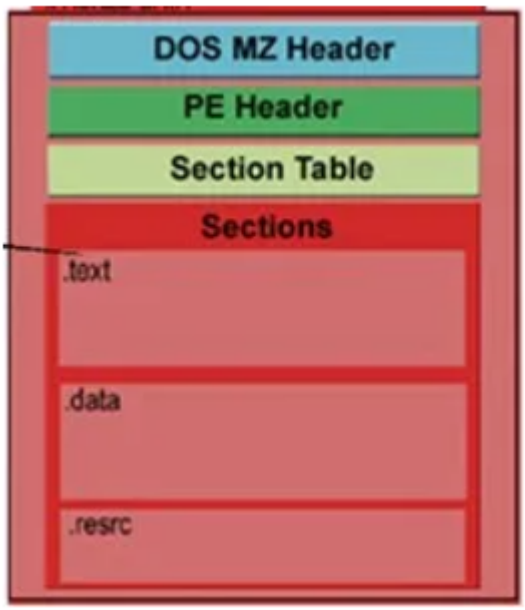
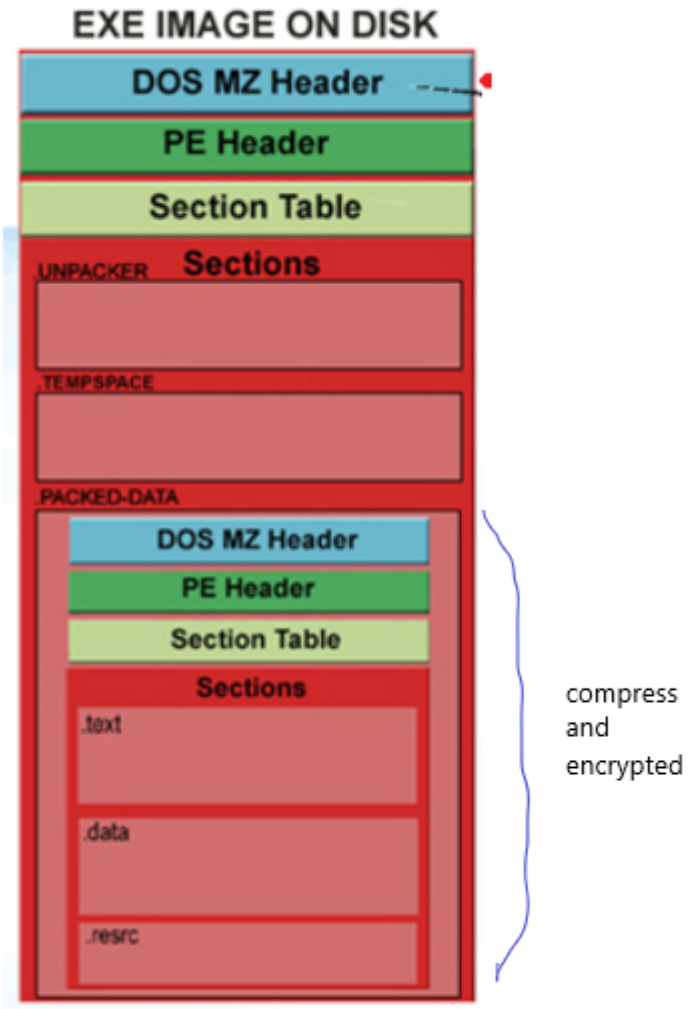
VCS_Báo cáo bài tập tuần 6 Reverse Engineering

0. Tổng quan packer

Cấu trúc file PE:

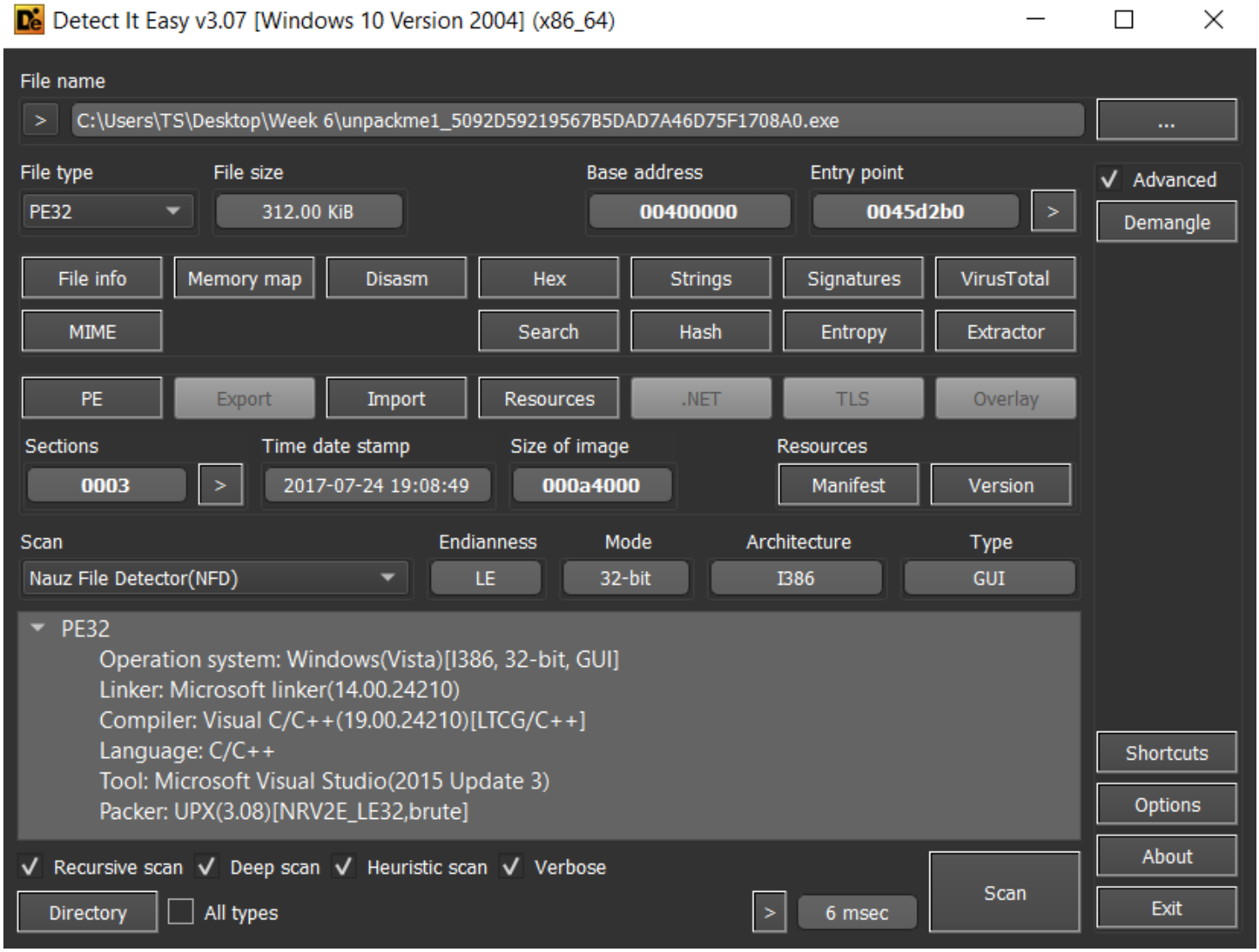


Cấu trúc file PE packed:



unpackme1_5092D59219567B5DAD7A46D75F1708A0.exe

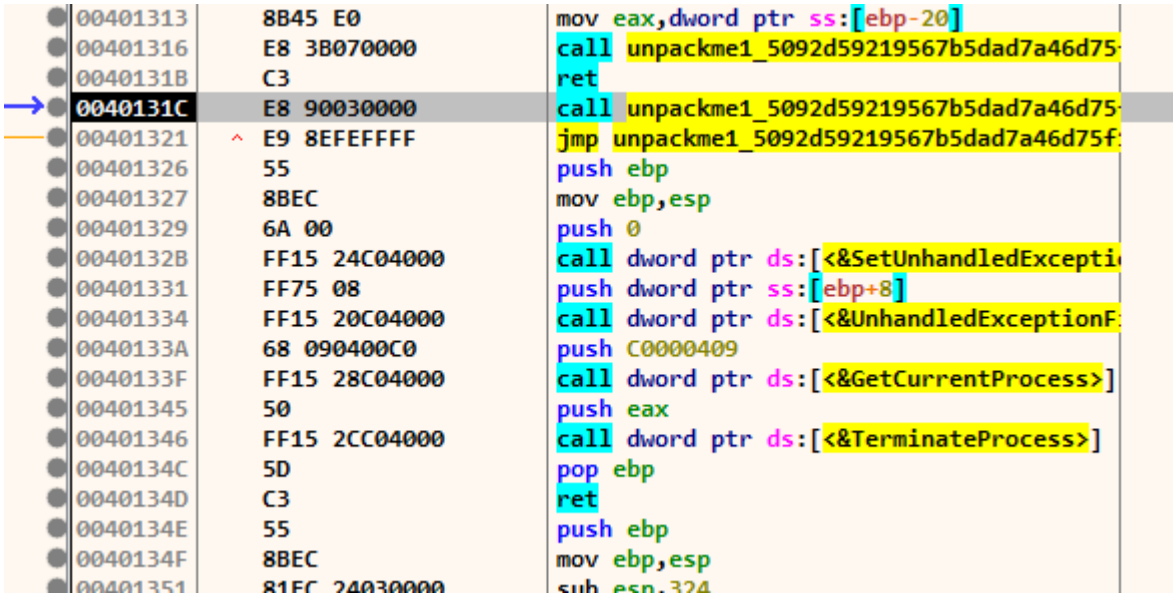
Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng UPX Packer:



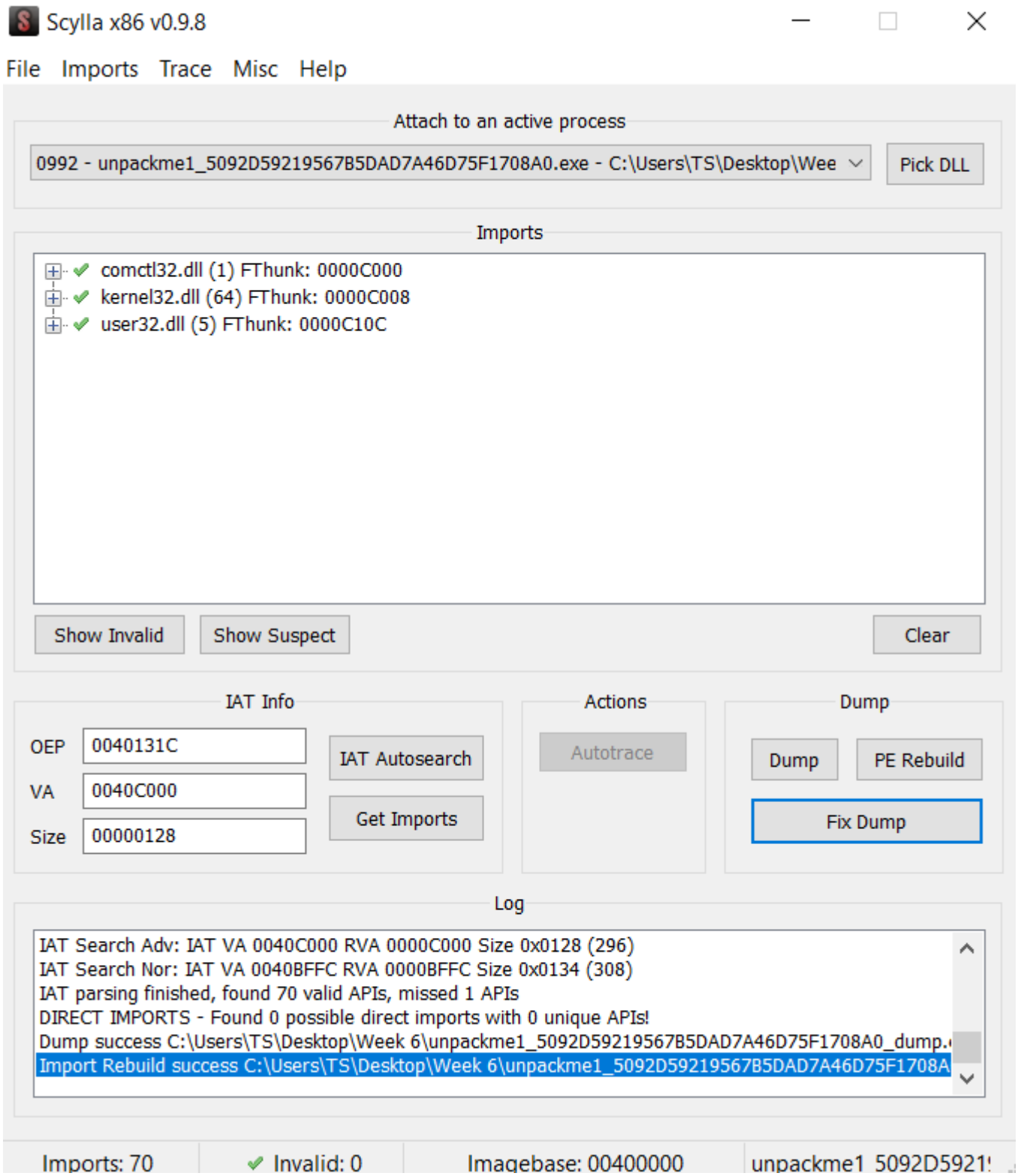
UPX hoạt động theo các bước:

1. Lưu tất cả các trạng thái register với PUSHAD
2. Giải nén tất cả các phần trong bộ nhớ
3. Sửa IAT
4. Khôi phục trạng thái register bằng POPAD
5. Chuyển đến OEP và thực thi chương trình gốc.

