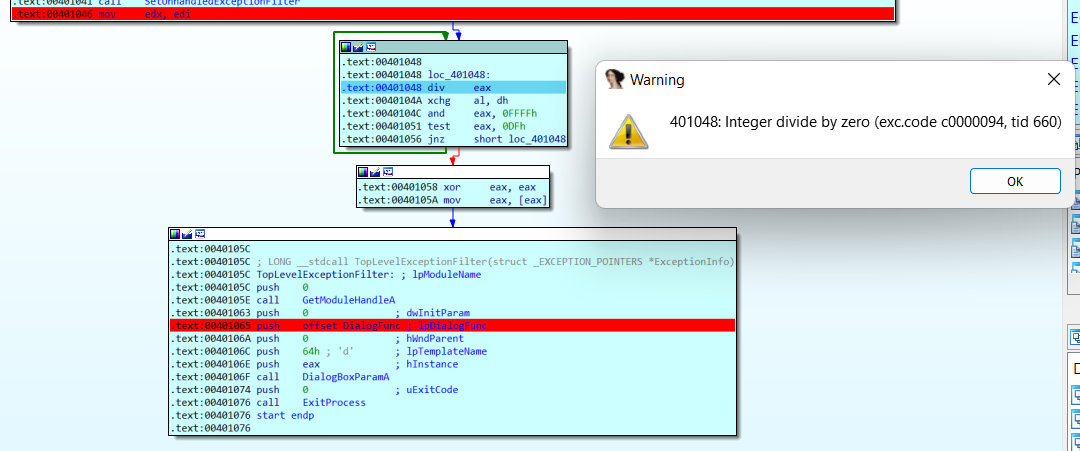
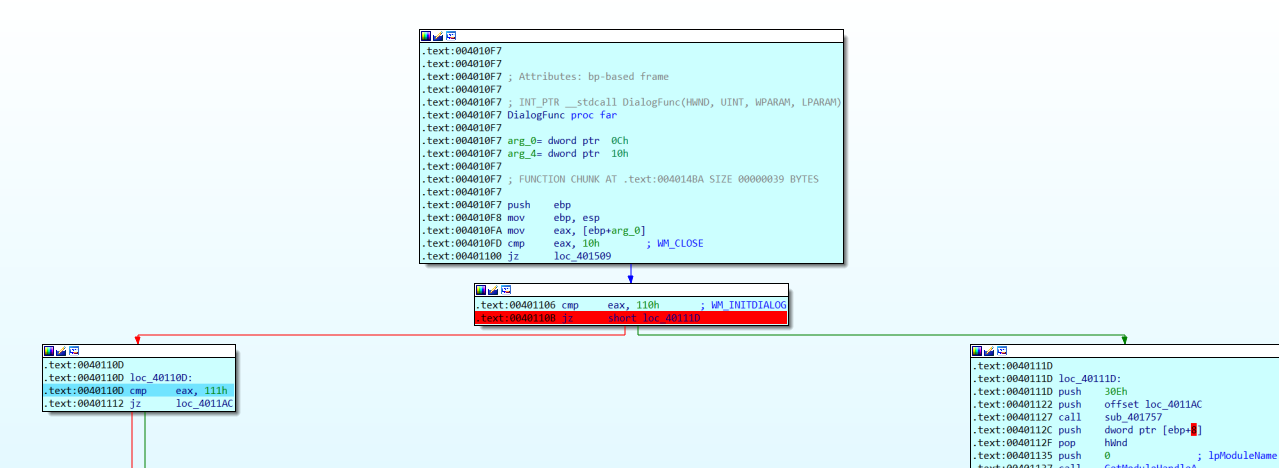
Dùng ida phân tích chương trình ta thấy có câu lệnh div eax báo lỗi lí do là eax= 0 thì ko thể chia cho 0.

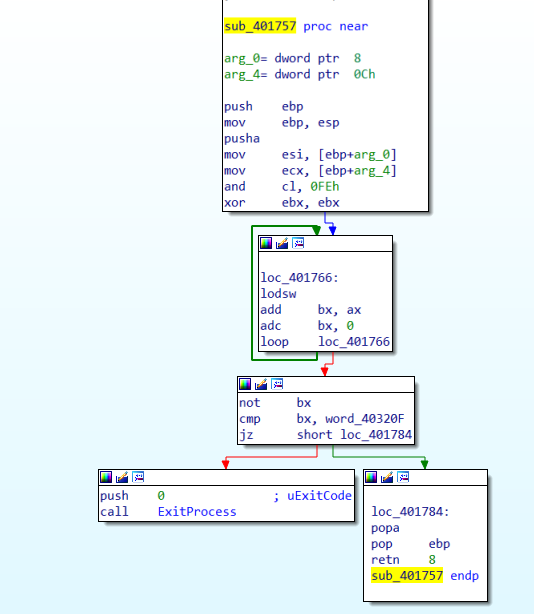


Chương trình sử dụng win API SetUnhandledExceptionFilter để xử lí ngoại lệ nếu chạy dưới debugger.



