

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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```
C hello.c  X
C hello.c
1 #include <stdio.h>
2
3 int main(void){
4     puts("Hello world");
5 }
```

Compile 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

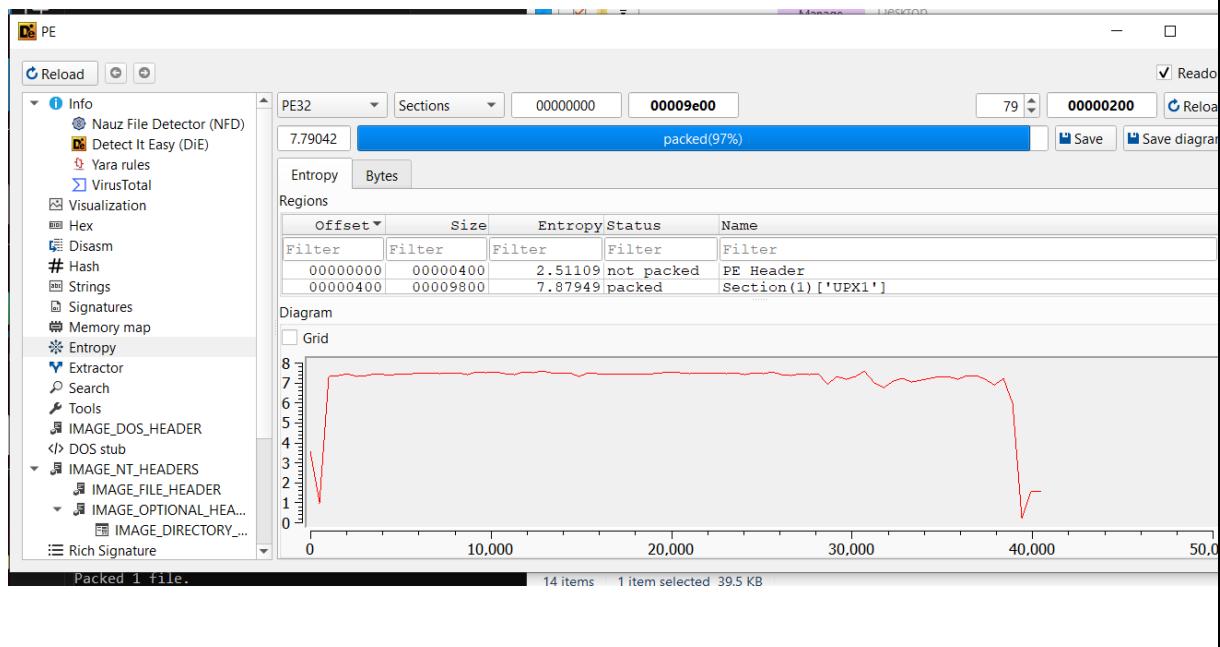
```
C:\Users\fare-vm\Desktop>upx hello.exe -o hello_upx.exe
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2025
UPX 5.0.2      Markus Oberhumer, Laszlo Molnar & John Reiser Jul 20th 2025

  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



Lab 4 – Kỹ thuật Phân tích tĩnh nâng 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

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumber
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word
UPX0	0000E000	00001000	00000000	00000400	00000000	00000000	0000	0000
UPX1	0000A000	0000F000	00009800	00000400	00000000	00000000	0000	0000
UPX2	00001000	00019000	00000200	00009C00	00000000	00000000	0000	0000

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.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
00009C3C	N/A	00009C00	00009C04	00009C08	00009C0C	00009C10
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.DLL	4	00000000	00000000	00000000	0001903C	00019028

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
N/A	00019068	0000	LoadLibraryA
N/A	0001904A	0000	ExitProcess
N/A	00019058	0000	GetProcAddress
N/A	00019076	0000	VirtualProtect

