

|  |  |
| --- | --- |
| Cách UPX pack 1 file | Cách 1 file đã được pack bằng UPX thực thi |
| * Sử dụng các thuật toán để nén dữ liệu   (LZMA > phiên bản 2.9; UCL: nhanh hơn đơn giản hơn LZMA)   * Thêm code mới vào cuối tệp ( mục đích là để giải nén code ban đầu trước khi chương trình thực thi) | * File sẽ thực thi phần tại phần code đã được thêm vào * Lưu lại trạng thái của các thanh ghi hiện tại bằng pushad * Giải nén phần code thực thi ban đầu * Khôi phục lại IAT * Khôi phục lại trạng thái thanh ghi bằng popad * Jump tới OEP gốc của file * Thực thi chương trình bình thường |

UPX

1. File sẽ thực thi phần tại phần code đã được thêm vào
2. Lưu lại trạng thái của các thanh ghi hiện tại bằng pushad
3. Giải nén phần code thực thi ban đầu

* Dựa trên thuật toán nén sẽ có thuật toán giải nén tương ứng
* Xác định được đoạn code chương trình dùng chỉ dùng các câu lệnh cơ bản

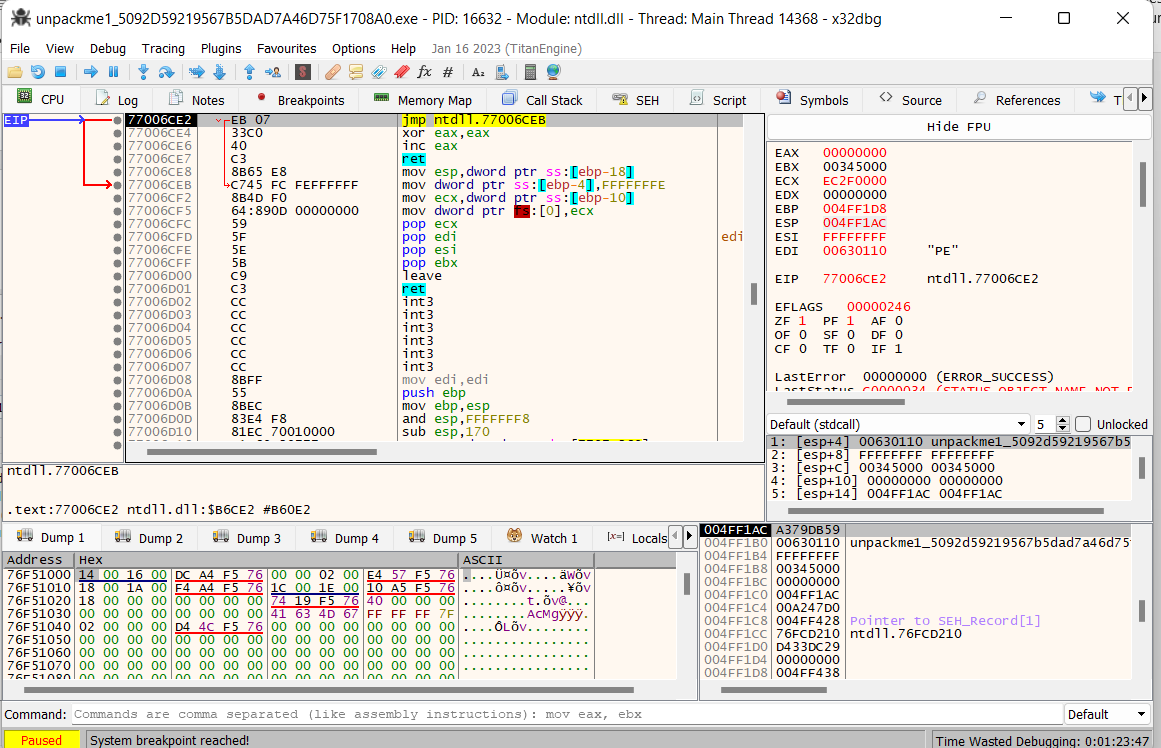
1. Khôi phục lại IAT

* LoadLibraryA
* GetprocessAddress

Sau khi khôi phục xong sử dụng VirtualProtect : thay đổi quyền truy cập ( hàm này sử dụng sau để đảm bảo code sau khi khôi phục sẽ được bảo vệ)

