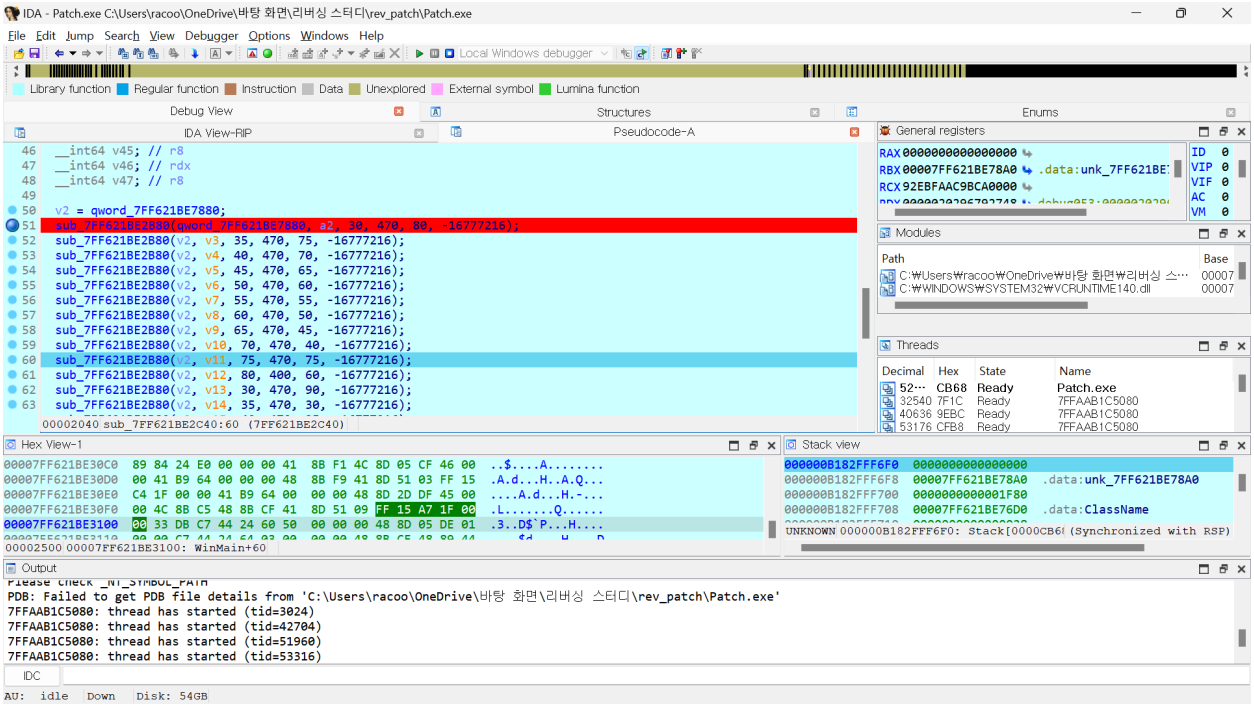
CreatePen, DrawLine, DeletePen, SmoothingMode 등 처음 보는 함수였지만

이 함수들이 좌표와 길이 같은 것을 받으면 선을 그려주는 방식임을 알 수 있었다.

`sub_140002B80`의 동적분석을 위해 해당 줄에 브레이크 포인트를 걸고 하나 하나 진행시켜보았다.



