Hàm API (Application Programming Interface) là các hàm được cung cấp bởi hệ điều hành hoặc thư viện phần mềm để cho phép các ứng dụng thực hiện các chức năng nhất định, từ giao tiếp với phần cứng đến thực hiện các thao tác xử lý dữ liệu. Dưới đây là một số nhóm hàm API phổ biến, đặc biệt là trong hệ điều hành Windows, nơi các hàm API thường được sử dụng trong lập trình hệ thống và ứng dụng:

|  |  |  |
| --- | --- | --- |
| **Hàm API Xử lý Tệp tin (File I/O APIs)** | Dùng để thao tác với tệp tin và thư mục như tạo, đọc, ghi, hoặc xóa. |  CreateFile(): Mở hoặc tạo một tệp tin hoặc thiết bị.   ReadFile(), WriteFile(): Đọc hoặc ghi dữ liệu vào tệp.   DeleteFile(): Xóa một tệp tin.   CopyFile(), MoveFile(): Sao chép hoặc di chuyển tệp tin. |
| **Hàm API Quản lý Bộ nhớ (Memory Management APIs)** | Cung cấp các thao tác quản lý bộ nhớ, như cấp phát hoặc giải phóng bộ nhớ. |  VirtualAlloc(), VirtualFree(): Cấp phát và giải phóng vùng nhớ.   GlobalAlloc(), GlobalFree(): Cấp phát và giải phóng bộ nhớ toàn cục. |
| **Hàm API Xử lý Quy trình và Luồng (Process and Thread Management APIs)** | Được sử dụng để tạo và quản lý các quy trình (process) và luồng (thread). |  CreateProcess(): Tạo một quy trình mới.   CreateThread(): Tạo một luồng mới.   TerminateProcess(): Kết thúc một quy trình.   WaitForSingleObject(): Chờ một quy trình hoặc luồng hoàn thành. |
| **Hàm API Quản lý Registry (Registry APIs)** | Cho phép thao tác với Windows Registry, hệ thống cơ sở dữ liệu để lưu trữ cấu hình hệ thống và ứng dụng. |  RegOpenKeyEx(): Mở một khóa trong Registry.   RegSetValueEx(): Thiết lập giá trị cho một khóa Registry.   RegDeleteKey(), RegDeleteValue(): Xóa khóa hoặc giá trị trong Registry. |
| **Hàm API Giao tiếp Mạng (Networking APIs)** | Hỗ trợ các thao tác giao tiếp mạng, bao gồm kết nối internet, gửi/nhận dữ liệu qua mạng. |  WSAStartup(), WSACleanup(): Khởi tạo và dọn dẹp thư viện Socket.   socket(): Tạo một socket để truyền thông mạng.   connect(), listen(), accept(): Thiết lập kết nối mạng.   send(), recv(): Gửi và nhận dữ liệu qua mạng |
| **Hàm API Xử lý Cửa sổ và Thao tác Giao diện (Window and User Interface APIs)** | Dùng để tạo, quản lý và tương tác với các cửa sổ giao diện người dùng. |  CreateWindowEx(): Tạo một cửa sổ mới.   ShowWindow(), UpdateWindow(): Hiển thị và cập nhật cửa sổ.   GetMessage(), DispatchMessage(): Xử lý thông điệp từ hệ thống.   MessageBox(): Hiển thị hộp thoại thông báo. |
| **Hàm API Mã hóa và Giải mã (Cryptography APIs)** | Cung cấp các chức năng mã hóa, giải mã, tạo hash để bảo mật dữ liệu. |  CryptAcquireContext(): Lấy ngữ cảnh mật mã.   CryptEncrypt(), CryptDecrypt(): Mã hóa và giải mã dữ liệu.   CryptGenRandom(): Sinh số ngẫu nhiên an toàn. |
| **Hàm API Hệ thống Thời gian và Lịch (Time and Date APIs)** | Cung cấp các thao tác để lấy và thiết lập thời gian, ngày tháng. |  GetSystemTime(), SetSystemTime(): Lấy và thiết lập thời gian hệ thống.   Sleep(): Tạm dừng thực thi trong khoảng thời gian nhất định. |
| **Hàm API Xử lý Sự kiện (Event APIs)** | Dùng để đồng bộ hóa và quản lý các sự kiện trong lập trình đa luồng. |  CreateEvent(): Tạo một sự kiện đồng bộ.   SetEvent(), ResetEvent(): Thiết lập hoặc reset một sự kiện.   WaitForMultipleObjects(): Chờ nhiều đối tượng sự kiện. |