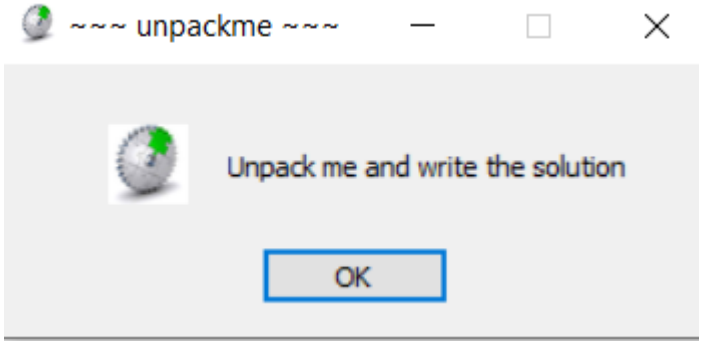
Do vậy đặt breakpoint sau lệnh POPAD và dump file ra để unpack. Thực hiện trong x32dbg:



Xóa dll bị lỗi, dump và fix dump bằng plugin scylla:

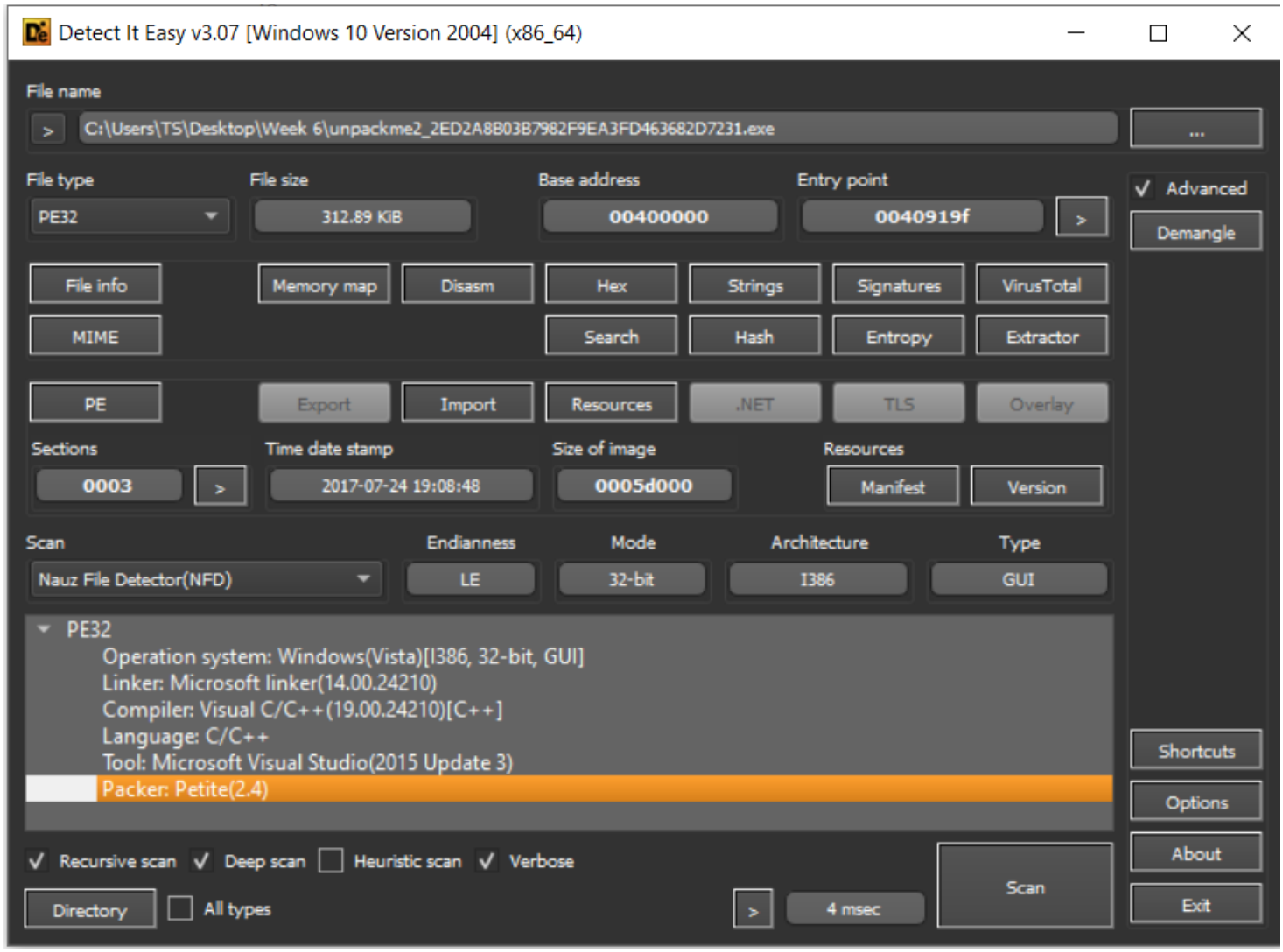


File unpack và chạy thành công:

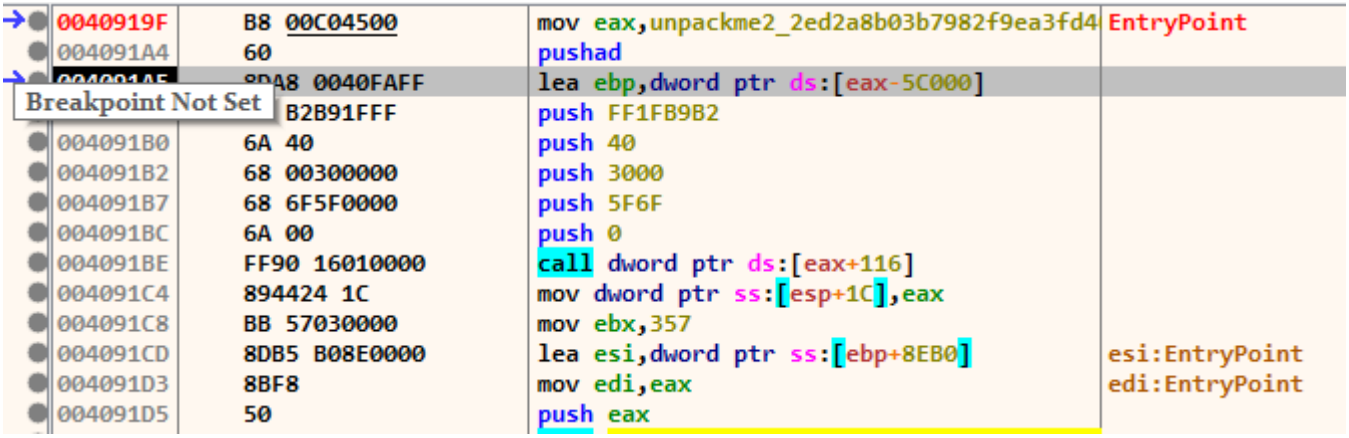


unpackme2_2ED2A8B03B7982F9EA3FD463682D7231.exe

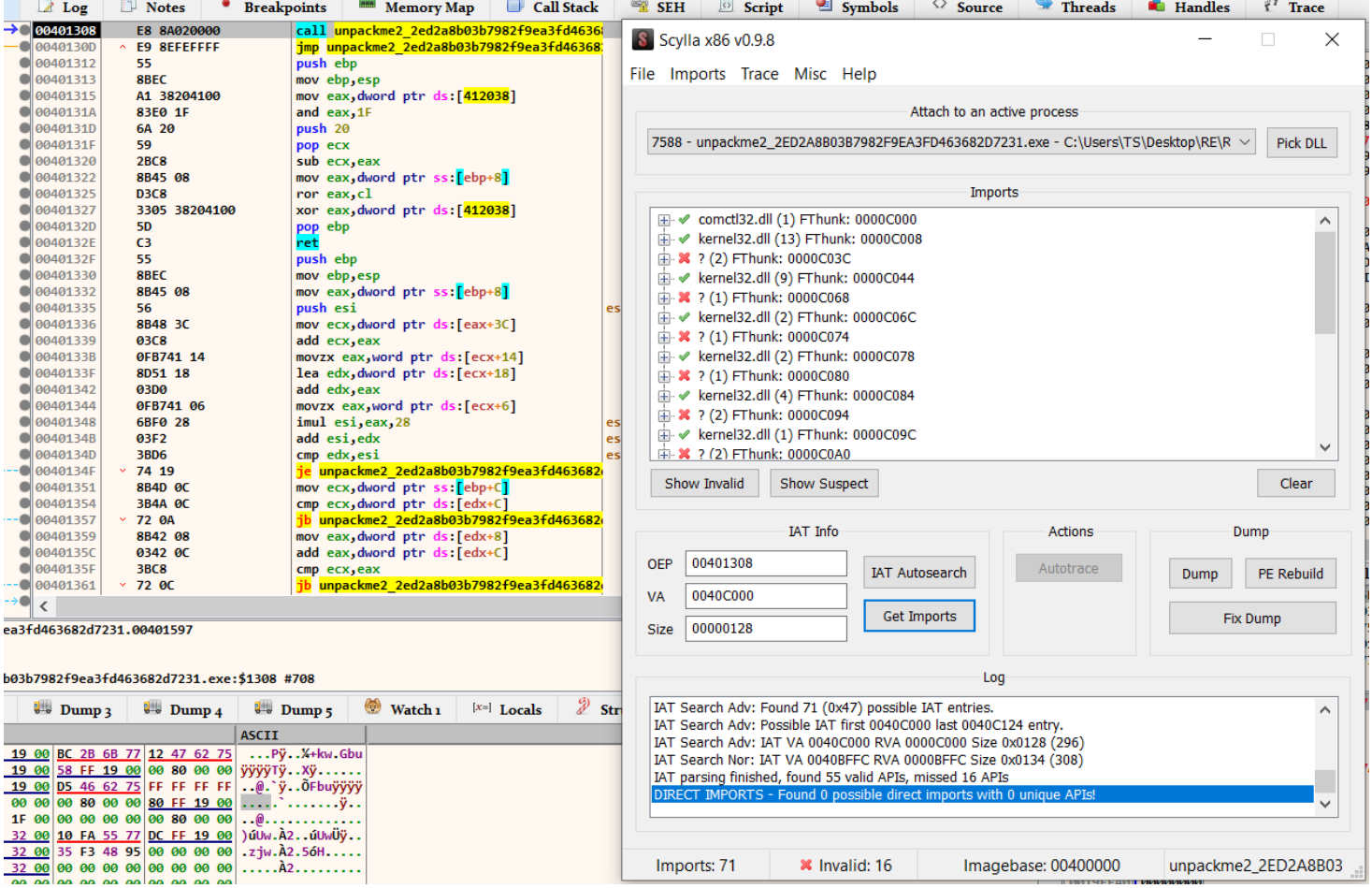
Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng Petite packer:



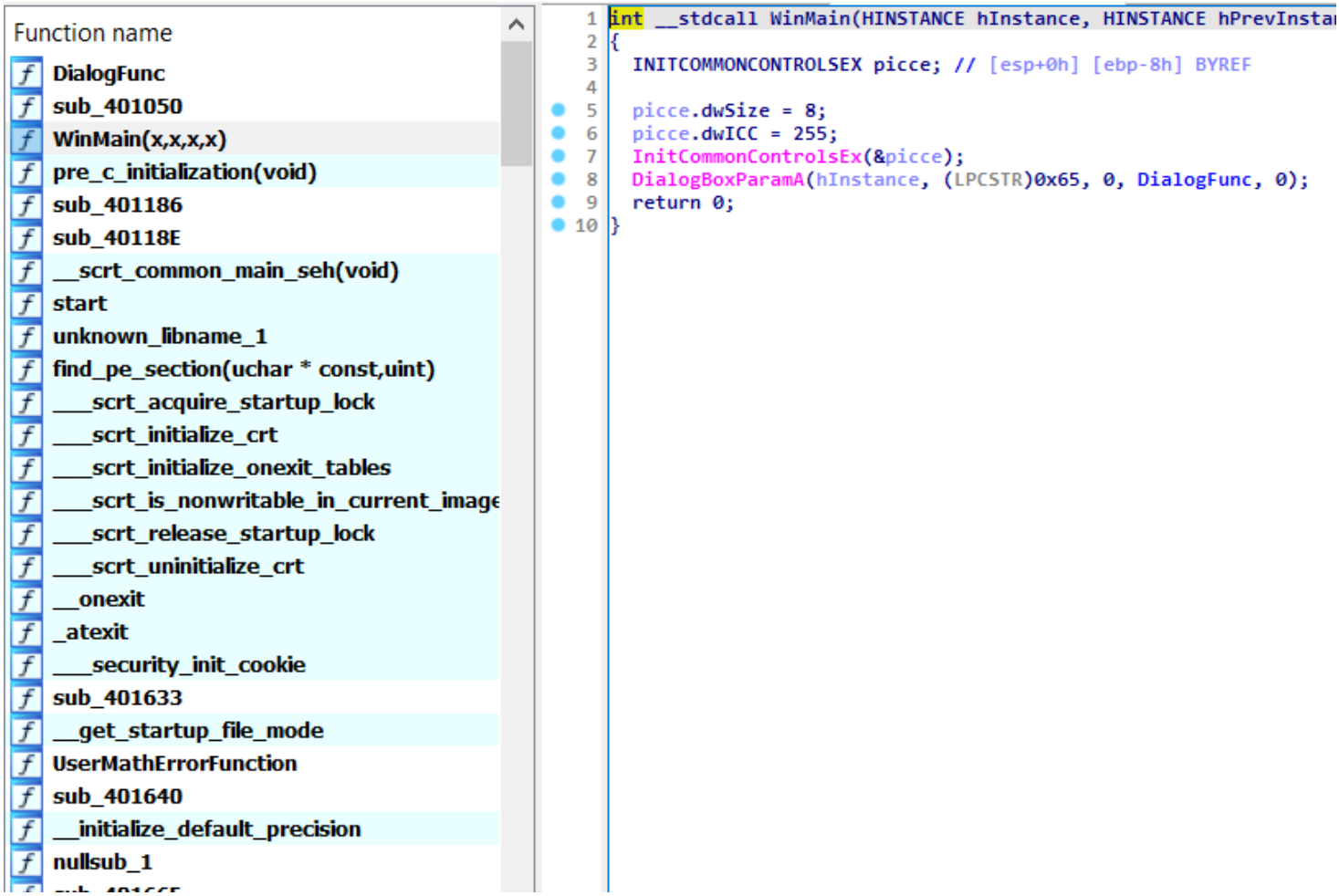
Debug file bằng x32dbg và f9 để đến EntryPoint. F7 qua và đặt hardware breakpoint ở địa chỉ ESP đang trỏ sau lệnh pushad:



execute till return qua hàm virtual alloc và tìm được OEP 0×401308, dump bằng Scylla:

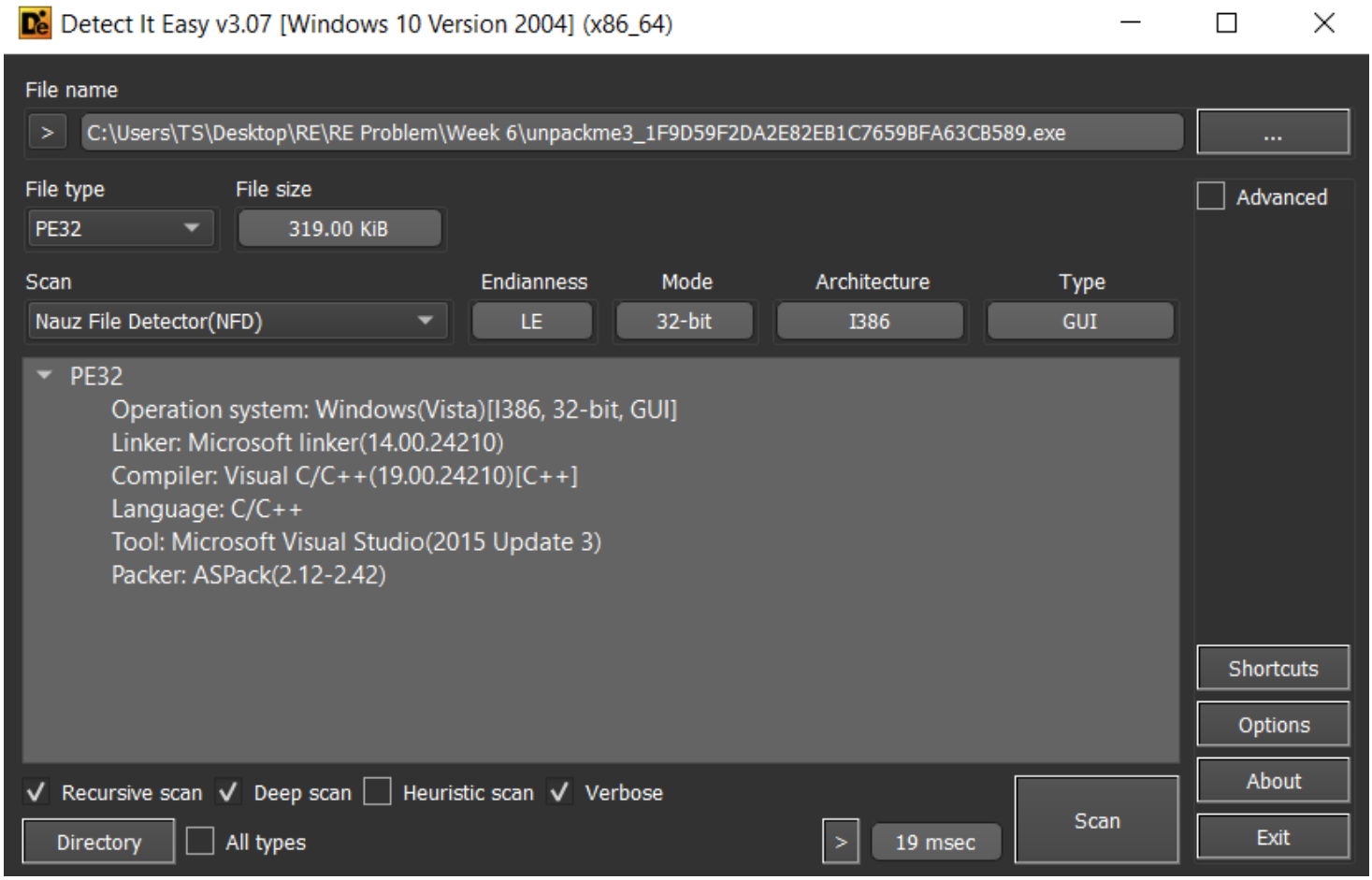


IAT bị hỏng nên file không chạy được nhưng có thể phân tích tĩnh:



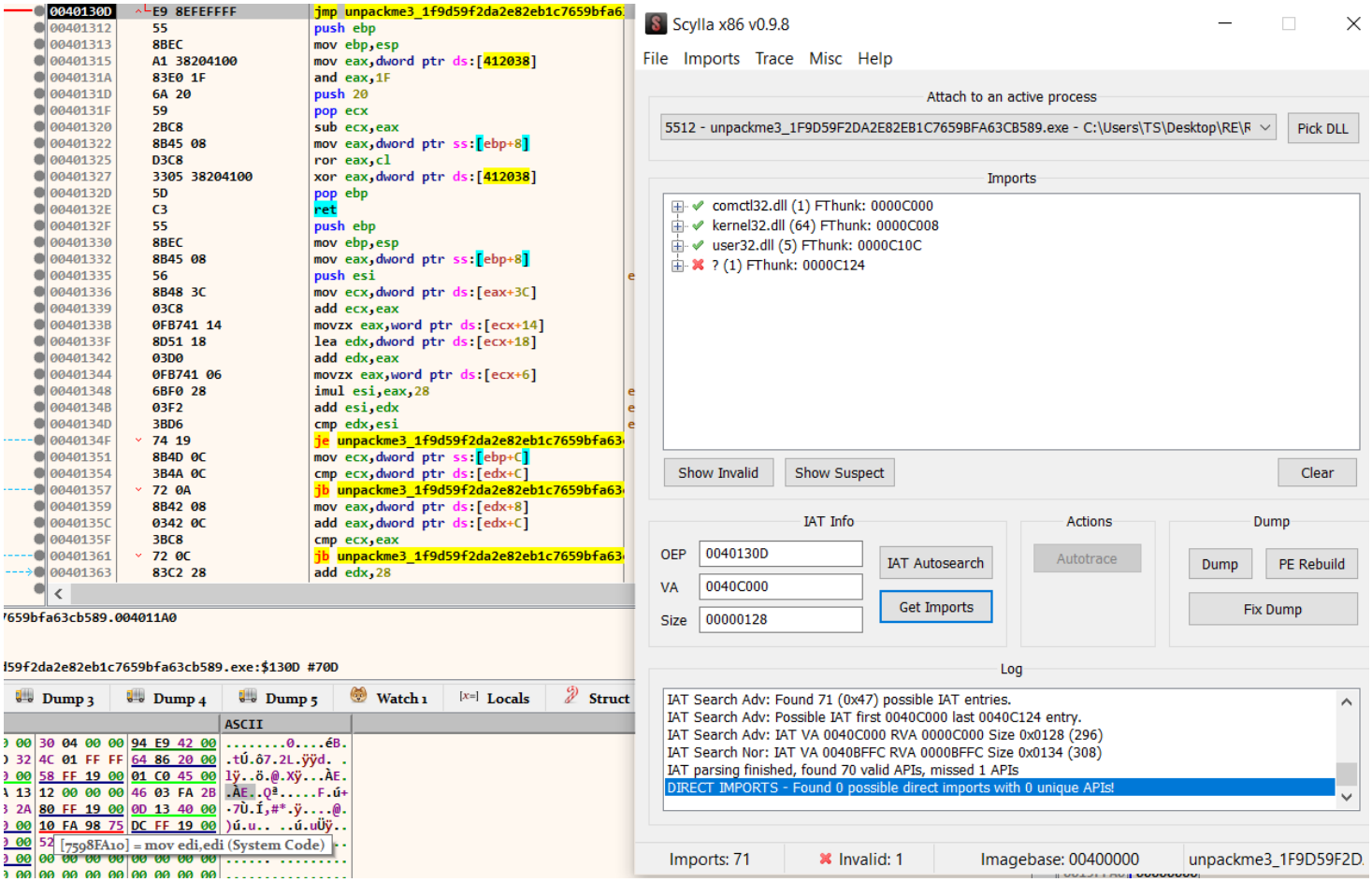
unpackme3_1F9D59F2DA2E82EB1C7659BFA63CB589.exe

Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng ASPack packer:



Debug file bằng x32dbg và f9 để đến EntryPoint. F7 qua và đặt hardware breakpoint ở địa chỉ ESP đang trỏ sau lệnh pushad.

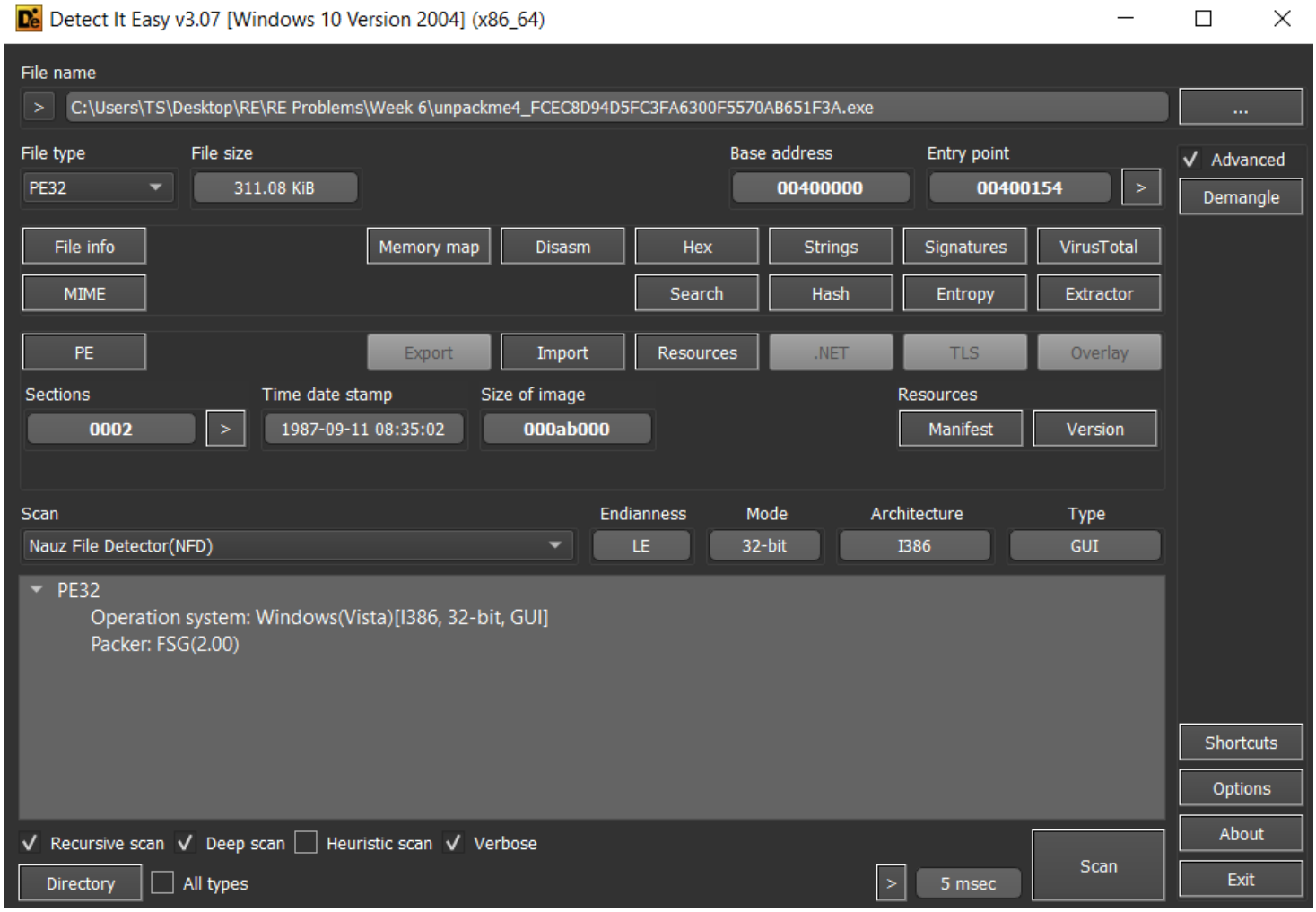
Dò đến đầu chương trình gốc của file và dump bằng Scylla(xóa import lỗi, dump và fix dump).



