LAB 09

Thầy Mai Hoàng Đỉnh
Trường đại học FPT

Người thực hiện

Đặng Hoàng Nguyên

Downloading and Installing IDA Pro

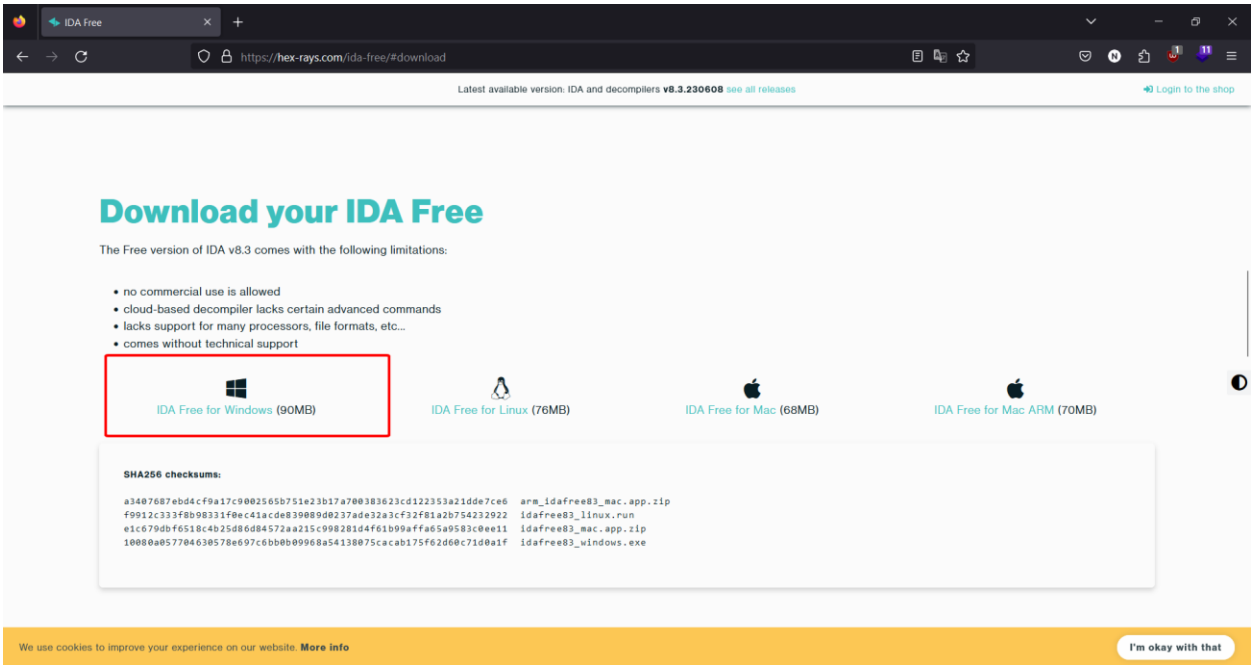
Trong máy Windows, mở trình duyệt Web và truy cập

- https://www.hexrayscomlproducts/ida/support/download_freeware.shtml

Nếu liên kết đó không hoạt động, hãy sử dụng liên kết tải xuống thay thế này:

- https://samsclass.info/126/prQj/idafree50 .exe

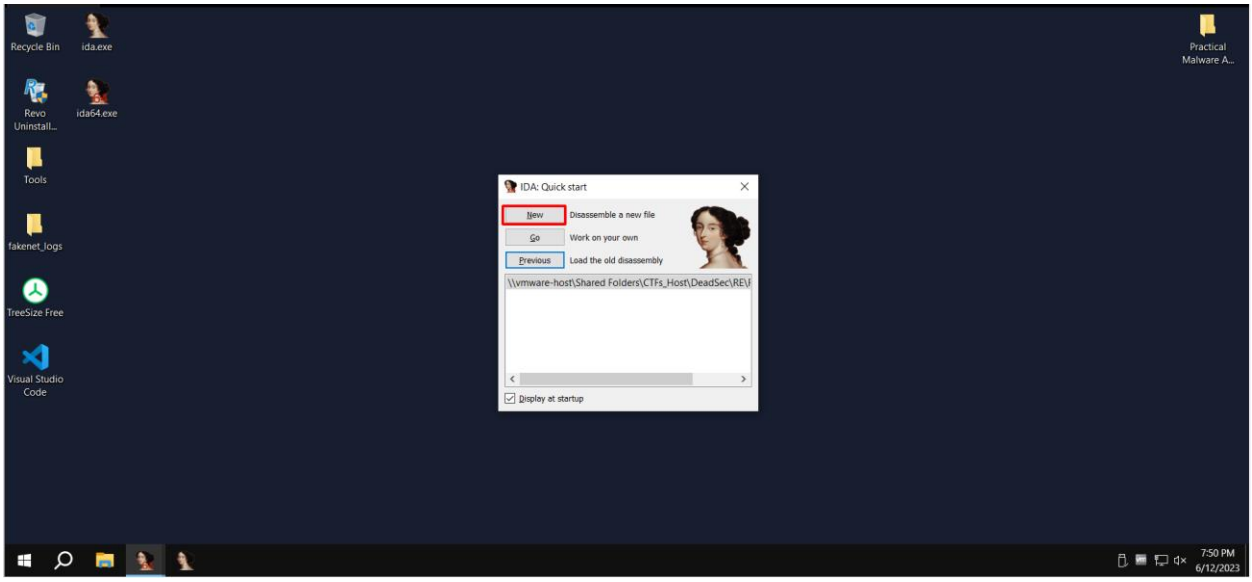
Vì IDA là phần mềm trả phí nên ta chỉ được sử dụng bản free của nó