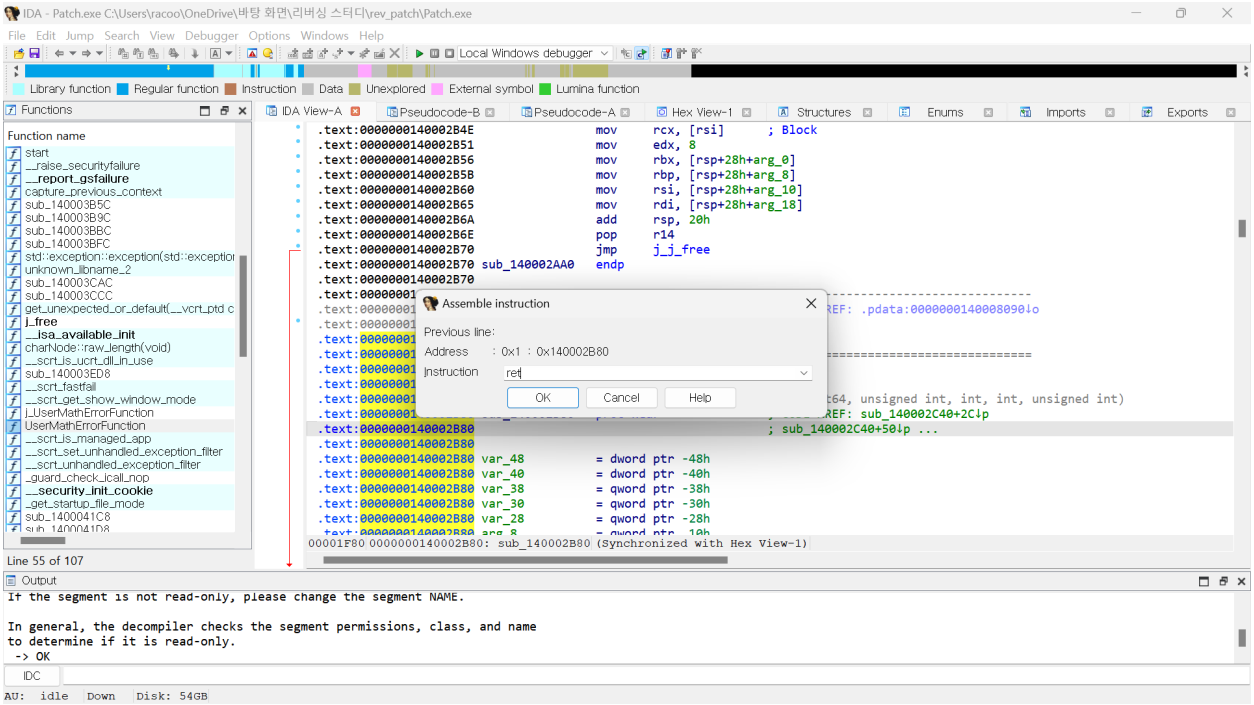
그 결과 선 하나 그어진 것을 볼 수 있다.



