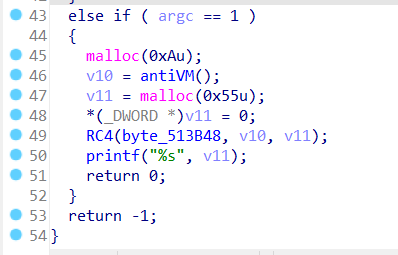
Dùng ida phân tích file

Chương trình có 2 luồng thực thi

1. Luồng thực thi với không có tham số đầu vào argc==1

Chương trình sẽ dùng hàm antiVM để kiểm tra xem có VM hay không nếu có thì sẽ thay đổi giá trị của v10=> làm cho chương trình giải mã ra kết quả bị sai. Cần phải làm cho v10 có giá trị chính xác việc giải mã ra v11 chính xác.



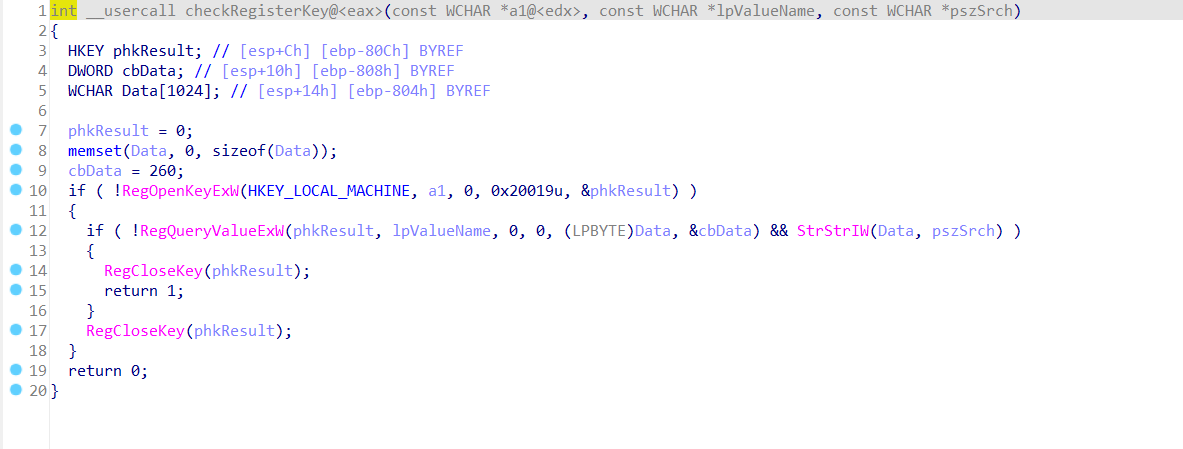
Tiến hành phân tích hàm antiVM

Có tất cả 10 key và key này sẽ ảnh hưởng tới giá trị trả về của hàm

* Key[0] **checkRegisterKey**

Kiểm tra bằng **RegOpenKeyExW** và **RegQueryValueExW**

**+ RegOpenKeyExW mở register key được chỉ định**

**+ RegQueryValueExW truy xuất loại và dữ liệu được liên kết với giá trị của khóa register key được chỉ định**

2 hàm này nếu thành công sẽ trả về **ERROR\_SUCCESS ( 0x0) => cả 2 câu lệnh if đều đúng => giá trị trả về là của hàm checkRegisterKey**. Mà trả về 1 dẫn tới v[0] tăng lên =>key[0] bị sai

Bypass: bỏ qua dòng 112: ++v0; =>làm cho v0 không đổi và có giá trị = 4





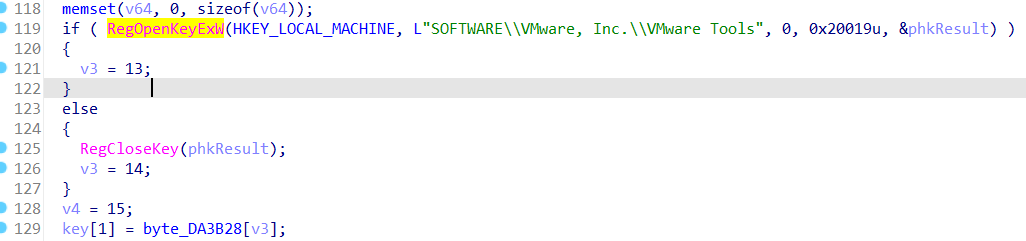
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