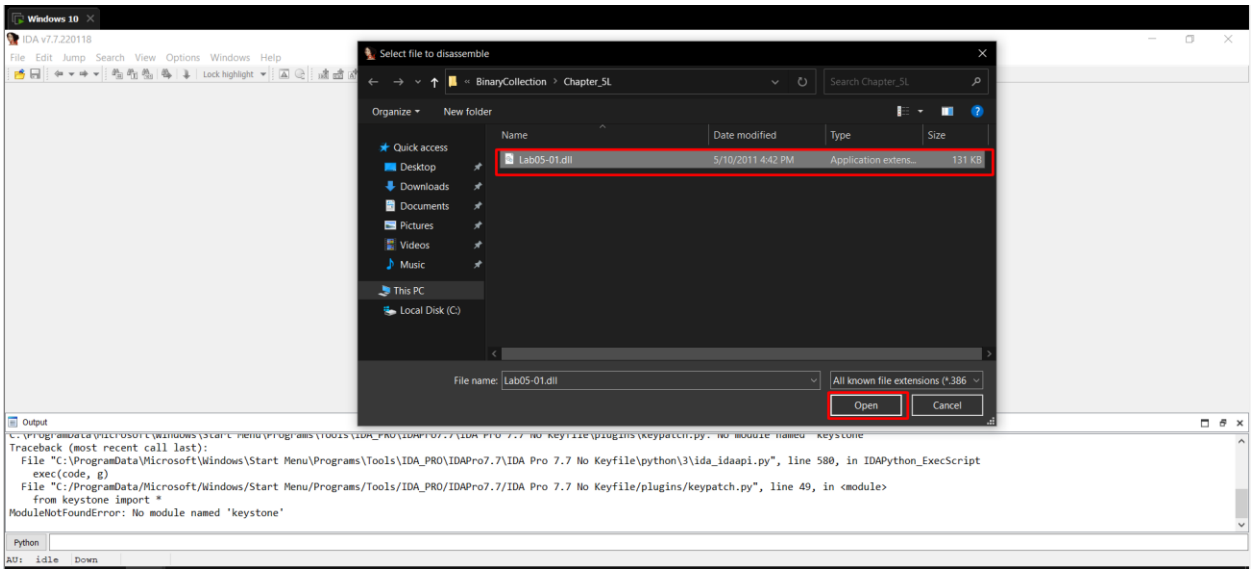


Opening Lab05-01.dll in IDA Pro

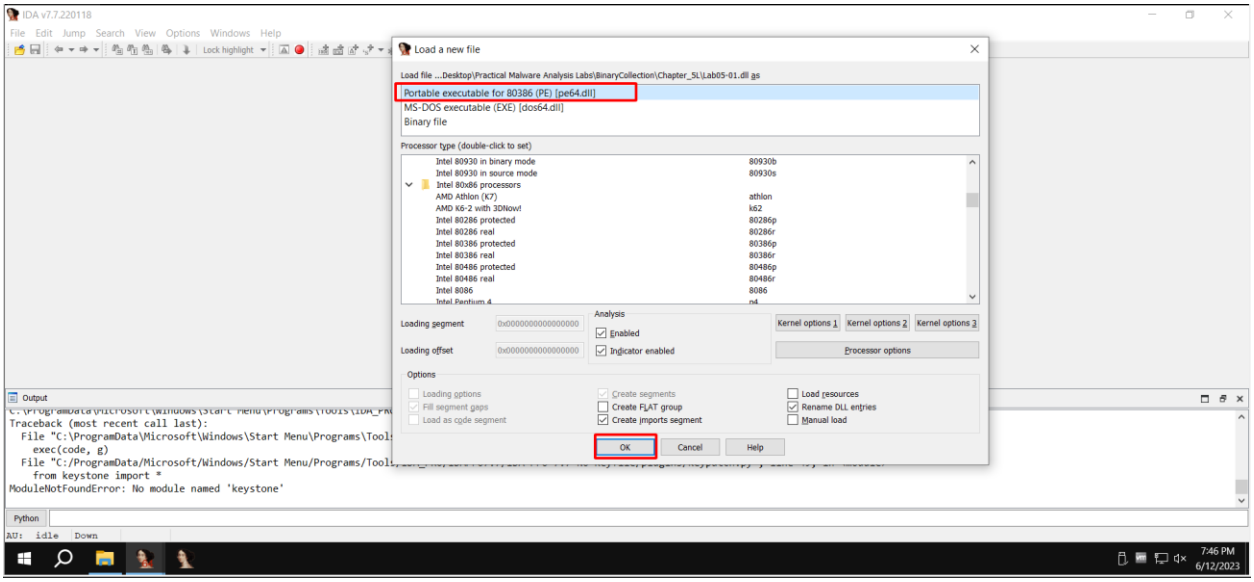
Sau khi download và cài đặt IDA xong, chúng ta sẽ bắt đầu khởi chạy IDA. Sau khi khởi chạy IDA, một khung cửa sổ như sau sẽ xuất hiện, ta sẽ chọn New → Browse ddenss file cần coi. Trong trường hợp này là

- C:\Users \Desktop\Practical Malware Analysis Labs\BinaryCollection\Chapter_5L\Lab05-01.dll



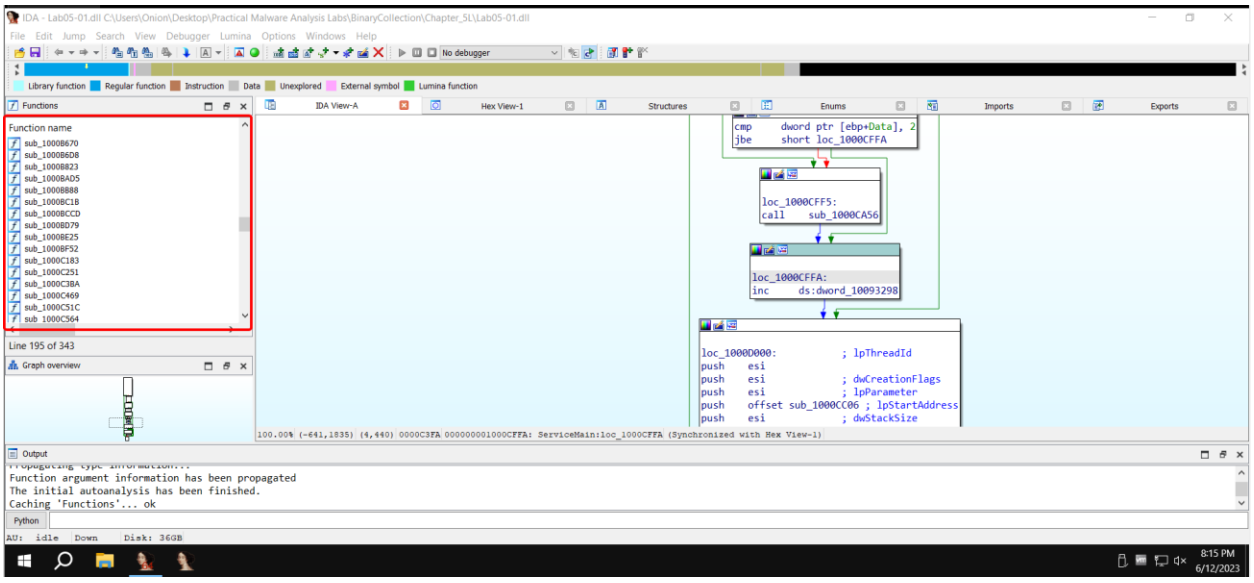


Sau đó chọn **Portable executable for 80386 (PE) [pe64.dlll]** và nhấn chọn **OK**



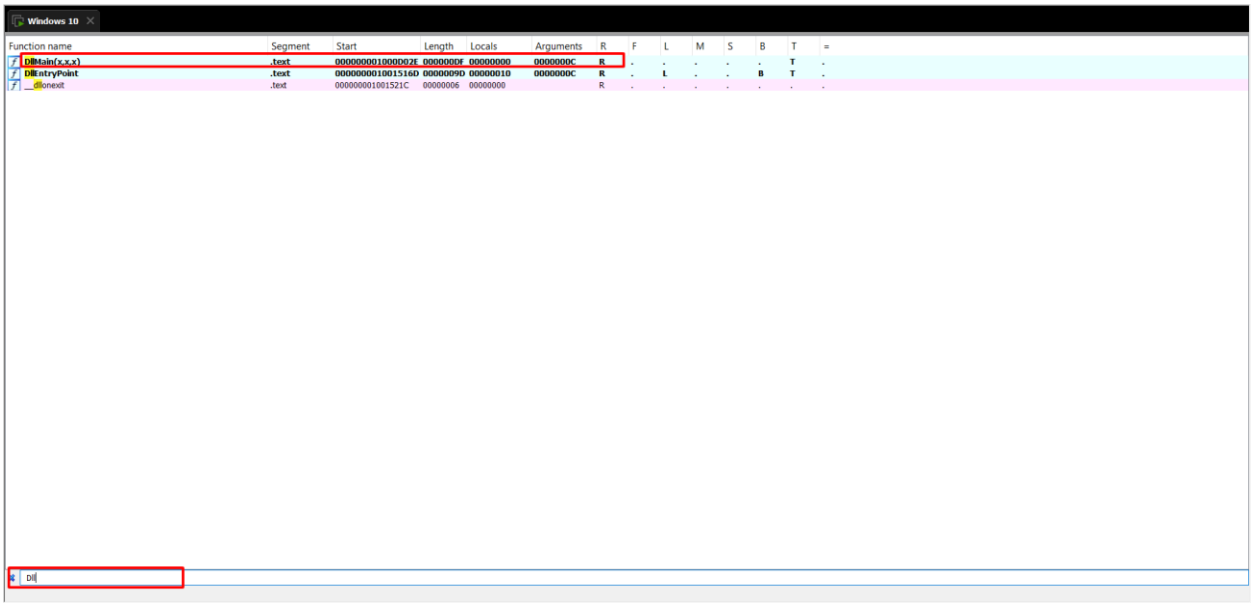
Q 1: Finding the Address of DLLMain

Trong IDA, để muốn xem địa chỉ của DLLmain thì ta có thể nhìn vào phía bên trái của IDA