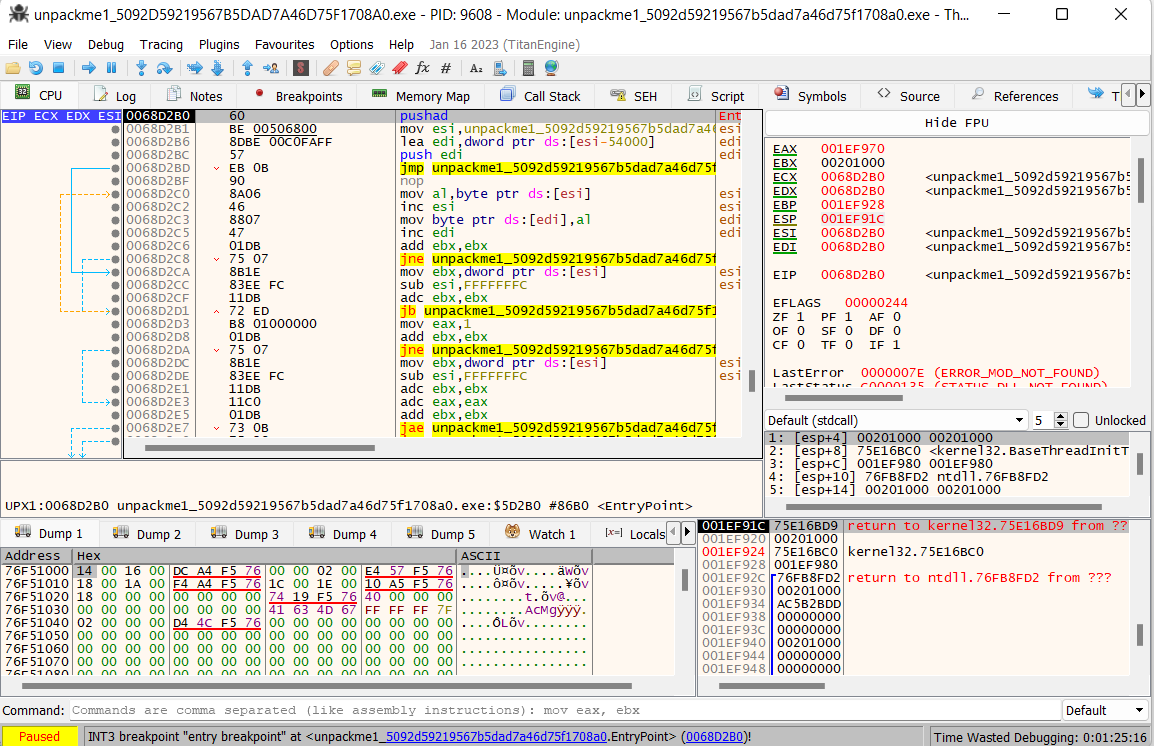
1. Khôi phục lại trạng thái thanh ghi bằng popad
2. Jump tới OEP gốc của file thực thi ban đầu

