BÁO CÁO THỰC HÀNH

Môn học: Kỹ thuật Phân tích mã độc

Kỳ báo cáo: Buổi 04

Tên chủ đề: Kỹ thuật phân tích tĩnh nâng cao (Advanced Static Analysis)

GVHD: Ngô Đức Hoàng Sơn

Ngày báo cáo: 8/12/2025

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT137.Q11.ANTT

STT	Họ và tên	MSSV	Email
1	Hà Minh Quân	22521177	22521177@gm.uit.edu.vn
2	Từ Chí Kiên	22520713	22520713@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN

STT	Công việc	Thành phần	Kết quả tự đánh giá
1	Phân tích tĩnh nâng cao 1	Packed PE	100%
2	Phân tích tĩnh nâng cao 2 basic-reverse.exe	Hard Coded	100%
		Username và Password	100%
3	Phân tích tĩnh nâng cao 3	unknown.exe	100%

Ghi chú:

Điểm của mỗi buổi Thực hành được tính như sau:

- Phần CP: điểm của phần thực hành bắt buộc trên lớp (+ điểm chuyên cần nếu có)
- Phần HP: điểm của phần thực hành ở nhà

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hành.

BÁO CÁO CHI TIẾT

1. Báo cáo Phân tích tĩnh nâng cao 1

BÁO CÁO PHÂN TÍCH TĨNH NÂNG CAO 1

I. Người phân tích:

Nhóm phân tích: Nhóm 7

Thành viên: Hà Minh Quân - 22521177, Từ Chí Kiên - 22520713

II. Tổng quan về file phân tích:

Tên file: hello.exe

Kích thước: 58KB

Định dạng: ***.exe***

III. Thời gian phân tích:

Ngày: 8/12/2025

Thời gian: 8:00

Tổng thời gian phân tích: 24 tiếng

IV. Thông tin phân tích tổng quan *(Gồm những thông tin cơ bản của file)*

4.1 Tình trạng file ban đầu: *file có bị nén packed hay rối mã obfuscated hay không?*

File

☐ Có

☒ **Không**

Phương pháp nhận biết: sử dụng công cụ **PEID**

4.2 Đặc điểm cài đặt/thực thi file:

☐ File thực thi trực tiếp

☒ **Cần một số bước cài đặt thêm (Nếu có điền Phần V. Cách thực thi file)**

V. Cách cài đặt/thực thi file

- File định dạng .exe, thực thi bằng cmd
- Cần phân tích mã nguồn để biết được cách thực thi.

VI. Phân tích hành vi của file (Phần 1)

- **Phương pháp:** Phân tích mã nguồn assembly của file với công cụ **(1.6.1)** IDA Pro phiên bản 6.6
- **Quá trình phân tích:**

Bước 6.1. Chuẩn bị chương trình

Chuẩn bị một chương hello.c đơn giản

```c
#include <stdio.h>

int main(void){
    puts("Hello world");
}
```

Complie sử dụng cl trong môi trường 32 bit

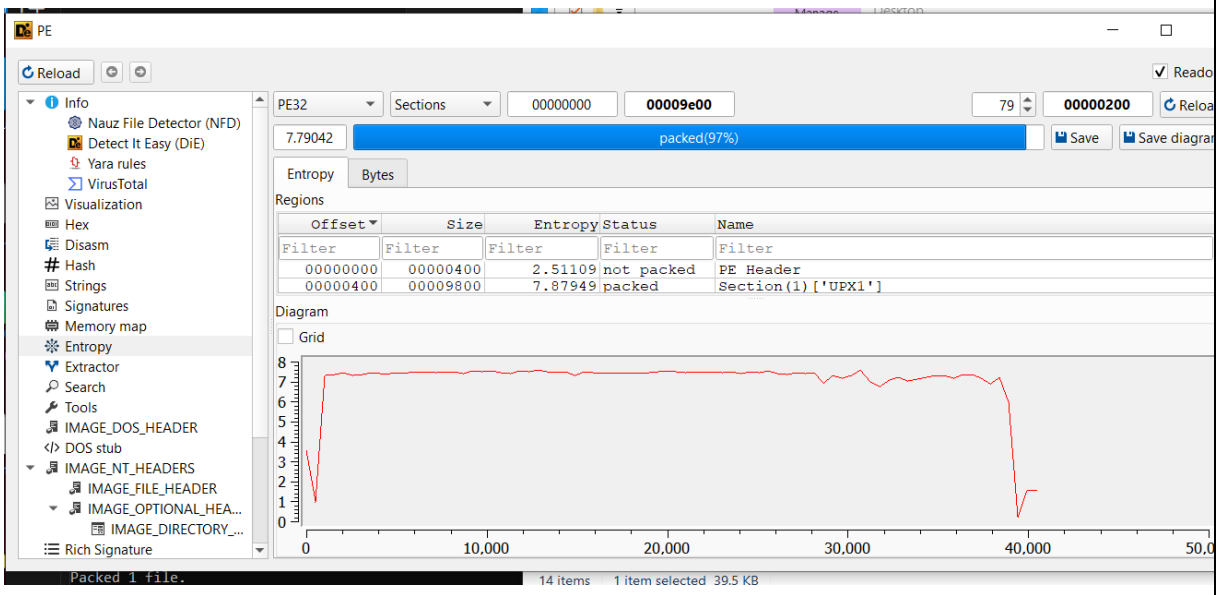
```
C:\Users\fare-vm\Desktop>cl hello.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.16.27054 for
Copyright (C) Microsoft Corporation.  All rights reserved.

hello.c
Microsoft (R) Incremental Linker Version 14.16.27054.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:hello.exe
hello.obj

FLARE-VM 08/12/2025  7:42:29.60
```

Sau đó pack dùng UPX

```
C:\Users\fare-vm\Desktop>upx hello.exe -o hello_upx.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2025
UPX 5.0.2       Markus Oberhumer, Laszlo Molnar & John Reiser   Jul 20th 2025

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     80384 ->     40448   50.32%    win32/pe     hello_upx.exe

Packed 1 file.
```

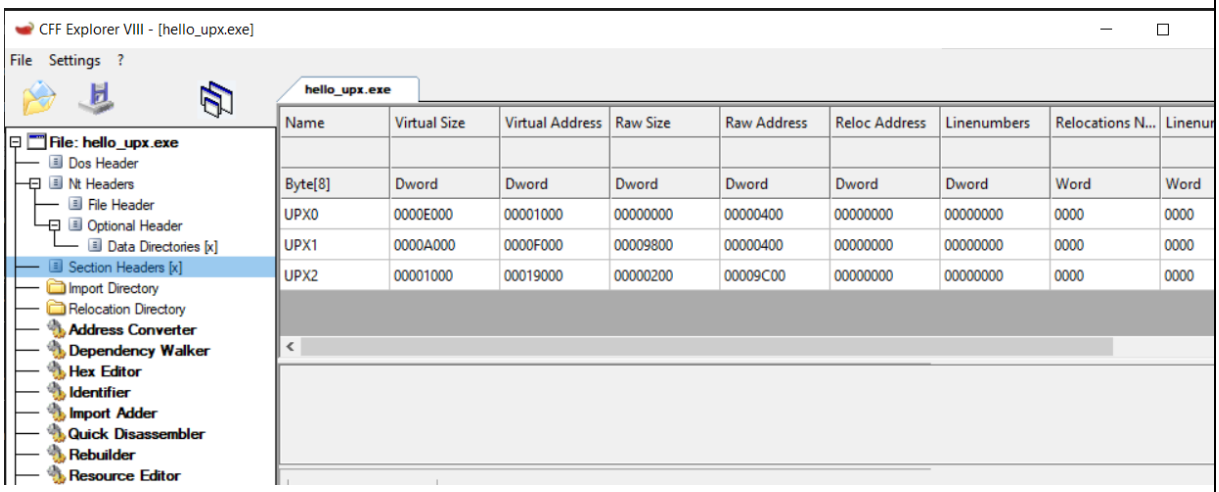
Bước 6.2. Xác định packer thông qua Entropy

Sử dụng tính năng Entropy trong DIE để phát hiện thấy được là entropy cao



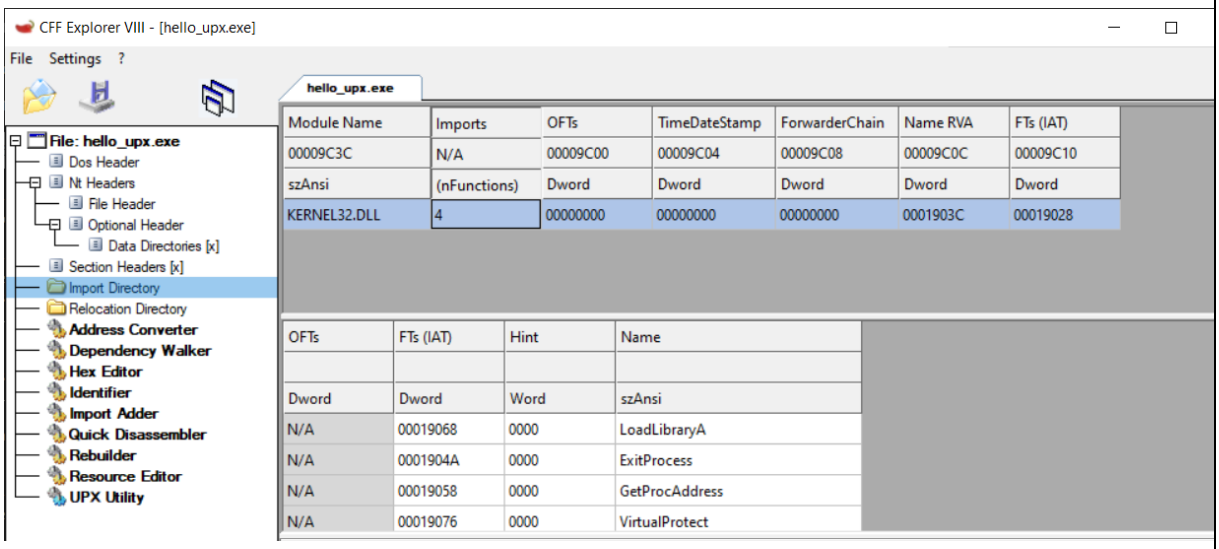
Bước 6.3. Kiểm tra tên và đặc điểm của các section

Sử dụng CFF explore trong section headers để xem các đặc điểm, thấy được các section name là UPX và Virtual Size lớn hơn Raw Size



Bước 6.4. Phân tích Import Table

Sử dụng CFF explore trong Import Directory thấy được chỉ sử dụng có KERNEL32.dll, các hàm sử dụng dll này chỉ để lấy địa chỉ và Load thư viện.

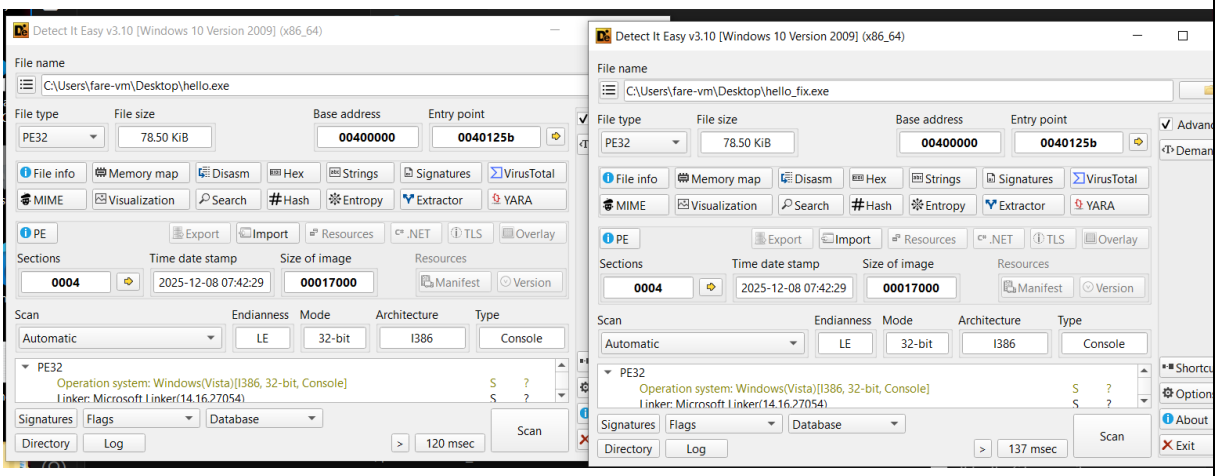


Bước 6.5. Sử dụng công cụ Unpacker

Sử dụng lại UPX

```
FLARE-VM 08/12/2025  7:45:36.93
C:\Users\fare-vm\Desktop>upx -d hello_upx.exe -o hello_fix.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2025
UPX 5.0.2       Markus Oberhumer, Laszlo Molnar & John Reiser   Jul 20th 2025

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     80384 <-     40448   50.32%    win32/pe     hello_fix.exe

Unpacked 1 file.
```

Sau đó sử dụng DIE để so sánh 2 file thì nó giống nhau:

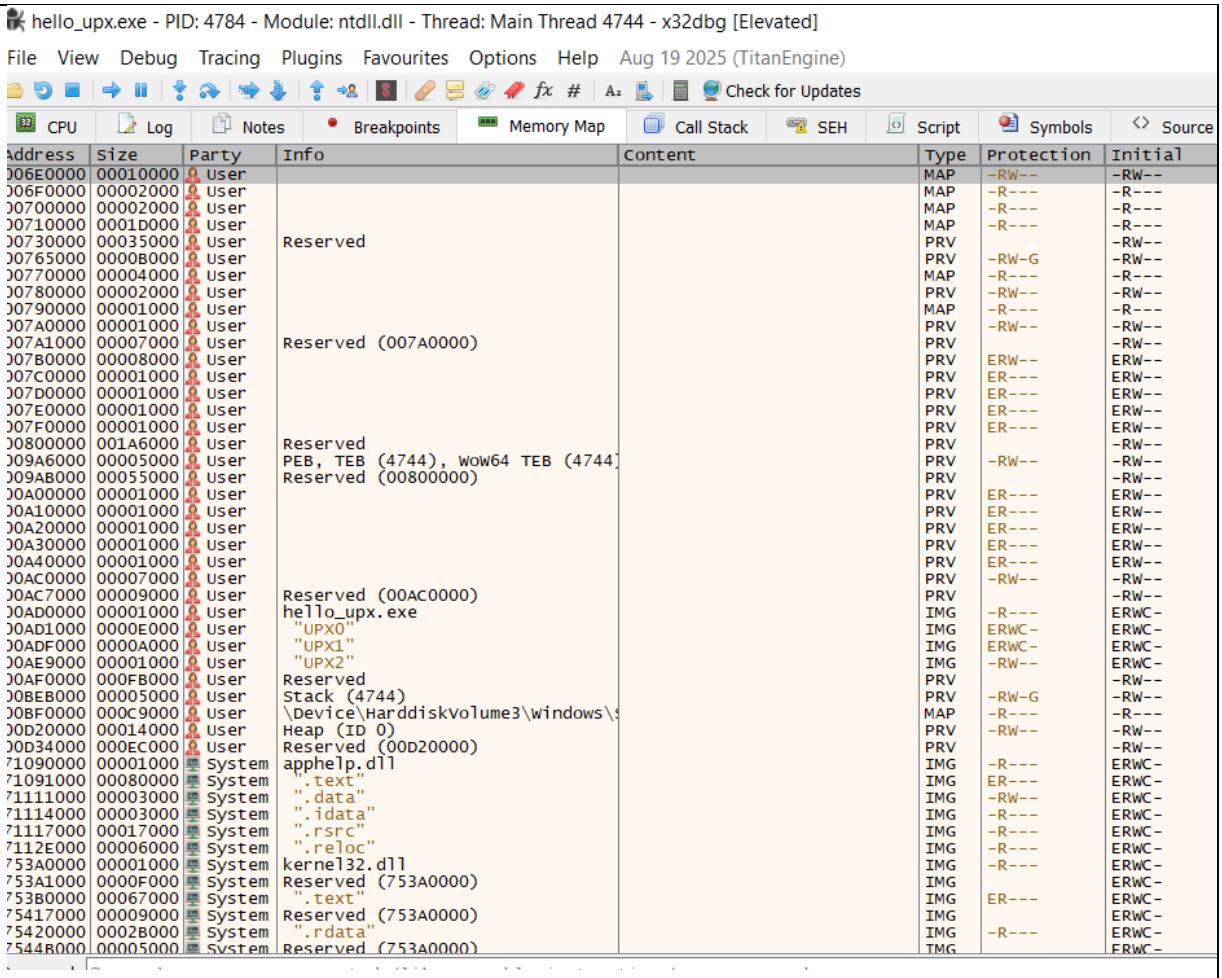


Bước 6.6. Unpacking thủ công

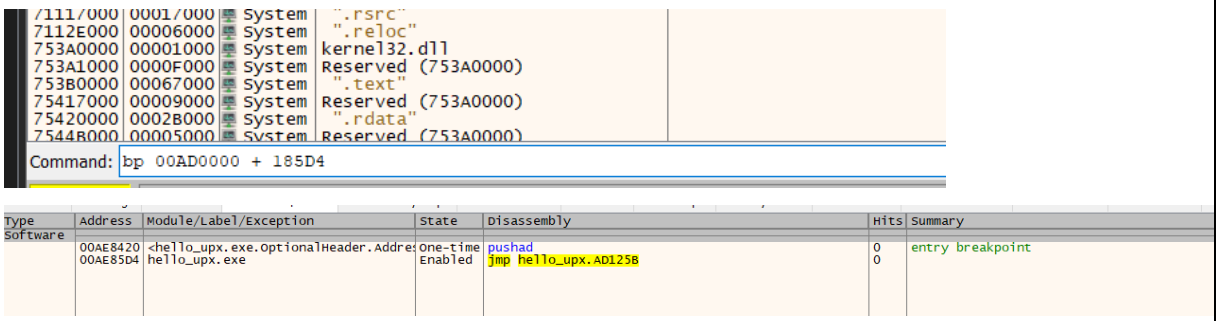
Sử dụng ida để xem các hoạt động của file bị pack, thấy địa chỉ thất bại là 004185D4

```
UPX1:004185CB loc_4185CB:                              ; CODE XREF: start+1AF↓j
UPX1:004185CB                 push    0
UPX1:004185CD                 cmp     esp, eax
UPX1:004185CF                 jnz     short loc_4185CB
UPX1:004185D1                 sub     esp, 0FFFFFF80h
UPX1:004185D4                 jmp     near ptr byte_40125B
UPX1:004185D4 start           endp ; sp-analysis failed
UPX1:004185D4
UPX1:004185D4 ; ---------------------------------------------------------------------------
UPX1:004185D9                 align 4
```

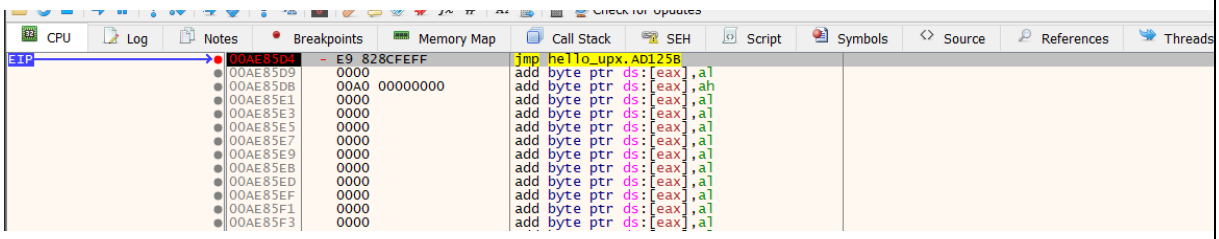
Vào x32dbg thấy được vị trí bắt đầu ở 00AD0000



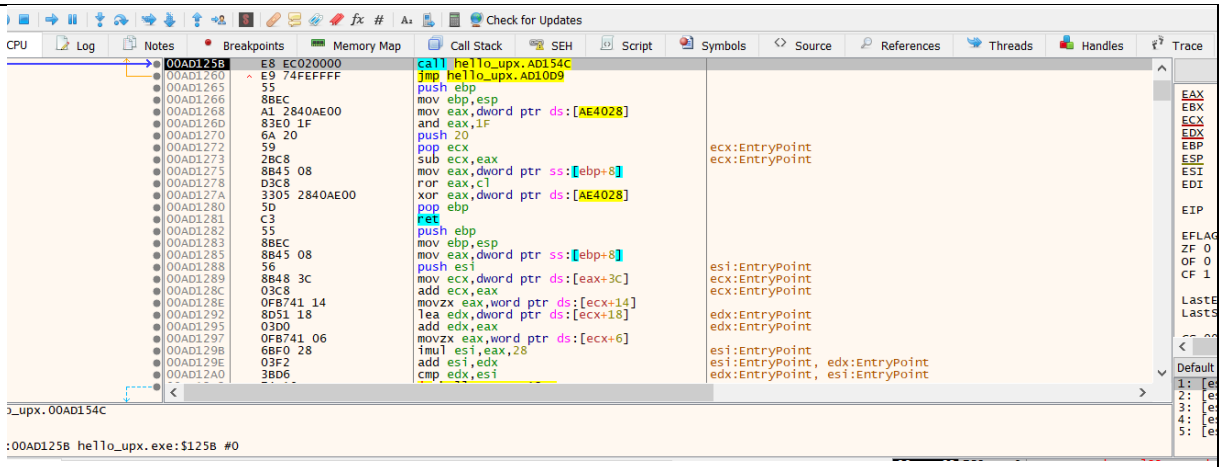
Đặt breakpoint tại vị trí bắt đầu cộng với vị trí tìm được trong IDA



Tiếp theo chạy chương trình đến breakpoint



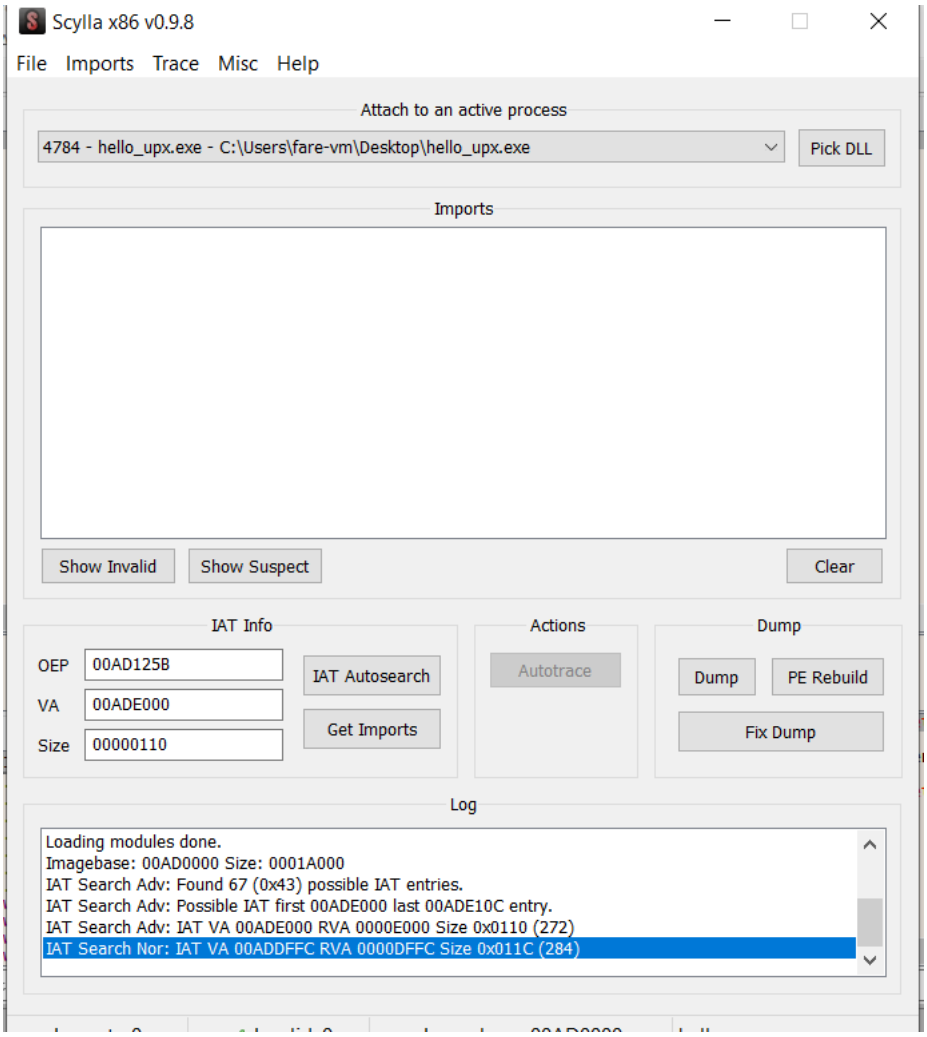
Bước qua lệnh trên



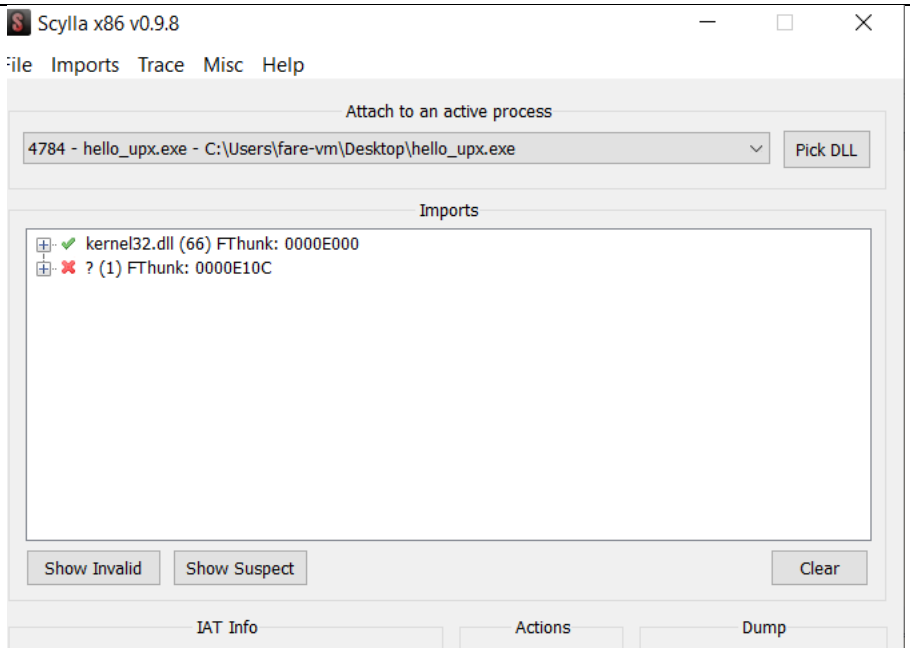
Đã tìm được phần chương trình đã được unpack

Bước 6.6. Xuất mã đã giải nén ra disk

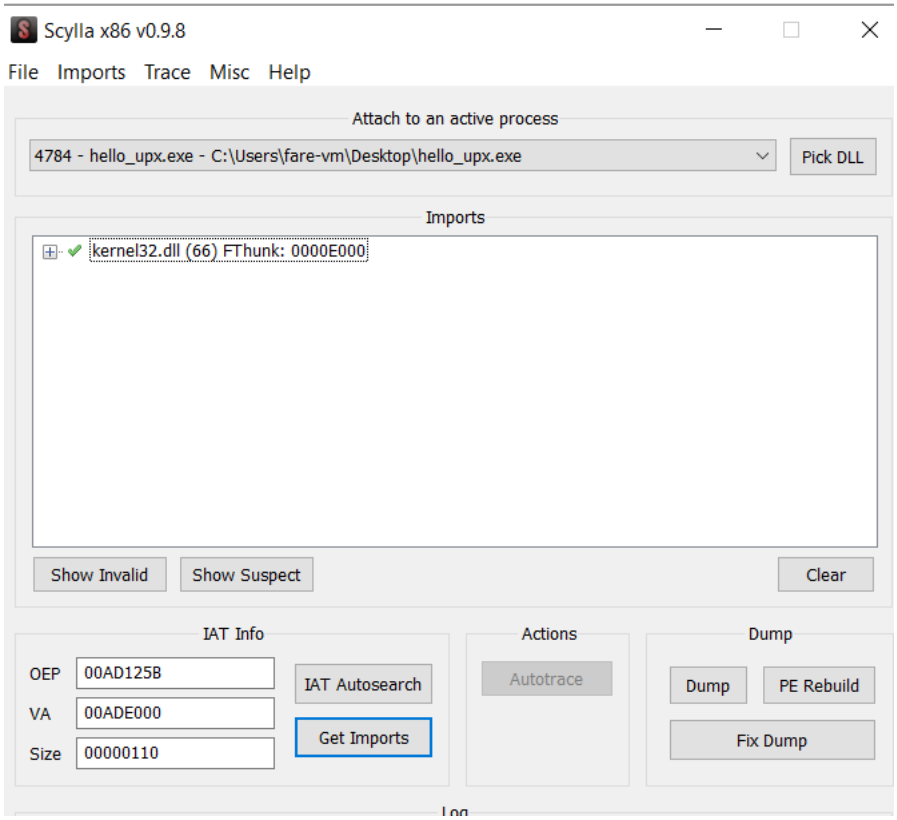
Sử dụng tính năng Entropy trong DIE để phát hiện thấy được là entropy caoSử dụng plugin Scylla để tìm bảng IAT



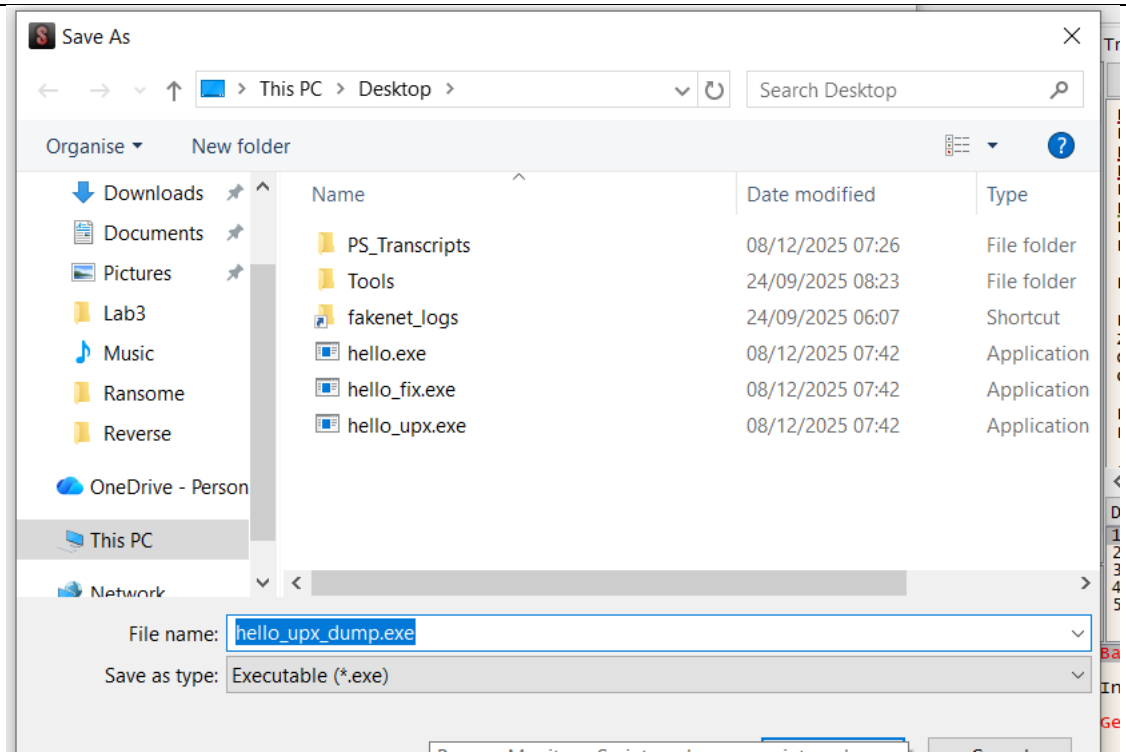
Nhấn get imports để lấy các thư viện cần nhập



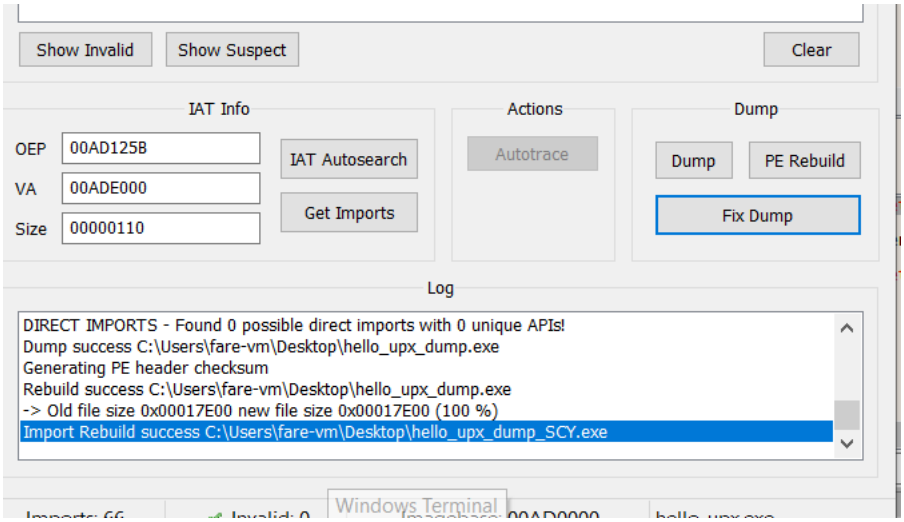
Xóa đi phần FThunk bị dư



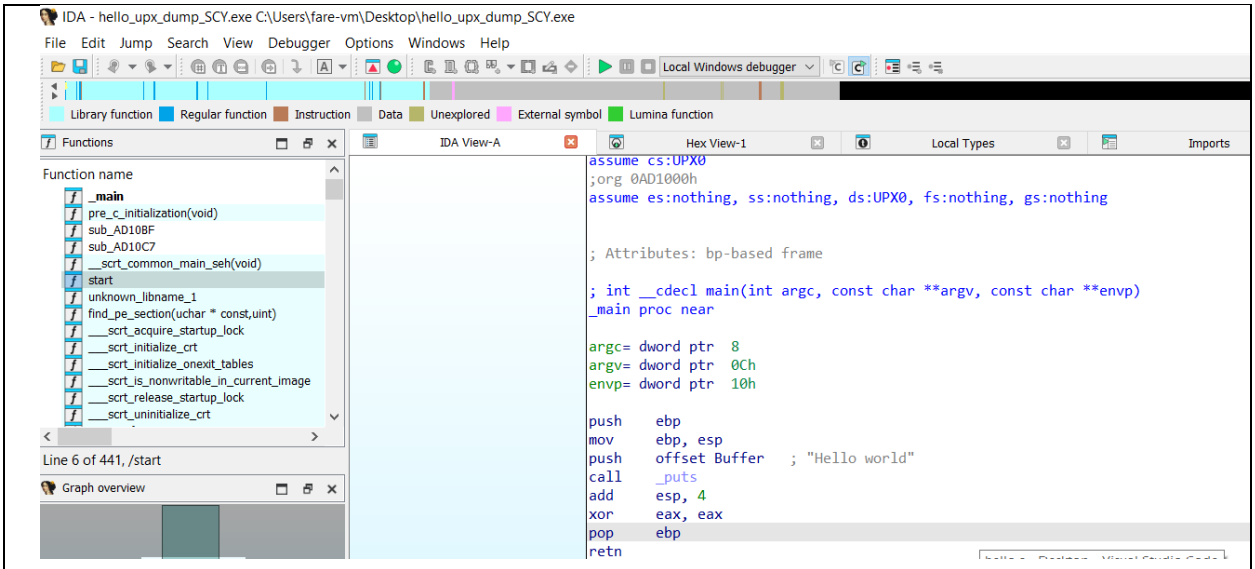
Dump file



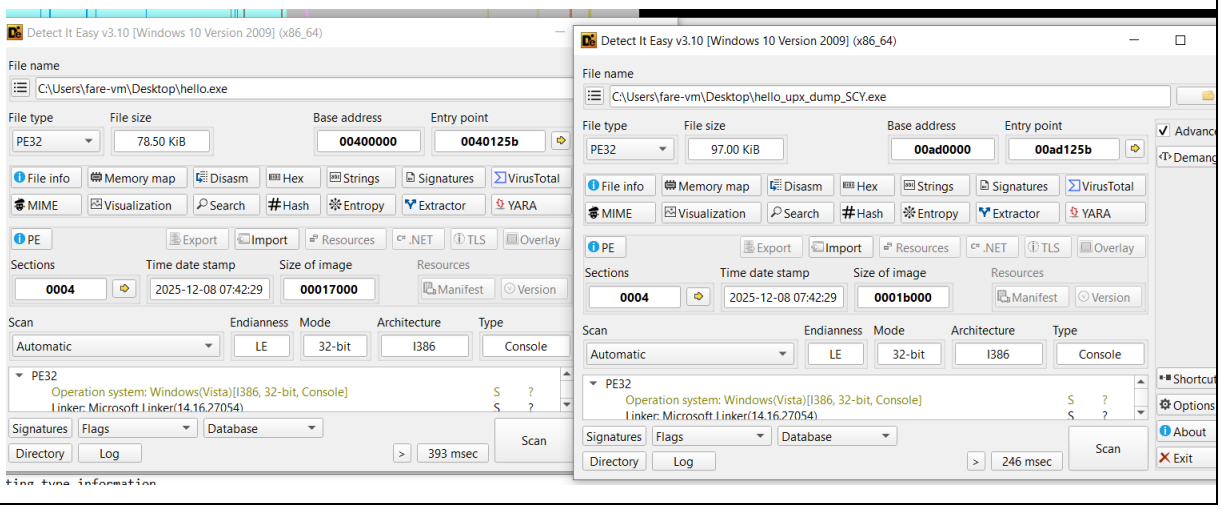
Và fix lại file



Kiểm tra trong IDA



Kiểm tra sử dụng DIE thì 2 file giống nhau



2. Báo cáo Phân tích tĩnh nâng cao 2

BÁO CÁO PHÂN TÍCH TĨNH NÂNG CAO 2
I. Người phân tích: Nhóm phân tích: Nhóm 7 Thành viên: Hà Minh Quân - 22521177, Từ Chí Kiên - 22520713
II. Tổng quan về file phân tích: Tên file: ***basic-reverse.exe*** Kích thước: 45 KB Định dạng: ***.exe***
III. Thời gian phân tích:

Ngày: 7/12/2025

Thời gian: 8:00

Tổng thời gian phân tích: 24 tiếng

IV. Thông tin phân tích tổng quan *(Gồm những thông tin cơ bản của file)*

4.3 Tình trạng file ban đầu: *file có bị nén packed hay rối mã obfuscated hay không?*

File basic-reverse.exe

☐ Có

☒ **Không**

Phương pháp nhận biết: sử dụng công cụ **PEID**

4.4 Đặc điểm cài đặt/thực thi file:

☐ File thực thi trực tiếp

☒ **Cần một số bước cài đặt thêm (Nếu có điền Phần V. Cách thực thi file)**

V. Cách cài đặt/thực thi file

- File định dạng .exe, thực thi bằng cmd
- Cần phân tích mã nguồn để biết được cách thực thi.

VI. Phân tích hành vi của file (Phần 1)

- **Phương pháp:** Phân tích mã nguồn assembly của file với công cụ **(2.6.1)** IDA Pro phiên bản 6.6
- **Quá trình phân tích:**

Bước 6.1. Xác định hướng phân tích thông qua phân tích tĩnh cơ bản

Công cụ sử dụng là **(2.6.2)** IDA Pro phiên bản 6.6

Dựa trên danh sách các hàm sử dụng và một số chuỗi tìm thấy, một số hoạt động dự đoán của chương trình gồm: **(2.6.3)** *(liệt kê thêm 2 hoạt động dự đoán được)*

- 3 hàm thực thi các thử thách khác nhau

(Hình ảnh đính kèm)

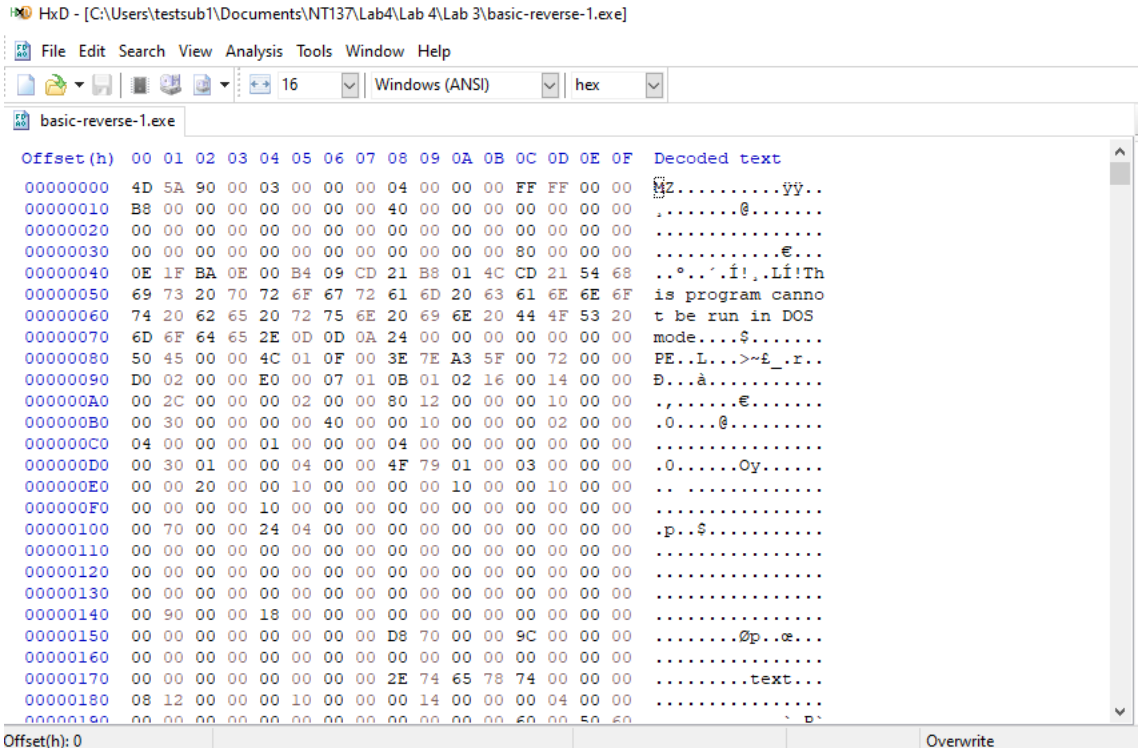
```
int main()
{
  int v1; // [sp+14h] [bp-Ch]@1

  __main();
  printf("Supported authentication methods:\n1. Hardcode\n2. Username/password\n3. Double keys\nEnter your choice: ");
  scanf("%d", &v1);
  fflush((FILE *)__iob[0]._ptr);
  if ( v1 == 1 )
  {
    hardCode();
  }
  else if ( v1 == 2 )
  {
    userpass();
  }
  else
  {
    if ( v1 != 3 )
    {
      puts("Invalid authentication method.");
      system("pause");
      exit(0);
    }
    double_key_check();
  }
  system("pause");
00000C33 main:16
```

Bước 6.2. Mở file cần phân tích với công cụ IDA Pro

Mở file cần phân tích với kiểu file **(2.6.4)** PE 32bit

(Hình ảnh đính kèm)

Sử dụng HxD cho thấy PE L cho biết là 32bit



Bước 6.3. Phân tích hoạt động chính của file với IDA Pro

Bắt đầu phân tích hoạt động của file bằng cách quan sát mã assembly của hàm **main**, là hàm được thực thi đầu tiên.

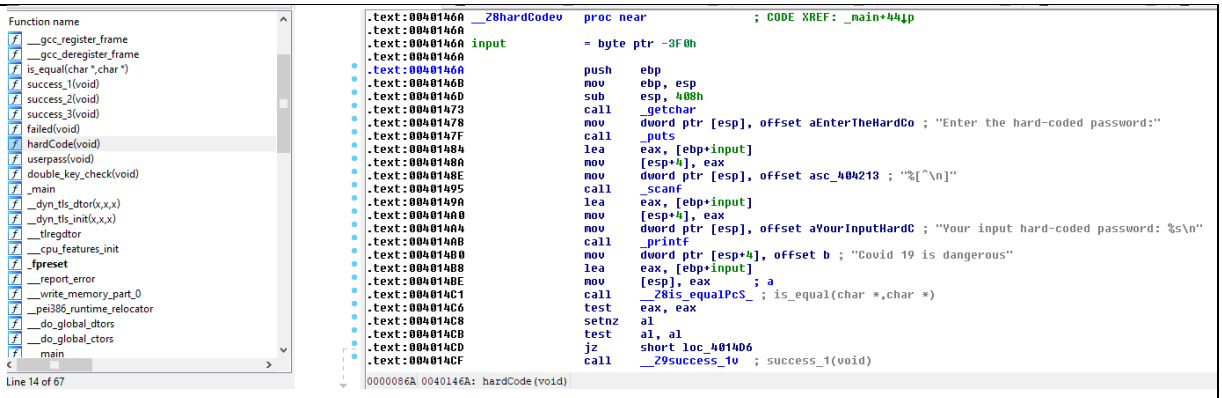
Dựa trên các câu lệnh, có thể thấy hoạt động chính của hàm **main** chia làm 3 trường hợp thực thi chương trình dựa trên số người dùng nhập là 1, 2, 3; nhưng đối với bài phân tích này chỉ tập trung trường hợp 1 và 2.

- **Trường hợp nhập số 1**

Chương trình sẽ thực thi hàm **(2.6.5)** hardCode trong đó có sử dụng hàm **(2.6.6)** getchar() để đọc 1 ký tự nhập từ bàn phím. Hàm **(2.6.7)** puts xuất chuỗi '**Enter the hard-coded password:**'. Hàm **(2.6.8)** scanf đọc chuỗi người dùng nhập nhưng với chuỗi '**%[^\n]**' , ở đây sẽ **(2.6.9)** đọc chuỗi người dùng nhập và ngắt khi phím Enter được nhập. Hàm prinf sẽ được chuỗi người dùng nhập rồi in ra sau chuỗi '**Your input hard-coded password:**'.

(Hình ảnh đính kèm)

Hàm hardCode được tìm thấy trong tab Function name



Hàm hardCode ở dạng mã giả

```
void hardCode()
{
  char input[1000]; // [sp+18h] [bp-3F0h]@1

  getchar();
  puts("Enter the hard-coded password:");
  scanf("%[^\n]", input);
  printf("Your input hard-coded password: %s\n", input);
  if ( is_equal(input, "Covid 19 is dangerous") != 0 )
    success_1();
  else
    failed();
}
```

Hàm is_equal nhận input người dùng và chuỗi **(2.6.10)** '**Covid 19 is dangerous**' là input:

- Trong đây chỉ sử dụng hàm strcmp để so sánh 2 chuỗi

(Hình ảnh đính kèm)

```
bool __cdecl is_equal(char *a, char *b)
{
  return strcmp(a, b) == 0;
}
```

Như vậy để có thể chạy thành công thì đầu tiên phải nhập **(2.6.11)** tín hiệu Enter để chương trình có thể đi qua hàm getchar() mà không bị mất một ký tự khi nhập chuỗi. Sau khi màn hình hiện '**Enter the hard-coded password:**' thì tiến hành nhập **(2.6.12)** chuỗi '**Covid 19 is dangerous**'.

(Hình ảnh đính kèm)

```
C:\Users\testsub1\Documents\NT137\Lab4\Lab 4\Lab 3>basic-reverse-1.exe
Supported authentication methods:
1. Hardcode
2. Username/password
3. Double keys
Enter your choice: 1

Enter the hard-coded password:
Covid 19 is dangerous
Your input hard-coded password: Covid 19 is dangerous
Congrats! You found the hard-coded secret :).
Hand in this to your instructor as a proof:
"Stay home for the safety of yourself and others."
Press any key to continue . . .
```

Ở đây tìm thấy chuỗi: Stay home for the safety of yourself and others.

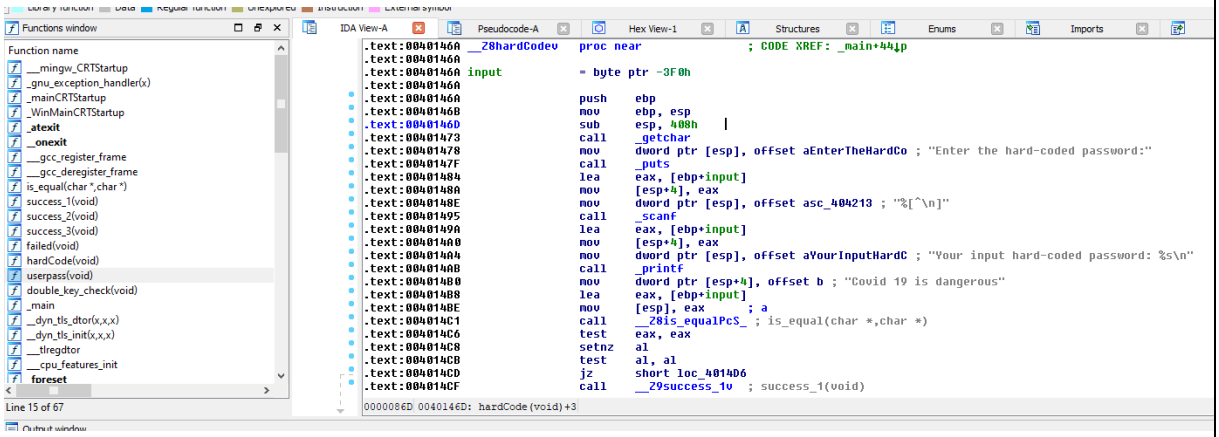
- **Trường hợp nhập số 2**

Chương trình sẽ thực thi hàm **(2.6.13)** userpass trong đó sử dụng một array tên là **(2.6.14)** SECRET có **(2.6.14)** 4 thành phần gồm 4 số nguyên là 99, 111, 118, 105, 100. Tiếp theo hàm **(2.6.15)** getchar() để đọc 1 ký tự nhập từ bàn phím ở 2 vị trí trước khi nhập username và pass.

Hàm **(2.6.16)** puts xuất chuỗi '**Enter your username:**' và '**Enter your password:**'. Hàm **(2.6.17)** scanf đọc chuỗi người dùng nhập nhưng với chuỗi '**%[^\n]**' , ở đây sẽ **(2.6.18)** đọc chuỗi người dùng nhập và ngắt khi phím Enter được nhập. Hàm prinf sẽ được chuỗi người dùng nhập gồm username và pass rồi in ra trong chuỗi '**Your input username: %s and password: %s**' .

(Hình ảnh đính kèm)

Hàm userpass được tìm thấy trong tab Function name



Hàm userpass ở dạng mã giả

```
1  void userpass()
2  {
3    size_t v0; // ebx@2
4    int v1; // ebx@13
5    int v2; // ebx@14
6    char your_secret[8]; // [sp+1Dh] [bp-2Bh]@6
7    char pass[9]; // [sp+25h] [bp-23h]@1
8    char name[9]; // [sp+2Eh] [bp-1Ah]@1
9    char SECRET[5]; // [sp+37h] [bp-11h]@1
10   int i; // [sp+3Ch] [bp-Ch]@4
11
12   SECRET[0] = 99;
13   SECRET[1] = 111;
14   SECRET[2] = 118;
15   SECRET[3] = 105;
16   SECRET[4] = 100;
17   getchar();
18   puts("Enter your username:");
19   scanf("%[^\n]", name);
20   getchar();
21   puts("Enter your password:");
22   scanf("%[^\n]", pass);
23   printf("Your input username: %s and password: %s\n", name, pass);
24   if ( strlen(name) == 8 && (v0 = strlen(name), v0 == strlen(pass)) )
25   {
26     for ( i = 0; i <= 7; ++i )
27     {
```

```
28      if ( i > 2 )
29        your_secret[i] = SECRET[i - 3];
30      else
31        your_secret[i] = name[i + 5];
32    }
33    for ( i = 0; ; ++i )
34    {
35      v1 = i;
36      if ( v1 >= strlen(name) || (name[i] + your_secret[i]) / 2 != pass[i] )
37        break;
38    }
39    v2 = i;
40    if ( v2 == strlen(name) )
41      success_2();
42    else
43      failed();
44  }
45  else
46  {
47    failed();
48  }
49}
```

Để tìm được giá trị của pass, cần phân tích các mã lệnh sau khi nhập mật khẩu. Các hàm strlen dùng trong điều kiện if được dùng để kiểm tra độ dài của chuỗi username và pass, từ đây biết được độ dài của username và pass là **(2.6.19)** 8 ký tự.

Theo yêu cầu đề bài là phải nhập username là 1177_0713 nhưng name chỉ nhận 8 ký tự nên đổi sang 11770713. Như vậy các ký tự trong dãy your_secret:

- your_secret[0] = name[0+5] = '7' = 55
- your_secret[1] = name[1+5] = '1' = 49
- your_secret[2] = name[2+5] = '3' = 51
- your_secret[3] = SECRET[3-3] = 99
- your_secret[4] = SECRET[4-3] = 111
- your_secret[5] = SECRET[5-3] = 118
- your_secret[6] = SECRET[6-3] = 105
- your_secret[7] = SECRET[7-3] = 100

Như vậy từ dãy trên thì suy ra các ký tự chuỗi pass cần tìm:

- pass[0] = ('1' + '7')/2 = (49 + 55)/2 = 52 = '4'
- pass[1] = ('1' + '1')/2 = (49 + 49)/2 = 49 = '1'
- pass[2] = ('7' + '3')/2 = (55 + 51)/2 = 53 = '5'
- pass[3] = ('7' + 99)/2 = (55 + 99)/2 = 77 = 'M'
- pass[4] = ('0' + 111)/2 = (48 + 111)/2 = 79 = 'O'
- pass[5] = ('7' + 118)/2 = (55 + 118)/2 = 86 = 'V'
- pass[6] = ('1' + 105)/2 = (49 + 105)/2 = 77 = 'M'
- pass[7] = ('3' + 100)/2 = (51 + 100)/2 = 75 = 'K'

Vậy chuỗi cần tìm là '415MOVMK'

(Hình ảnh đính kèm)

```
C:\Users\testsub1\Documents\NT137\Lab4\Lab 4\Lab 3>basic-reverse-1.exe
Supported authentication methods:
1. Hardcode
2. Username/password
3. Double keys
Enter your choice: 2

Enter your username:
11770713
Enter your password:
415MOVMK
Your input username: 11770713 and password: 415MOVMK
Congrats! You found your own username/password pair.
Hand in this to your instructor as a proof:
"Vietnam can win over SARS-CoV-2."
Press any key to continue . . .
```

Chuỗi tìm được 'Vietnam can win over SARS-CoV-2.'

3. Báo cáo Phân tích tĩnh nâng cao 3

BÁO CÁO PHÂN TÍCH TĨNH NÂNG CAO 3

I. Người phân tích:

Nhóm phân tích: Nhóm 7

Thành viên: Hà Minh Quân - 22521177, Từ Chí Kiên - 22520713

II. Tổng quan về file phân tích:

Tên file: ***unknown.exe***

Kích thước: 39KB

Định dạng: ***.exe***

III. Thời gian phân tích:

Ngày: 7/12/2025

Thời gian: 8:00

Tổng thời gian phân tích: 24 tiếng

IV. Thông tin phân tích tổng quan *(Gồm những thông tin cơ bản của file)*

4.5 Tình trạng file ban đầu: *file có bị nén packed hay rối mã obfuscated hay không?*

File unknown.exe

☒ **Có**

☐ Không

Phương pháp nhận biết: sử dụng công cụ **CFF explore**

V. Phân tích hành vi của file (Phần 1)

- **Phương pháp:** Phân tích mã nguồn assembly của file với công cụ **(1.6.1)** IDA Pro phiên bản 6.6 và CFF explore

- **Quá trình phân tích:**

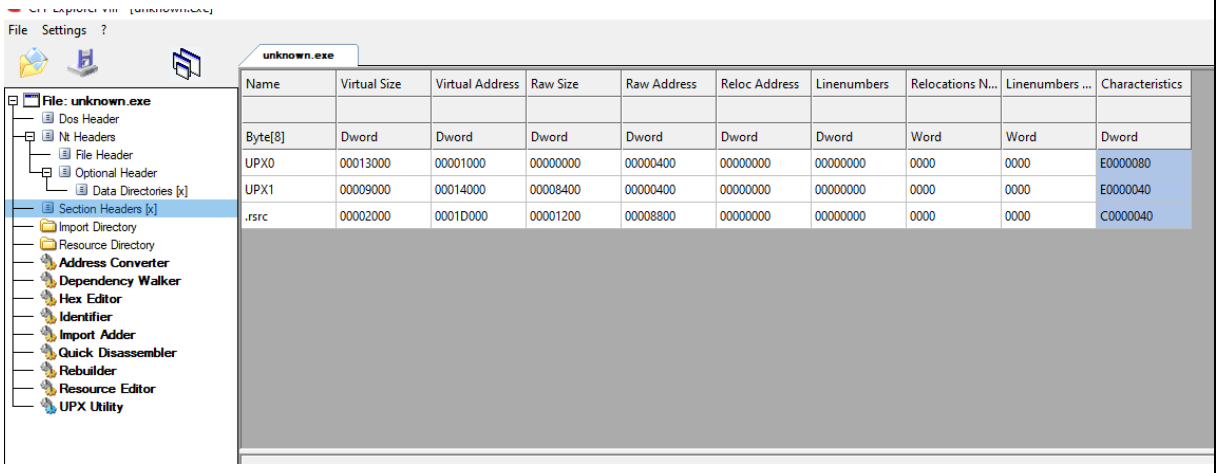
Bước 6.1. Xác định chương trình có bị packed không?

Công cụ sử dụng là **(1.6.2)** CFF explore

Dựa trên kết quả ở phần section header của cff biết được tệp exe ở dạng **(1.6.3)** bị đóng gói và công cụ sử dụng UPX.

(Hình ảnh đính kèm)

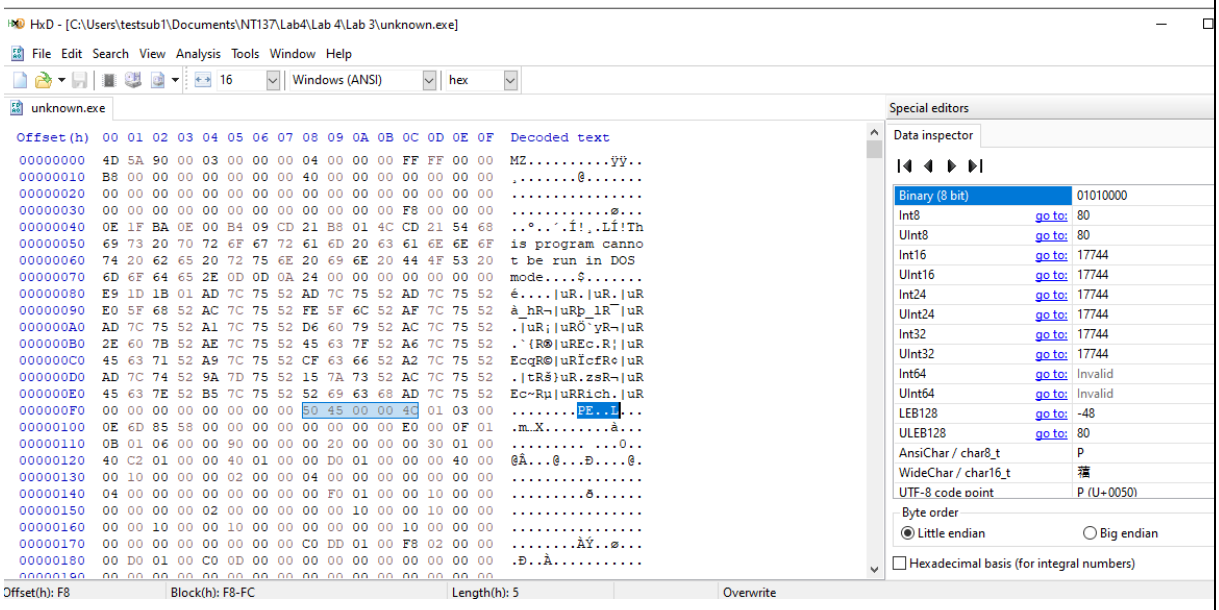
Hình thể hiện có thể thực thi nhiều câu lệnh và mật khẩu bảo vệ



Bước 6.2. Xác định chương trình là loại PE gì

Mở file cần phân tích với kiểu file **(1.6.4)** PE 32 bit sử dụng công cụ **(1.6.5)** HxD cho thấy PE L đồng nghĩa với 32bit.

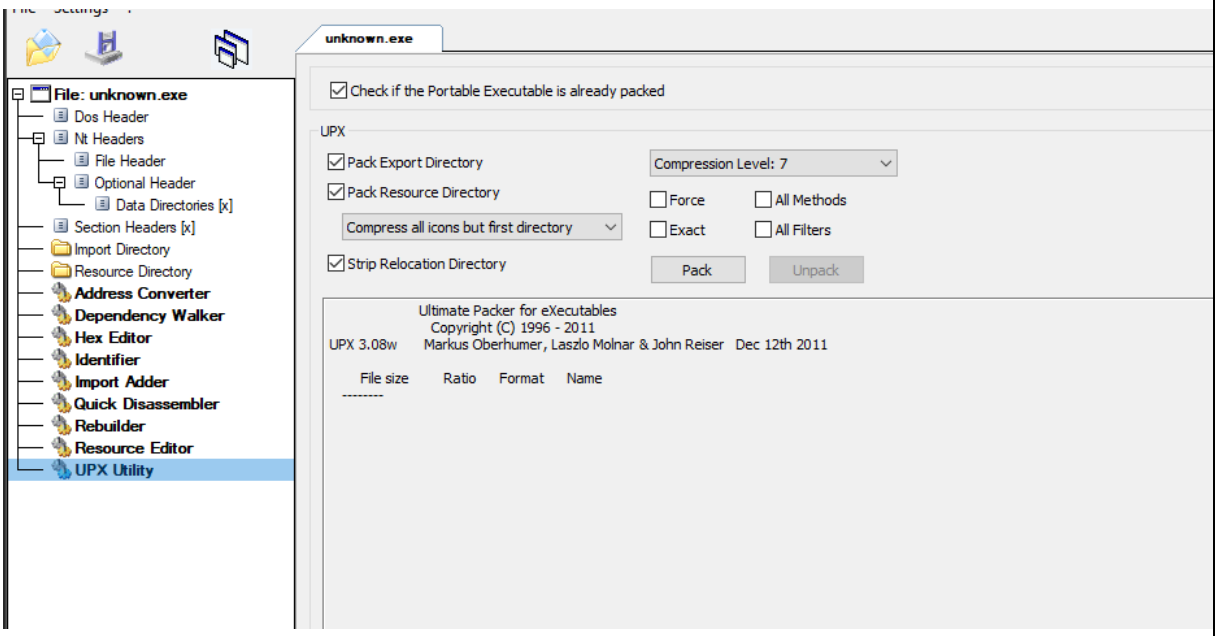
(Hình ảnh đính kèm)



Bước 6.3. Unpack gói tin và tìm kích thước raw của section .text

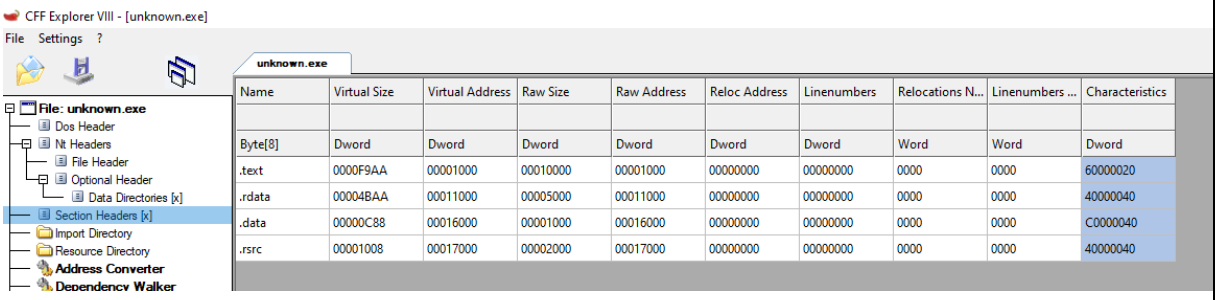
Do CFF có hỗ trợ unpack loại UPX nên sử dụng **(1.6.6)** UPX utility để giải nén

(Hình ảnh đính kèm)



Sau khi giải nén, sử dụng công cụ **(1.6.7)** CFF explore xem trong Section Headers sẽ thấy được kích thước raw của .text là **(1.6.8)** Dword 00010000

(Hình ảnh đính kèm)



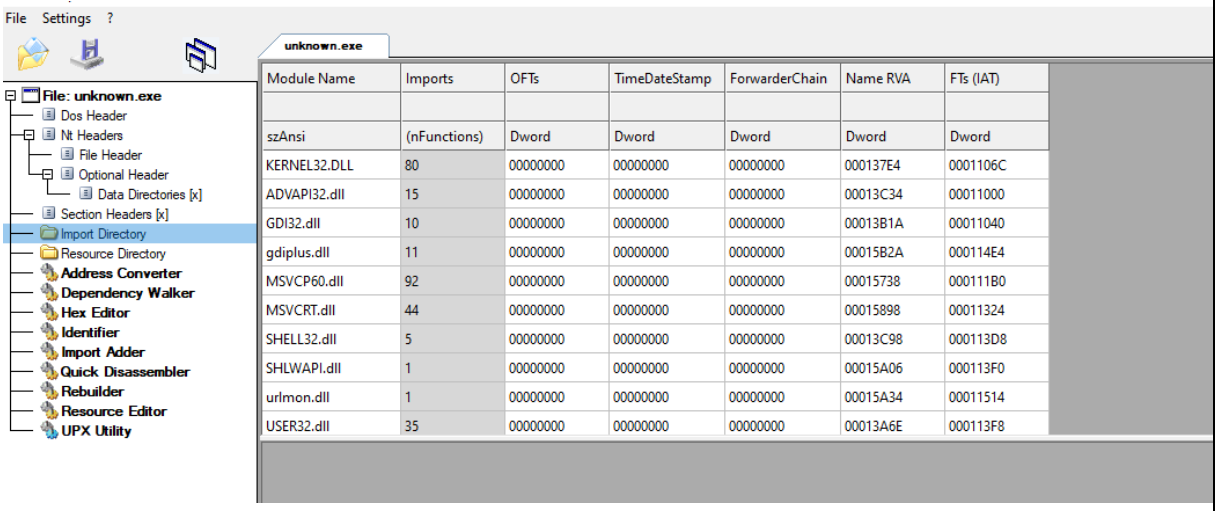
Bước 6.4. Mẫu này import những DLL nào? Có bất kỳ hàm (function) đáng ngờ nào được gọi từ các DLL này không?

Sử dụng CFF ở phần import Directory và trong IDA sẽ tìm được các dll:

- 1. ADVAPI32.dll - Windows Advanced Services API
- 2. GDI32.dll - Graphics Device Interface
- 3. KERNEL32.dll - Core Windows API
- 4. MSVCP60.dll - Microsoft C++ Runtime
- 5. MSVCRT.dll - Microsoft C Runtime
- 6. SHELL32.dll - Shell API
- 7. SHLWAPI.dll - Shell Lightweight Utility API
- 8. USER32.dll - User Interface API

- 9. WININET.dll - Windows Internet API
- 10. WINMM.dll - Windows Multimedia API
- 11. WS2_32.dll - Windows Sockets 2 API
- 12. gdiplus.dll - GDI+ Graphics API
- 13. urlmon.dll - URL Moniker API

(Hình ảnh đính kèm)



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.DLL	80	00000000	00000000	00000000	000137E4	0001106C
ADVAPI32.dll	15	00000000	00000000	00000000	00013C34	00011000
GDI32.dll	10	00000000	00000000	00000000	00013B1A	00011040
gdiplus.dll	11	00000000	00000000	00000000	00015B2A	000114E4
MSVCP60.dll	92	00000000	00000000	00000000	00015738	000111B0
MSVCRT.dll	44	00000000	00000000	00000000	00015898	00011324
SHELL32.dll	5	00000000	00000000	00000000	00013C98	000113D8
SHLWAPI.dll	1	00000000	00000000	00000000	00015A06	000113F0
urlmon.dll	1	00000000	00000000	00000000	00015A34	00011514
USER32.dll	35	00000000	00000000	00000000	00013A6E	000113F8

00411488		InternetCloseHandle	WININET
0041148C		InternetOpenUrlA	WININET
00411490		InternetOpenA	WININET
00411494		InternetReadFile	WININET
0041149C		waveInOpen	WINMM
004114A0		waveInStop	WINMM
004114A4		waveInClose	WINMM
004114A8		waveInAddBuffer	WINMM
004114AC		waveInPrepareHeader	WINMM
004114B0		waveInUnprepareHeader	WINMM
004114B4		waveInStart	WINMM
004114BC	9	_imp_htons	WS2_32
004114C0	52	_imp_gethostbyname	WS2_32
004114C4	3	_imp_closesocket	WS2_32
004114C8	23	_imp_socket	WS2_32
004114CC	19	_imp_send	WS2_32
004114D0	111	_imp_WSAGetLastError	WS2_32
004114D4	4	_imp_connect	WS2_32
004114D8	16	_imp_recv	WS2_32
004114DC	115	imp_WSAStartup	WS2_32

Trong đó

- CreateRemoteThread, WriteProcessMemory, VirtualAllocEx - Process Injection
- OpenProcess, TerminateProcess - Process Manipulation
- CreateToolhelp32Snapshot, Process32First/Next - Process Enumeration
- OpenProcessToken, AdjustTokenPrivileges - Privilege Escalation
- RegCreateKeyExA, RegSetValueExA - Registry Persistence
- InternetOpenA, socket, connect, send - Network Communication
- URLDownloadToFileA (urlmon.dll) - Payload Download
- CreateDC, CreateCompatibleBitmap, GetDIBits - Screen Capture

Bước 6.5. Nếu có các hàm đáng ngờ, hãy nêu tên một hàm và các tham số mà nó chấp nhận từ hàm đã gọi.

Ví dụ hàm WriteProcessMemory lấy tham số là:

- Process: Xử lý được lấy thông qua OpenProcess() đến tiến trình mục tiêu (ví dụ: explorer.exe, svchost.exe)
- lpBaseAddress: Địa chỉ bộ nhớ được cấp phát bởi VirtualAllocEx() trong tiến trình mục tiêu
- lpBuffer: Con trỏ đến shellcode/payload độc hại
- nSize: Kích thước của shellcode/payload
- lpNumberOfBytesWritten: Thường là NULL (phần mềm độc hại không quan tâm đến việc xác minh)

Bước 6.6. Cái nhìn tổng quan ngắn gọn về khả năng của phần mềm độc hại này

Cơ chế duy trì:

Sửa đổi Registry: Tạo/chỉnh sửa khóa registry thông qua RegCreateKeyExA, RegSetValueExA

Registry StartUp: Có khả năng tự thêm vào phần khởi động thông qua khóa Run hoặc dịch vụ

Tạo Mutex: CreateMutexA để kiểm tra từng phiên bản

Leo thang đặc quyền:

Thao tác Token: OpenProcessToken, AdjustTokenPrivileges để giành quyền hệ thống/quản trị viên

Tiêm tiến trình: Tiêm vào các tiến trình hợp lệ để ẩn và nâng cao quyền

Các hoạt động hệ thống tệp:

Tải xuống Tải trọng: URLDownloadToFileA để tìm nạp phần mềm độc hại bổ sung

Thao tác tệp: DeleteFileA/W, CopyFileA, liệt kê tệp

Tự sao chép: Có thể tự sao chép vào các thư mục hệ thống

Đánh cắp thông tin:

Ghi lại phím: SetWindowsHookExA để hook bàn phím, kiểm tra sử dụng GetKeyState

Chụp Màn hình: Các hàm GDI để chụp ảnh màn hình

Giám sát Bảng tạm: Các hàm API Bảng tạm để đánh cắp dữ liệu

Giám sát Cửa sổ: GetForegroundWindow, GetWindowTextA để giám sát hoạt động của người dùng

Khả năng mạng:
C2 Giao tiếp: Raw socket (socket, connect, send) hoặc HTTP (InternetOpenA)
Lọc dữ liệu: Gửi dữ liệu bị đánh cắp đến máy chủ từ xa
Tiếp nhận lệnh: Nhận lệnh từ máy chủ C2

Tính năng chống phân tích:
Phát hiện trình gỡ lỗi: Process32First/Next để tìm kiếm trình gỡ lỗi/công cụ
Phát hiện máy ảo: GetDriveTypeA, kiểm tra thời gian bằng GetTickCount
Kết thúc quy trình: TerminateProcess để tắt phần mềm bảo mật

Phương thức thực thi:
Throwing quy trình: Kết hợp CreateRemoteThread, WriteProcessMemory
Tiêm luồng: Tiêm vào các luồng hiện có của các quy trình hợp lệ
Tiêm DLL: Có thể tiêm các DLL độc hại vào các quy trình

Thông tin mục tiêu:
Thông tin hệ thống: GetUserNameW, GetModuleFileNameA/W
Thông tin mạng: Có thể thu thập IP, cấu hình mạng
Thông tin phần cứng: Thông qua nhiều API hệ thống khác nhau