Và để tìm DLLMain, ta phải biết rằng DllMain sẽ là DllEntryPoint ben trong IDA, việc của chúng ta chỉ có việc tìm đến phần DllEntryPoint là chúng ta biết được địa chỉ của nó là bao nhiêu

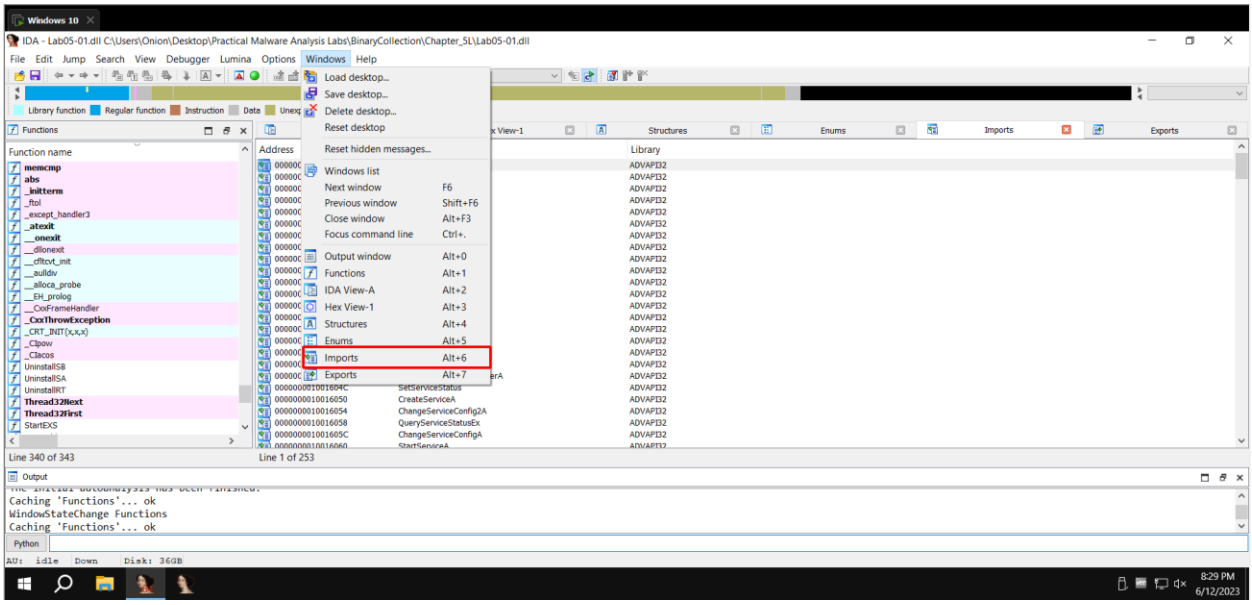
Ta nhấn tổ hợp **Ctrl + F** sau đó search DllEntryPoint là sẽ ra được kết quả



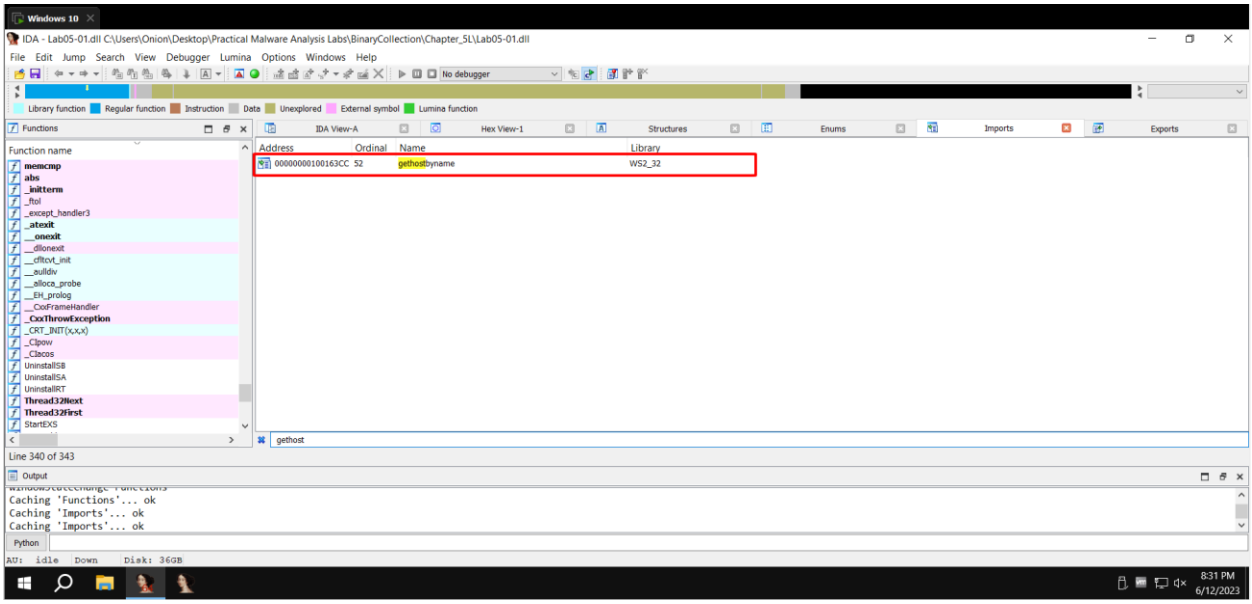
Q 2: Find the import for gethostbyname

Chức năng import cho chúng ta có thể biết rằng thư viện nào được sử dụng và hàm nào trong thư viện hàm đó được gọi ra, đồng thời là địa chỉ hàm được gọi.

Bằng cách nhấn tổ hợp phím **Alt + 6** hoặc trên toolbar, nhấn chọn **Windows → Import**



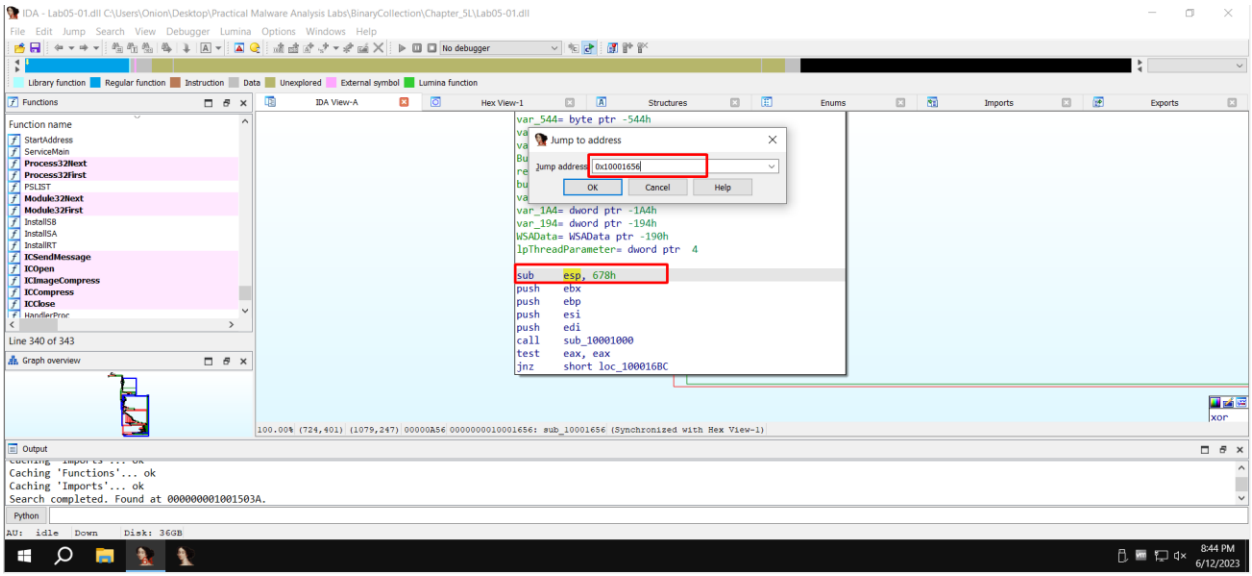
Sau đó chúng ta sẽ sử dụng tổ hợp **Ctrl + F** và tìm đến hàm getbyhostname