Bước 6.5. Sử dụng công cụ Unpacker

Sử dụng lại UPX

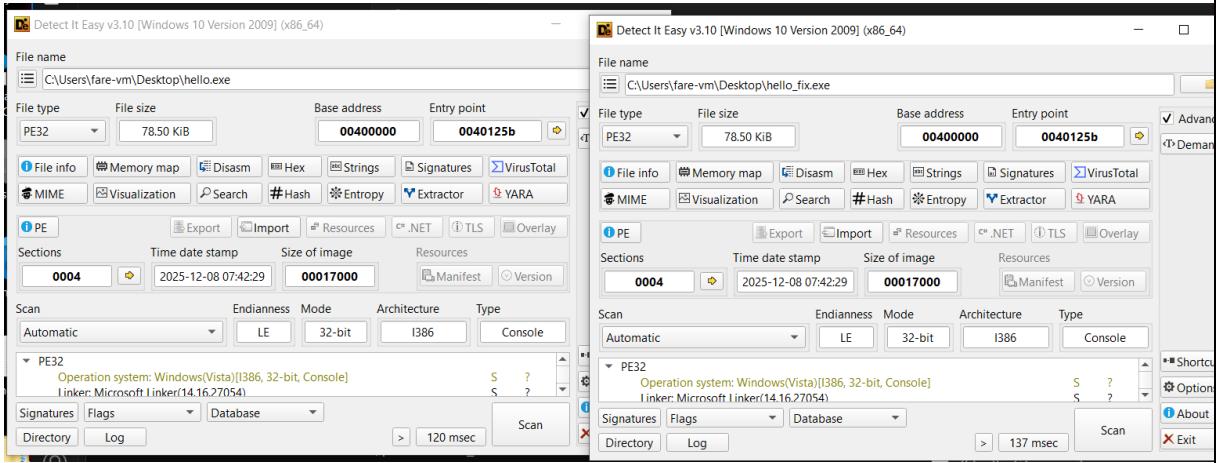
Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```
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
JPX 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 thắt 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, 0FFFFFFF80h
• UPX1:004185D4          jmp   near ptr byte_40125B
UPX1:004185D4 start       endp ; sp-analysis failed
UPX1:004185D4
UPX1:004185D4 ; -----
UPX1:004185D9 alien 4
```

Vào x32dbg thấy được vị trí bắt đầu ở 00AD0000

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

hello_upx.exe - PID: 4784 - Module: ntdll.dll - Thread: Main Thread 4744 - x32dbg [Elevated]

File View Debug Tracing Plugins Favourites Options Help Aug 19 2025 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source

Address	size	Party	Info	Content	Type	Protection	Initial
006E0000	00010000	User			MAP	-RW--	-RW--
006F0000	00002000	User			MAP	-R---	-R---
00700000	00002000	User			MAP	-R---	-R---
00710000	0001D000	User			MAP	-R---	-R---
00730000	00035000	User	Reserved		PRV	-RW-G	-RW--
00765000	0000B000	User			PRV	-RW--	-RW--
00770000	00004000	User			PRV	-RW--	-RW--
00780000	00002000	User			PRV	-RW--	-RW--
00790000	00001000	User			PRV	-R---	-R---
007A0000	00001000	User			PRV	-RW--	-RW--
007A1000	00007000	User	Reserved (007A0000)		PRV	-RW--	-RW--
007B0000	00008000	User			PRV	ERW--	ERW--
007C0000	00001000	User			PRV	ER---	ERW--
007D0000	00001000	User			PRV	ER---	ERW--
007E0000	00001000	User			PRV	ER---	ERW--
007F0000	00001000	User			PRV	ER---	ERW--
00800000	001A6000	User	Reserved		PRV	-RW--	-RW--