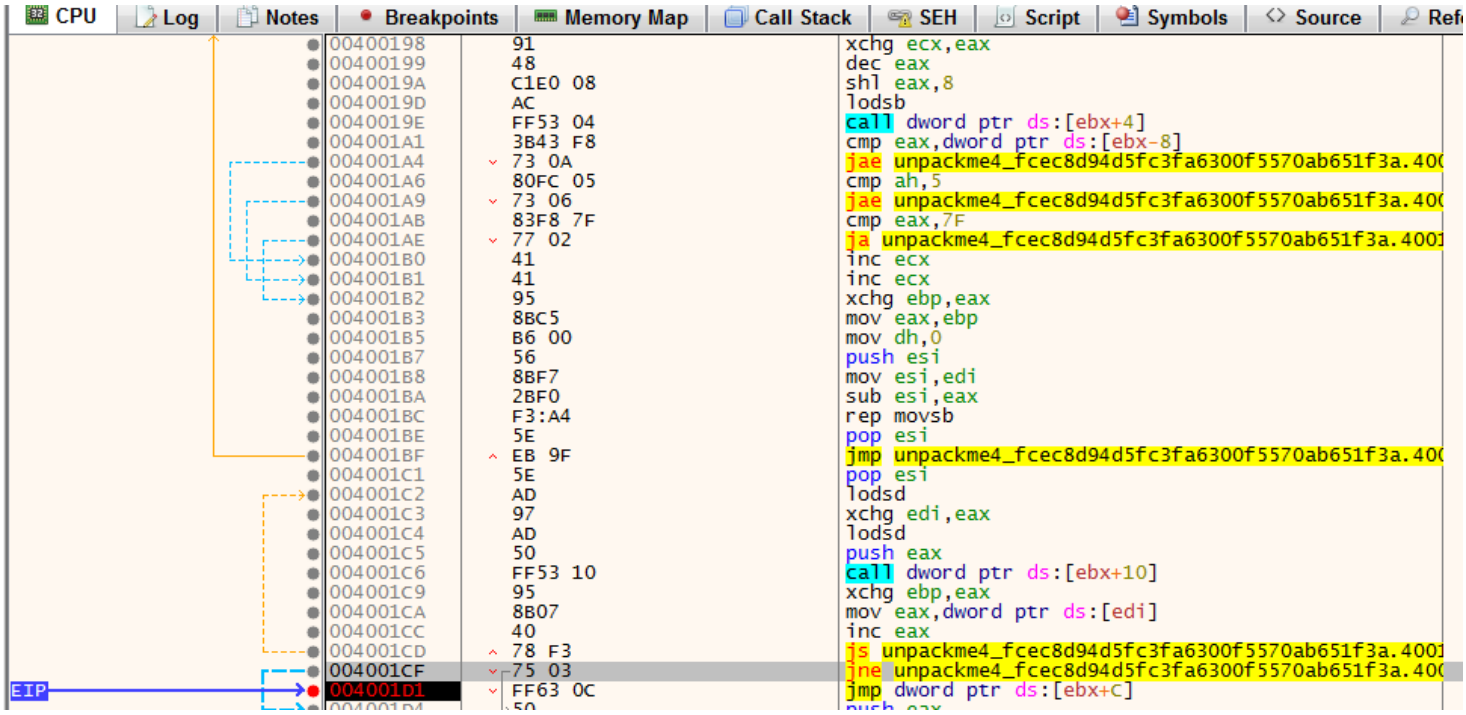
Sau đó chạy thành công chương trình gốc.

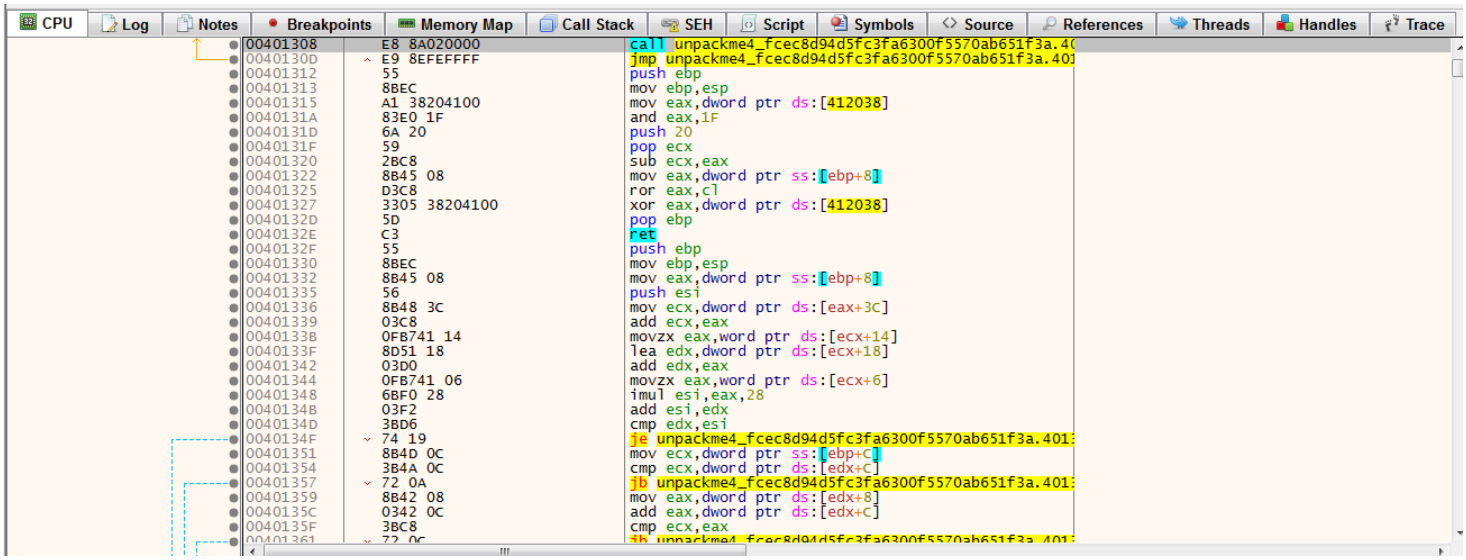
unpackme4_FCEC8D94D5FC3FA6300F5570AB651F3A.exe

Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng FSG packer:

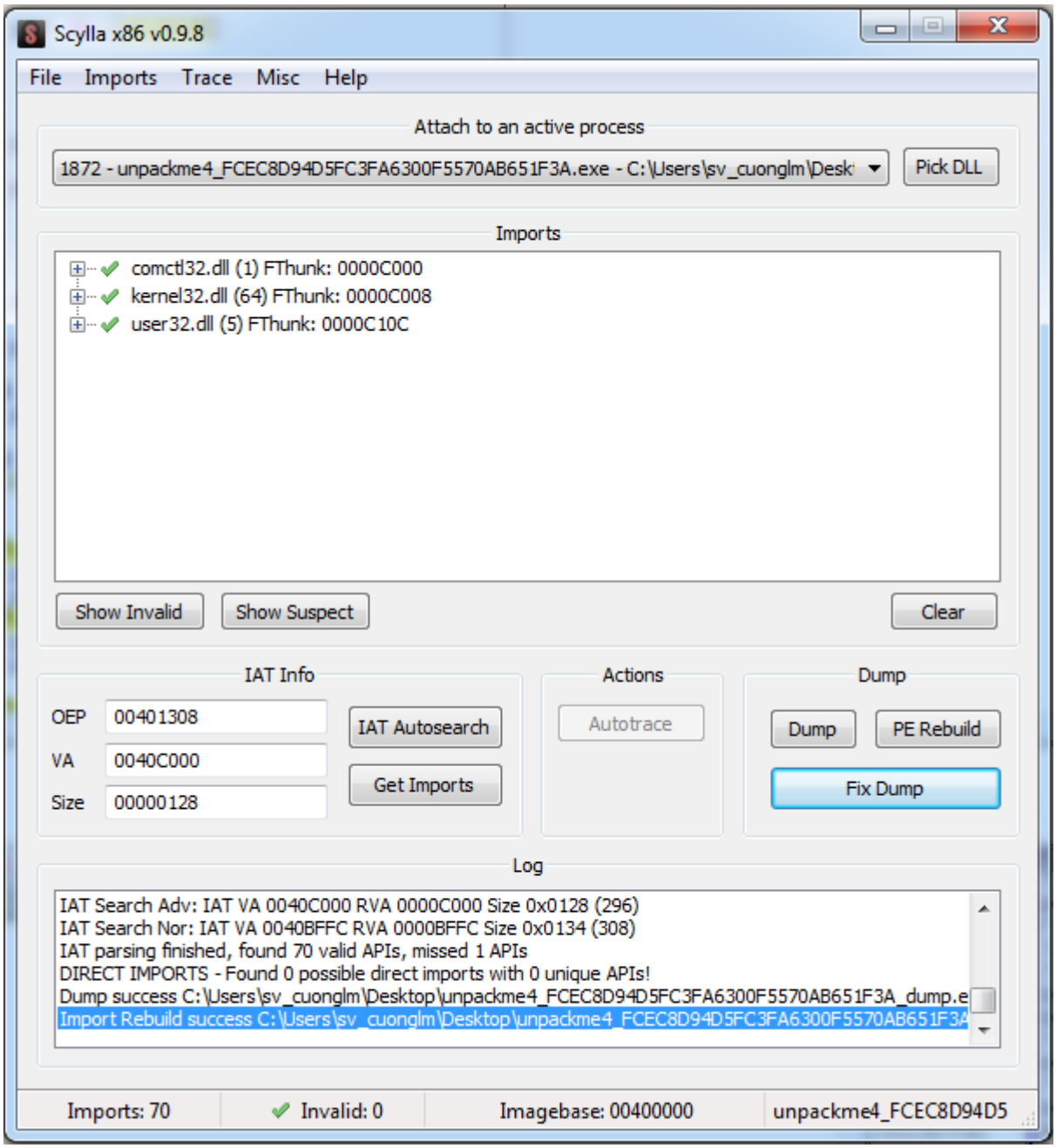


Sau entrypoint có một vòng lặp, đặt breakpoint sau vòng lặp(địa chỉ 0×4001D1) thì nhảy đến OEP:



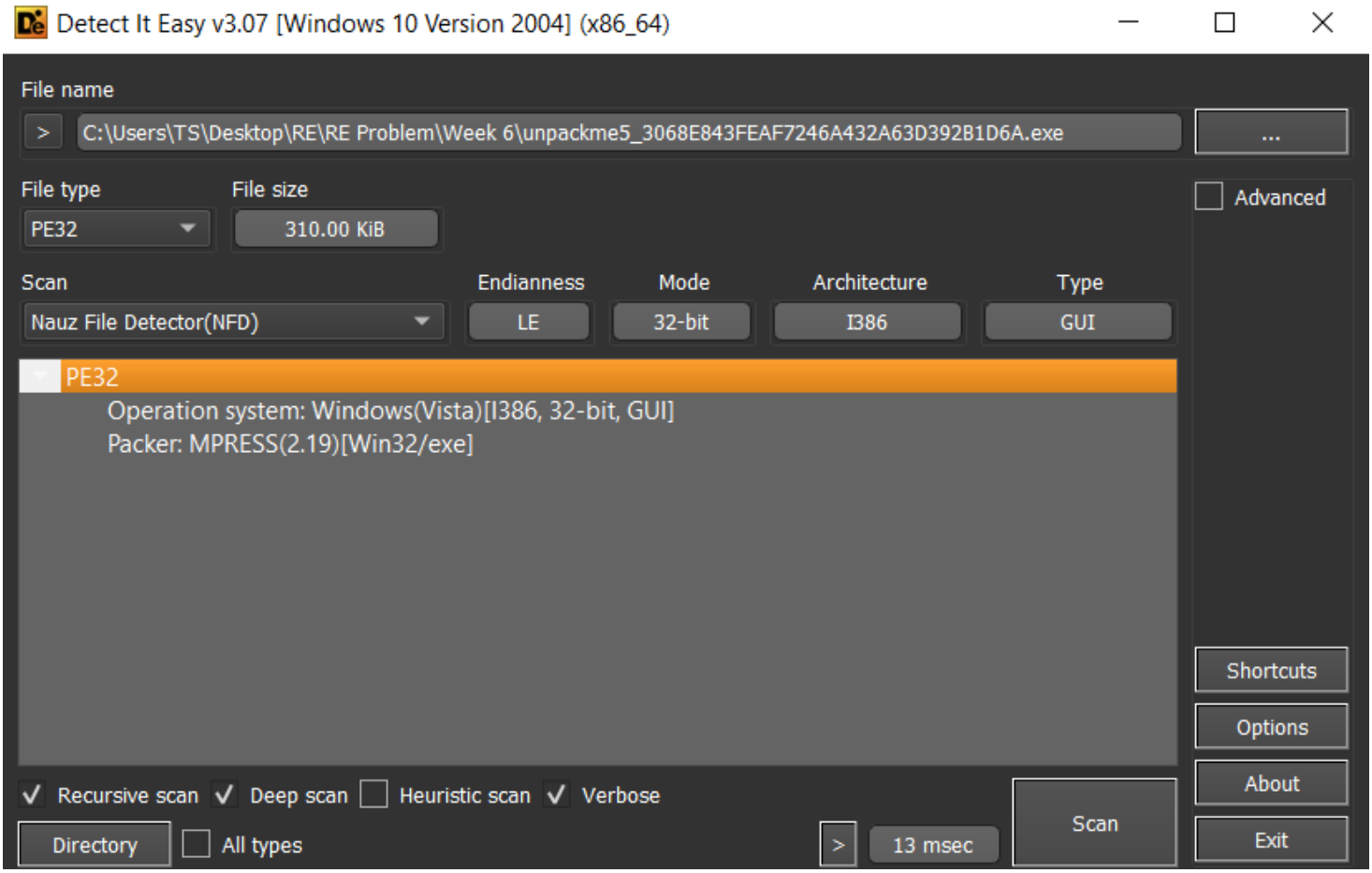


Dump và fix dump bằng Scylla:

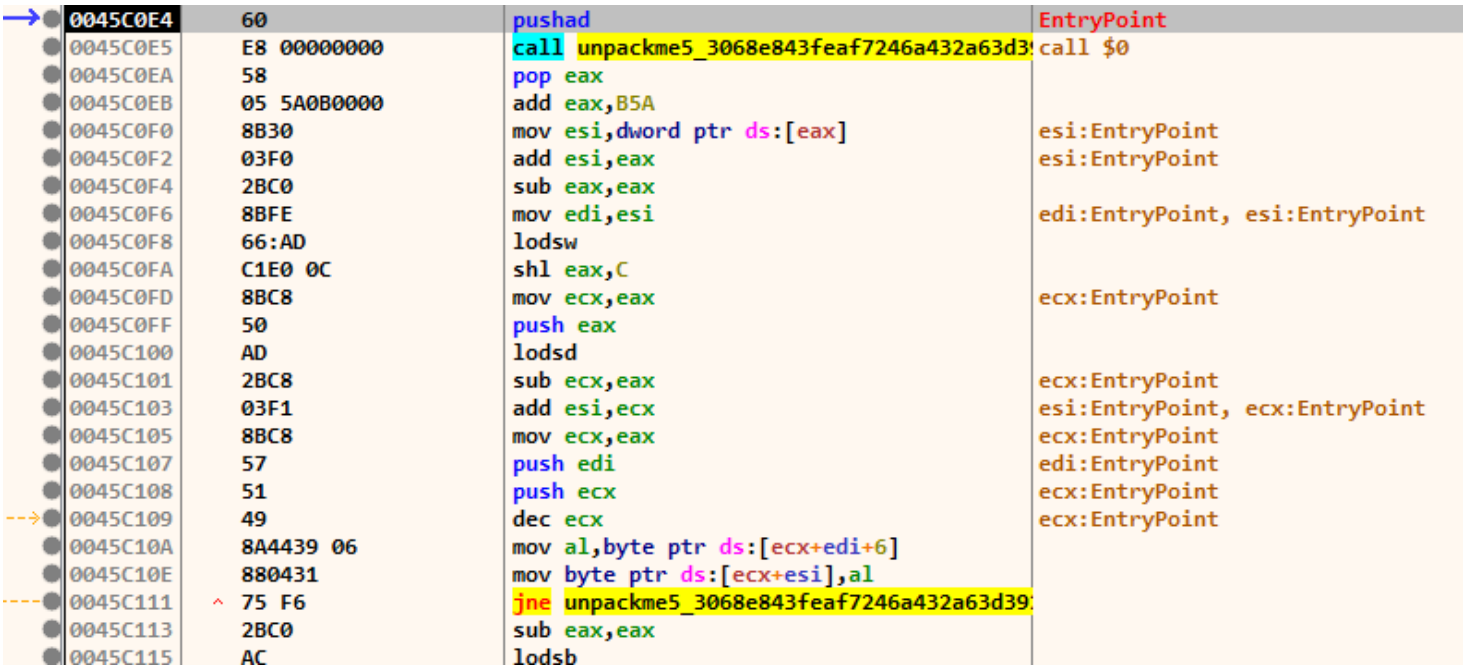


unpackme5_3068E843FEAF7246A432A63D392B1D6A.exe

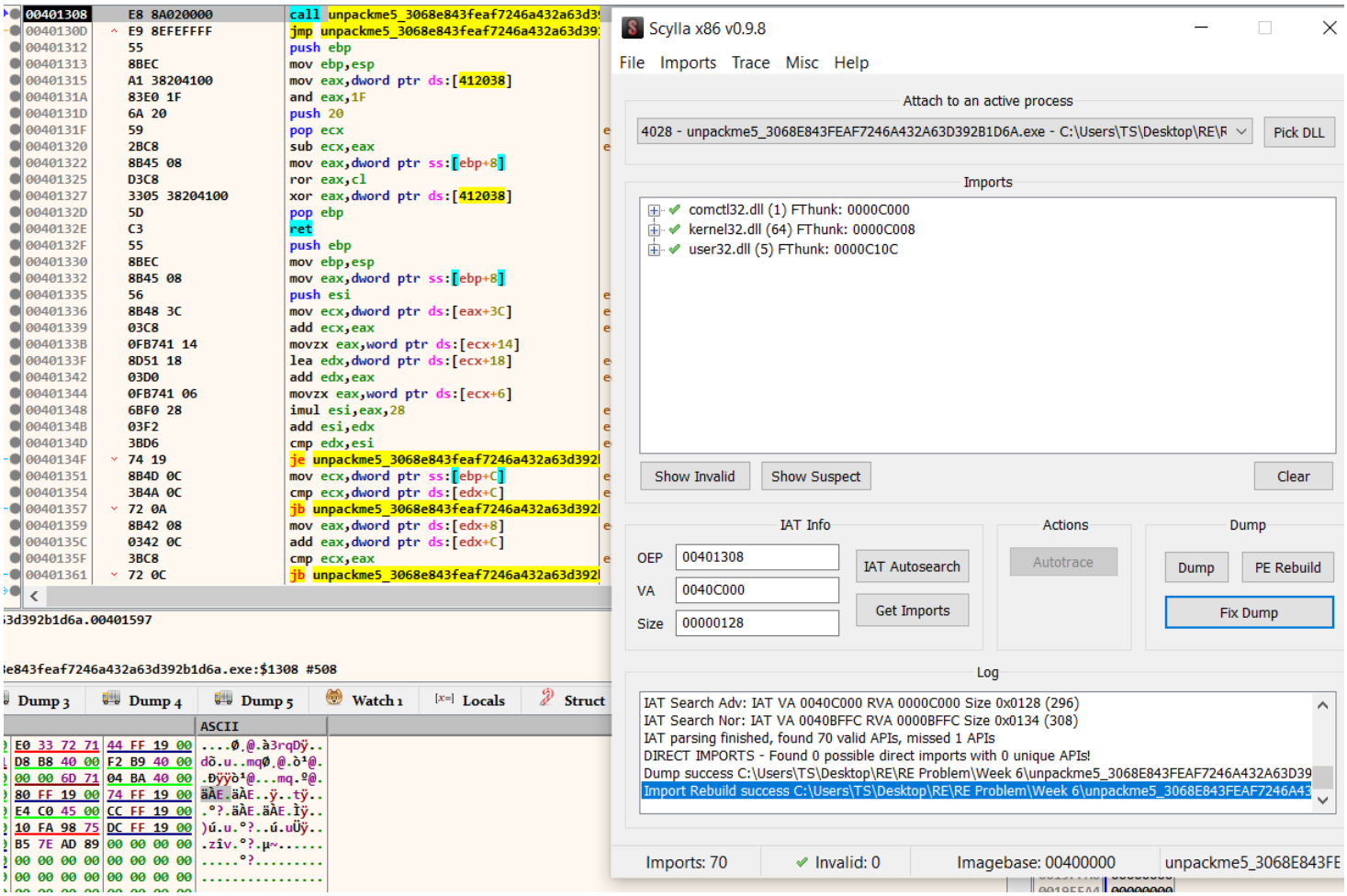
Làm tương tự unpack3.
Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng MPRESS packer:



Debug file bằng x32dbg và f9 để đến EntryPoint. F7 qua và đặt hardware breakpoint ở địa chỉ ESP đang trỏ(sau lệnh pushad):

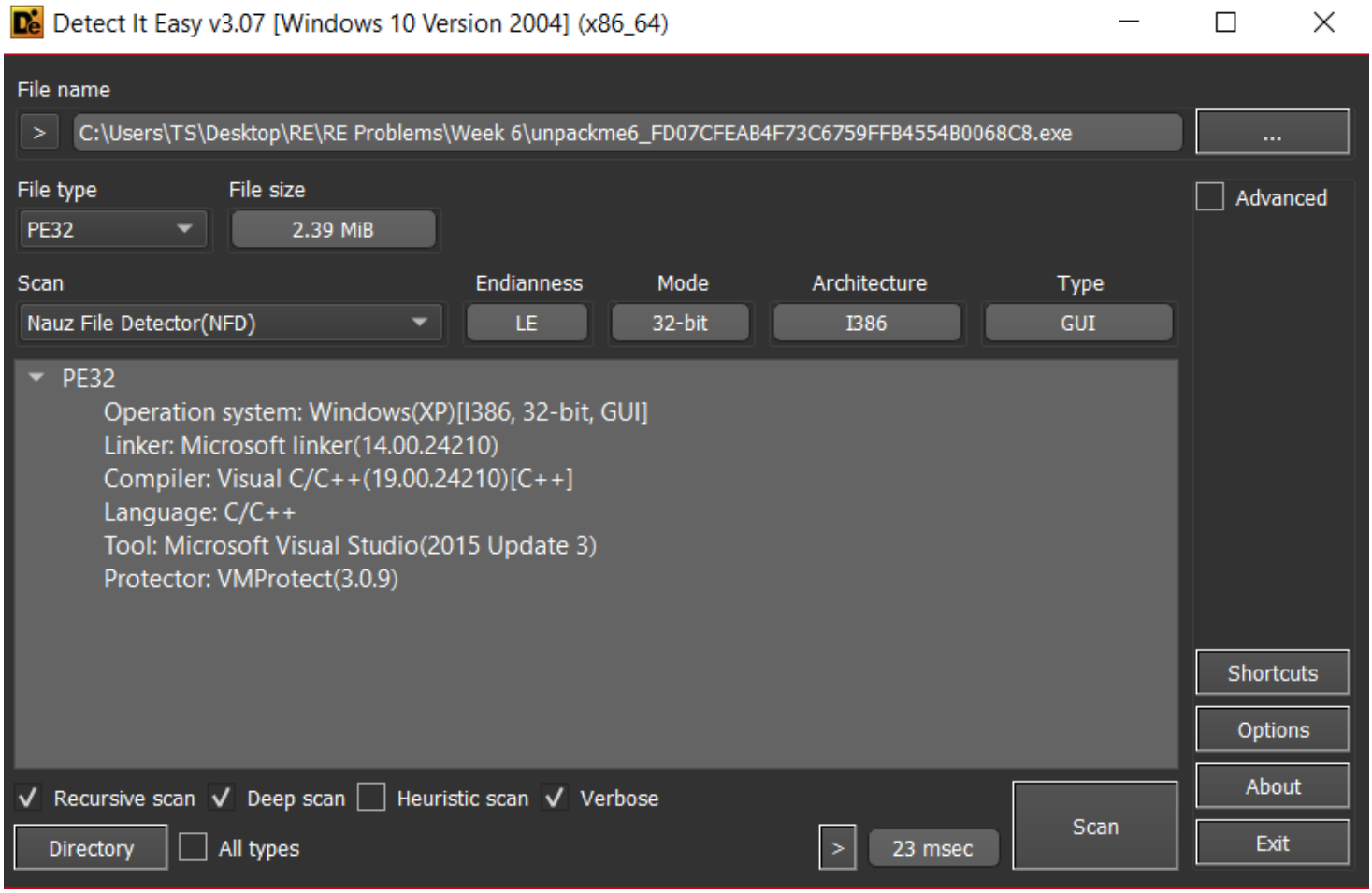


Dò đến đầu chương trình gốc của file và dump bằng Scylla(xóa import lỗi, dump và fix dump).