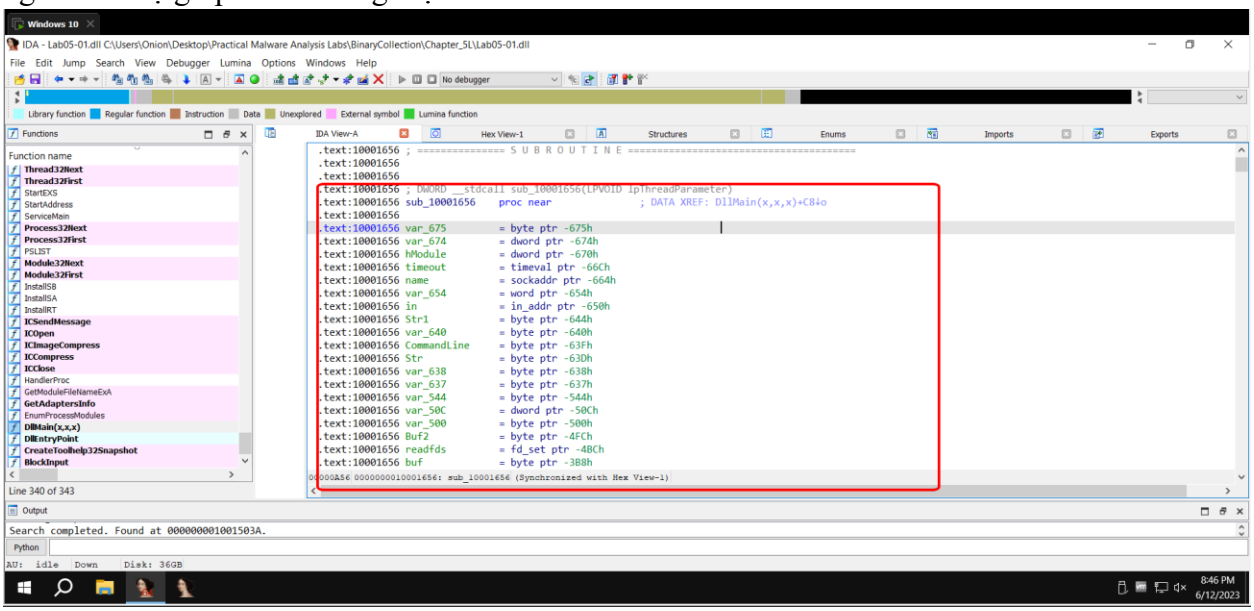
Hàm gethostbyname được sử dụng bởi thư viện WS2_32 được xuất phát từ addres 100163CC

Q 5: Count Local Variables for the Subroutine at 0x10001656

Bâu giờ chúng ta sẽ xem coi có bao nhiêu beiens cục bộ có trong Subroutine tại **0x10001656**. Trong phần IDA-ViewA, chúng ta nhấn **G** và sau đó paste địa chỉ ở bên trên vào.



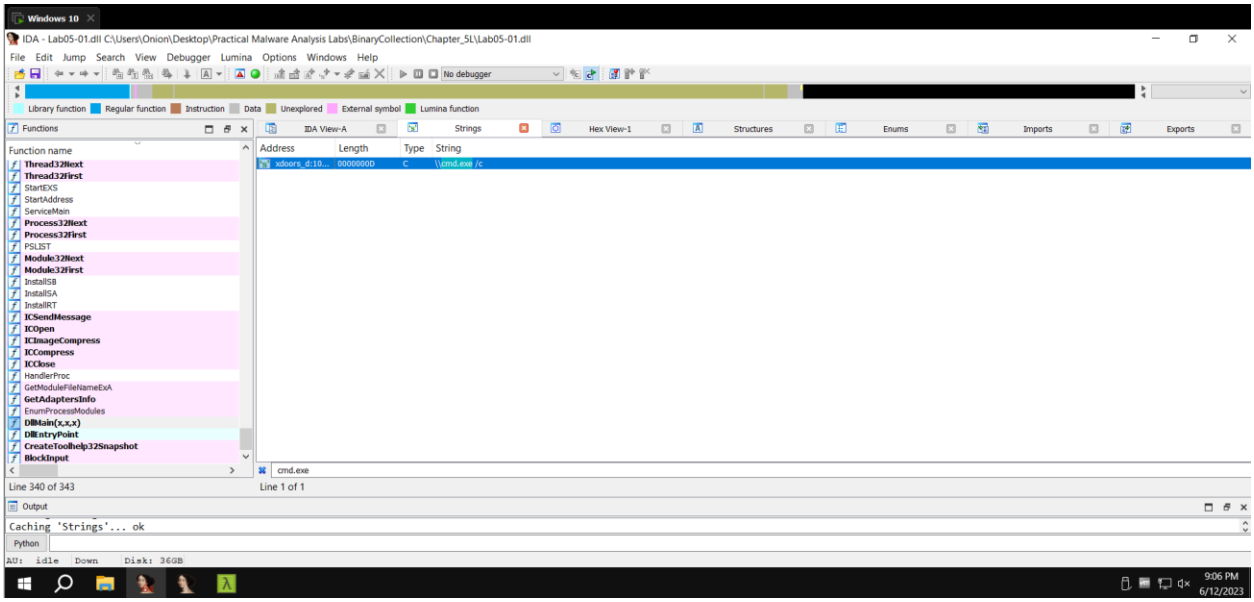
Khi xác định được đâu là phần subroutine, ta nhấn **Spaceboard** để có thể chuyển chế độ khác ngoài chế độ graph view đang hiện hành