* **WM\_INITDIALOG**

sub\_401757





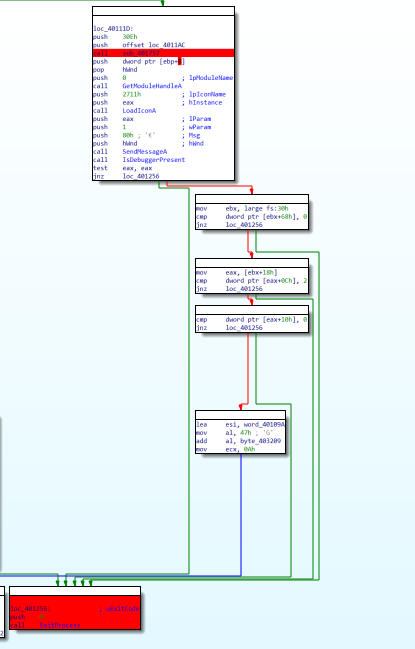
IsDebuggerPresent kiểm tra flag **BeingDebugged** của PEBcó được bật hay không ( kĩ thuật tương tự trong bài simple antidebug )

NtGlobalFlag : offset 0x68 trong PEB

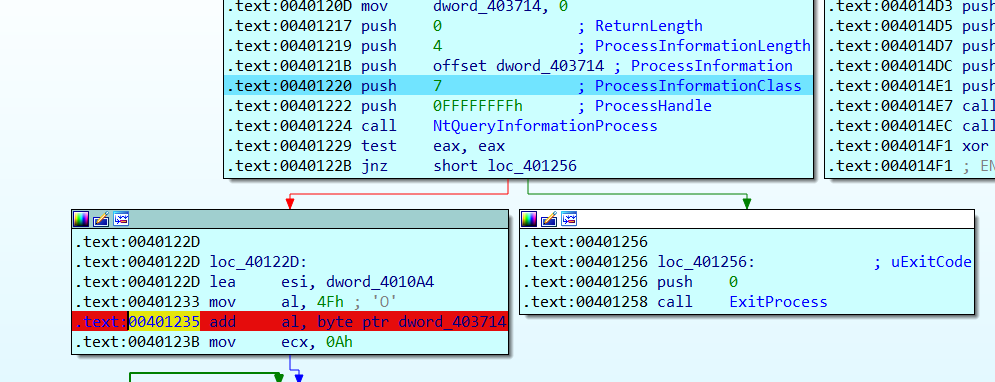
ProcessHeap :

* Flags (offset 0x0C ) set HEAP\_GROWABLE(2) nếu debug
* ForceFlags (offset 0x10) set 0 nếu debug

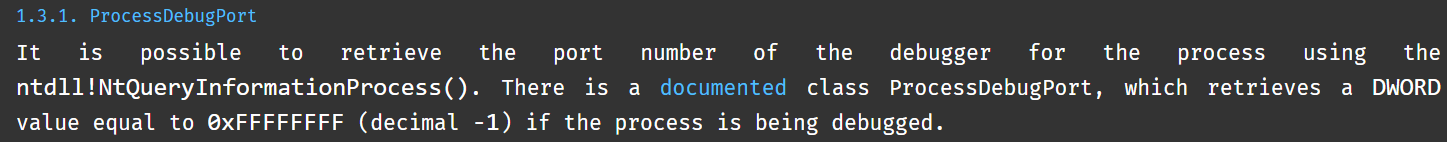
Bypass



Ta thấy có các lệnh nhảy dẫn tới các luồng thực thi kết thúc chương trình => ko cho thực hiện các lệnh nhảy đó



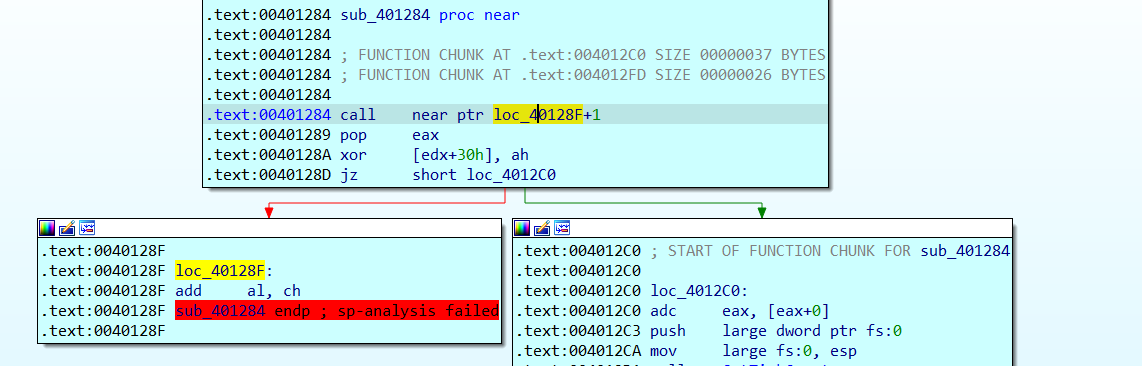
Sử dụng  **NtQueryInformationProcess** để kiểm tra debugger. với trường **ProcessInformationClass** = 7 ProcessDebugPort ( kĩ thuật tương tự trong bài simple antidebug)