009A6000	00005000	User	PEB, TEB (4744), wow64 TEB (4744)		PRV	-RW--	-RW--
009AB000	00055000	User	Reserved (00800000)		PRV	-RW--	-RW--
00A00000	00001000	User			PRV	ER---	ERW--
00A10000	00001000	User			PRV	ER---	ERW--
00A20000	00001000	User			PRV	ER---	ERW--
00A30000	00001000	User			PRV	ER---	ERW--
00A40000	00001000	User			PRV	ER---	ERW--
00AC0000	00007000	User	Reserved (00AC0000)		PRV	-RW--	-RW--
00AC7000	00009000	User	hello_upx.exe		PRV	-RW--	-RW--
00AD0000	00001000	User	"UPX0"		IMG	-R---	ERWC-
00AD1000	00000E00	User	"UPX1"		IMG	ERWC-	ERWC-
00ADF000	00000A00	User	"UPX2"		IMG	ERWC-	ERWC-
00AF0000	000FB000	User	Reserved		PRV	-RW--	ERWC-
00BE8000	00005000	User	Stack (4744)		PRV	-RW-G	-RW--
00BF0000	000C9000	User	\Device\Harddiskvolume3\windows\\$		MAP	-R---	-R---
00D20000	00014000	User	Heap (ID 0)		PRV	-RW--	-RW--
00D34000	0000EC00	User	Reserved (00D20000)		PRV	-RW--	-RW--
71090000	00001000	System	apphelp.dll		IMG	-R---	ERWC-
71091000	00080000	System	".text"		IMG	ER---	ERWC-
71110000	00003000	System	".data"		IMG	-RW--	ERWC-
71114000	00003000	System	".idata"		IMG	-R---	ERWC-
71117000	00017000	System	".rsrc"		IMG	-R---	ERWC-
7112E000	00006000	System	".reloc"		IMG	-R---	ERWC-
753A0000	00001000	System	kernel32.dll		IMG	-R---	ERWC-
753A1000	0000F000	System	Reserved (753A0000)		IMG	-R---	ERWC-
753B0000	00067000	System	".text"		IMG	ER---	ERWC-
75417000	00009000	System	Reserved (753A0000)		IMG	ER---	ERWC-
75420000	0002B000	System	".rdata"		IMG	-R---	ERWC-
7544B000	00005000	System	Reserved (753A0000)		IMG	-R---	ERWC-

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