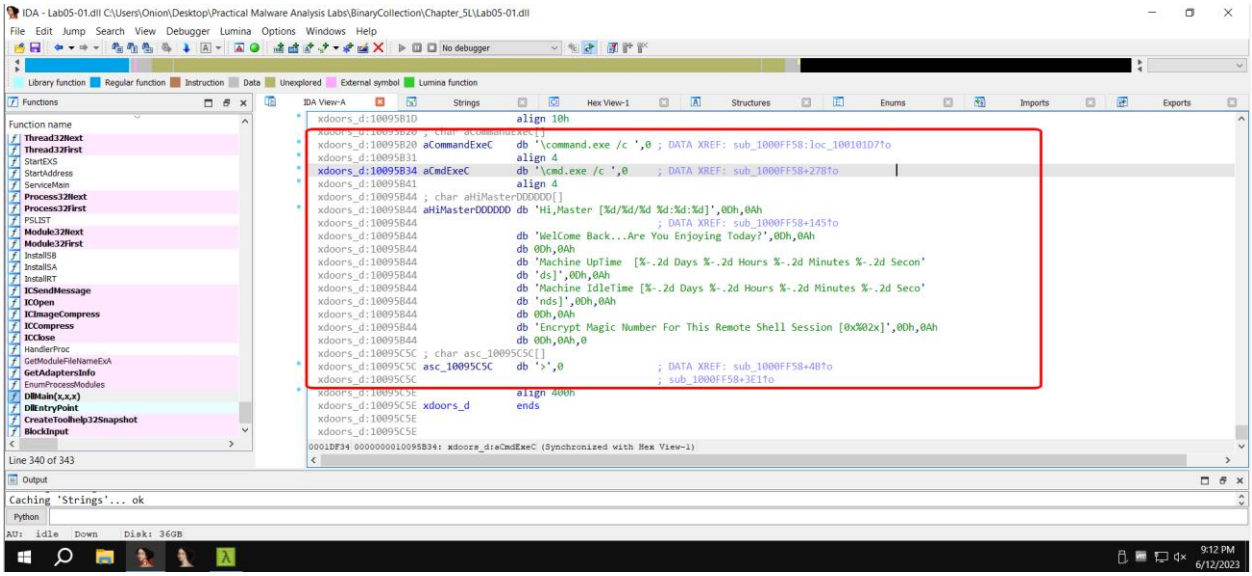
Q 8: Finding the Purpose of the Code that References \cmd.exe /c

Bây giờ, chungs ta sẽ tìm nhwunxg đoạn code nào có string là cmd.exe. Sử dụng cách như sau: **View > Open Subviews > Strings** hoặc có thể sử dụng câu lệnh **Shift + F12**

Sau đó sử dụng câu lệnh Ctrl + F và tìm string “cmd.exe” thì ta thấy rằng ở đây có một string trùng với cmd.exe



Nhấn vào và đến bên trong cmd.exe ta sẽ thấy những gì diễn ra bên trong nó:



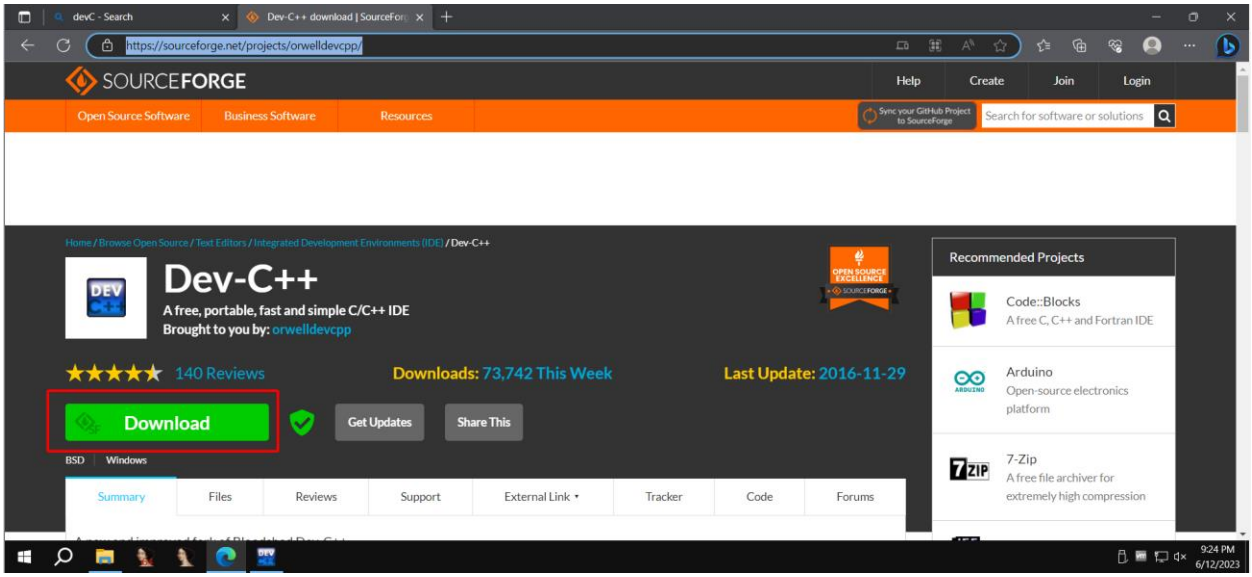
Ta thấy rằng chương trình sẽ chạy bộ đếm thời gian uptime và idletime, sau đó thì Encypr Magic Number.

Disassembling C on Windows

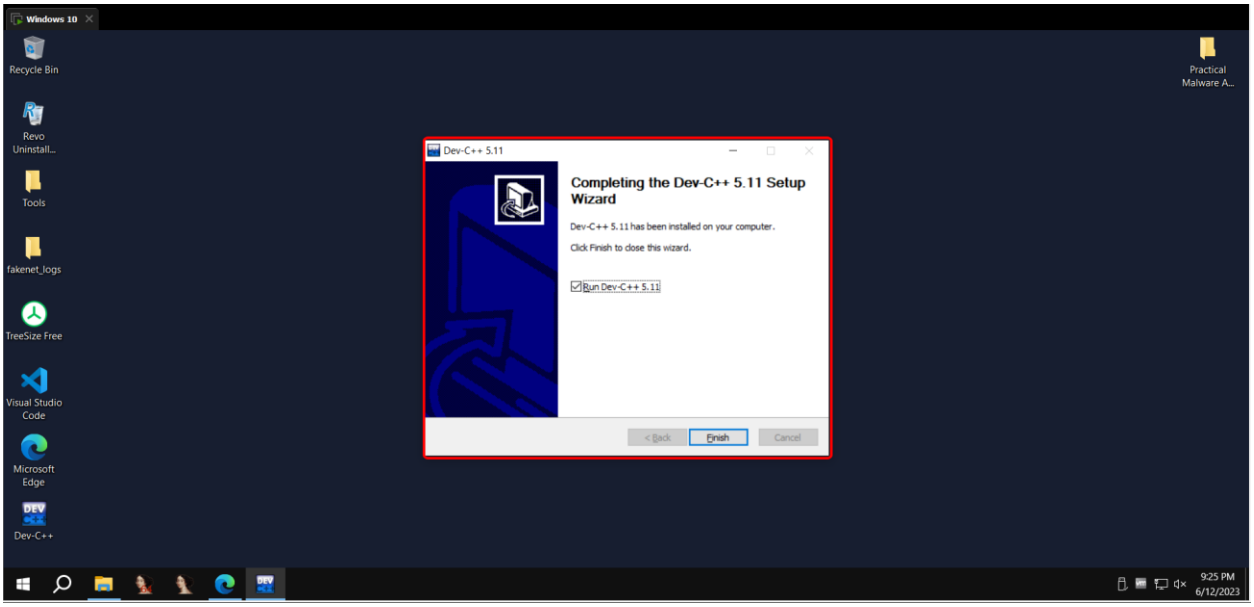
Tại bài lab này, chúng ta phải có một file exe được tạo ra bởi chương trình C. Để làm bài này, ta sẽ download DevC, một công cụ code C.