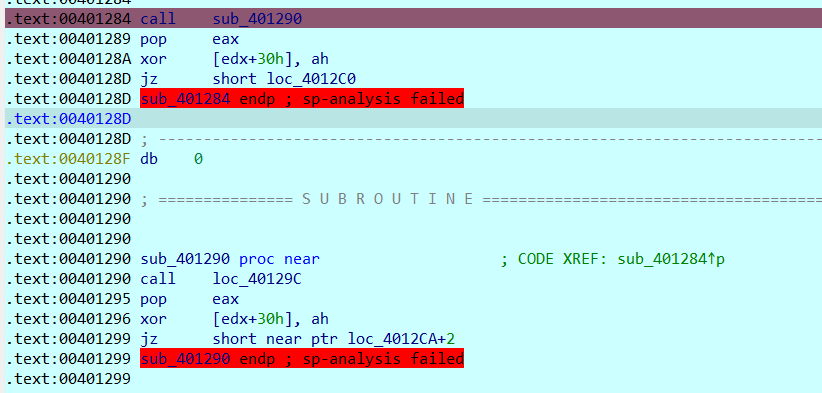
Nếu phát hiện debugger thì giá trị **dword\_303714** sẽ khác không giá trị **dword\_4010A4** sẽ bị thay đổi ảnh hưởng tới flag sinh ra sau này.

Bypass nop dòng này

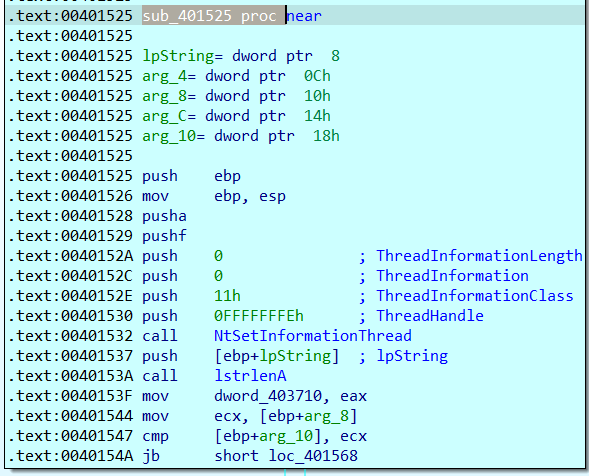
Tiếp tục chương trình có hàm sub\_401284 hàm này có vẻ IDA phân tích sai



Ta define lại code chương trình

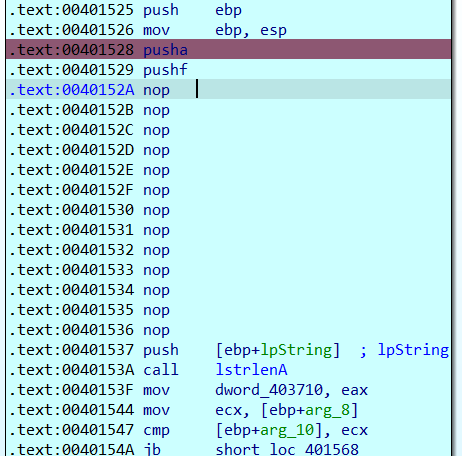


Hàm **sub\_401525**



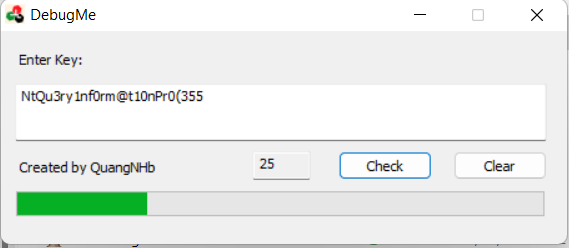
**NtSetInformationThread** có thể ẩn 1 luồng khỏi debugger (**ThreadInformationClass = 0x11)** sau khi ẩn thì ctrinh tiếp tục thực thi nhưng ko nhận được các sự kiện liên quan tới luồng đã ẩn. nếu ta đặt breakpoint trong luồng đã ẩn chương trình sẽ bị lỗi.

Bypass



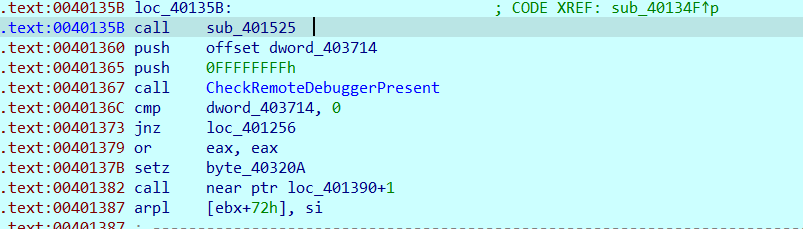
Hàm này cũng sẽ show ra đoạn flag đầu tiên