Command: bp 00AD0000 + 185D4

Type	Address	Module/Label/Exception	State	Disassembly	Hits	Summary
Software	00AE8420	<hello_upx.exe.OptionalHeader.Address>	One-time	pushad	0	entry breakpoint
	00AE8504	hello_upx.exe	Enabled	jmp hello_upx.A01258	0	

Tiếp theo chạy chương trình đến breakpoint

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads

BP → 00AE8504 - E9 828CFEFF jmp hello_upx.A01258

00AE8509	0000	add byte ptr ds:[eax],al
00AE85DB	00A0 00000000	add byte ptr ds:[eax],ah
00AE85E1	0000	add byte ptr ds:[eax],al
00AE85E3	0000	add byte ptr ds:[eax],al
00AE85E5	0000	add byte ptr ds:[eax],al
00AE85E7	0000	add byte ptr ds:[eax],al
00AE85E9	0000	add byte ptr ds:[eax],al
00AE85ED	0000	add byte ptr ds:[eax],al
00AE85EF	0000	add byte ptr ds:[eax],al
00AE85F1	0000	add byte ptr ds:[eax],al
00AE85F3	0000	add byte ptr ds:[eax],al

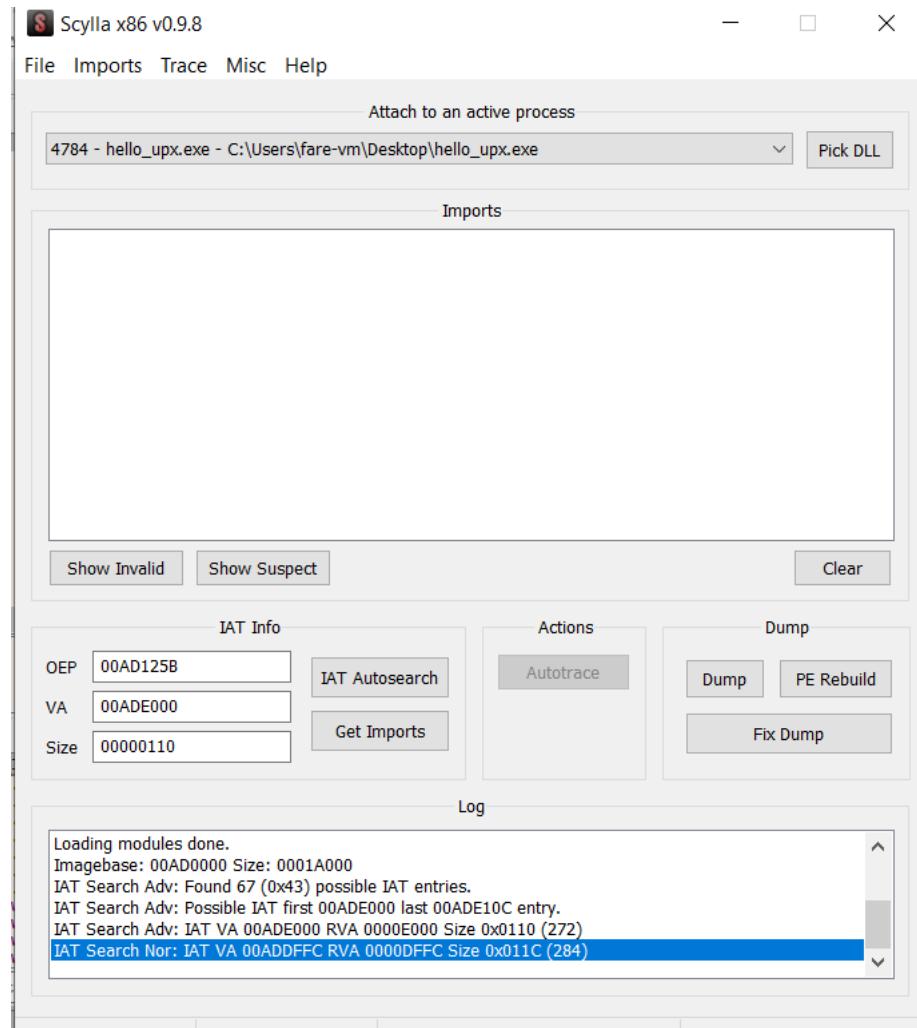
Bước qua lệnh trên

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

Đã tìm được phần chương trình đã được unpack

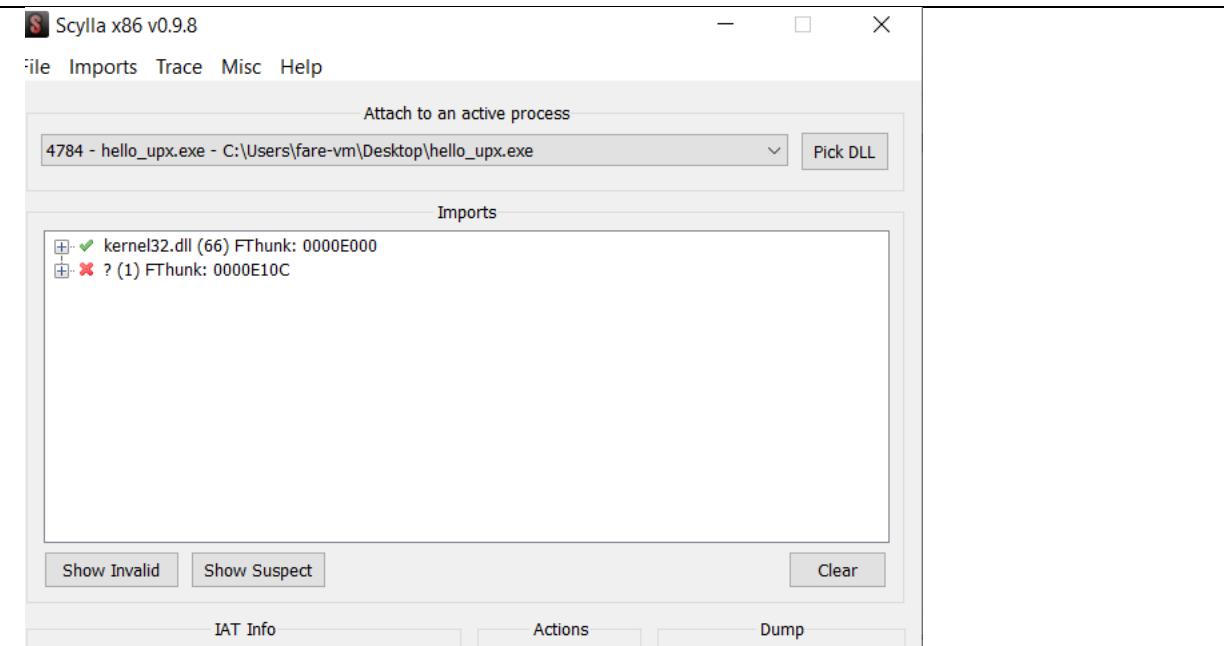
Bước 6.6. Xuất mã đã giải nén ra disk

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

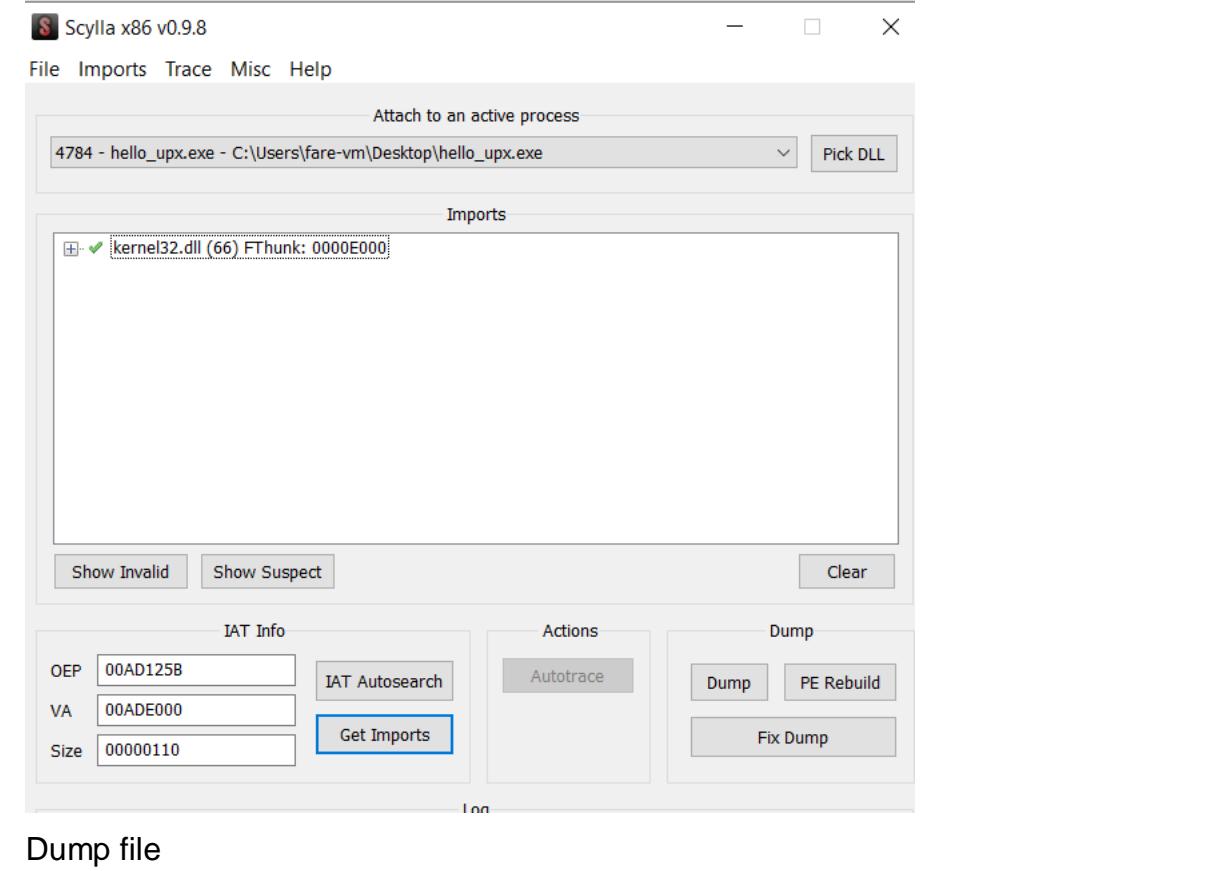


Nhấn `get imports` để lấy các thư viện cần nhập

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

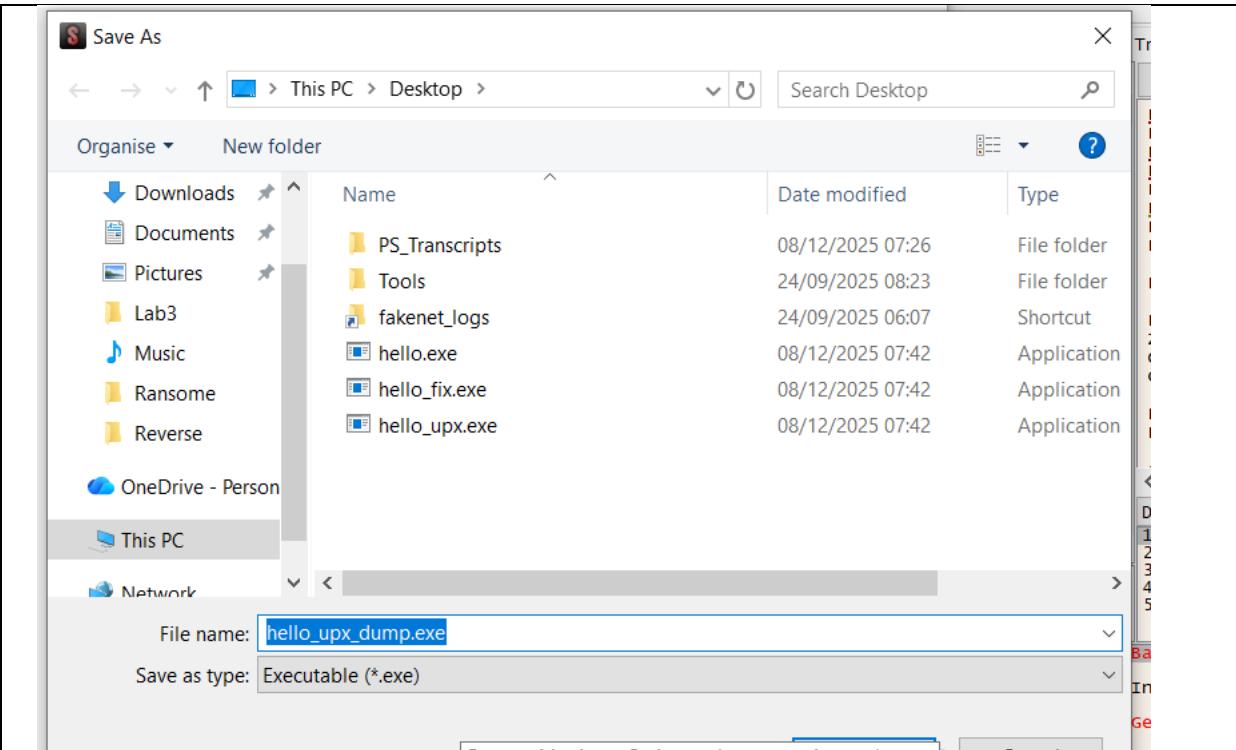


Xóa đi phần FThunk bị dư



Dump file

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao



Và fix lại file

Show Invalid Show Suspect Clear

IAT Info

OEP 00AD125B
VA 00ADE000
Size 00000110

IAT Autosearch Get Imports

Actions

Autotrace

Dump

Dump PE Rebuild Fix Dump

Log