[**https://blogchiasekienthuc.com/thu-thuat-may-tinh/huong-dan-toan-tap-ve-registry.html**](https://blogchiasekienthuc.com/thu-thuat-may-tinh/huong-dan-toan-tap-ve-registry.html)

* Key[1] **checkRegisterKey**

Kiểm tra VM bằng cách check register key của VM ware tool

Kiểm tra **SOFTWARE\\VMware, Inc.\\VMware Tools** nếu mở được thì v3 = 14 key[1]=byte\_403B28[14] nếu ko mở được thì key[1]=byte\_403B28[13]



Bypass : cả 2 th dù đều cho v3 =13

|  |  |
| --- | --- |
|  |  |

* Key[2] **GetFileAttributesW**

Kiểm tra trong drivers system32 có vm hay ko bằng hàm **GetFileAttributesW**

**GetFileAttributesW** truy xuất các thuộc tính hệ thống tệp cho 1 tệp hoặc thư mục chỉ định

Nếu thất bại trả về **INVALID\_FILE\_ATTRIBUTES.** Nếu chức năng thành công, giá trị trả về chứa các thuộc tính của tệp hoặc thư mục đã chỉ định. Nếu thành công và trả về **FILE\_ATTRIBUTE\_DIRECTORY** 16 (0x10) thì xác định đây là một thư mục

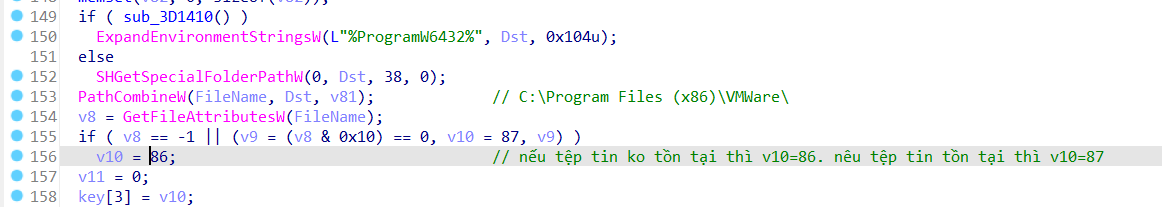


Bypass bằng cách dòng dòng 169 chỉnh thành v7= v3

|  |  |
| --- | --- |
|  |  |

* Key[3] **GetFileAttributesW**

Dùng **GetFileAttributesW** tương tự Key[2] kiểm tra tệp tin **C:\Program Files (x86)\VMWare\** có tồn tại hay ko



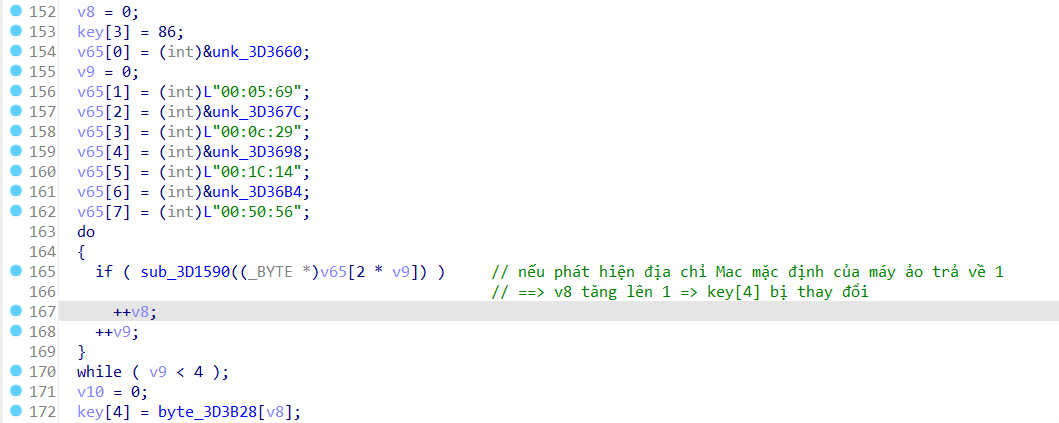
Bypass bằng cách cho v12=86 =>(key[3]= 86)