- https://sourceforge.net/projects/orwelldevcpp/

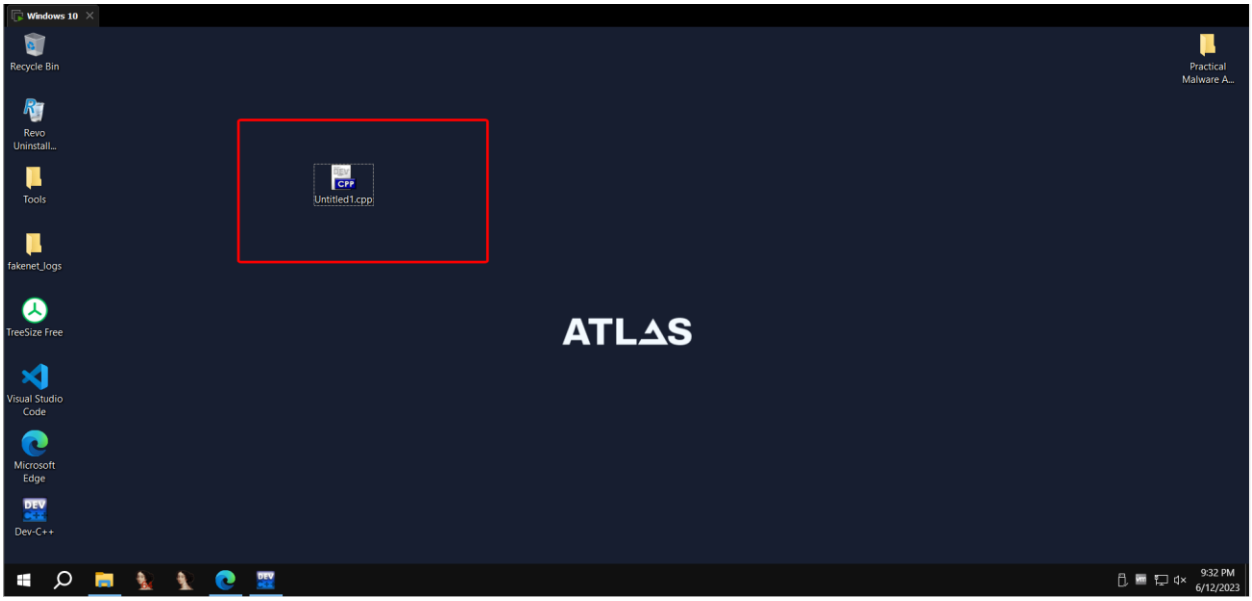
Sau khi tải xong, file download sẽ ở bên trong thư mục Download, ta chỉ cần cho nó chạy cài đặt thôi là được



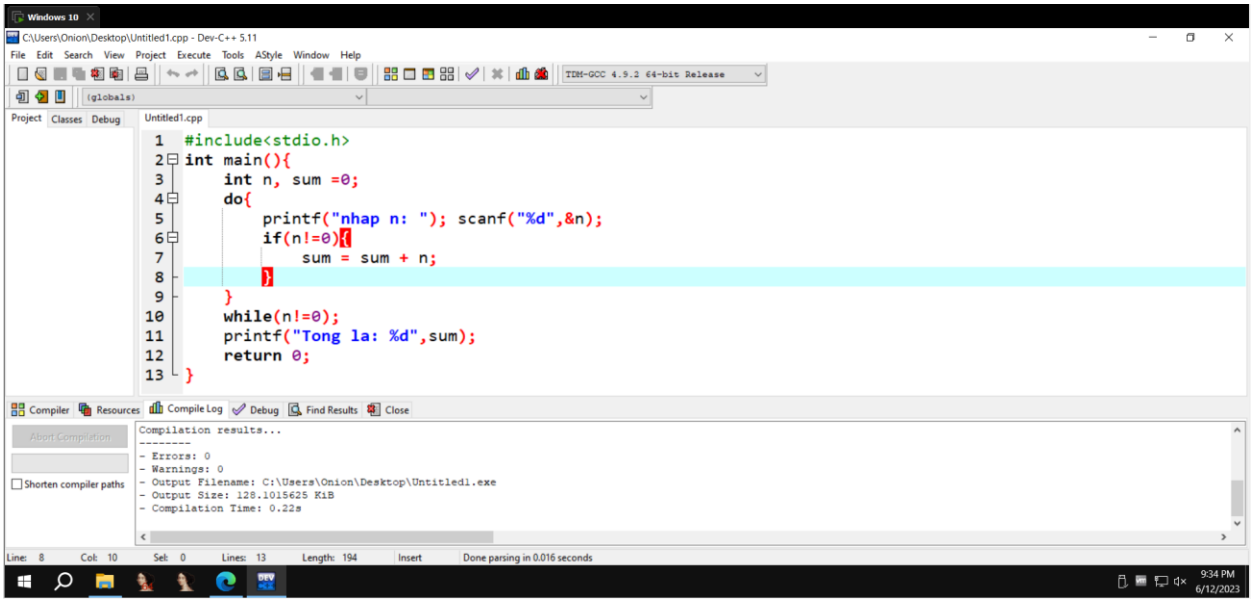
Sau đó ta chỉ đợi cho file cài đặt thôi là được, trong lúc cài đặt, chỉ cần yes hết tất cả là được. Sa khi cài đặt xong hiện ra thông báo như sau:



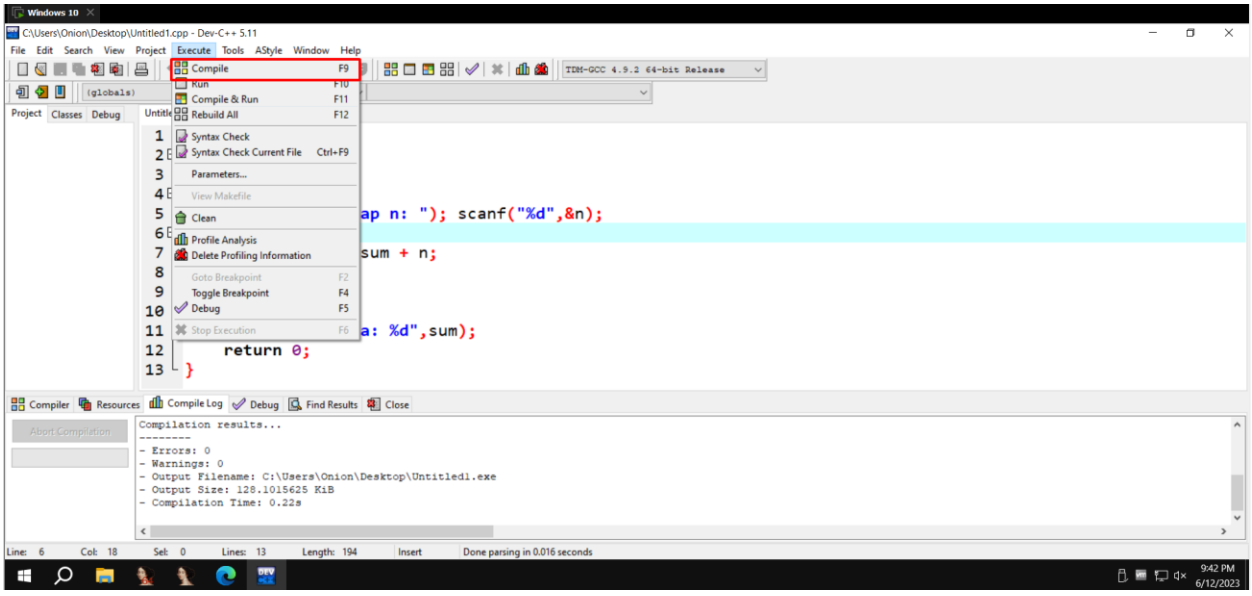
Sau đó, ta sẽ load một file C vào bên trong chương trình DevC.