```
DIRECT IMPORTS - Found 0 possible direct imports with 0 unique APIs!
Dump success C:\Users\fare-vm\Desktop\hello_upx_dump.exe
Generating PE header checksum
Rebuild success C:\Users\fare-vm\Desktop\hello_upx_dump.exe
-> Old file size 0x00017E00 new file size 0x00017E00 (100 %)
Import Rebuild success C:\Users\fare-vm\Desktop\hello_upx_dump_SCY.exe
```

Imports SCY 0x00000000 0x00000000 hello_upx.exe

Windows Terminal 00AD0000 hello_upx.exe

Kiểm tra trong IDA

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```

assume cs:UPX0
;org 0AD100h
assume es:nothing, ss:nothing, ds:UPX0, fs:nothing, gs:nothing

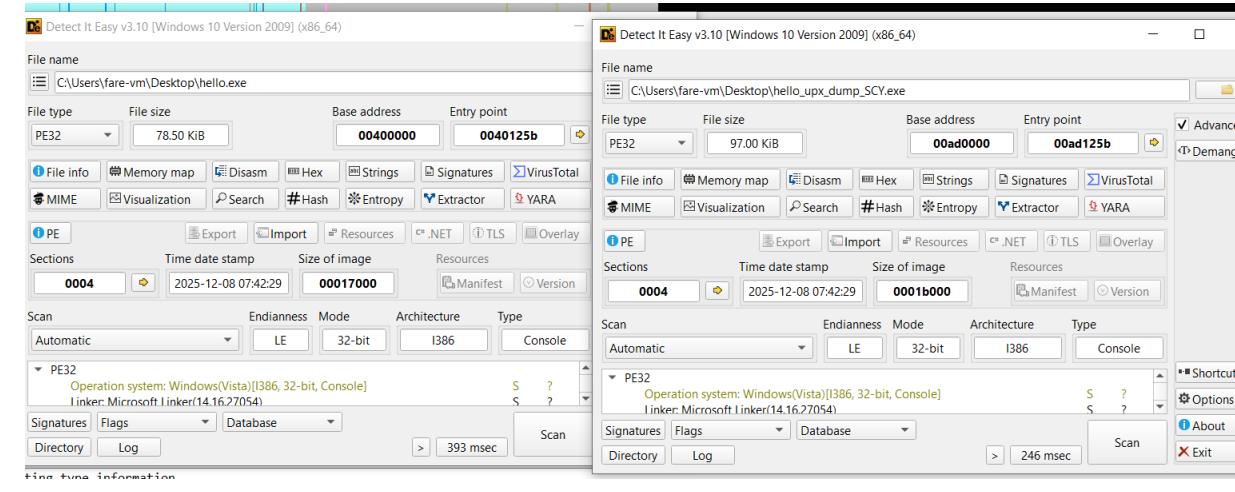
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h

    push    ebp
    mov     ebp, esp
    push    offset Buffer ; "Hello world"
    call    _puts
    add    esp, 4
    xor    eax, eax
    pop    ebp
    retn