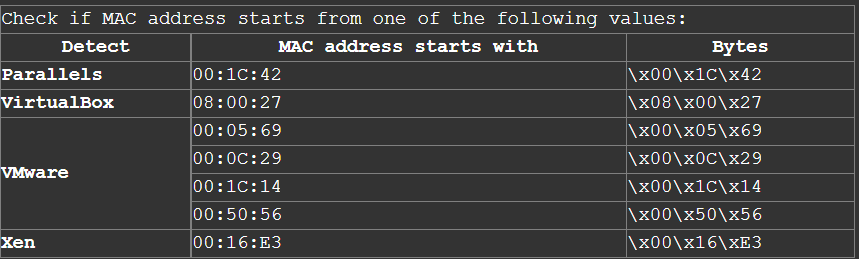
|  |  |
| --- | --- |
|  |  |

* Key[4] **GetAdaptersInfo**

Dùng **GetAdaptersInfo** để kiểm tra địa chỉ MAC của máy.

**GetAdaptersInfo** truy xuất thông tin Adapter của máy



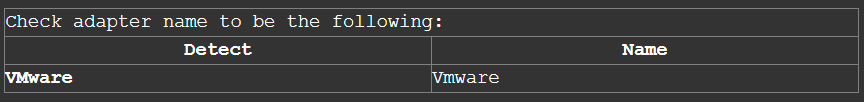


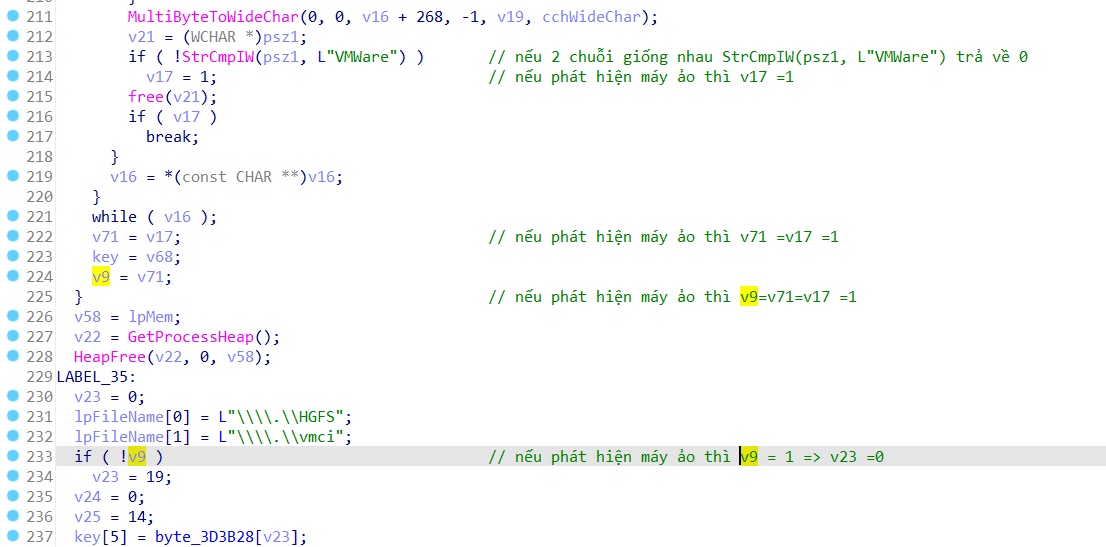
Bypass: bằng cách bỏ qua dòng 167 làm cho v8 ko bị thay đổi và v8 =0

|  |  |
| --- | --- |
|  |  |

* Key[5] **GetAdaptersInfo**

Tương tự Key[4] nhưng kiểm tra tên Adapter





Bypass bằng cách làm v23 luôn bằng 19

|  |  |
| --- | --- |
|  |  |

* Key[6] **CreateFileW**

1. Sử dụng **CreateFileW** để kiểm tra sự tồn tại của VMWare Host Guest File System (HGFS), VMWare Virtual Machine Communication Interface (vmci)

Với dwCreationDisposition =3 OPEN\_EXISTING hàm sẽ thực hiện chức năng mở tệp nếu tệp tồn tại nếu thành công trả về 1 handle. Nếu không thành công thì trả về **INVALID\_HANDLE\_VALUE** và trong trường hợp dwCreationDisposition = 3 OPEN\_EXISTING khi nhận thông tin mở rộng về lỗi **ERROR\_FILE\_NOT\_FOUND** (2).