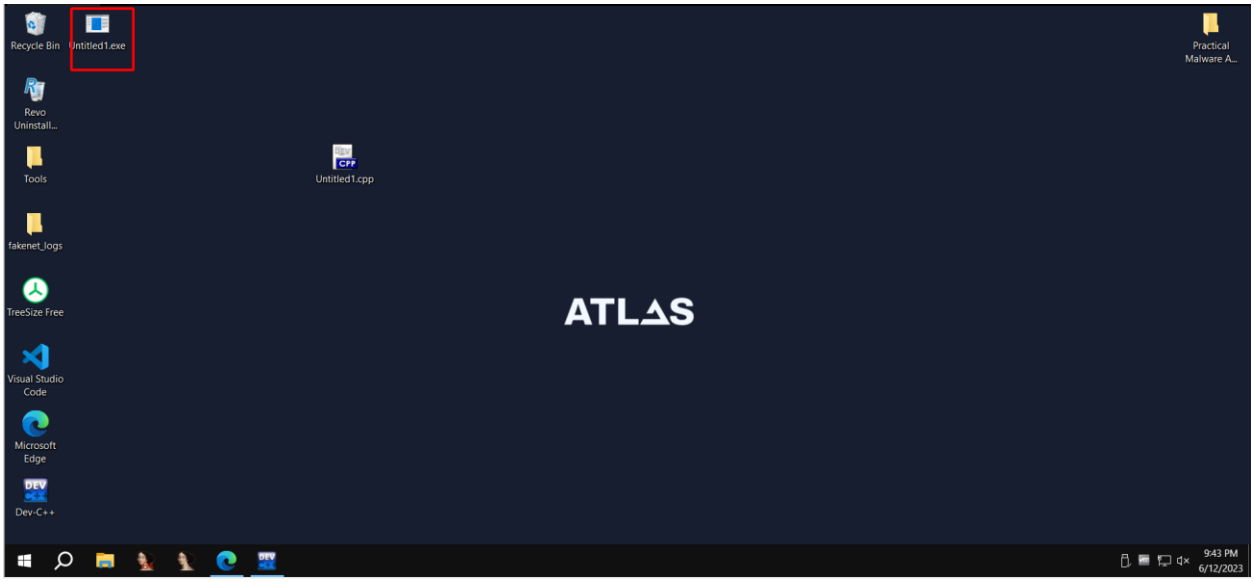
Ta có thể double click vào file Untitled.cpp để file tự load vào DevC++



Ta có thể thấy đây là chương trình C đơn giản, nhập số n và tính tổng cho tới khi nào nhập 0 vào thì chương trình dừng lại và in ra tổng. Sau khi load xong, chúng ta chọn trên thanh toolbar **Execute → Compile** để có thể tạo thành file exe hoặc nhấ phím F9 để chương trình compile

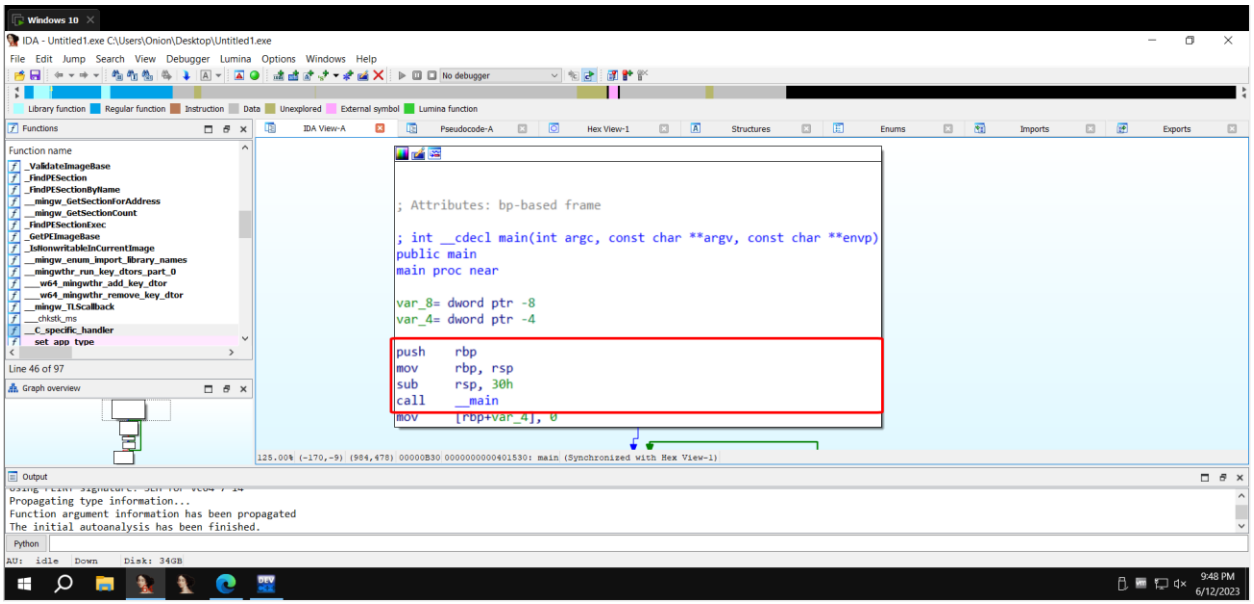


Sau khi compile xong, chương trình sẽ được compile ra cùng thư mục với file code. Sau đó chúng ta bỏ vào IDA để bắt đầu phân tích chương trình



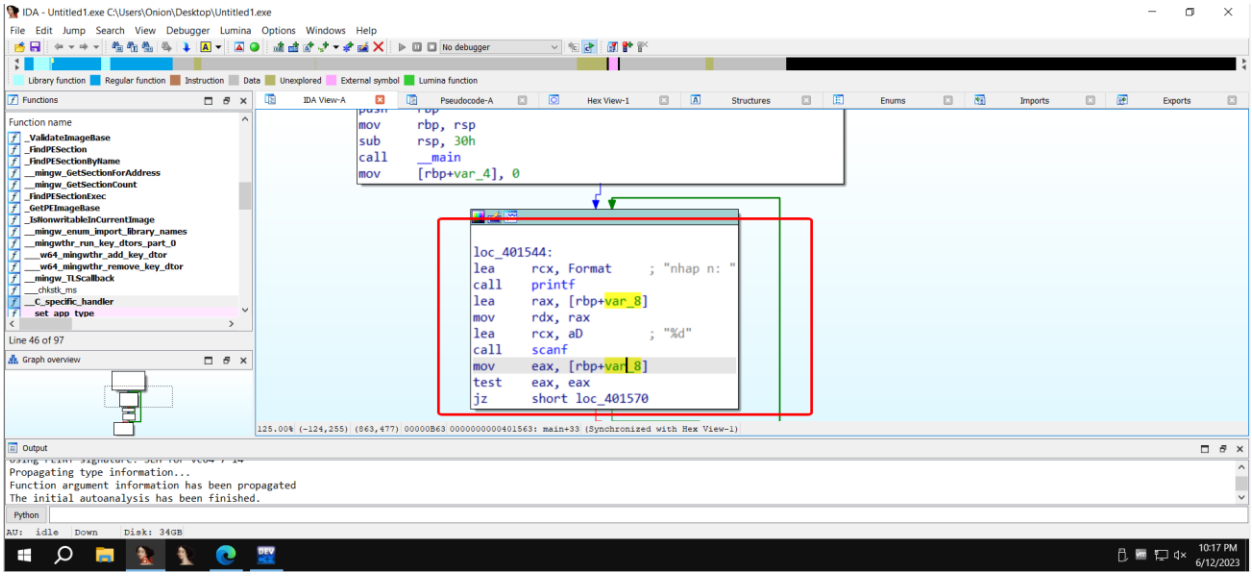
Chúng ta sẽ load chương trình vào bên trong IDA và thực hiện phân tích thông qua HexView.

Ở trong chương trình này, khúc khonah đỏ có nghĩa là tạo ra vùng stack 48 bytes(30 hệ hex) và sau đó sẽ gọi tới hàm main.