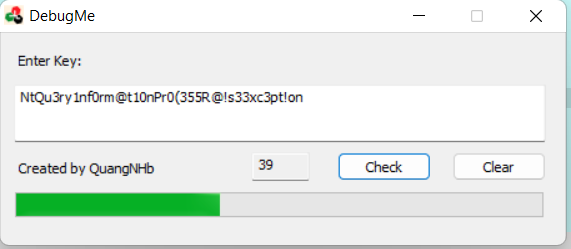
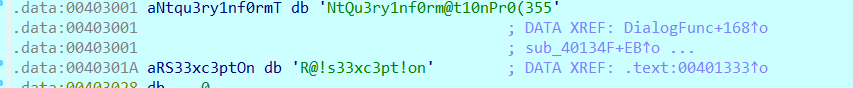




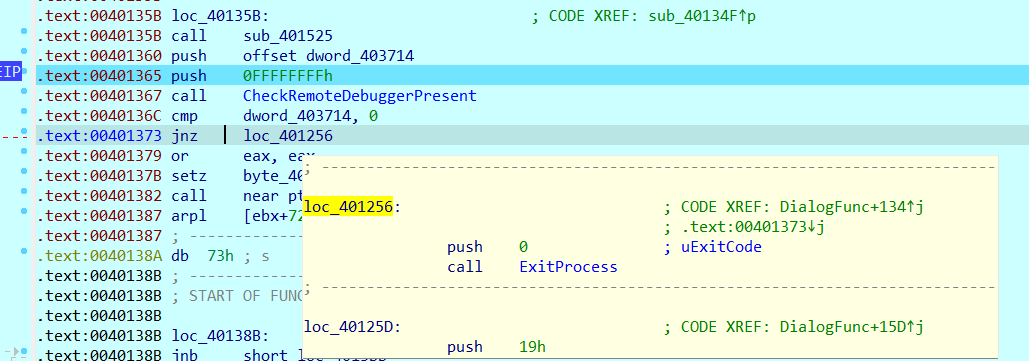
Kiểm tra thì thấy flag này chưa hoàn chỉnh

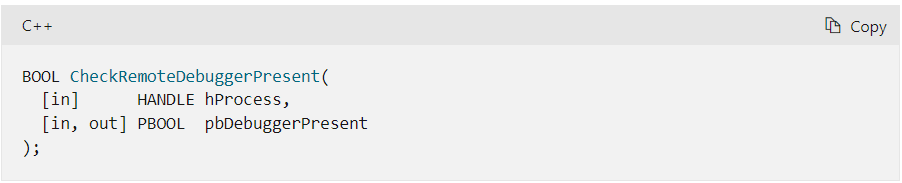
Tiếp tục thì chương trình lại gọi tới hàm **sub\_401525** ( giống hàm phía trên) hàm này sẽ cho ta được đoạn flag tiếp theo





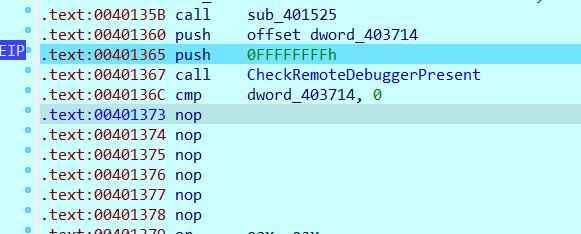
**CheckRemoteDebuggerPresent**



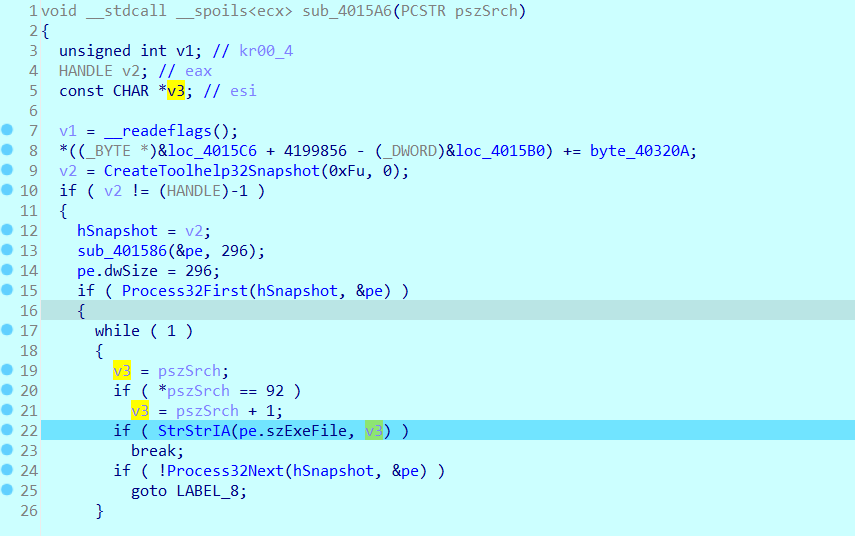


Nếu phát hiện debugger thì tham số **pbDebuggerPresent** giá trị khác 0 => chương trình nhảy vào luồng kết thúc