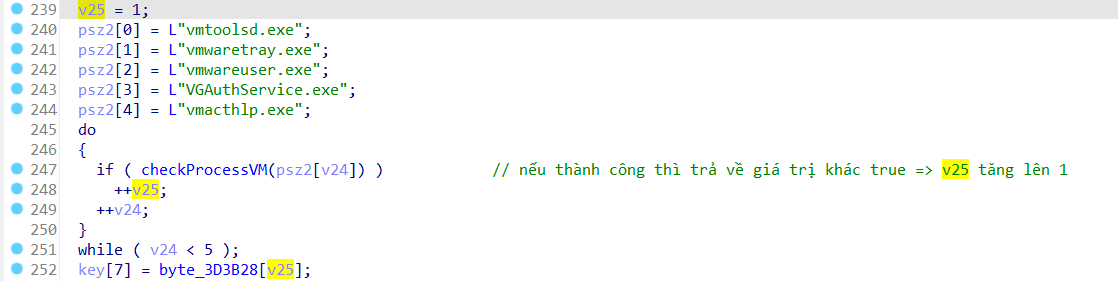


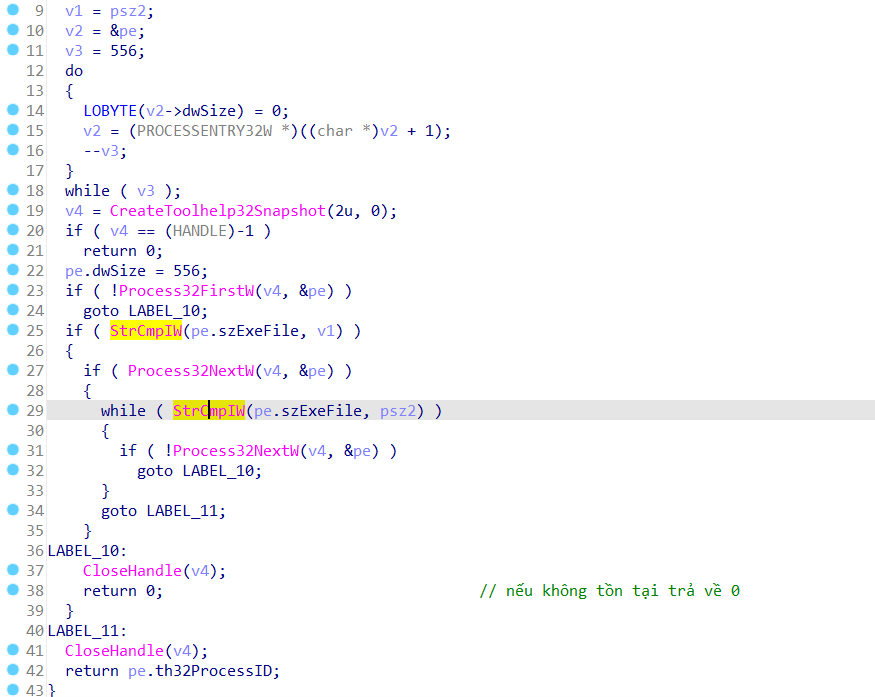
Bypass bằng cách cho v22 ko đổi

|  |  |
| --- | --- |
|  |  |

* key[7] **CreateToolhelp32Snapshot -** **Process32FirstW-** **Process32NextW**

Kiểm tra sự tồn tại của các process Vmware

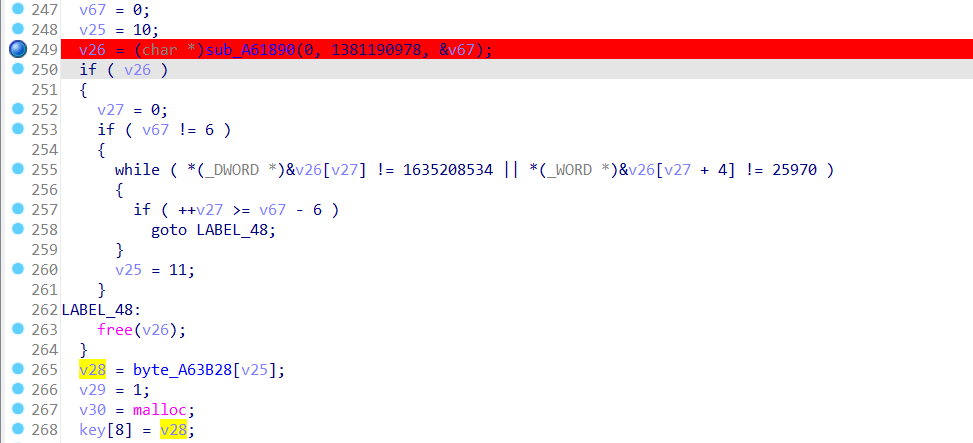




Bypass bằng cách cho v25 ko đổi

|  |  |
| --- | --- |
|  |  |

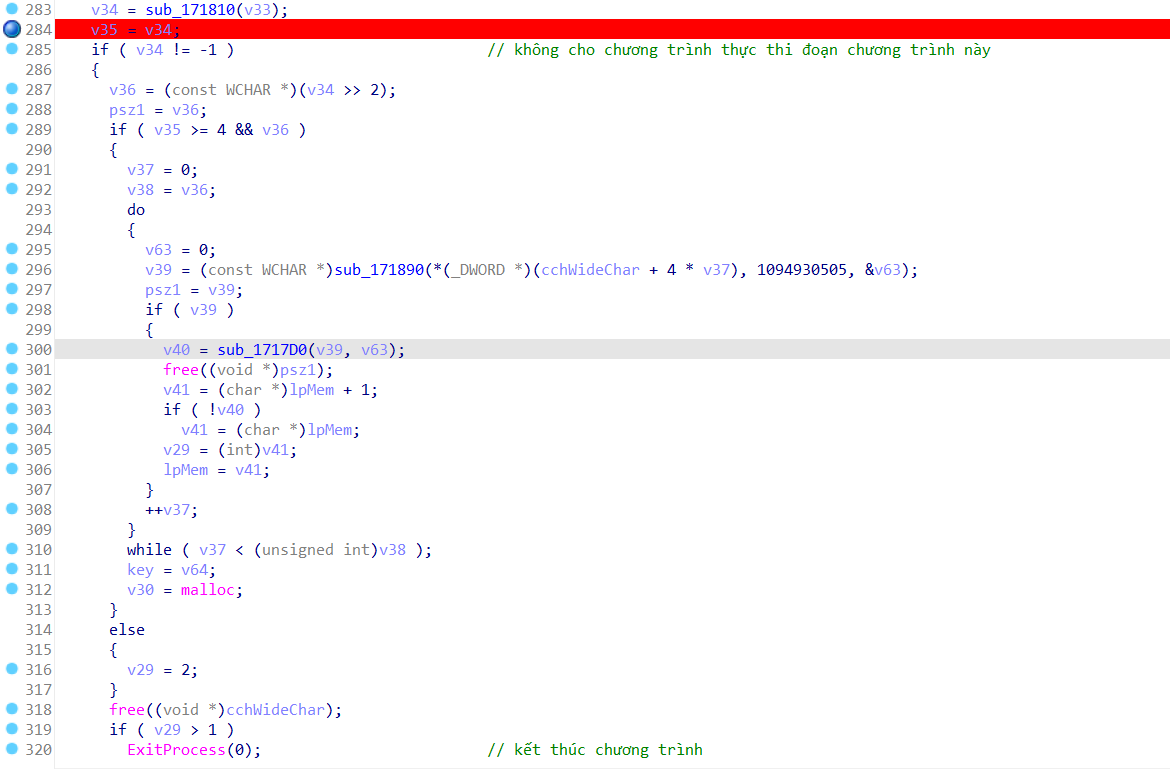
* key[8]



Bypass bằng cách làm cho v25 luôn bằng 10

* key 9 thì mặc định là 0 nên ko cần phải làm gì cả





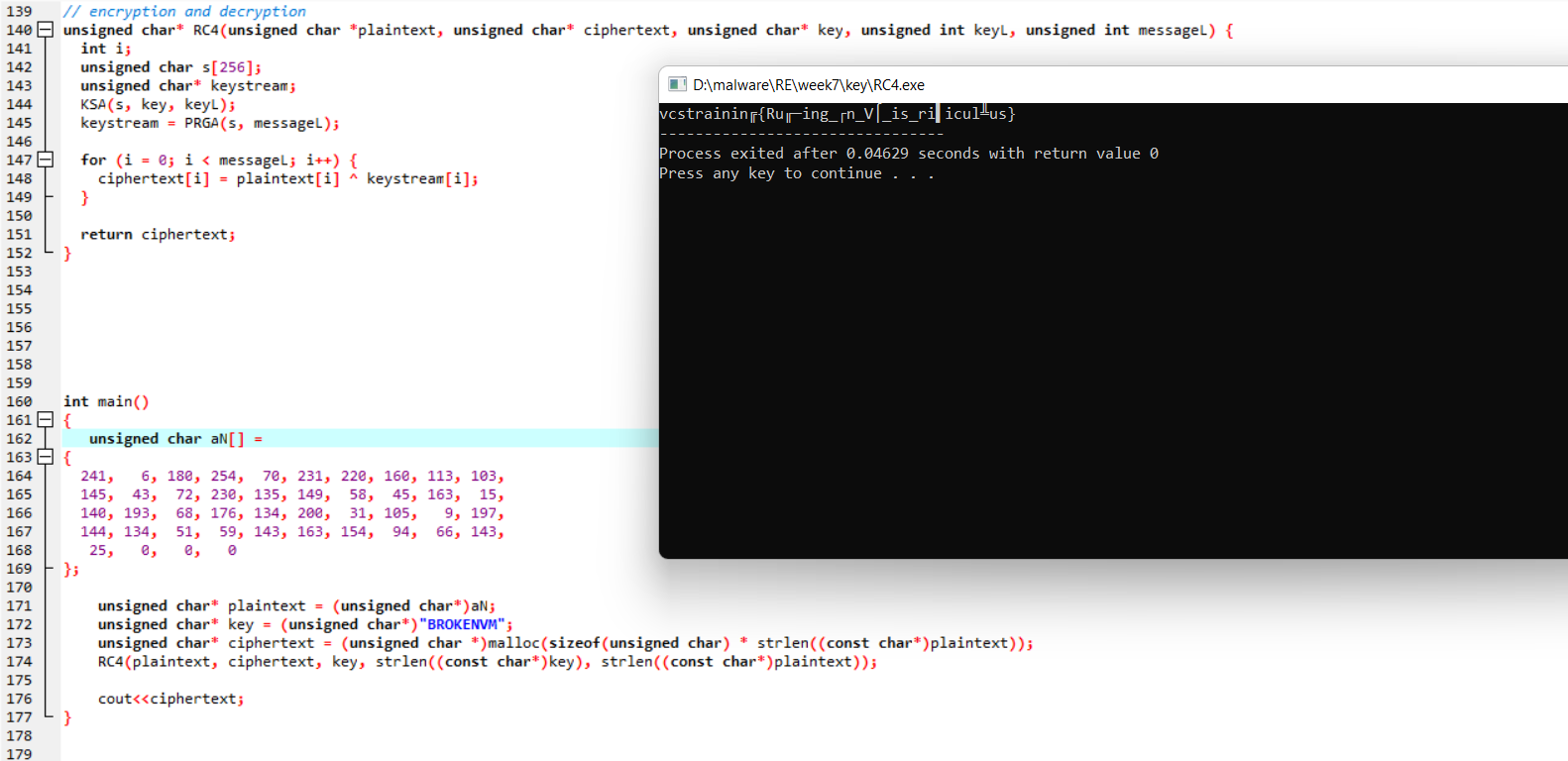
Sau khi vượt qua các kĩ thuật anti VM ở luồng thực thi này ta chạy chương trình có được



ở đây ta tìm được Key là **BROKENVM** ta sẽ sử dụng key này để phân tích luồng thực thi còn lại và tìm ra flag

1. Nếu argc==2 tức là có tham số đầu vào. Chương trình tiền hành biển đổi dựa trên key nhập vào và chuỗi aN đã được cấp phát tiến hành tạo ra flag. Nếu key nhập vào đúng thì sẽ in ra flag đúng. Nếu phân tích hàm này thì hàm này sử dụng mã RC4 để biến đổi mảng aN với key là “BROKENVM” để ra flag





Thực chất sau khi giải mã xong luồng thực thi đầu tiên(luồng thực thi ko tham số) thì đã có thể tìm được flag của bài này mà ko cần quan tâm tới đoạn luồng thực thi này.