Trong kỹ thuật phân tích dịch ngược, **API Hooking** là một phương pháp giúp người phân tích hoặc nhà phát triển có thể "chen ngang" và can thiệp vào quá trình gọi hàm API trong một ứng dụng hoặc hệ thống. Điều này cho phép họ theo dõi, kiểm soát hoặc thay đổi hành vi của các hàm API mà ứng dụng sử dụng. API Hooking thường được sử dụng để:

* **Theo dõi** các hàm API mà ứng dụng gọi để hiểu luồng hoạt động của nó.
* **Phân tích mã độc** bằng cách giám sát các API liên quan đến hệ thống file, registry, và mạng để hiểu hành vi của mã độc.
* **Kiểm tra bảo mật** và bảo vệ hệ thống bằng cách ngăn chặn các hành động không mong muốn hoặc gian lận.
* **Thay đổi hành vi** của các ứng dụng bằng cách điều chỉnh đầu vào/đầu ra của các hàm API.

**Vậy có những loại API HOOKING nào?**

**Các loại API Hooking**

API Hooking có thể được thực hiện bằng nhiều kỹ thuật khác nhau, tùy thuộc vào cách mà lập trình viên muốn can thiệp vào quá trình gọi API. Dưới đây là các loại API Hooking chính:

**1. Inline Hooking**

* **Cách thức hoạt động**: Inline Hooking thay đổi mã nguồn của hàm API trực tiếp trong bộ nhớ. Điều này được thực hiện bằng cách sửa đổi một số byte đầu tiên của hàm API để chuyển hướng nó đến hàm tùy chỉnh (custom function).
* **Ưu điểm**: Đơn giản và hiệu quả, ít ảnh hưởng đến hiệu suất.
* **Nhược điểm**: Phụ thuộc vào kiến trúc CPU, dễ bị phát hiện bởi các phần mềm chống gian lận hoặc phần mềm bảo mật.
* **Ứng dụng**: Phổ biến trong các phần mềm phân tích mã độc hoặc điều tra hành vi của chương trình.

**2. Import Address Table (IAT) Hooking**

* **Cách thức hoạt động**: IAT Hooking thay đổi địa chỉ hàm trong bảng IAT của một module, nơi lưu trữ địa chỉ của các hàm API được ứng dụng gọi đến. Khi ứng dụng cố gắng gọi API, nó sẽ được chuyển hướng đến hàm tùy chỉnh.
* **Ưu điểm**: Không cần thay đổi mã nguồn của hàm API, dễ triển khai.
* **Nhược điểm**: Chỉ hiệu quả khi ứng dụng sử dụng bảng IAT để tra cứu hàm API; dễ bị phát hiện.
* **Ứng dụng**: Theo dõi các API được gọi trong ứng dụng, đặc biệt trong các ứng dụng Windows.

**3. Export Address Table (EAT) Hooking**

* **Cách thức hoạt động**: EAT Hooking can thiệp vào bảng EAT của DLL để thay đổi địa chỉ các hàm API ngay tại nơi export của thư viện, không chỉ ảnh hưởng đến một module mà đến tất cả các module sử dụng API từ thư viện đó.
* **Ưu điểm**: Có phạm vi rộng hơn IAT Hooking; hữu dụng khi ứng dụng gọi trực tiếp hàm từ DLL.
* **Nhược điểm**: Khó triển khai hơn IAT Hooking và có thể gây ra lỗi ngoài ý muốn cho các module khác.
* **Ứng dụng**: Theo dõi hoặc ngăn chặn các API trên toàn hệ thống, ví dụ như kiểm tra việc sử dụng các API mạng hoặc file.

**4. Virtual Table (VTable) Hooking**

* **Cách thức hoạt động**: Được sử dụng với các đối tượng trong lập trình hướng đối tượng, VTable Hooking thay đổi địa chỉ hàm trong bảng hàm ảo (vtable) của một đối tượng để chuyển hướng các cuộc gọi đến hàm khác.
* **Ưu điểm**: Rất hữu dụng khi làm việc với các đối tượng COM trong Windows.
* **Nhược điểm**: Giới hạn trong các đối tượng có VTable, chủ yếu trong các ngôn ngữ hướng đối tượng như C++.
* **Ứng dụng**: Thường dùng trong các ứng dụng Windows có sử dụng đối tượng COM.

**5. Detour Hooking**

* **Cách thức hoạt động**: Detour Hooking chèn mã để chuyển hướng luồng điều khiển của một hàm API đến hàm tùy chỉnh, rồi từ đó có thể gọi API gốc hoặc thực hiện các tác vụ khác.
* **Ưu điểm**: Phổ biến và hiệu quả cao, Detours Library của Microsoft là một trong những thư viện thực hiện detour phổ biến nhất.
* **Nhược điểm**: Có thể ảnh hưởng đến hiệu suất nếu không được triển khai đúng cách.
* **Ứng dụng**: Thường dùng để giám sát hoặc thay đổi hành vi của các API hệ thống.

Đâu tiên mình tìm uesr32.dll để mò ra MessageBoxA

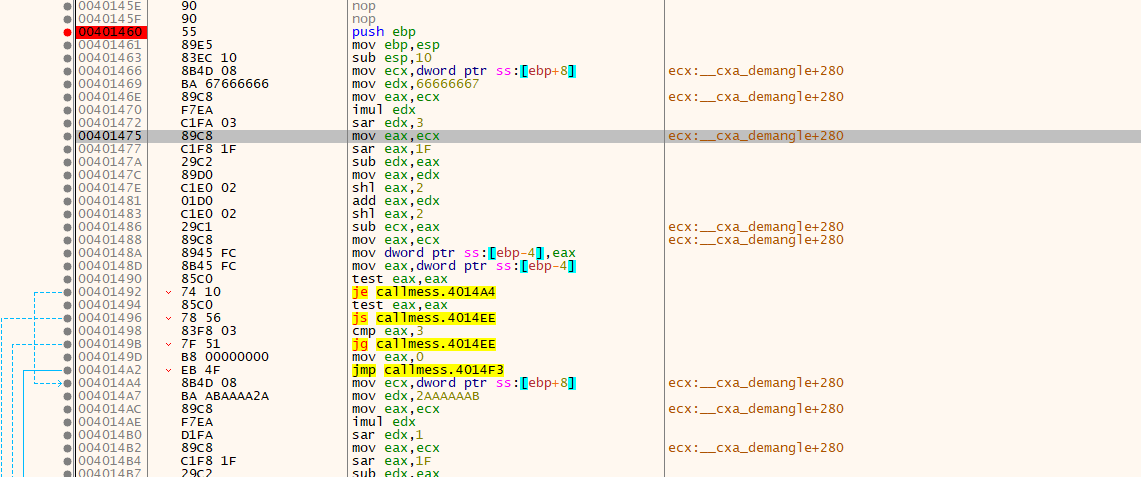
Nó có địa chỉ là 7760ADC0

Rất khó để có thể xác định được nơi chương trình mính bắt đâu, nến tôi sẽ tìm kiếm theo string “Enter” để nhảy đến vị trí của thằng main(). Sau đó tôi cũng xác định vị trí các điểm nhảy và lệnh call các fuction()

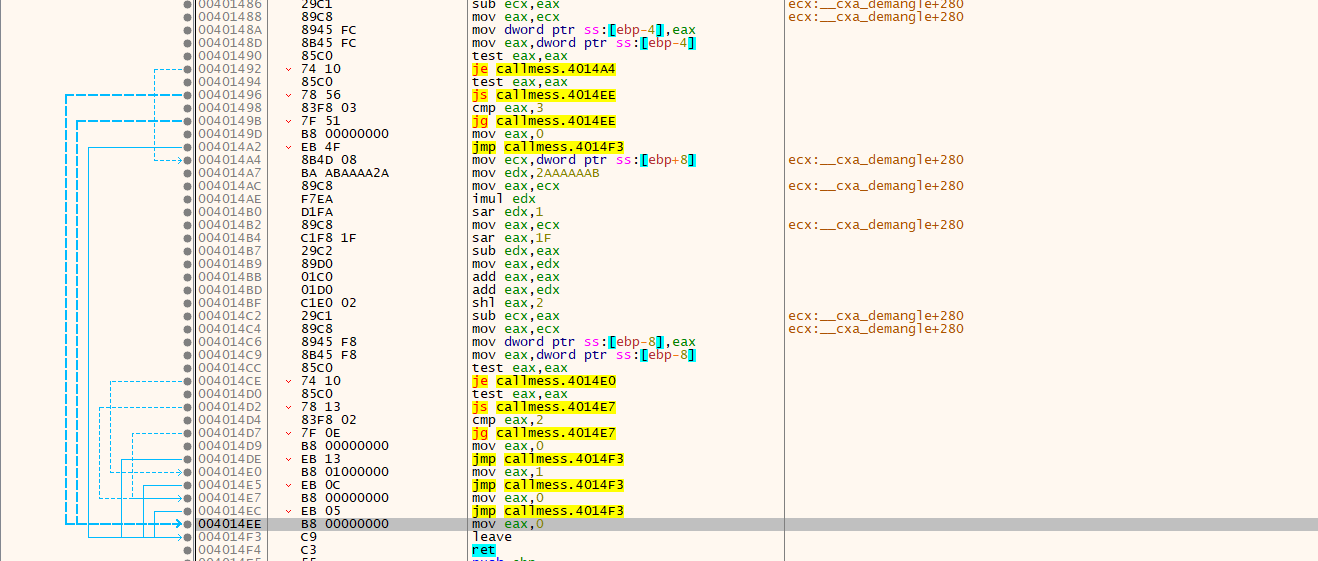
A screenshot of a computer program

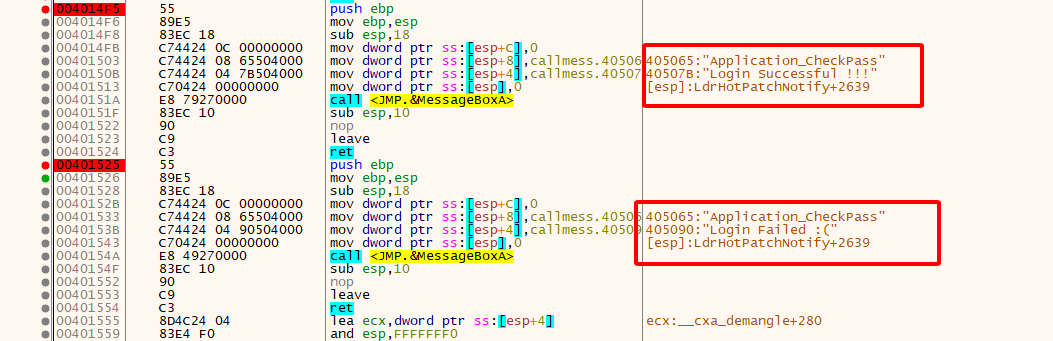
Description automatically generated

Sau đó mình sẽ tiếp tục tìm chỗ *checkpass()* với điểm chỉ bắt đầu là 00401460



Chúng ta sẽ thấy các chỗ tô vàng sẽ là các điểm nhảy với điệu kiện So Sánh:



Tôi cũng xác định luôn được 2 function gọi messagebox ouput kết quả đúng hoặc sai:

**VÀ** bây giờ tôi sẽ tiến hành API Hooking theo kiểu **Inline Hooking**

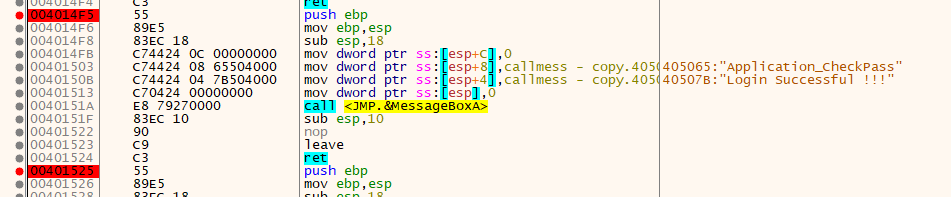
Ở trên tôi đã ghi lại địa chỉ của fucntion checkpass() với mục đích:

1. Nhảy đến đẩu fucntion này
2. Thay đổi mà sửa thành 1 lệnh JMP xxxxxxxnx
3. Tại địa chỉ xxxxxxxnx thì là địa chỉ nơi không có code, nên tôi sẽ viết 1 đoạn ASM để hiển thị messageBOX rằng tôi đã “Application is hacked 😊))))))))”
4. Tiến hành làm.

Đây là trên X32-x64Dbg  
Còn thêm code tôi sẽ **sử** **dụng** **IDA**

*checkpass()* với điểm chỉ bắt đầu là 00401460

Đây là chô fuction hiển thị messageBox và tôi sẽ sẽ thêm 1 đoạn code mới tương tự ở địa chỉa khác:

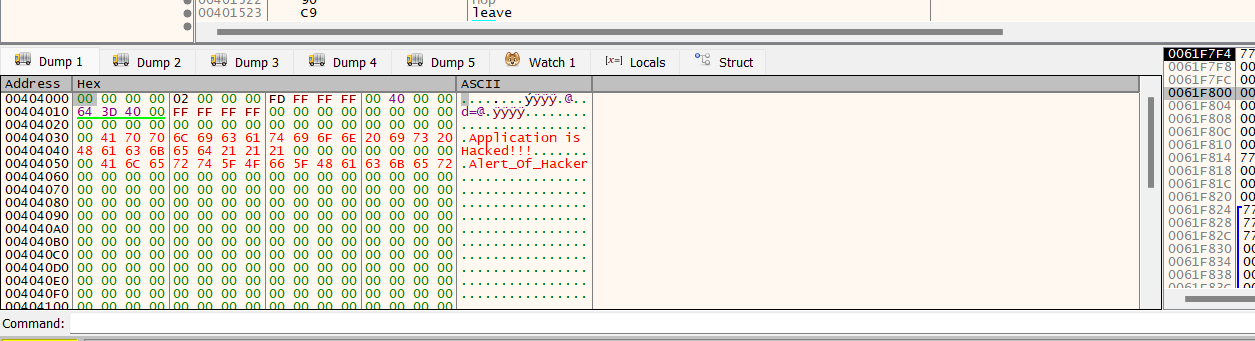


Đầu tiên thêm 2 segment data và code, nên nhớ là phải thêm quyền cho nó:

A screenshot of a computer

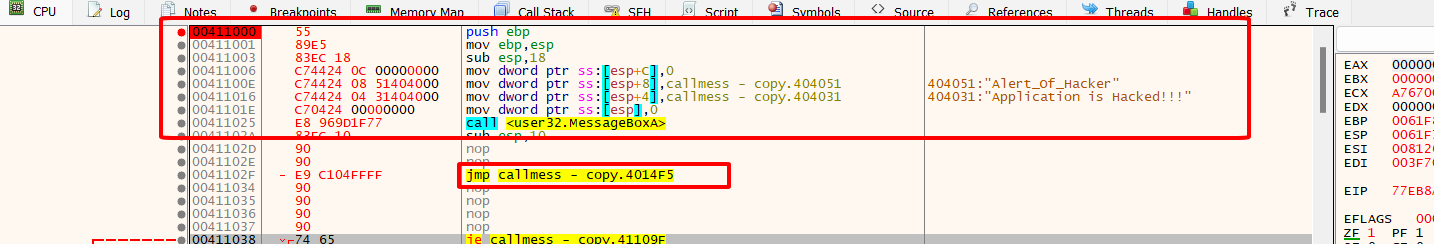
Description automatically generated

Theem title vaf mess:



App = 404031

Alert = 404051

Bắt đầu thêm code tại địa chỉ: 0x00411000 :