Bypass bỏ qua lệnh jump tới ExitProcess



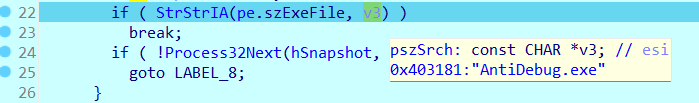
**CreateToolhelp32Snapshot**

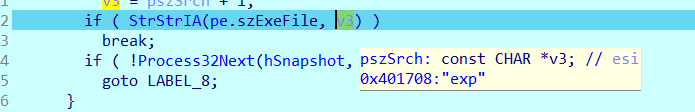
****

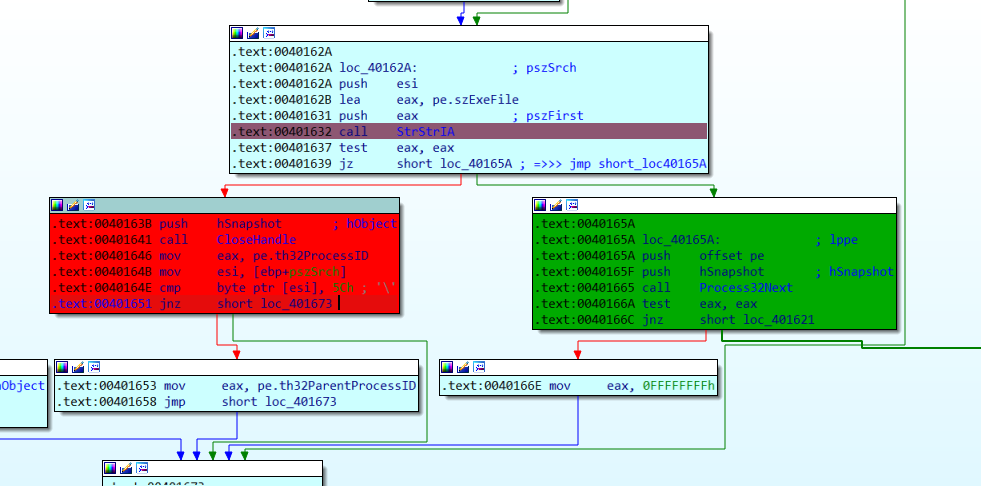
Lấy ra tên tiến trình con và ID tiến trình con

Ở đây chương trình kiểm tra nhưng ở đây chương trình kiểm tra xem tên các tiến trình có tồn tại

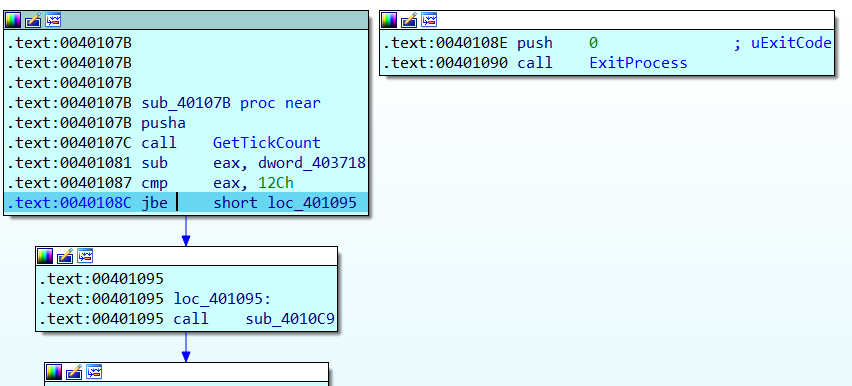
“Antidebug.exe;exp; csrss ” (hàm này được gọi 3 lần)



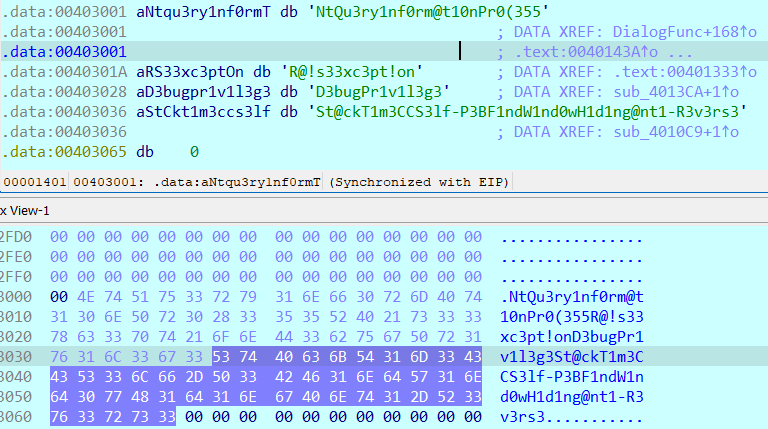


Bypass 

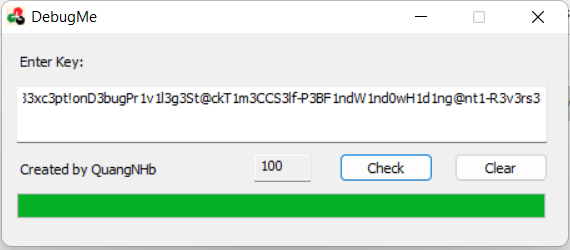
Tiếp theo chương trình sử dụng hàm **GetTickCount** hàm này sẽ kiểm tra xem chương trình nếu như đã chạy lâu bất thường thì có khả năng đây là debugger làm cho tiến trình bị thoát



Ở đây sửa thành **jmp short loc\_401095** đây cũng là hàm cho ra đoạn flag cuối

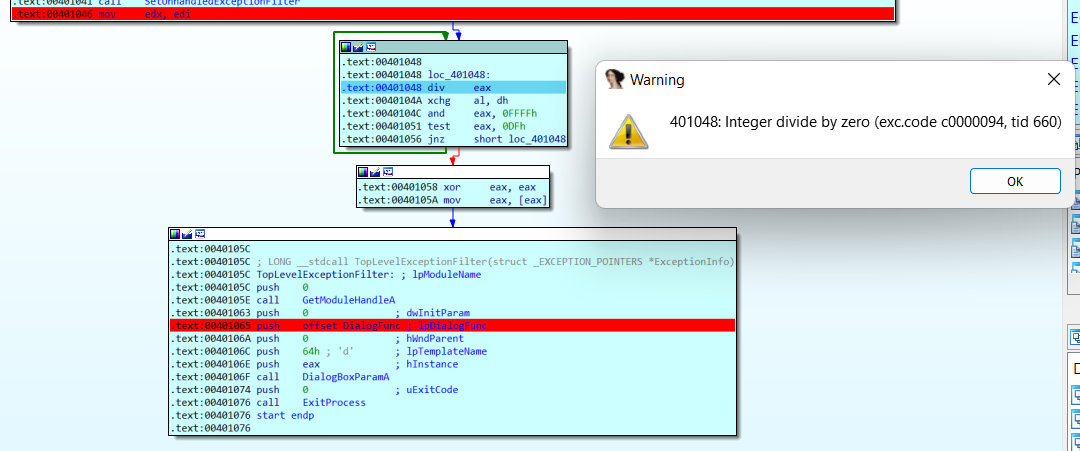


Flag hoàn chỉnh : **NtQu3ry1nf0rm@t10nPr0(355R@!s33xc3pt!onD3bugPr1v1l3g3St@ckT1m3CCS3lf-P3BF1ndW1nd0wH1d1ng@nt1-R3v3rs3**

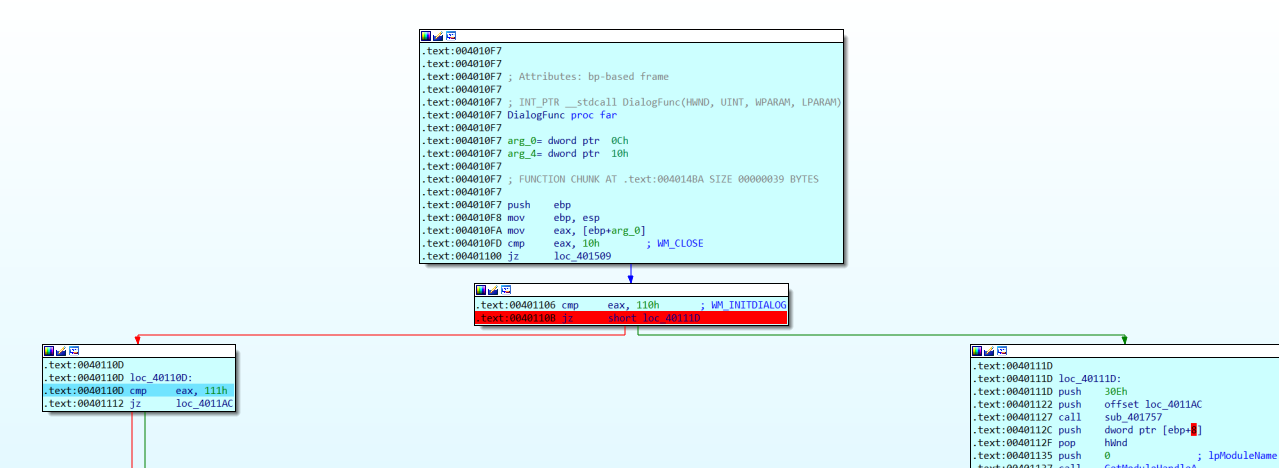


**############# HOÀN THIỆN BỔ SUNG ##############**

Dùng ida phân tích chương trình ta thấy có câu lệnh div eax báo lỗi lí do là eax= 0 thì ko thể chia cho 0.

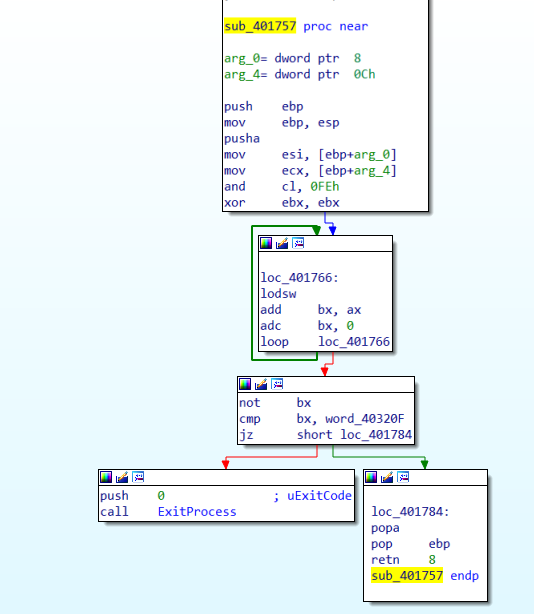


Chương trình sử dụng win API SetUnhandledExceptionFilter để xử lí ngoại lệ nếu chạy dưới debugger.



* **WM\_INITDIALOG**

sub\_401757





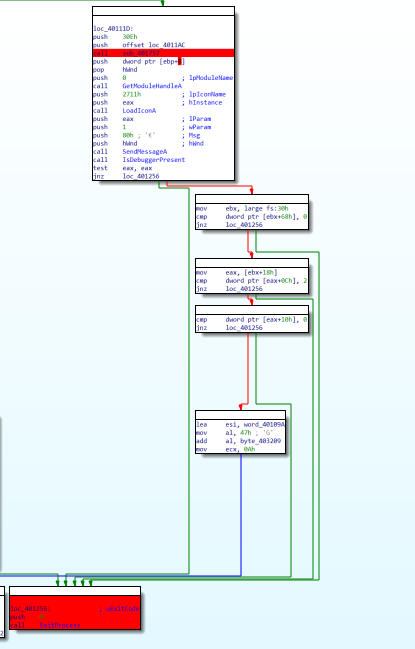
IsDebuggerPresent kiểm tra flag **BeingDebugged** của PEBcó được bật hay không

NtGlobalFlag : offset 0x68 trong PEB

ProcessHeap :

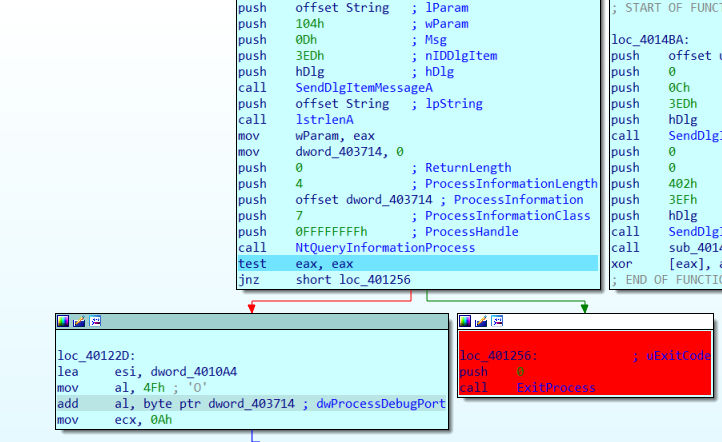
* Flags (offset 0x0C ) set HEAP\_GROWABLE(2) nếu debug
* ForceFlags (offset 0x10) set 0 nếu debug

Bypass



Ta thấy có các lệnh nhảy dẫn tới các luồng thực thi kết thúc chương trình => ko cho thực hiện các lệnh nhảy đó

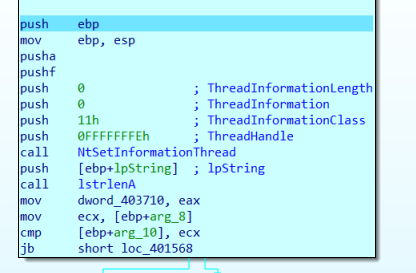
**NtQueryInformationProcess**



ProcessInformationClass = 7 => ProcessDebugPort trả về 0xFFFFFFFF (decimal -1) nếu phát hiện debug

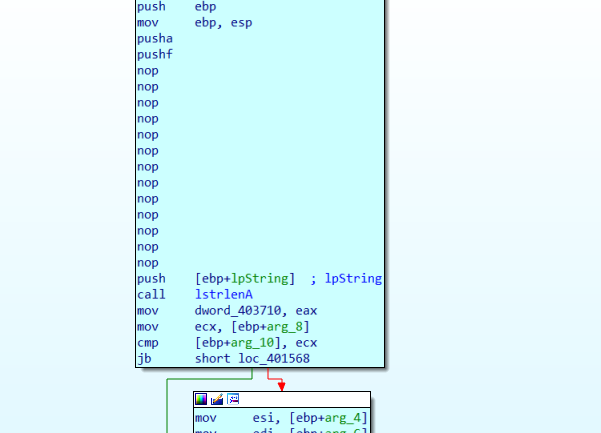
Ngoài ra dwProcessDebugPort cũng sẽ có giá trị khác 0 nếu phát hiện debug => giá trị dword\_4010A4 cũng bị thay đổi

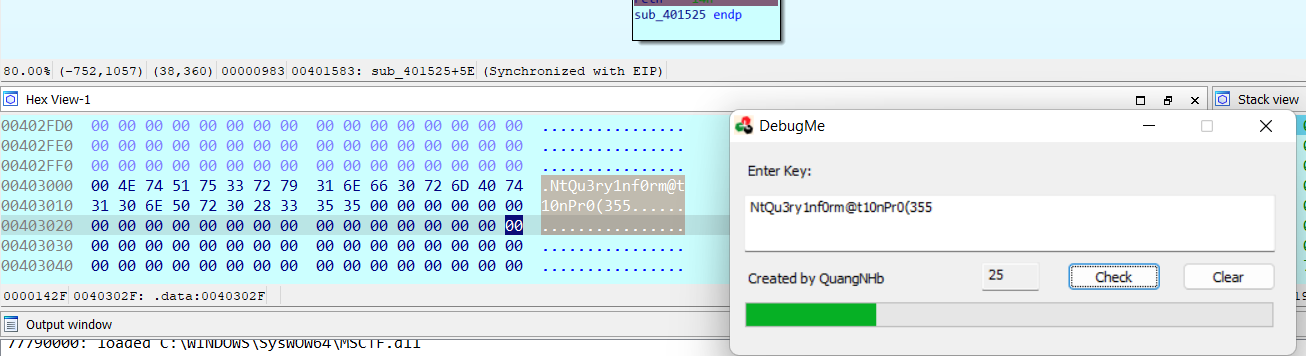
NtSetInformationThread



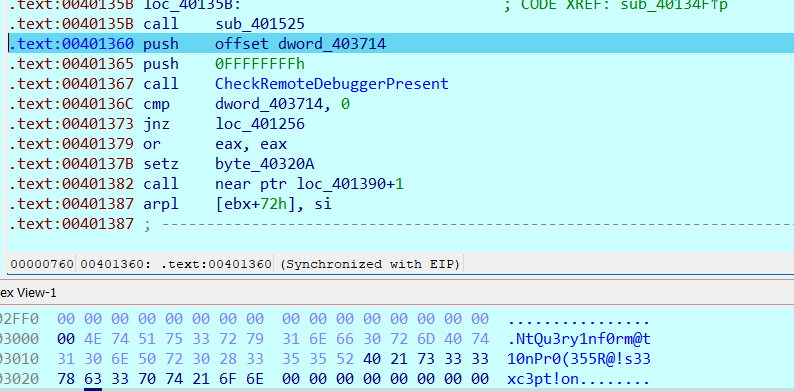
Sử dụng NtSetInformationThread để ẩn các luồng khỏi debugger bằng cách đặt ThreadInfoClass = 0x11- ThreadHideFromDebugger

Bypass



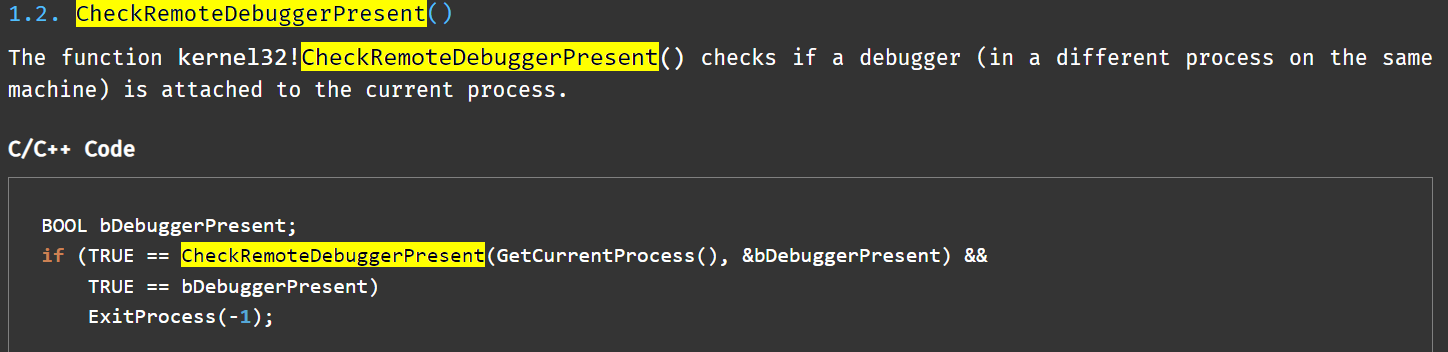


Qua hàm này cũng tìm được 1 phần của key đầu tiên **NtQu3ry1nf0rm@t10nPr0(355**

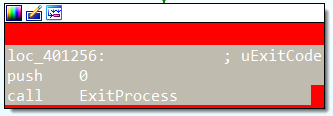


Hàm loc\_401358 có hàm sub\_401525 qua hàm này ta tìm được thêm 1 phần của key NtQu3ry1nf0rm@t10nPr0(355R@!s33xc3pt!on

**CheckRemoteDebuggerPresent** xác định xem quy trình được chỉ định có đang được debug hay ko



Nếu phát hiện thì nhảy tới loc\_401256 đây là đoạn kết thúc chương trình



Bypass