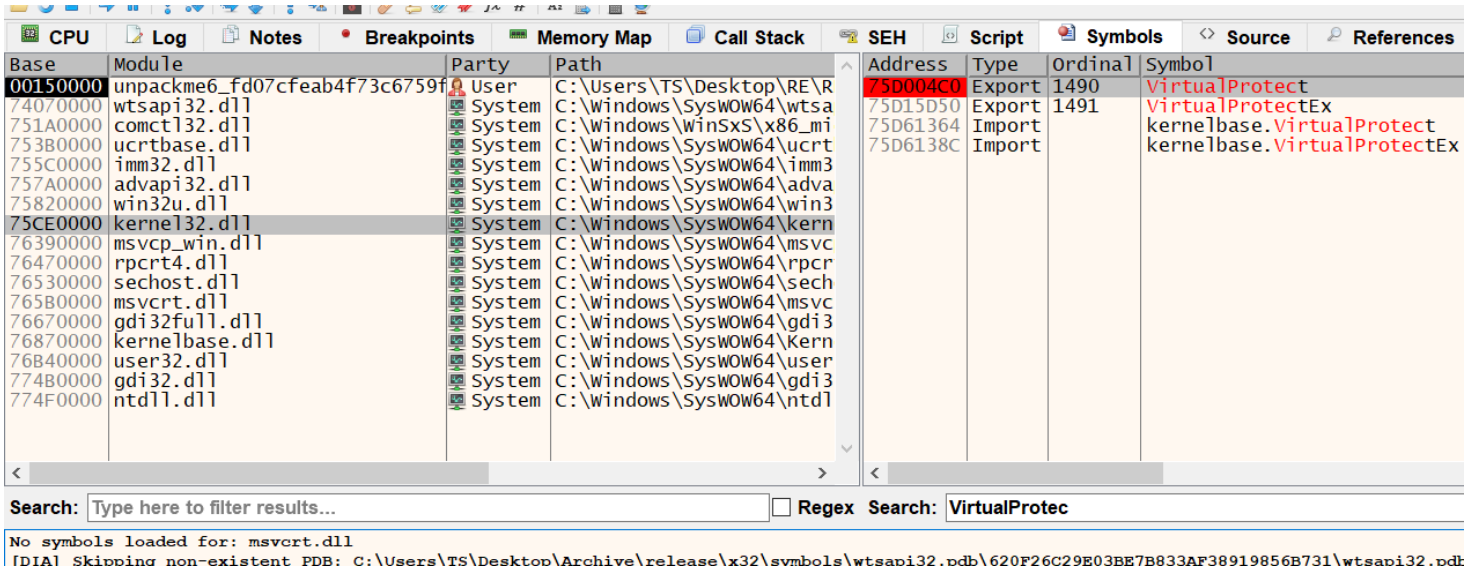


unpackme6_FD07CFEAB4F73C6759FFB4554B0068C8.exe

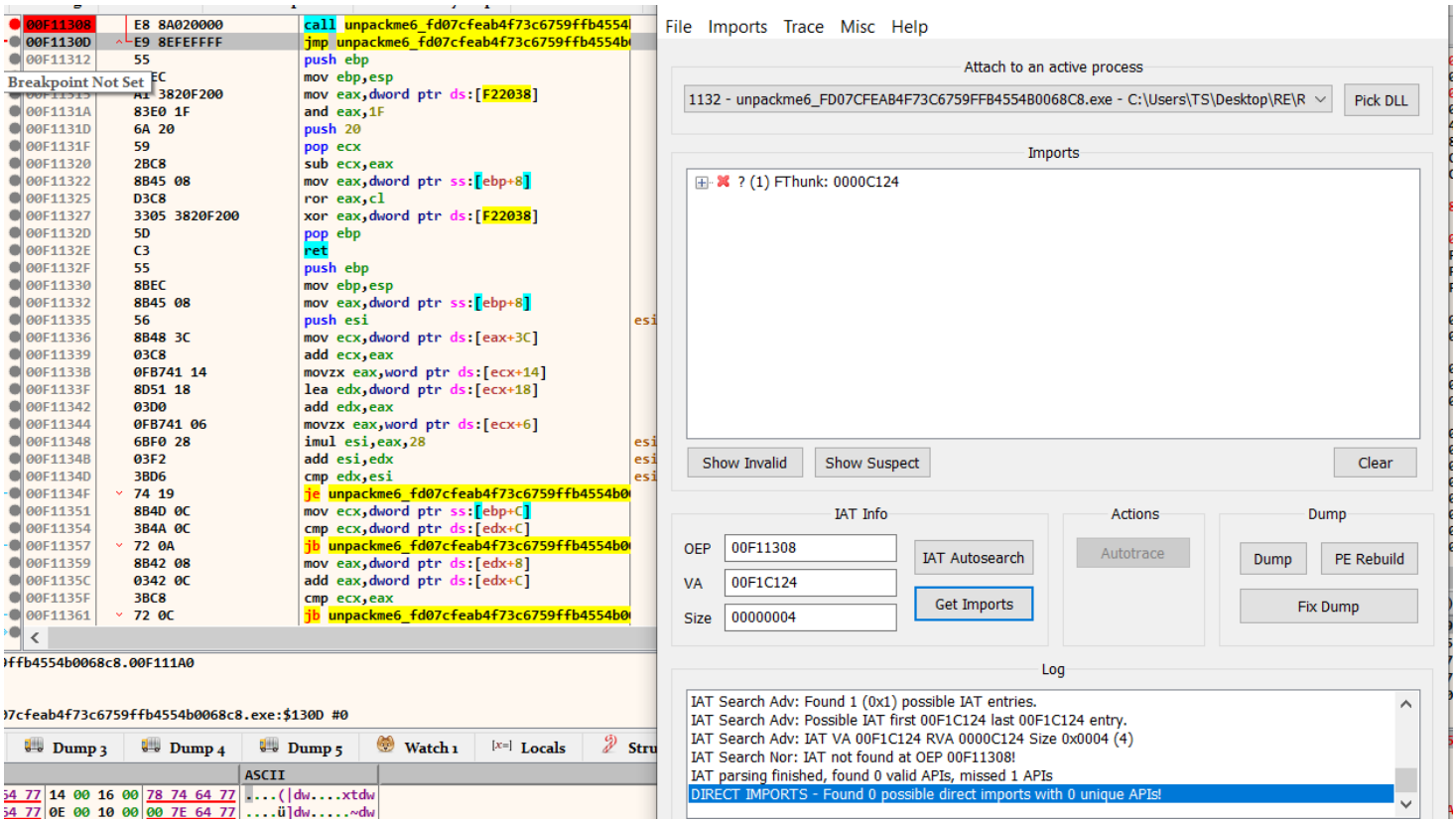
Phân tích bằng PEiD cho thấy file pack bằng VMProtect packer:



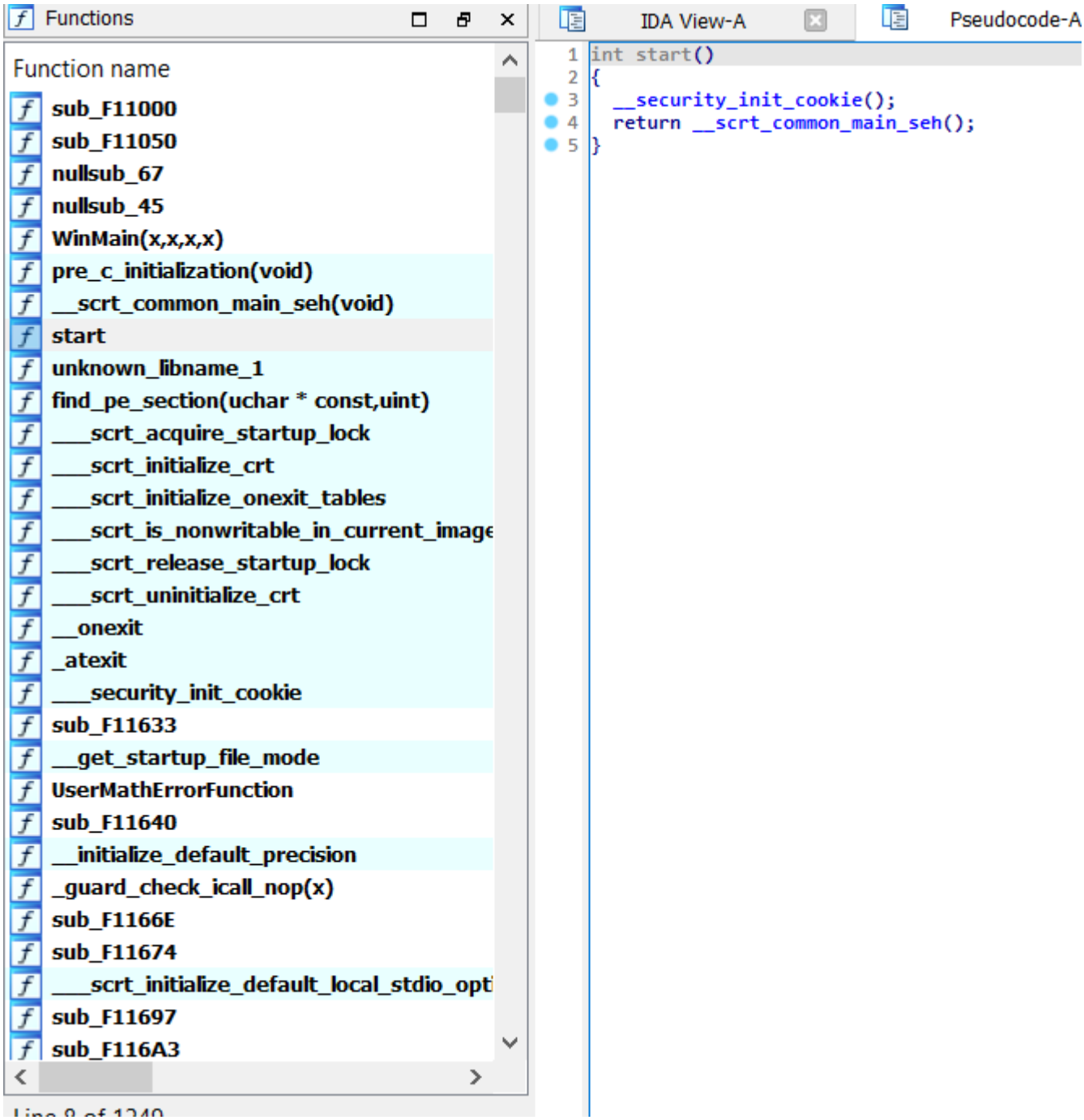
Debug file bằng x64dbg và đặt breakpoint ở kernel32.VirtualProtect:



,sau đó F9 8 lần. Sau đó code sẽ được giải nén ở .text. đặt memory breakpoint ở .text và F9 đến khi tìm được OEP đúng là 0xF11308, dump ra bằng Scylla:

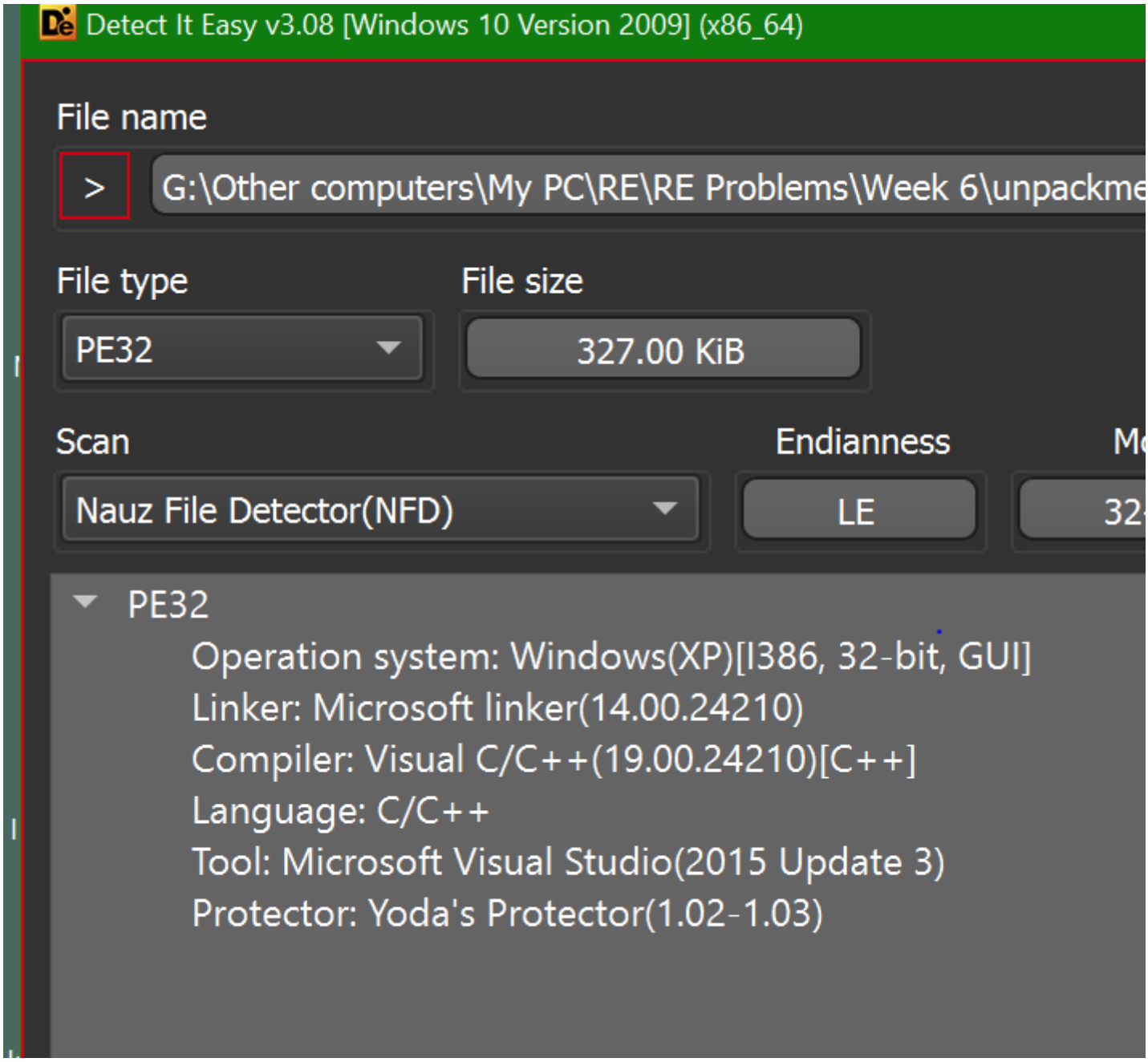


Phân tích tĩnh được code đã dump nhưng file không chạy được vì IAT bị hỏng:



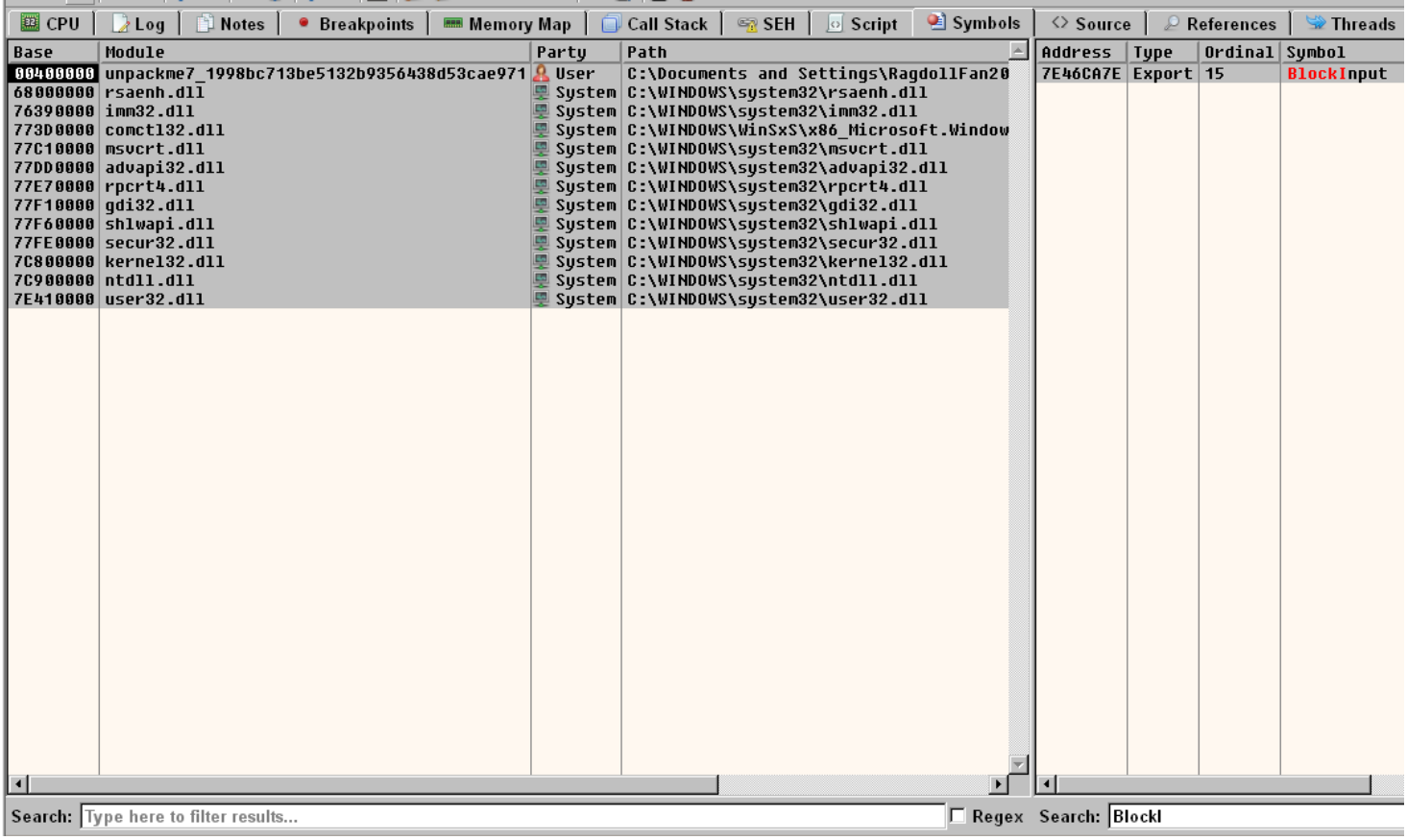
unpackme7_1998BC713BE5132B9356438D53CAE971.exe

Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng Yoda's Protecter packer:



Tính năng của Yoda's Protector:
-Anti debugging: JMP vào giữa một hàm, raise exception INT 3, BlockInput(), IsDebuggerPresent(), BlockInput(), CreateToolhelp32Snapshot(), Process32First(), và Process32Next(), GetCurrentProcessId(),...
-Anti-SoftICE
-Chống sửa đổi(Checksum Check)
-Load API bằng LoadLibraryA() và GetProcAddress()
-Chống dump
-Xóa thông tin Import
-Xóa PE header
(ref: https://sanseolab.tistory.com/11)

File chỉ chạy trên Windows XP. Dùng x32dbg để chạy chương trình sẽ bị thoát ra do chương trình phát hiện debugger. Khắc phục bằng cách đặt breakpoint ở hàm LoadLibraryA(), chạy đến khi chương trình load đầy đủ thư viện:



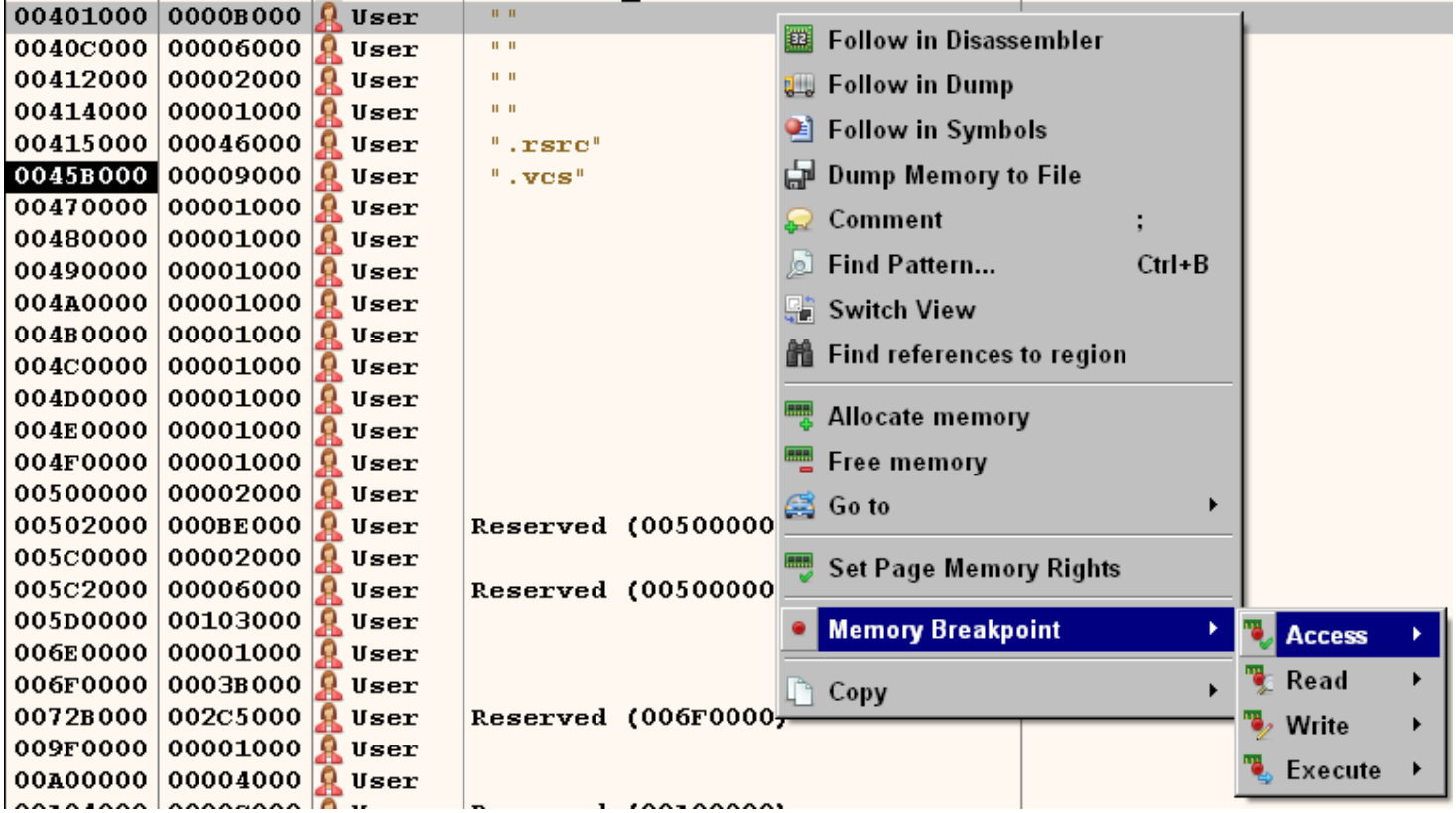
Patch nop hàm BlockInput(), patch xor eax, eax IsDebuggerPresent(), patch mov eax, {pid của x64 dbg} trong GetCurrentProcessId() để bypass:

```
7E46CA7E   90              nop                          BlockInput
7E46CA7F   90              nop
7E46CA80   90              nop
7E46CA81   90              nop
7E46CA82   90              nop
7E46CA83   90              nop
7E46CA84   90              nop
7E46CA85   90              nop
7E46CA86   90              nop
7E46CA87   90              nop
7E46CA88   90              nop
7E46CA89   90              nop
7E46CA8A   C2 0400         ret 4
```

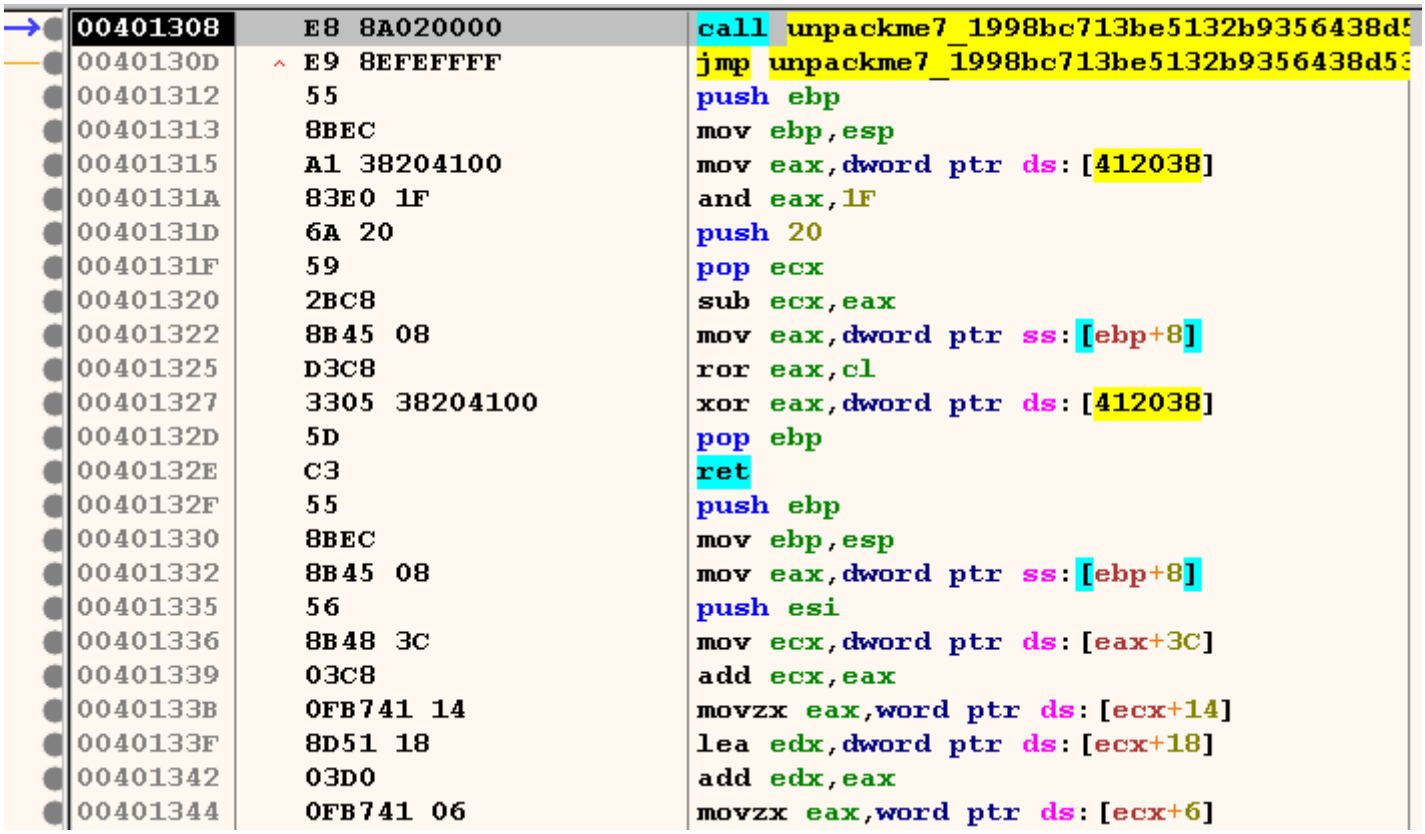
```
7C8099C0   90              nop                          GetCurrentProcessId
7C8099C1   90              nop
7C8099C2   90              nop
7C8099C3   B8 8D030000     mov eax,38D
7C8099C8   90              nop
7C8099C9   C3              ret
```

```
7C82F6EF   90              nop                          IsDebuggerPresent
7C82F6F0   90              nop
7C82F6F1   90              nop
7C82F6F2   90              nop
7C82F6F3   90              nop
7C82F6F4   90              nop
7C82F6F5   90              nop
7C82F6F6   90              nop
7C82F6F7   90              nop
7C82F6F8   33C0            xor eax,eax
7C82F6FA   90              nop
7C82F6FB   90              nop
7C82F6FC   C3              ret
```

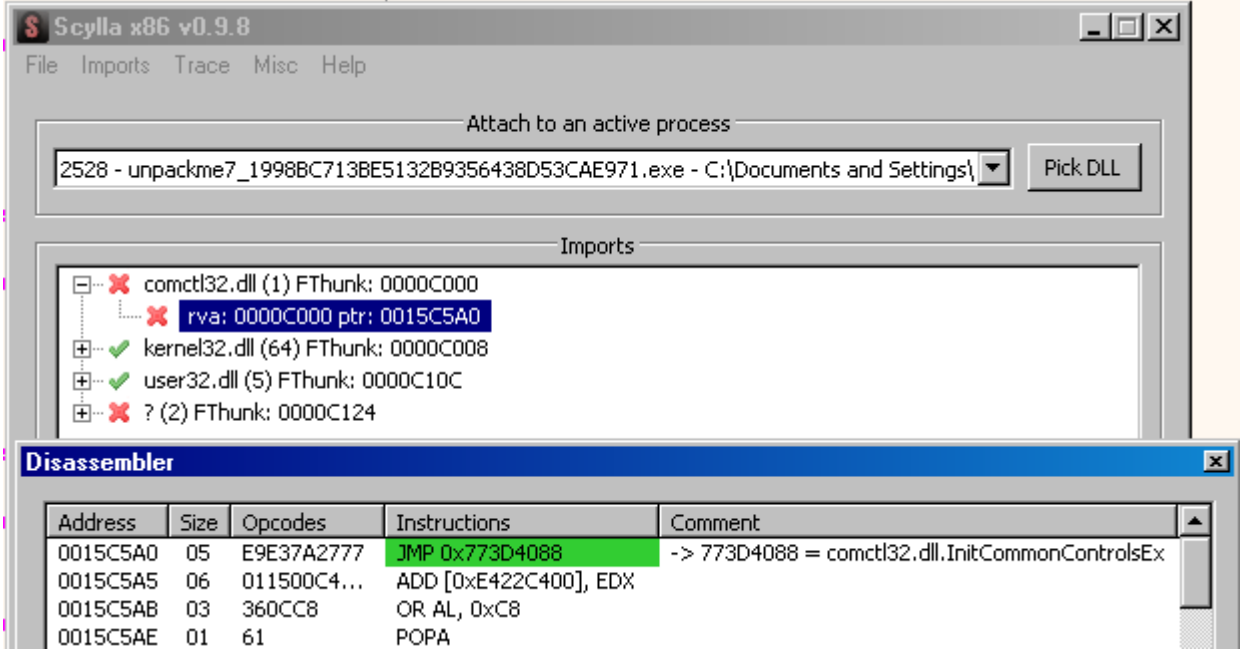
Hoặc đơn giản hơn là dùng plugin ScyllaHide để debugger không bị phát hiện. Khi debug với ScyllaHide, f9 9 lần thì chương trình chạy ra cửa sổ, do vậy OEP được chạy giữa lần f9 thứ 8 và 9. f9 8 lần đến đây rồi đặt memory access breakpoint tại section không tên đầu tiên(khả năng cao là .text):