```

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:

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

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)

```

1 int main()
2 {
3     int v1; // [sp+14h] [bp-Ch]@1
4
5     __main();
6     printf("Supported authentication methods:\n1. Hardcode\n2. Username/password\n3. Double keys\nEnter your choice: ");
7     scanf("%d", &v1);
8     fflush((FILE *)__iob[0]->_ptr);
9     if ( v1 == 1 )
10    {
11        hardCode();
12    }
13    else if ( v1 == 2 )
14    {
15        userpass();
16    }
17    else
18    {
19        if ( v1 != 3 )
20        {
21            puts("Invalid authentication method.");
22            system("pause");
23            exit(0);
24        }
25        double_key_check();
26    }
27    system("pause");
28
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

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
00000010	B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030	00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00€...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°..Í!.Í!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 0D 0A 24 00 00 00 00 00 00 00 00	mode....\$.....
00000080	50 45 00 00 04 01 0F 00 03 7E A3 5F 00 72 00 00	PE..L...>~£_r..
00000090	D0 02 00 00 E0 00 07 01 0B 01 02 16 00 14 00 00	Đ...à.....
000000A0	00 2C 00 00 00 02 00 00 80 12 00 00 00 10 00 00€.....
000000B0	00 30 00 00 00 00 40 00 00 10 00 00 00 02 00 00	.0.....@.....
000000C0	04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00
000000D0	00 30 01 00 00 04 00 00 4F 79 01 00 03 00 00 00	.0.....Oy.....
000000E0	00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00
000000F0	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000100	00 70 00 00 24 04 00 00 00 00 00 00 00 00 00 00	.p..\$.....
00000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140	00 90 00 00 18 00 00 00 00 00 00 00 00 00 00 00
00000150	00 00 00 00 00 00 00 00 D8 70 00 00 9C 00 00 00Øp..æ...
00000160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170	00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00text...
00000180	08 12 00 00 00 10 00 00 00 14 00 00 00 04 00 00
00000190	00 00 00 00 00 00 00 00 00 00 00 00 60 00 50 00

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 printf 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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```

Function name: _Z8hardCodev
.text:0000146A _Z8hardCodev proc near ; CODE XREF: _main+444p
    .text:0000146A     = byte ptr -3F0h
    .text:0000146A     push    ebp
    .text:0000146A     mov     ebp, esp
    .text:0000146A     sub     esp, 408h
    .text:0000146A     call    _getchar
    .text:0000146A     mov     dword ptr [esp], offset aEnterTheHardCo ; "Enter the hard-coded password:"
    .text:0000146A     call    _puts
    .text:0000146A     lea     eax, [ebp+input]
    .text:0000146A     mov     [esp+4], eax
    .text:0000146A     mov     dword ptr [esp], offset asc_404213 ; "%[^\\n]"
    .text:0000146A     call    _scanf
    .text:0000146A     lea     eax, [ebp+input]
    .text:0000146A     mov     [esp+4], eax
    .text:0000146A     mov     dword ptr [esp+4], offset aYourInputHardC ; "Your input hard-coded password: %s\n"
    .text:0000146A     call    _printf
    .text:0000146A     lea     eax, [ebp+input]
    .text:0000146A     mov     [esp+4], eax
    .text:0000146A     mov     [esp], eax
    .text:0000146A     call    _Z8is_equalPcs ; is_equal(char *,char *)
    .text:0000146A     test   eax, eax
    .text:0000146A     setnz  al
    .text:0000146A     jz     short loc_401406
    .text:0000146A     call    _Z9success_1v ; success_1(void)
0000086A 0040146A: hardCode(void)

Line 14 of 67

```

Hàm hardCode ở dạng mã giả

```

1 void hardCode()
2 {
3     char input[1000]; // [sp+18h] [bp-3F0h]@1
4
5     getchar();
6     puts("Enter the hard-coded password:");
7     scanf("%[^\\n]", input);
8     printf("Your input hard-coded password: %s\\n", input);
9     if (is_equal(input, "Covid 19 is dangerous") != 0)
10        success_1();
11     else
12        Failed();
13 }

```

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)

```

1 bool __cdecl is_equal(char *a, char *b)
2 {
3     return strcmp(a, b) == 0;
4 }

```

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 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.