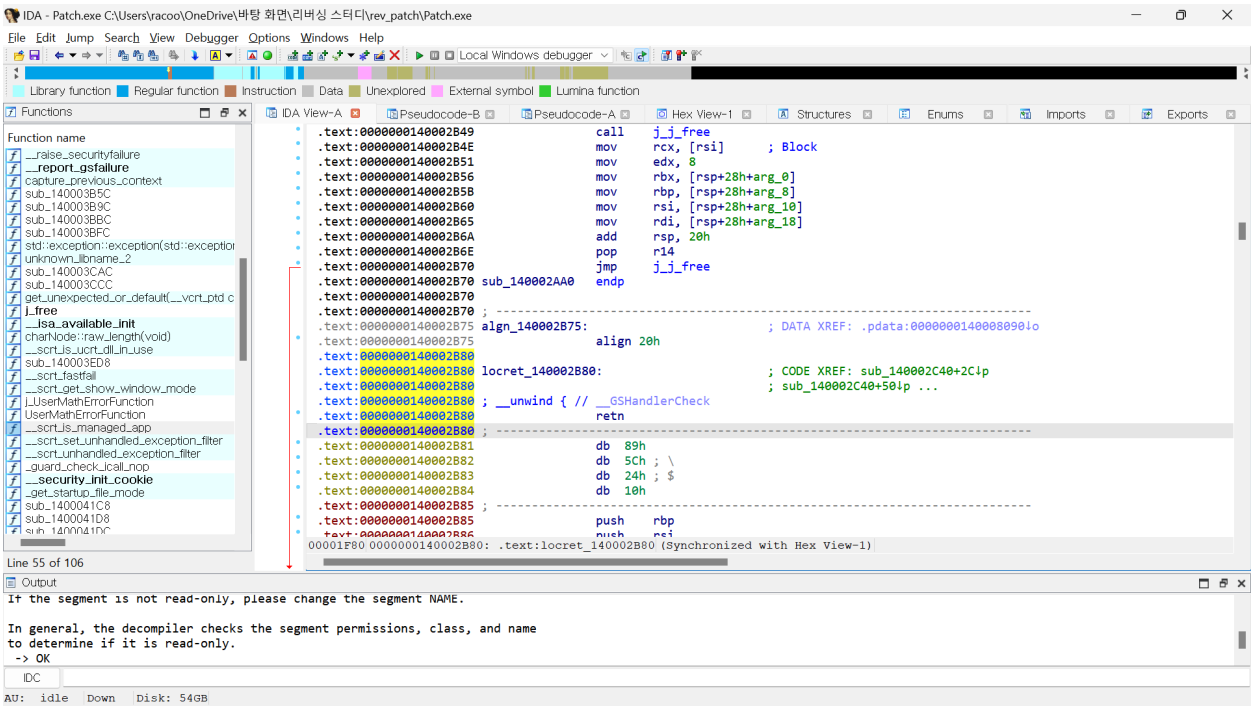
몇 번 더 진행한 결과 선은 계속해서 늘어나고 이를 통해

`sub_140002B80` 코드는 플래그를 가리는 선을 만들어내는 코드임을 알 수 있다.

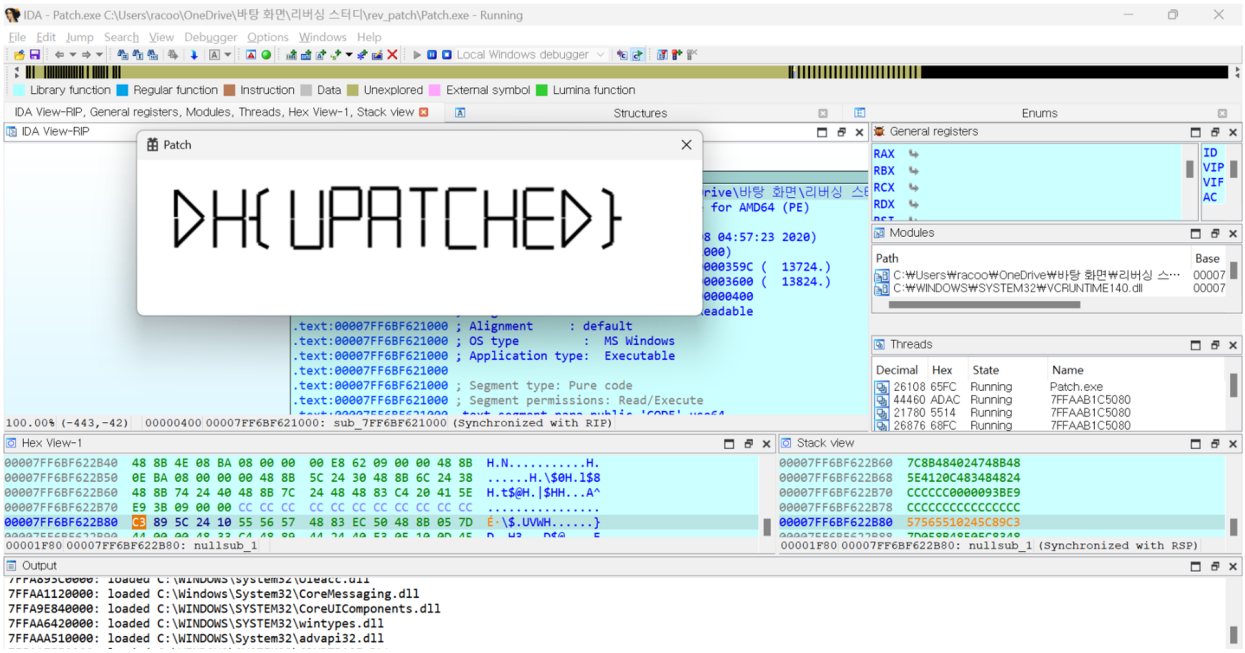
해당 함수에서 선을 그리는 작동이 일어나지 않도록 `ret` 값을 넣어 함수 패치를 해준다.



`sub_140002B80`에서 어셈블리 코드를 고쳐주고 적용시킨다.



이후 실행시키면 플래그 값을 알 수 있다.


IDA - Patch.exe C:\Users\racoo\OneDrive\바탕 화면\리버싱 스터디\rev_patch\Patch.exe - Running
Patch
DH{UPATCHED}
IDA View-RIP
Hex View-1
Stack view
Output
7FFAA1120000: loaded C:\Windows\System32\CoreMessaging.dll
7FFA9E840000: loaded C:\WINDOWS\SYSTEM32\CoreUIComponents.dll
7FFAA6420000: loaded C:\WINDOWS\SYSTEM32\wintypes.dll
7FFAAA510000: loaded C:\WINDOWS\System32\advapi32.dll