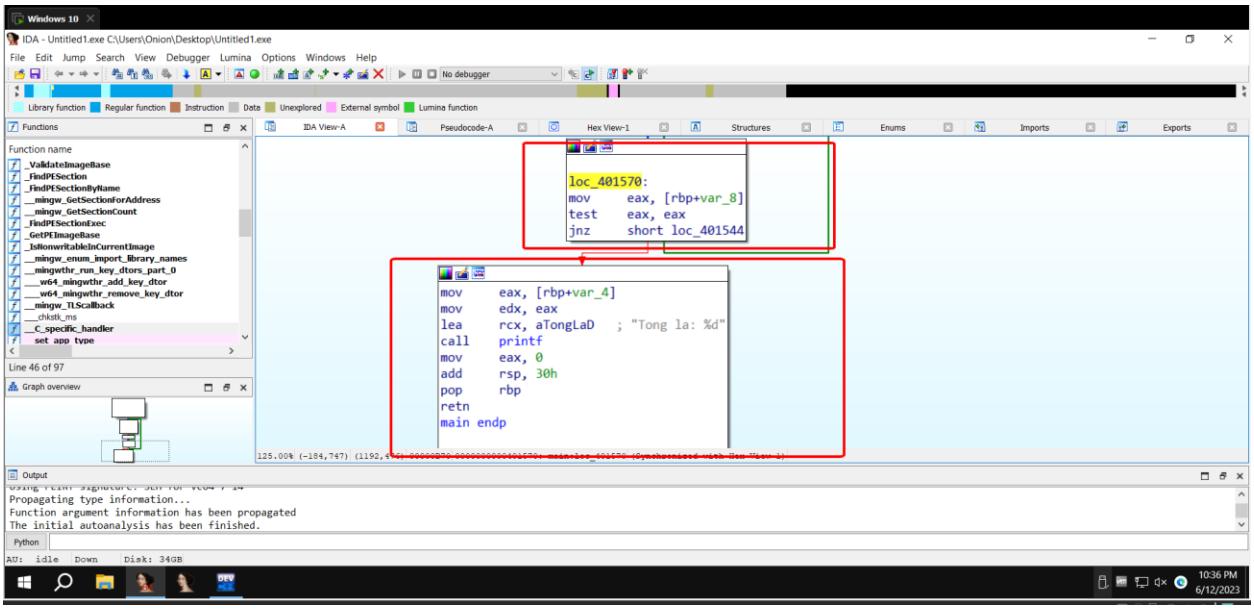


Sau đó tới câu lệnh tiếp theo, ta sẽ thấy nó dung hàm call printf để in ra chữ **“nhap n:”** sau đó tải địa chỉ của biến tại [rbp + var_8] vào thanh ghi `rax`. Biến này sẽ chứa giá trị nhập từ người dùng.

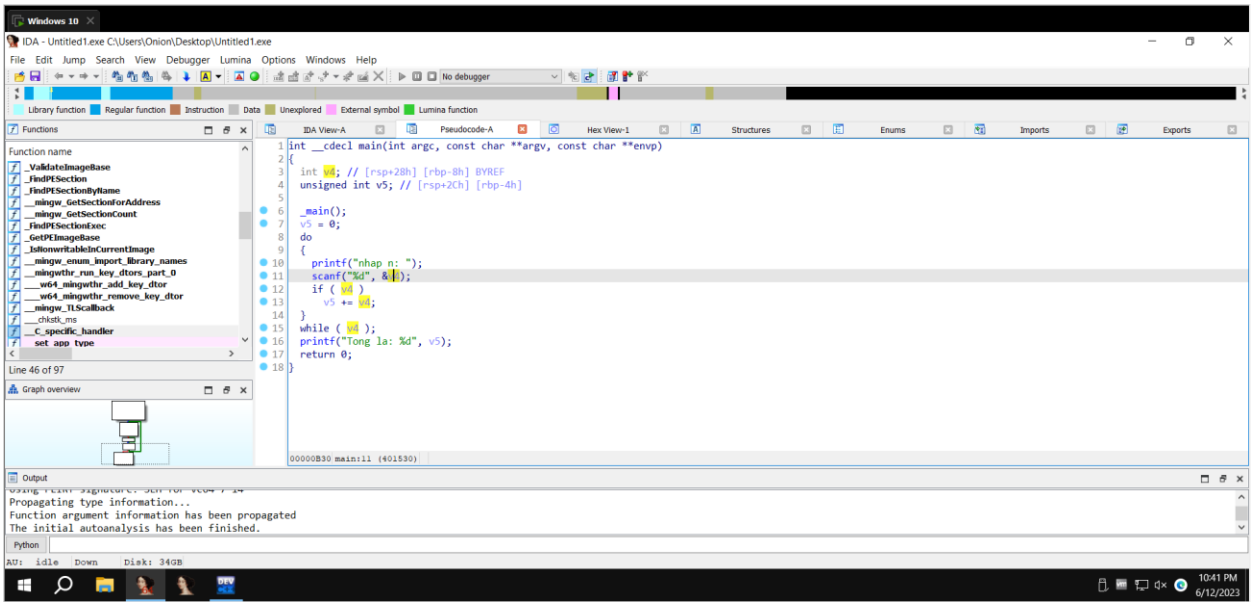
Sau đó giá trị trong thanh ghi `rax` vào thanh ghi `rdx`, được sử dụng như đối số thứ hai cho cuộc gọi hàm `scanf`. Tiếp theo địa chỉ của chuỗi "%d" vào thanh ghi `rcx`, được sử dụng như đối số đầu tiên cho cuộc gọi hàm `scanf`. Và sau đó nó gọi hàm `scanf`, đọc một giá trị số nguyên từ đầu vào của người dùng và lưu trữ nó trong vị trí bộ nhớ được trỏ tới bởi thanh ghi `rdx`. Sau khi làm xong tại rbp-8 sẽ được load vào bên trong eax, và sử dụng hàm test – một phép AND để check xem có phải bằng 0 hay không thông qua loc_401570.



Nhảy đến loc_401570, nó sẽ test xem là có khác không hay không. Nếu đúng thì chương trình sẽ dừng lại, còn không thì xuống dưới, nó sẽ dung hàm printf để in ra “Tong la:” và in số ra, sau đó nso sẽ truyền giá trị eax = 0 và truyền vào stack 48 byte trong stack và pop stack ra khỏi và kết thúc chương trình tại main endp. Nếu trường hợp check eax khác không nó sẽ theo mũi tên chỉa lên trên lại khu scanf và thực hiện tiếp chương trình



Nếu chúng ta muốn ta có thể xem mã giả của chúng bằng cách nhấn C. Tại đây v4 là input của chúng ta và v5 là sum sẽ được in ra


Windows 10
IDA - Untitled1.exe C:\Users\Onion\Desktop\Untitled1.exe
File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
Function name
_ValidateImageBase
_FindPESection
_FindPESectionByName
__mingw_GetSectionForAddress
__mingw_GetSectionCount
_FindPESectionExec
_GetPEImageBase
_IsNonwritableInCurrentImage
__mingw_enum_import_library_names
__mingwthr_run_key_dtors_part_0
__w64_mingwthr_add_key_dtor
__w64_mingwthr_remove_key_dtor
__mingw_TLScallback
___chkstk_ms
__C_specific_handler
__set_app_type
Line 46 of 97
Graph overview
IDA View-A
Pseudocode-A
Hex View-1
Structures
Enums
Imports
Exports
int __cdecl main(int argc, const char **argv, const char **envp)
{
int v4; // [rsp+28h] [rbp-8h] BYREF
unsigned int v5; // [rsp+2Ch] [rbp-4h]
_main();
v5 = 0;
do
{
printf("nhap n: ");
scanf("%d", &v4);
if (v4)
v5 += v4;
}
while (v4);
printf("Tong la: %d", v5);
return 0;
}
00000B30 main:11 (401530)
Output
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 34GB
10:41 PM
6/12/2023