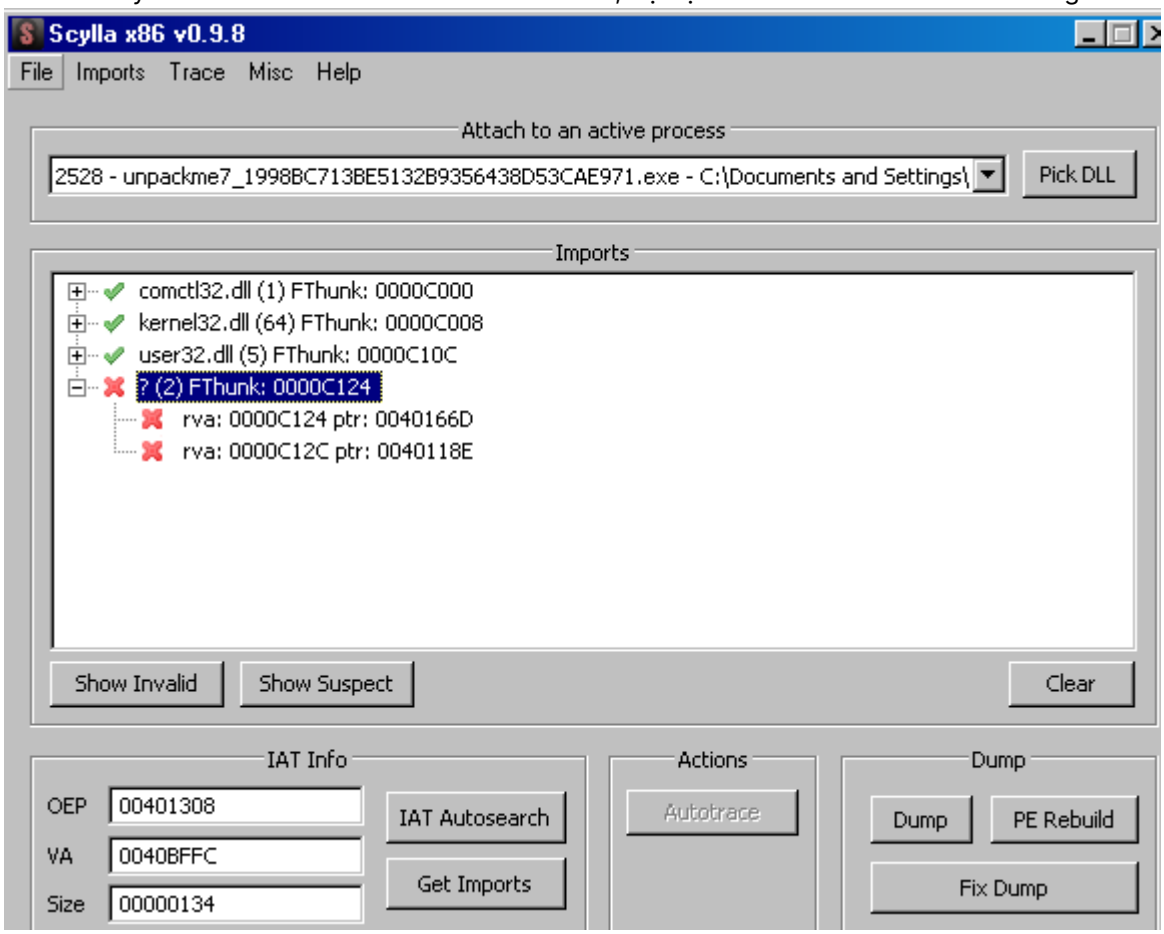
Chương trình dừng tại địa chỉ 0×401308, chính là OEP:



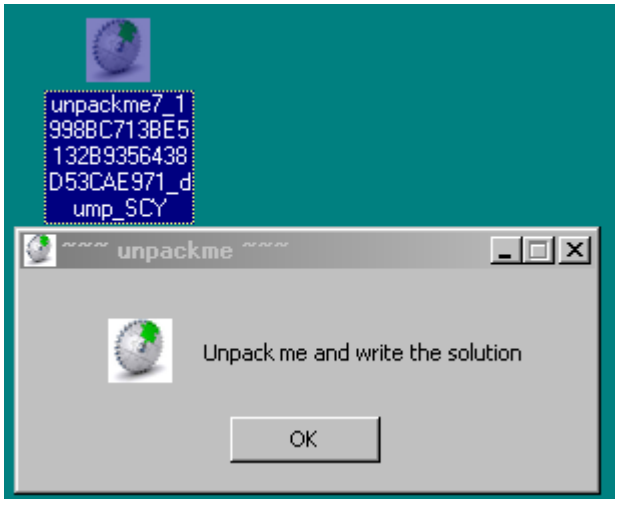
Dùng Scylla IAT Auto Search chế độ normal(chế độ advanced gặp lỗi), get imports và disassemble function chưa được nhận diện:



→ Cho thấy function trên là InitCommonControlEx, đặt lại con trỏ của function cho đúng:

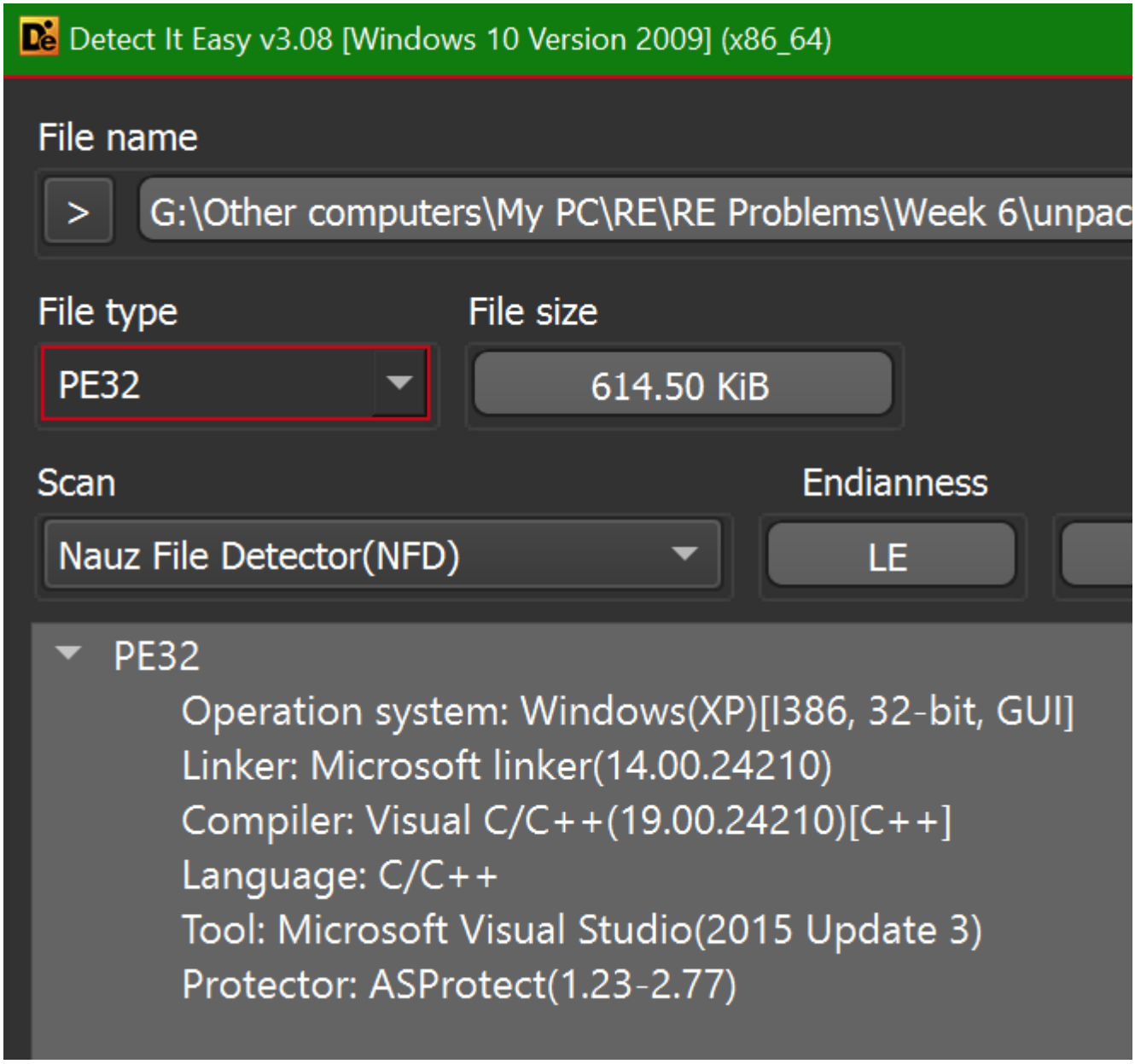


và cut function lỗi còn lại, sau đó dump và fix dump. Kết quả đạt được là file unpack và chạy thành công:

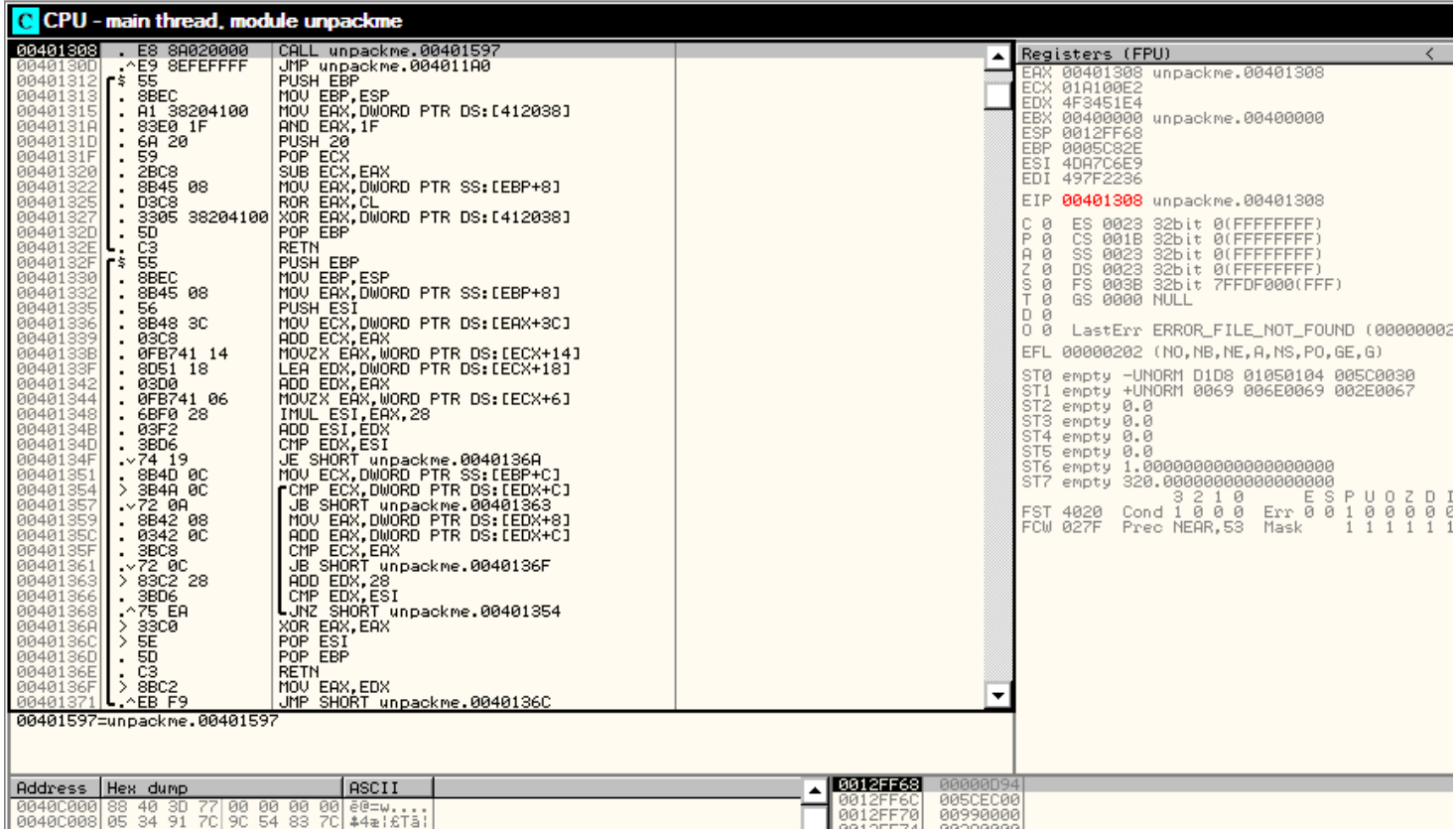


unpackme8_0E29BE1D445143C4135AA9108DF327C4.exe

Kiểm tra file bằng Detect It Easy cho thấy file được pack bằng ASProtect packer:



Cài OllyDBG Script và dùng script Aspr2.XX_unpacker_v1.15E.osc, tìm được OEP:



Dump ra và dùng ImpRec fix IAT, sau đó file chạy thành công: