PE FILE STRUCTURE :

A screenshot of a computer

Description automatically generated

**1. Vai trò của DOS Header**

* **Khả năng tương thích ngược**:
  + DOS Header được thiết kế để file PE có thể chạy được trên các hệ thống MS-DOS cũ.
  + Nếu file PE được mở trên MS-DOS, phần này sẽ in ra thông báo: "This program cannot be run in DOS mode."
* **Chứa thông tin định vị PE Header**:
  + Một trong những trường quan trọng nhất trong DOS Header là **e\_lfanew**. Trường này lưu trữ **offset** (khoảng cách) đến **PE Header**, nơi bắt đầu thực sự của cấu trúc file PE trên Windows.

**2. Thành phần chính của DOS Header**

DOS Header có cấu trúc được định nghĩa trong Windows như sau (định dạng IMAGE\_DOS\_HEADER):

| **Trường** | **Kích thước** | **Mô tả** |
| --- | --- | --- |
| e\_magic | 2 bytes | Dấu hiệu nhận diện file DOS, thường có giá trị "MZ". |
| e\_cblp | 2 bytes | Số byte còn lại trong sector cuối cùng. |
| e\_cp | 2 bytes | Số sector trong file. |
| e\_lfanew | 4 bytes | **Offset đến PE Header** (thường là 0x3C), nơi định nghĩa chính của file PE bắt đầu. |

**3. Trong hình**

* **Dòng đầu tiên**:
  + MZ (dạng ASCII) là **magic number** cho biết đây là một file DOS hoặc tương thích với DOS.
* **Thông báo "This program cannot be run in DOS mode":**
  + Được sử dụng để cảnh báo người dùng rằng file này không thể chạy trên hệ điều hành DOS.
* **e\_lfanew:**
  + Giá trị này nằm trong DOS Header, chỉ ra offset tới **PE Header** trong file (thường ở vị trí 0x3C).

A screenshot of a computer

Description automatically generated

Hình ảnh hiển thị phần **NT Headers** của file thực thi CrackMe4.exe trong **CFF Explorer**. Đây là cấu trúc quan trọng trong định dạng **PE (Portable Executable)**, chứa thông tin chi tiết về file thực thi, bao gồm **File Header**, **Optional Header**, và các bảng dữ liệu khác.

**Phân tích chi tiết hình ảnh**

**1. NT Headers**

* **NT Headers** là phần chính của cấu trúc file PE, bắt đầu tại offset được chỉ định bởi trường e\_lfanew trong **DOS Header**.
* NT Headers bao gồm:
  + **Signature**: Xác định định dạng file.
  + **File Header**: Thông tin cơ bản về file PE (số lượng section, loại file...).
  + **Optional Header**: Thông tin chi tiết về cách nạp file vào bộ nhớ.

**2. Signature**

* **Offset**: 0x00000100
  + Đây là vị trí của **NT Headers**, được xác định bởi giá trị e\_lfanew trong **DOS Header**.
* **Kích thước**: 4 bytes (Dword).
* **Giá trị**: 0x00004550
  + Đây là giá trị nhận diện của file PE.
  + Giá trị này tương ứng với chuỗi "PE\0\0" (PE00 trong ASCII), xác nhận rằng file này tuân theo định dạng PE (Portable Executable).

**Vai trò của NT Headers**

* NT Headers đóng vai trò chính trong việc cung cấp thông tin cấu trúc file PE cho hệ điều hành:
  + Hệ điều hành dựa trên các thông tin trong NT Headers để nạp (load) và thực thi file.
  + Bao gồm các thông tin về Entry Point, không gian bộ nhớ, và các thư viện được sử dụng.

**Mối liên hệ với Reverse Engineering**

* **Signature** (PE\0\0):
  + Giúp xác định một file có phải là file PE hợp lệ hay không.
* **Offset** (e\_lfanew từ DOS Header chỉ vào đây):
  + Là bước đầu tiên trong quá trình phân tích file PE, giúp bạn truy cập các thông tin quan trọng trong File Header và Optional Header.

A screenshot of a computer

Description automatically generated

Hình ảnh hiển thị phần **File Header** của file thực thi CrackMe4.exe trong **CFF Explorer**. Đây là một phần của **NT Headers** trong cấu trúc file PE, cung cấp thông tin cơ bản về file thực thi.

**Giải thích từng trường trong File Header**

| **Trường** | **Offset** | **Kích thước** | **Giá trị** | **Ý nghĩa** |
| --- | --- | --- | --- | --- |
| **Machine** | 0x00000104 | 2 bytes (Word) | 014C | Xác định kiến trúc CPU mà file này hỗ trợ. Giá trị 014C đại diện cho **Intel 386** (32-bit). |
| **NumberOfSections** | 0x00000106 | 2 bytes (Word) | 0004 | Số lượng **sections** trong file. File này có 4 section (thường bao gồm .text, .data...). |
| **TimeDateStamp** | 0x00000108 | 4 bytes (Dword) | 5EB0163F | Dấu thời gian (UNIX timestamp) khi file được biên dịch. |
| **PointerToSymbolTable** | 0x0000010C | 4 bytes (Dword) | 00000000 | Offset tới bảng biểu tượng (symbol table). Hiện tại không sử dụng (giá trị là 0). |
| **NumberOfSymbols** | 0x00000110 | 4 bytes (Dword) | 00000000 | Số lượng biểu tượng trong symbol table. Hiện tại không có biểu tượng nào (giá trị là 0). |
| **SizeOfOptionalHeader** | 0x00000114 | 2 bytes (Word) | 00E0 | Kích thước của **Optional Header** (224 bytes, giá trị này là cố định trong PE Header). |
| **Characteristics** | 0x00000116 | 2 bytes (Word) | 0103 | Các thuộc tính của file, ví dụ: **EXE**, **32-bit**, hoặc **DLL**. |

**Chi tiết các trường đáng chú ý**

1. **Machine (014C)**:
   * Giá trị này cho biết file được thiết kế để chạy trên kiến trúc **Intel 386** (32-bit).
   * Nếu giá trị là 8664, file sẽ dành cho **x64 (64-bit)**.
2. **NumberOfSections (0004)**:
   * Xác định số lượng section trong file PE. Mỗi section lưu trữ mã lệnh, dữ liệu, hoặc thông tin tài nguyên (ví dụ: .text, .data, .rsrc).
3. **TimeDateStamp (5EB0163F)**:
   * Đây là một **UNIX timestamp** (số giây từ 1/1/1970).
   * Giá trị này đại diện cho thời điểm file được biên dịch. Bạn có thể chuyển đổi nó sang thời gian thực để xác định ngày biên dịch file.
4. **SizeOfOptionalHeader (00E0)**:
   * Kích thước của **Optional Header**, phần chứa các thông tin quan trọng như **Entry Point**, **Image Base**, và **Section Alignment**.
5. **Characteristics (0103)**:
   * Trường này mô tả các đặc điểm của file. Một số cờ phổ biến:
     + 0x0002: File này là thực thi (Executable).
     + 0x0100: File này là 32-bit.

**Liên quan đến Reverse Engineering**

1. **Machine**:
   * Giúp xác định file này là dành cho kiến trúc 32-bit hay 64-bit, từ đó chọn công cụ phân tích phù hợp.
2. **TimeDateStamp**:
   * Hữu ích khi cần kiểm tra lịch sử biên dịch, thời điểm file được tạo, hoặc phát hiện hành vi giả mạo (timestamp giả).
3. **NumberOfSections**:
   * Giúp bạn biết có bao nhiêu section để phân tích chi tiết. Thông tin này được liên kết với phần **Section Headers**.
4. **Characteristics**:
   * Cho biết đây là file thực thi (EXE) hoặc thư viện (DLL).

A screenshot of a computer

Description automatically generated

Hình ảnh hiển thị phần **Optional Header** của file thực thi CrackMe4.exe trong **CFF Explorer**. **Optional Header** là một phần quan trọng trong **NT Headers** của định dạng file PE (Portable Executable), chứa các thông tin chi tiết cần thiết để nạp và thực thi file.

**Giải thích các trường trong Optional Header**

| **Trường** | **Offset** | **Kích thước** | **Giá trị** | **Ý nghĩa** |
| --- | --- | --- | --- | --- |
| **Magic** | 0x00000118 | 2 bytes (Word) | 010B | Xác định định dạng file PE. 010B đại diện cho PE32 (32-bit). |
| **MajorLinkerVersion** | 0x0000011A | 1 byte | 0E | Phiên bản chính của trình liên kết (linker). |
| **MinorLinkerVersion** | 0x0000011B | 1 byte | 10 | Phiên bản phụ của trình liên kết (linker). |
| **SizeOfCode** | 0x0000011C | 4 bytes (Dword) | 00008400 | Kích thước của section chứa mã thực thi (.text). |
| **SizeOfInitializedData** | 0x00000120 | 4 bytes (Dword) | 0001E000 | Kích thước của section chứa dữ liệu đã khởi tạo (ví dụ: .data, .rdata). |
| **SizeOfUninitializedData** | 0x00000124 | 4 bytes (Dword) | 00000000 | Kích thước của dữ liệu chưa khởi tạo (ví dụ: .bss). |
| **AddressOfEntryPoint** | 0x00000128 | 4 bytes (Dword) | 0000138E | Địa chỉ Entry Point (offset từ Image Base). Đây là nơi CPU bắt đầu thực thi chương trình. |
| **BaseOfCode** | 0x0000012C | 4 bytes (Dword) | 00001000 | Địa chỉ bắt đầu của mã thực thi (offset từ Image Base). |
| **BaseOfData** | 0x00000130 | 4 bytes (Dword) | 00002000 | Địa chỉ bắt đầu của dữ liệu (offset từ Image Base). |
| **ImageBase** | 0x00000134 | 4 bytes (Dword) | 00400000 | Địa chỉ cơ sở của file trong bộ nhớ (Image Base). |
| **SectionAlignment** | 0x00000138 | 4 bytes (Dword) | 00001000 | Kích thước căn chỉnh của section trong bộ nhớ (thường là 4KB = 0x1000). |
| **FileAlignment** | 0x0000013C | 4 bytes (Dword) | 00000200 | Kích thước căn chỉnh của section trong file (thường là 512 bytes = 0x200). |
| **SizeOfImage** | 0x00000150 | 4 bytes (Dword) | 002D0000 | Tổng kích thước của file sau khi được nạp vào bộ nhớ (căn chỉnh theo SectionAlignment). |
| **Subsystem** | 0x0000015C | 2 bytes (Word) | 0002 | Loại subsystem. 0002 là **Windows GUI** (ứng dụng có giao diện đồ họa). |
| **DllCharacteristics** | 0x0000015E | 2 bytes (Word) | 8000 | Đặc điểm của file. 8000 cho biết file hỗ trợ ASLR (Address Space Layout Randomization). |
| **SizeOfStackReserve** | 0x00000160 | 4 bytes (Dword) | 00100000 | Kích thước stack được dự trữ (1MB). |
| **SizeOfStackCommit** | 0x00000164 | 4 bytes (Dword) | 00001000 | Kích thước stack được cam kết (4KB). |
| **SizeOfHeapReserve** | 0x00000168 | 4 bytes (Dword) | 00100000 | Kích thước heap được dự trữ (1MB). |
| **SizeOfHeapCommit** | 0x0000016C | 4 bytes (Dword) | 00001000 | Kích thước heap được cam kết (4KB). |
| **NumberOfRvaAndSizes** | 0x00000174 | 4 bytes (Dword) | 00000010 | Số lượng mục trong bảng Data Directory. |

**Chi tiết các trường quan trọng**

1. **Magic (010B)**:
   * Xác định định dạng file:
     + 010B: PE32 (32-bit).
     + 020B: PE32+ (64-bit).
   * File này là ứng dụng 32-bit.
2. **AddressOfEntryPoint (0000138E)**:
   * Đây là **Entry Point** của chương trình, nơi CPU bắt đầu thực thi mã lệnh.
   * Địa chỉ thực trong bộ nhớ là ImageBase + AddressOfEntryPoint = 0x0040138E.
3. **ImageBase (00400000)**:
   * Là địa chỉ cơ sở mặc định của file khi được nạp vào bộ nhớ.
   * Các địa chỉ khác (Entry Point, BaseOfCode, BaseOfData) được tính tương đối so với Image Base.
4. **SectionAlignment và FileAlignment**:
   * **SectionAlignment** (0x1000 = 4KB): Kích thước căn chỉnh của section trong bộ nhớ.
   * **FileAlignment** (0x200 = 512 bytes): Kích thước căn chỉnh của section trong file trên đĩa.
5. **Subsystem (0002)**:
   * Chỉ ra đây là một ứng dụng **Windows GUI**, sử dụng giao diện đồ họa.
6. **DllCharacteristics (8000)**:
   * Chỉ ra file hỗ trợ ASLR (Address Space Layout Randomization), giúp tăng tính bảo mật bằng cách thay đổi vị trí nạp của file trong bộ nhớ.

**Liên quan đến Reverse Engineering**

1. **AddressOfEntryPoint**:
   * Đây là vị trí đầu tiên cần phân tích khi dịch ngược, vì CPU sẽ bắt đầu thực thi từ đây.
   * Thường là nơi chứa mã khởi tạo hoặc mã thực thi chính.
2. **ImageBase**:
   * Dùng để tính toán các địa chỉ thực trong bộ nhớ.
3. **Subsystem**:
   * Giúp xác định file là ứng dụng giao diện đồ họa (GUI) hay giao diện dòng lệnh (Console).
4. **DllCharacteristics**:
   * Giá trị này cho thấy các cơ chế bảo mật như ASLR, DEP có được kích hoạt hay không.

A screenshot of a computer

Description automatically generated

Hình ảnh hiển thị **Data Directories** trong **Optional Header** của file thực thi CrackMe4.exe thông qua **CFF Explorer**. **Data Directories** là bảng chứa thông tin về các phần (directories) đặc biệt trong file PE, như các thư viện được nhập (import), xuất (export), và các cấu trúc dữ liệu quan trọng khác.

**RVA là gì?**

**RVA (Relative Virtual Address)** là một thuật ngữ thường được sử dụng trong định dạng file PE (Portable Executable) trên Windows. Nó là **địa chỉ tương đối** của một thành phần trong file PE khi được nạp vào bộ nhớ.

**Giải thích từng mục trong Data Directories**

| **Mục** | **Offset** | **Size** | **Value** | **Ý nghĩa** |
| --- | --- | --- | --- | --- |
| **Export Directory RVA** | 0x178 | 4 bytes (Dword) | 00000000 | Địa chỉ RVA của bảng export (không có export trong file này, giá trị là 0). |
| **Export Directory Size** | 0x17C | 4 bytes (Dword) | 00000000 | Kích thước của bảng export. |
| **Import Directory RVA** | 0x180 | 4 bytes (Dword) | 00012324 | Địa chỉ RVA của bảng import, nơi chứa danh sách các thư viện và hàm được nạp. |
| **Import Directory Size** | 0x184 | 4 bytes (Dword) | 0000003C | Kích thước của bảng import. |
| **Resource Directory RVA** | 0x188 | 4 bytes (Dword) | 00015000 | Địa chỉ RVA của bảng resource (chứa tài nguyên như biểu tượng, hình ảnh, v.v.). |
| **Resource Directory Size** | 0x18C | 4 bytes (Dword) | 000171B0 | Kích thước của bảng resource. |
| **Exception Directory RVA** | 0x190 | 4 bytes (Dword) | 00000000 | Không có cấu trúc xử lý ngoại lệ (exception handling). |
| **Security Directory RVA** | 0x198 | 4 bytes (Dword) | 00000000 | Không có thông tin bảo mật (chữ ký số). |
| **Relocation Directory RVA** | 0x1A0 | 4 bytes (Dword) | 00000000 | Không có thông tin về relocation (do file sử dụng ImageBase cố định). |
| **Debug Directory RVA** | 0x1A8 | 4 bytes (Dword) | 00011B50 | Địa chỉ RVA của bảng debug (chứa thông tin phục vụ gỡ lỗi). |
| **Debug Directory Size** | 0x1AC | 4 bytes (Dword) | 00000070 | Kích thước của bảng debug. |
| **TLS Directory RVA** | 0x1C0 | 4 bytes (Dword) | 00000000 | Không có dữ liệu TLS (Thread Local Storage). |
| **Import Address Table RVA** | 0x1D8 | 4 bytes (Dword) | 00012354 | Địa chỉ RVA của bảng IAT (Import Address Table). |
| **Import Address Table Size** | 0x1DC | 4 bytes (Dword) | 00000130 | Kích thước của bảng IAT. |

**Chi tiết một số mục quan trọng**

1. **Import Directory**
   * **Import Directory RVA**: 00012324
     + Đây là nơi lưu trữ danh sách các thư viện và hàm được chương trình nạp vào lúc chạy.
     + Đây là một phần quan trọng để hiểu chương trình sử dụng những hàm hoặc thư viện nào (ví dụ: kernel32.dll, user32.dll).
   * **Import Directory Size**: 0000003C
     + Kích thước của bảng import nhỏ, vì file chỉ sử dụng một số ít hàm từ thư viện.
2. **Resource Directory**
   * **Resource Directory RVA**: 00015000
     + Đây là nơi chứa các tài nguyên của chương trình, như biểu tượng, hình ảnh, chuỗi văn bản.
     + Tài nguyên thường được sử dụng trong ứng dụng GUI.
3. **Debug Directory**
   * **Debug Directory RVA**: 00011B50
     + Đây là nơi chứa thông tin gỡ lỗi, như tên file nguồn hoặc phiên bản trình biên dịch.
     + Dữ liệu debug có thể bị loại bỏ (strip) khỏi file khi phát hành.
4. **Import Address Table (IAT)**
   * **Import Address Table RVA**: 00012354
     + Bảng này chứa địa chỉ của các hàm đã được nạp vào lúc chạy.
     + Đây là phần quan trọng để phân tích mã độc hoặc chương trình, vì nó cho biết các hàm hệ thống mà chương trình gọi.

**Ý nghĩa đối với Reverse Engineering**

* **Import Directory**:
  + Dùng để xác định chương trình sử dụng những hàm nào từ các thư viện (ví dụ: các hàm API Windows).
  + Giúp phát hiện các hành vi đáng ngờ trong mã độc, chẳng hạn các hàm liên quan đến mạng (Wininet, Sockets) hoặc quản lý tiến trình (CreateProcess, OpenProcess).
* **Resource Directory**:
  + Quan trọng để phân tích giao diện người dùng hoặc tài nguyên nhúng trong chương trình (như thông điệp, biểu tượng).
* **Debug Directory**:
  + Thông tin debug có thể tiết lộ thông tin nhạy cảm, như tên file nguồn hoặc các hàm được sử dụng trong quá trình phát triển.

A screenshot of a computer

Description automatically generated

Hình ảnh hiển thị chi tiết **Section Headers** của file thực thi CrackMe4.exe trong **CFF Explorer**, cùng với nội dung của một trong các section. **Section Headers** là một phần quan trọng của file PE, giúp xác định các vùng dữ liệu hoặc mã lệnh trong file và cách chúng được ánh xạ vào bộ nhớ.

**Phân tích các cột trong Section Headers**

| **Cột** | **Ý nghĩa** |
| --- | --- |
| **Name** | Tên của section, ví dụ: .text, .data, .rdata, .rsrc. |
| **Virtual Size** | Kích thước thực tế của section khi được nạp vào bộ nhớ (RAM). |
| **Virtual Address** | Địa chỉ bắt đầu (RVA - Relative Virtual Address) của section trong không gian bộ nhớ ảo. |
| **Raw Size** | Kích thước của section trên đĩa (đã căn chỉnh theo File Alignment, thường là bội số 512). |
| **Raw Address** | Vị trí bắt đầu của section trên file (offset từ đầu file). |
| **Characteristics** | Các thuộc tính của section, ví dụ: chứa mã thực thi, dữ liệu chỉ đọc, hoặc dữ liệu ghi/đọc. |

**Phân tích các Section trong file CrackMe4.exe**

1. **.text**
   * **Chứa mã thực thi của chương trình** (code).
   * **Virtual Size**: 8247 (khoảng 33 KB).
   * **Virtual Address**: 00001000 (offset từ Image Base, khi nạp vào bộ nhớ sẽ là ImageBase + 0x1000).
   * **Raw Size**: 8400 (kích thước trên đĩa đã căn chỉnh).
   * **Characteristics**: 60000020:
     + Section này có thể thực thi (Executable) và chỉ đọc (Read).
2. **.rdata**
   * **Chứa dữ liệu chỉ đọc**, ví dụ: chuỗi ký tự, bảng Import/Export.
   * **Virtual Size**: 5EE (khoảng 1.5 KB).
   * **Virtual Address**: 0000D000.
   * **Raw Address**: 0000B000 (offset từ đầu file).
   * **Characteristics**: 40000040:
     + Section này chỉ có quyền **Read**.
3. **.data**
   * **Chứa dữ liệu có thể đọc/ghi** (ví dụ: biến toàn cục).
   * **Virtual Size**: 130C (khoảng 5 KB).
   * **Virtual Address**: 00013000.
   * **Raw Address**: 00011200 (offset từ đầu file).
   * **Characteristics**: C0000040:
     + Section này có quyền **Read/Write**.
4. **.rsrc**
   * **Chứa tài nguyên của chương trình**, như biểu tượng, hình ảnh, hoặc thông điệp.
   * **Virtual Size**: 171B0 (khoảng 92 KB).
   * **Virtual Address**: 00015000.
   * **Raw Address**: 00017200.
   * **Characteristics**: 40000040:
     + Section này chỉ có quyền **Read**.

**Dữ liệu trong Section**

* Ở dưới cùng của hình, dữ liệu trong **.data** section được hiển thị ở dạng **hexadecimal** và **ASCII**.
* Dữ liệu này có thể là:
  + Các biến toàn cục.
  + Dữ liệu cấu hình.
  + Chuỗi ký tự hoặc giá trị nhị phân.

**Mối liên hệ giữa Section và Reverse Engineering**

1. **.text**:
   * Là nơi chứa mã thực thi, thường là mục tiêu chính khi phân tích mã nguồn hoặc tìm Entry Point.
   * Được kiểm tra đầu tiên khi phân tích hành vi của chương trình.
2. **.rdata**:
   * Là nơi chứa bảng Import, giúp xác định các thư viện và hàm mà chương trình sử dụng.
   * Quan trọng khi phân tích mã độc hoặc chương trình để hiểu các phụ thuộc.
3. **.data**:
   * Là nơi chứa dữ liệu có thể ghi, thường chứa các biến hoặc trạng thái của chương trình.
   * Các giá trị quan trọng hoặc mã hóa thường được tìm thấy ở đây.
4. **.rsrc**:
   * Là nơi chứa tài nguyên giao diện, như biểu tượng hoặc thông báo.
   * Có thể chứa thông tin nhạy cảm hoặc bị nhúng mã độc (trong mã độc phức tạp).

**Tóm tắt**

* **Section Headers** cung cấp thông tin về cấu trúc của file PE và cách các phần được ánh xạ vào bộ nhớ.
* Các section phổ biến:
  + .text: Chứa mã thực thi (code).
  + .rdata: Dữ liệu chỉ đọc, bảng Import/Export.
  + .data: Dữ liệu có thể đọc/ghi.
  + .rsrc: Tài nguyên (resources).
* Việc hiểu rõ các section là bước quan trọng khi phân tích mã độc hoặc thực hiện reverse engineering.