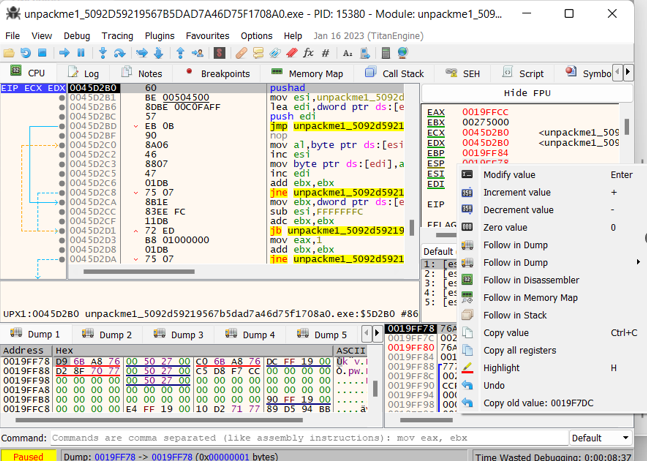
Load file x64dbg



F9 để nhảy tới entry point (EP)

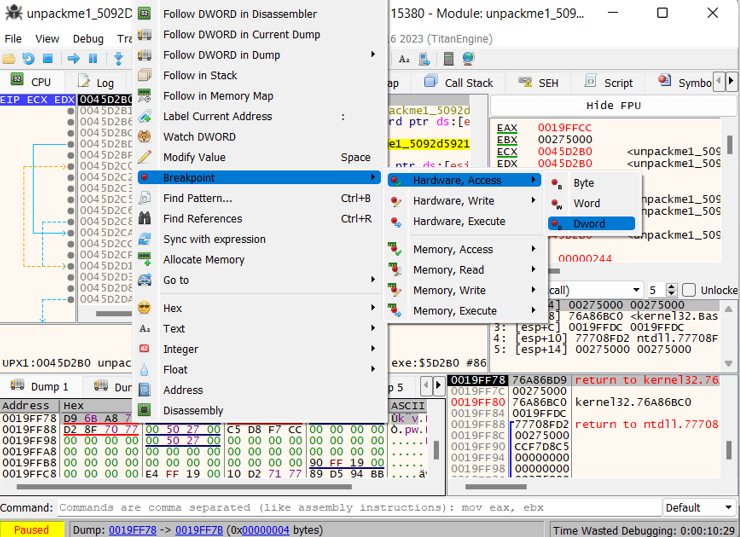
Ở đây ta thấy câu lệnh pushad đây là dấu hiệu tốt khi unpack



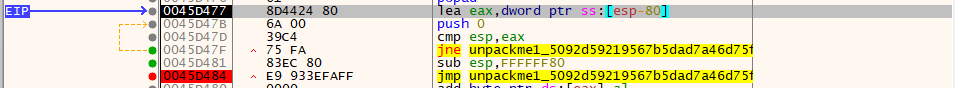


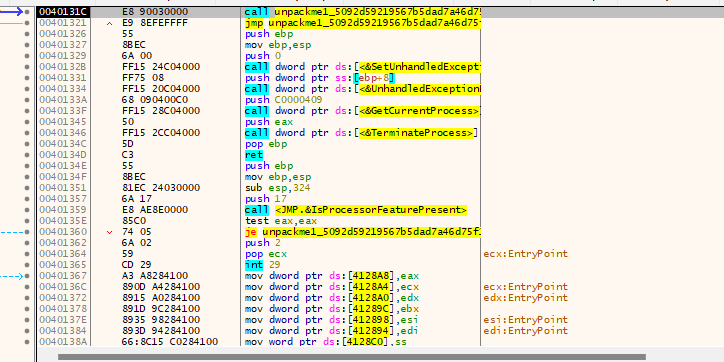
Ta đặt 1 memory breakpoint on access tại ESP

" memory breakpoint on access" đặt một breakpoint tại một địa chỉ bộ nhớ. Khi truy cập vào địa chỉ bộ nhớ này được thực hiện, chương trình sẽ dừng lại.



F9 1 lẫn nữa thì chương trình tới lệnh này. Dưới câu lệnh này có 1 lệnh jmp => khả năng cao chính là oep





Dump thử file bằng IDA thì chương trình chạy tốt. dùng IDA cũng thấy được hàm main