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

- 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 printf 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

```

Functions window          IDA View-A          Pseudocode-A          Hex View-1          Structures          Enums          Imports
Line 15 of 67
000008ED 0040146D: hardCode(void)+3

```

Hàm userpass ở dạng mã giả

```

1 void userpass()
2 {
3     size_t v0; // ebx@0
4     int v1; // ebx@13
5     int v2; // ebx@14
6     char your_secret[8]; // [sp+10h] [bp-28h]@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    }

```

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```

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 '415MOVVK'

(Hình ảnh đính kèm)

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

```
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:
415MOVVK
Your input username: 11770713 and password: 415MOVVK
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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

- Quá trình phân tích:

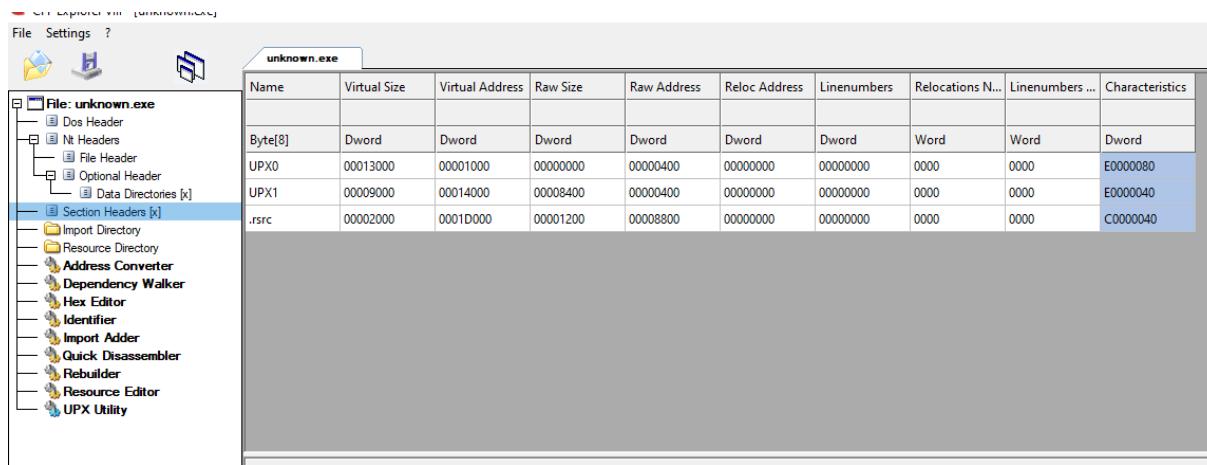
Bước 6.1. Xác định chương trình có bị packed không?

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)

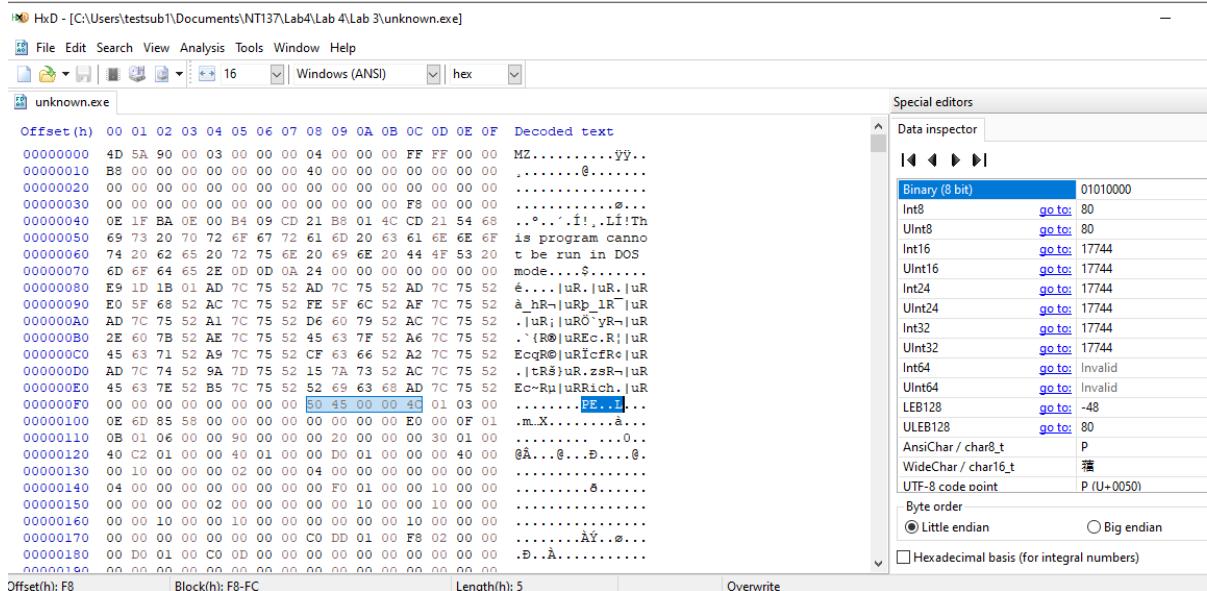
Hình thể hiện có thể thực thi nhiều câu lệnh và mật khẩu bảo vệ



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)

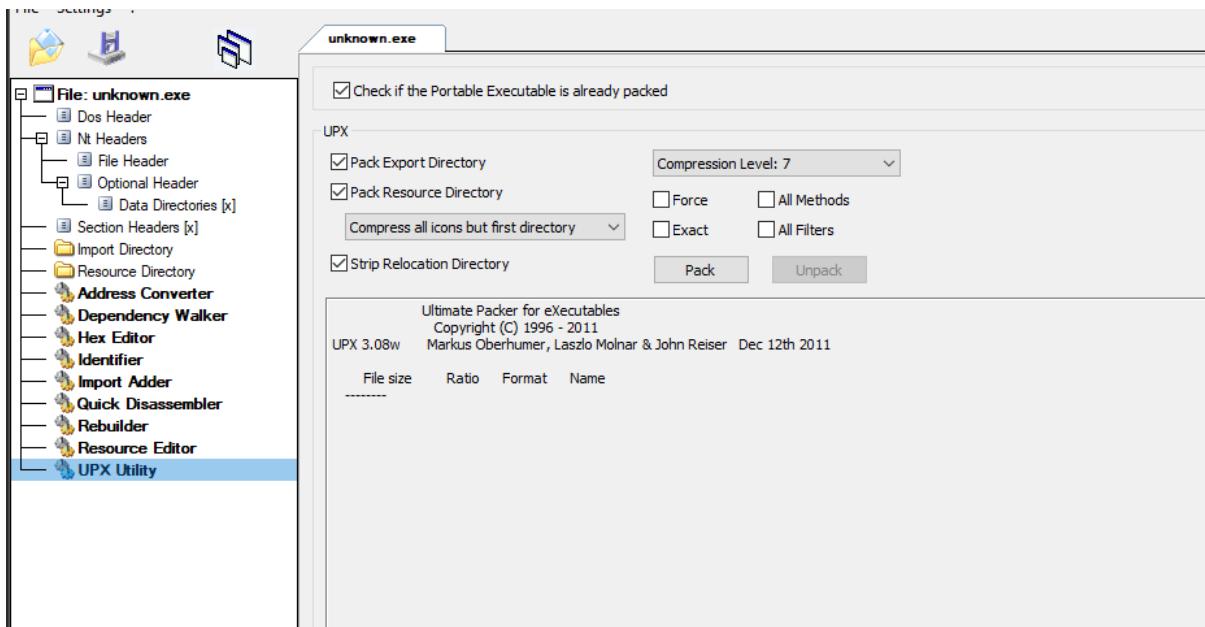


Bước 6.3. Unpack gói tin và tìm kích thước raw của section .text

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

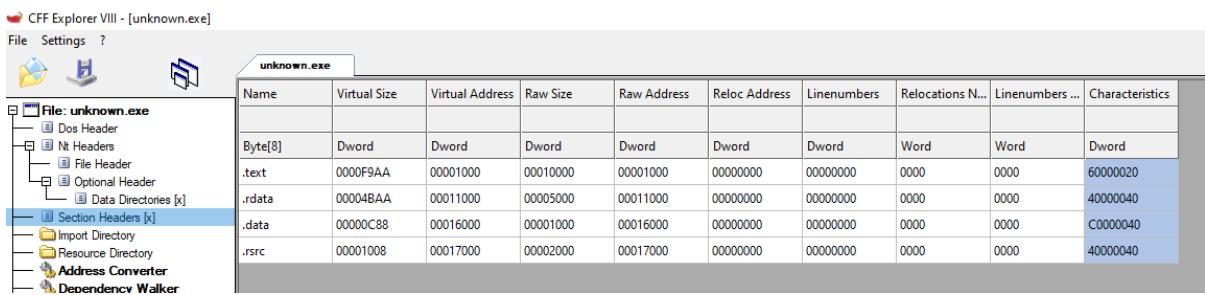
Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

(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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

- 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	TimeStamp	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

Address	Function Name	Library
00411488	InternetCloseHandle	WININET
0041148C	InternetOpenUrlA	WININET
00411490	InternetOpenA	WININET
00411494	InternetReadFile	WININET
0041149C	wavelnOpen	WINMM
004114A0	wavelnStop	WINMM
004114A4	wavelnClose	WINMM
004114A8	wavelnAddBuffer	WINMM
004114AC	wavelnPrepareHeader	WINMM
004114B0	wavelnUnprepareHeader	WINMM
004114B4	wavelnStart	WINMM
004114BC	_imp_htons	WS2_32
004114C0	_imp_gethostbyname	WS2_32
004114C4	_imp_closesocket	WS2_32
004114C8	_imp_socket	WS2_32
004114CC	_imp_send	WS2_32
004114D0	_imp_WSAGetLastError	WS2_32
004114D4	_imp_connect	WS2_32
004114D8	_imp_recv	WS2_32
004114DC	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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

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

Lab 4 – Kỹ thuật Phân tích tĩnh nâng cao

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 gõ lỗi: Process32First/Next để tìm kiếm trình gõ 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