TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

# BÁO CÁO

## PBL6 – Dự Án Chuyên Ngành

## An Toàn Thông Tin

## Phần Mềm Phát Hiện Mã Độc Bằng Học Máy

**Giảng viên hướng dẫn** : ThS. Nguyễn Thị Lệ Quyên

**Sinh viên thực hiện** : Nguyễn Hồng Trường
**Lớp học phần** : 20.14D
**Lớp sinh hoạt** : 20TCLC_DT2

ĐÀ NẴNG, 03/01/2024

# TÓM TẮT ĐỒ ÁN

Mã độc (Malware) là một trong những mối đe dọa nghiêm trọng đối với bảo mật của hệ thống máy tính, thiết bị thông minh và nhiều ứng dụng. Nó có thể gây nguy hiểm cho dữ liệu quan trọng, nhạy cảm bằng cách đánh cắp, sửa đổi, mã hoá hoặc phá hủy dữ liệu. Vậy nên việc nhanh chóng phát hiện và tiêu diệt mã độc luôn là một trong những đề tài được quan tâm nhất trong lĩnh vực an toàn thông tin.

Hiện nay đã có rất nhiều phầm mềm phát hiện mã độc anti virus nhưng chủ yếu sử dụng cách đối chiếu với dữ liệu đã lưu trước điều này dẫn đến một nhược điểm không thể phán đoán trước một mã độc hoàn toàn mới. Từ đó dẫn đến hướng tiếp cận mới trong lĩnh vực này là sử dụng học máy để phát hiện mã độc. Sự kết hợp giữa sức mạnh của máy học và khả năng phân tích sâu rộng của các hệ thống thông tin có thể mở ra cánh cửa cho việc phát hiện kịp thời và ngăn chặn các loại mã độc nguy hiểm.

Trên cơ sở tìm hiểu và nghiên cứu sâu rộng về các phương pháp học máy và an ninh mạng, báo cáo này sẽ đề cập đến những chiến lược, kỹ thuật và kết quả thu được từ việc áp dụng học máy vào việc phát hiện các loại mã độc trong môi trường thực tế.

Trong quá trình thực hiện đồ án, em đã được nhận sự giúp đỡ, hỗ trợ tận tình từ các thầy cô giáo, bạn bè trong khoa, đặc biệt là cô Nguyễn Thị Lệ Quyên - giảng viên hướng dẫn đồ án PBL6 – Dự án chuyên ngành An Toàn Thông Tin. Em xin chân thành cảm ơn! Vì đây là một đề tài khá rộng kết hợp nhiều vùng kiến thức nên khi thực hiện không thể tránh khỏi những thiếu sót về kĩ thuật, rất mong được thầy cô bỏ qua cho em. Kính chúc thầy cô luôn mạnh khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy của mình.

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

**GIẢNG VIÊN HƯỚNG DẪN**

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# I. TỔNG QUAN ĐỀ TÀI

## 1. Giới thiệu

Trong thời đại công nghệ thông tin phát triển như hiện nay, các cuộc tấn công mạng ngày càng trở nên phổ biến và tinh vi hơn. Mã độc là một trong những mối đe dọa lớn nhất đối với an ninh mạng, gây ra nhiều thiệt hại về tài chính, dữ liệu và uy tín cho các tổ chức, doanh nghiệp và cá nhân.

Phát hiện mã độc là một nhiệm vụ quan trọng trong bảo vệ an ninh mạng. Các phương pháp phát hiện mã độc truyền thống thường dựa trên các dấu hiệu nhận dạng tĩnh, như tên, hàm, địa chỉ IP,... Tuy nhiên, các mã độc hiện nay ngày càng tinh vi hơn, có thể thay đổi các dấu hiệu nhận dạng tĩnh để tránh bị phát hiện.

Học máy là một công nghệ đang được ứng dụng rộng rãi trong nhiều lĩnh vực, trong đó có bảo mật thông tin. Học máy có thể học hỏi và phát hiện các mối đe dọa mới từ dữ liệu thực tế.

## 2. Mục tiêu

Mục tiêu của đề tài là nghiên cứu và xây dựng hệ thống phát hiện mã độc sử dụng học máy. Hệ thống phát hiện mã độc sử dụng học máy sẽ có những ưu điểm sau:

- Có thể phát hiện các mã độc mới, chưa có trong cơ sở dữ liệu dấu hiệu nhận dạng.
- Có thể phát hiện các mã độc đã thay đổi các dấu hiệu nhận dạng tĩnh.

## 3. Nội dung nghiên cứu

Đề tài sẽ nghiên cứu các vấn đề sau:

- Tổng quan về mã độc và phát hiện mã độc.
- Tổng quan về học máy.
- Nghiên cứu các kỹ thuật học máy cho phát hiện mã độc.
- Xây dựng hệ thống phát hiện mã độc bằng học máy.

## 4. Kết quả mong đợi

Kết quả mong đợi của đề tài là xây dựng được hệ thống phát hiện mã độc sử dụng học máy có hiệu quả cao, có thể phát hiện các mã độc mới và mã độc đã thay đổi các dấu hiệu nhận dạng tĩnh.

# II. Tổng quan mã độc

## 1. Mã độc

Mã độc (malware) là một loại phần mềm độc hại được tạo ra với mục đích gây hại cho máy tính hoặc hệ thống mạng. Mã độc có thể được sử dụng để đánh cắp dữ liệu, phá hoại hệ thống, hoặc chiếm quyền điều khiển máy tính.

Mã độc được phân loại thành nhiều loại khác nhau, dựa trên chức năng, cách thức lây nhiễm, hoặc mục tiêu tấn công. Một số loại mã độc phổ biến bao gồm:

- Virus: Virus là một loại mã độc có thể tự sao chép và lây lan sang các máy tính khác.
- Worm: Worm là một loại mã độc có thể tự lây lan qua mạng.
- Trojan: Trojan là một loại mã độc được ngụy trang dưới dạng một phần mềm hoặc ứng dụng hợp pháp.
- Rootkit: Rootkit là một loại mã độc có thể xâm nhập vào hệ thống và cấp cho kẻ tấn công quyền truy cập đặc quyền.
- Adware: Adware là một loại mã độc hiển thị quảng cáo trên máy tính của người dùng.
- Spyware: Spyware là một loại mã độc thu thập thông tin cá nhân của người dùng.
- Ransomware: Ransomware là một loại mã độc tống tiền, khóa hệ thống của người dùng và yêu cầu tiền chuộc để giải mã.

## 2. Phân tích mã độc

Phân tích mã độc là việc sử dụng các công cụ, kỹ thuật phân tích nhằm xác định hành vi "độc hại" của mã độc đến hệ thống máy tình. Phân tích mã độc là có mục tiêu nữa là từ kết quả của việc phần tích đưa ra cách để gỡ bỏ và phòng tránh mã độc đó. Về phương pháp phân tích, có hai phương pháp phân tích mã độc là phân tích tĩnh và phân tích động.

### 2.1. Phân tích tĩnh

Phân tích tĩnh mô tả việc phân tích mã, cấu trúc của một phần mềm mà không cần thực thi chúng nhằm mục đích kiểm tra xem tập tin, phần mềm có phải mã độc hay không, và cố gắng xác định hành vi của mã độc. Phân tích tĩnh có thể chia làm hai mức độ: cơ bản và nâng cao.

Ở mức độ cơ bản, người phân tích sẽ không đi sâu vào việc phân tích các đoạn mã của mã độc mà có thể sử dụng một số công cụ để tìm kiếm, kiểm tra một số thông tin hữu ích như: định dạng tập tin, trích xuất các chuỗi, xem tập tin

có bị làm rối không, các hàm và thư viện mà tập tin có thể sử dụng, … Một số kỹ thuật sử dụng dùng trong phân tích tĩnh như:

Cơ bản:

- Hàm băm
- Hàm và thư viện liên kết
- Tìm kiếm Strings
- Kiểm tra định dạng, thông tin tệp tin
- Packing và Obfuscation

Nâng cao:

- Disassembly / Reverse

Các phương pháp phân tích tĩnh cơ bản rất tốt cho việc phân loại ban đầu, tuy nhiên chúng không cung cấp đủ thông tin để phân tích hoàn toàn phần mềm độc hại. Có thể sử dụng phân tích tĩnh để rút ra một số kết luận sơ bộ, nhưng cần phải phân tích sâu hơn để nắm được toàn bộ hành vi của mã độc. Ví dụ: có thể thấy rằng một API cụ thể đã được import, nhưng sẽ không biết nó được sử dụng như thế nào hoặc liệu nó có được sử dụng hay không.

Ở mức độ cao hơn, phức tạp hơn, phân tích tĩnh chính là việc dịch ngược (reverse) chương trình để đọc các đoạn mã thực thi nó (thường ở dạng assembly), từ đó biết được logic và mục đích của phần mềm. Đây là một phương pháp đòi hỏi người phân tích có kỹ năng chuyên môn, nó cũng là phương thức thường dùng và đáng tin cậy nhất trong phân tích tĩnh. Việc phân tích các đoạn mã assembly sẽ cho biết tất cả các kịch bản thực thi có thể có của mã độc, tất cả hành vi mã độc và không hề bị hạn chế bởi điều kiện gì. Tuy vậy, để đọc, phân tích toàn bộ mã assembly tốn rất nhiều thời gian. Còn một ưu điểm nữa của phân tích tĩnh đó là không cần thực thi trực tiếp mã độc, điều này có nghĩa là hệ thống sẽ không bị ảnh hưởng và gây nguy hiểm.

### 2.2. Phân tích động

Phân tích động là việc phân tích các hành vi, chức năng của mã độc bằng cách thực thi phần mềm độc hại. Không giống như phân tích tĩnh, phân tích động cho phép quan sát chức năng thực sự của phần mềm độc hại, ví dụ: sự tồn tại của một chuỗi hành động trong mã nhị phân không có nghĩa là hành động sẽ thực sự thực thi. Phân tích động cũng là một cách hiệu quả để xác định chức năng của phần mềm độc hại. Ví dụ: nếu phần mềm độc hại là keylogger, phân tích động có thể cho phép xác định vị trí tập tin ghi nhật ký của keylogger trên hệ thống, các loại bản ghi mà nó lưu giữ, giải mã nơi nó gửi thông tin, …. Loại thông tin chi tiết này sẽ khó đạt được nếu chỉ sử dụng các kỹ thuật tĩnh cơ bản.

Trong phân tích động, những thông tin cần được theo dõi trong suốt quá trình thực thi mã độc như: các hành vi với tập tin, các tiến trình, hành vi liên quan đến registry, hành vi liên quan đến mạng, hay lắng nghe các gói tin, … Việc theo dõi các hành vi và tác động của việc thực thi lên hệ thống rất hữu ích, Microsoft cũng có một bộ công cụ hỗ trợ theo dõi, phân tích, đó là bộ Windows Sysinternals Suite. Một số công cũ hỗ trợ việc phân tích động chẳng hạn như Proces Monitor (Procmon), Process Explorer, Autoruns, TCPview, Regshot, Wireshark, …

- Process Monitor, hay procmon là một công cụ giám sát nâng cao dành cho Windows cung cấp một cách để giám sát hoạt động registry, file system, network, process và thread nhất định. Nó kết hợp và nâng cao chức năng của hai công cụ kế thừa: FileMon và RegMon.
- Process Explorer là phần mềm miễn phí của Microsoft quản lý tác vụ cực kỳ mạnh mẽ sẽ chạy khi thực hiện phân tích động. Nó có thể cung cấp cái nhìn chi tiết về các tiến trình hiện đang chạy trên hệ thống. Process Explorer có thể liệt kê các tiền trình đang hoạt động, các tập tin DLL được tải bởi một tiến trình, các thuộc tính quy trình khác nhau và thông tin hệ thống tổng thể. Cũng có thể sử dụng nó để hủy một quy trình, trích xuất một tiến trình đang chạy,…
- TCPView là công cụ có giao diện, thân thiện với người dùng, nó cung cấp được nhiều thông tin chi tiết về các kết nối.
- Công cụ phổ biến để kiểm tra thông tin startup là Autoruns được Microsoft cung cấp trong bộ Sysinternal Suite.
- Regshot là một công cụ so sánh registry mã nguồn mở cho phép tạo và so sánh hai registry snapshot. Để sử dụng Regshot để phân tích phần mềm độc hại, Regshot sẽ so sánh hai snapshot trước và sau khi thực thi phần mềm độc hại.

Mặc dù các kỹ thuật phân tích động cực kỳ mạnh mẽ, chúng chỉ nên được thực hiện sau khi phân tích tĩnh cơ bản đã hoàn thành vì phân tích động có thể khiến mạng và hệ thống gặp rủi ro. Các kỹ thuật động có những hạn chế bởi vì không phải tất cả các đoạn mã đều có thể thực thi khi một phần mềm độc hại được chạy. Ví dụ: trong trường hợp phần mềm độc hại dòng lệnh (command-line) yêu cầu tham số truyền vào, mỗi tham số có thể thực thi chức năng chương trình khác nhau và nếu không biết các tùy chọn đó, sẽ không thể kiểm tra tất cả chức năng của chương trình.

**Debug**

Ở mức độ cao hơn, phân tích động là việc tiến hành debug một chương trình. Debugger – trình gỡ lỗi là công cụ hữu ích để phân tích phần mềm độc hại. Chúng cho phép kiểm tra mã ở cấp độ chi tiết hơn và cung cấp toàn quyền kiểm soát các hành vi có thể xảy ra trong thời gian chạy của phần mềm độc hại. Sử dụng debugger, có thể thực hiện phân tích từng lệnh hay một hàm chức năng một cách thuận tiện. Nói cách khác, có thể thực thi chương trình ở chế độ chuyển động chậm trong khi nghiên cứu mọi hành động của nó. Cũng có thể sử dụng trình gỡ lỗi để thực thi một vài chức năng được chọn thay vì toàn bộ chương trình, điều này rất hữu ích nếu cần bỏ qua mã chống gỡ lỗi hoặc thực thi với một tham số truyền vào nào đó.

**Sandbox**

Để hạn chế các rủi ro trong việc thực thi mã độc đối với hệ thống, thì phân tích động thường được thực hiện trên môi trường ảo, cách li với môi trường thật.

Sandbox là một trong những công cụ được sử dụng khá phổ biến cho việc theo dõi, kiểm tra và trích xuất các thông tin thực thi của một tập tin trên hệ thống. Bên cạnh đó Sandbox là một cơ chế bảo mật để chạy các chương trình không đáng tin cậy trong một môi trường an toàn mà không sợ làm hại hệ thống thực. Sandbox bao gồm các môi trường ảo hóa thường mô phỏng hệ thống máy tính thật, các dịch vụ mạng theo một số cách để đảm bảo rằng phần mềm hoặc phần mềm độc hại đang được kiểm tra sẽ hoạt động bình thường.

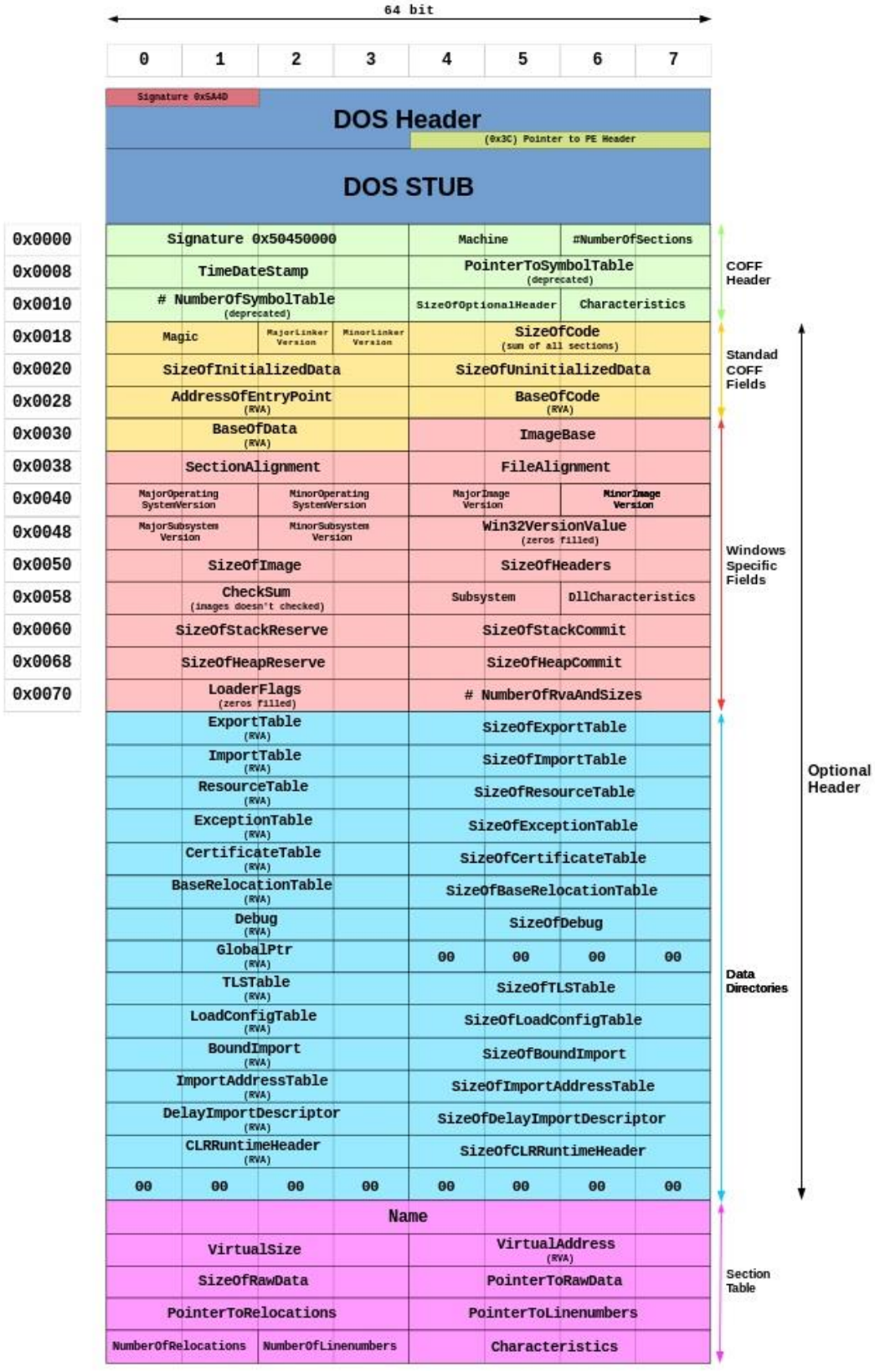
Có khá nhiều phần mềm Sandbox khác nhau, phổ biến nhất có thể kể đến Cuckoo Sandbox. Các thông tin có thể được trích xuất từ báo cáo Cuckoo:

- Registry
- Thông tin về tệp tin
- Tiến trình
- Địa chỉ IP và truy vấn DNS
- Dlls, API calls

❖ *Trong đồ án này em tập trung vào phân tích tĩnh cụ thể là phân tích tiêu đề PE (Portable Executable) Header của các tệp thực thi (EXE, DLL)*

## 2.3. Phân tích cấu trúc định dạng PE Header

### 2.3.1. Giới thiệu




*Hình 1: Cấu trúc PE file*

PE header là một phần của định dạng file thực thi Windows (PE). Nó chứa thông tin quan trọng về file, chẳng hạn như tên file, loại file, địa chỉ entry point, và các thư viện được tải.

PE header được chia thành hai phần chính:

- DOS header: Phần này chứa thông tin cơ bản về file, chẳng hạn như kích thước file, thời gian tạo, và thời gian sửa đổi.
- NT header: Phần này chứa thông tin chi tiết hơn về file, chẳng hạn như các section, thư viện được tải, và các thuộc tính khác.

### 2.3.2. DOS header

Bắt đầu với 64 bytes (0x40), DOS MZ header chứa các thông tin cơ bản về của một PE file hợp lệ.

Header này chúng ta chỉ cần quan tâm đến e_lfanew tại offset 0x3C, chứa offset bắt đầu của PE header. Windows loader sẽ nhìn vào trường này để bỏ qua DOS stub và đi đến PE header.

| | | | |
|---|---|---|---|
| +00 | WORD | e_magic | Magic Number (“MZ”) |
| +02 | WORD | e_cblp | Byte on last page of file |
| +04 | WORD | e_cp | Pages in file |
| +06 | WORD | e_crlc | Relocations |
| +08 | WORD | e_cparhdr | Size of header in paragraphs |
| +0A | WORD | e_minalloc | Minimum extra paragraphs needed |
| +0C | WORD | e_maxalloc | Maximum extra paragraphs needed |

| | | | |
|---|---|---|---|
| +0E | WORD | e_ss | Initial (relative) SS value |
| +10 | WORD | e_sp | Initial SP value |
| +12 | WORD | e_csum | Checksum |
| +14 | WORD | e_ip | Initial IP value |
| +16 | WORD | e_cs | Initial (relative) CS value |
| +18 | WORD | e_lfarlc | File address of relocation table |
| +1A | WORD | e_ovno | Overlay number |
| +1C | Array[4] of WORD | e_res | Reserved words |
| +24 | WORD | e_oemid | OEM identifier (for e_oeminfo) |
| +26 | WORD | e_oeminfo | OEM information; e_oemid specific |
| +28 | WORD | e_res2 | Reserved words |
| +3C | DWORD | e_lfanew | File address of new exe header |

### 2.3.3. NT header

| | | |
|---|---|---|
| +00 | DWORD | Signature (“PE”) |
| +04 | WORD | Machine |
| +06 | WORD | Number of Sections |
| +08 | DWORD | TimeDateStamp |
| +0C | DWORD | PointerToSymbolTable |
| +10 | DWORD | NumberOfSymbols |
| +14 | WORD | SizeOfOptionalHeader |
| +16 | WORD | Characteristics |

- **Signature**: Một chuỗi 4 byte xác định định dạng file.
- **File header size**: Kích thước của NT header.
- **Machine**: Kiểu máy mà file được thiết kế để chạy.
- **NumberOfSections**: Số lượng section của file.
- **Time stamp**: Thời gian tạo file.
- **Date stamp**: Thời gian sửa đổi file.
- **Pointer to symbol table**: Địa chỉ của bảng symbol.
- **NumberOfSymbols**: Số lượng symbol trong bảng symbol.
- **Size of optional header**: Kích thước của phần header tùy chọn.
- **Characteristics**: Các thuộc tính của file.

### 2.3.4. Phần header tùy chọn

Phần header tùy chọn chứa các thông tin bổ sung về file, chẳng hạn như:

- Data directory table: Bảng chứa thông tin về các thư viện được tải, các section, và các dữ liệu khác.
- Image base: Địa chỉ cơ sở của file trong bộ nhớ.
- Section alignment: Độ lệch của các section.
- File alignment: Độ lệch của các byte trong file.
- Major version number: Phiên bản chính của file.
- Minor version number: Phiên bản phụ của file.
- Subsystem: Loại hệ thống con mà file được thiết kế để chạy.
- Dll characteristics: Các thuộc tính của file DLL.

# III. Tổng quan học máy

## 1. Giới thiệu

Máy học là một thuật ngữ đề cập đến các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm, đồng thời cải thiện hiệu suất theo thời gian.

Học máy là một thành phần quan trọng của lĩnh vực khoa học dữ liệu đang phát triển. Thông qua việc sử dụng phương pháp thống kê, các thuật toán được đào tạo để phân loại hoặc dự đoán và khám phá những thông tin chi tiết trong các dự án khai thác dữ liệu.

Những thông tin chi tiết này hỗ trợ, thúc đẩy việc đưa ra quyết định trong các ứng dụng, công cụ hỗ trợ doanh nghiệp, người dùng. Khi khối lượng dữ liệu tiếp tục mở rộng và phát triển, khả năng dự đoán, phân tích chính xác của máy học sẽ tăng lên.

## 2. Phân loại

Các mô hình học máy phổ biến được chia thành hai loại chính:

- **Mô hình học máy có giám sát**: Các mô hình này được đào tạo trên dữ liệu có nhãn, trong đó mỗi mẫu dữ liệu được gắn nhãn với một lớp hoặc giá trị.

- **Mô hình học máy không có giám sát**: Các mô hình này được đào tạo trên dữ liệu không có nhãn, trong đó các mẫu dữ liệu không được gắn nhãn với bất kỳ lớp hoặc giá trị nào.

**Mô hình học máy có giám sát**

- **Phân loại**: Phân loại là một loại học máy có giám sát, trong đó mục tiêu là dự đoán lớp của một mẫu dữ liệu. Một số mô hình phân loại phổ biến bao gồm:
    - **Cây quyết định:** Cây quyết định là một mô hình phân loại đơn giản nhưng hiệu quả. Nó hoạt động bằng cách tạo một cây phân nhánh, trong đó mỗi nhánh đại diện cho một quyết định.
    - **Máy hỗ trợ vector (SVM):** SVM là một mô hình phân loại mạnh mẽ có thể được sử dụng để phân loại các mẫu dữ liệu phức tạp. Nó hoạt động bằng cách tìm ra một đường phân cách tối ưu giữa các lớp.
    - **Random forest**: Random forest là một mô hình phân loại kết hợp kết quả của nhiều cây quyết định. Nó có độ chính xác cao và khả năng chống nhiễu tốt.
    -
- **Dự đoán**: Dự đoán là một loại học máy có giám sát, trong đó mục tiêu là dự đoán một giá trị của một mẫu dữ liệu. Một số mô hình dự đoán phổ biến bao gồm:
    - Hồi quy tuyến tính: Hồi quy tuyến tính là một mô hình dự đoán đơn giản nhưng hiệu quả. Nó hoạt động bằng cách tìm ra một đường thẳng phù hợp nhất với dữ liệu.
    - Hồi quy logistic: Hồi quy logistic là một mô hình dự đoán mạnh mẽ có thể được sử dụng để dự đoán các giá trị rời rạc. Nó hoạt động bằng cách tìm ra một đường cong logistic phù hợp nhất với dữ liệu.
    - Recurrent neural network (RNN): RNN là một mô hình học sâu có thể được sử dụng để dự đoán các giá trị theo thời gian. Nó hoạt động bằng cách lưu trữ thông tin từ các mẫu dữ liệu trước đó.

**Mô hình học máy không có giám sát**

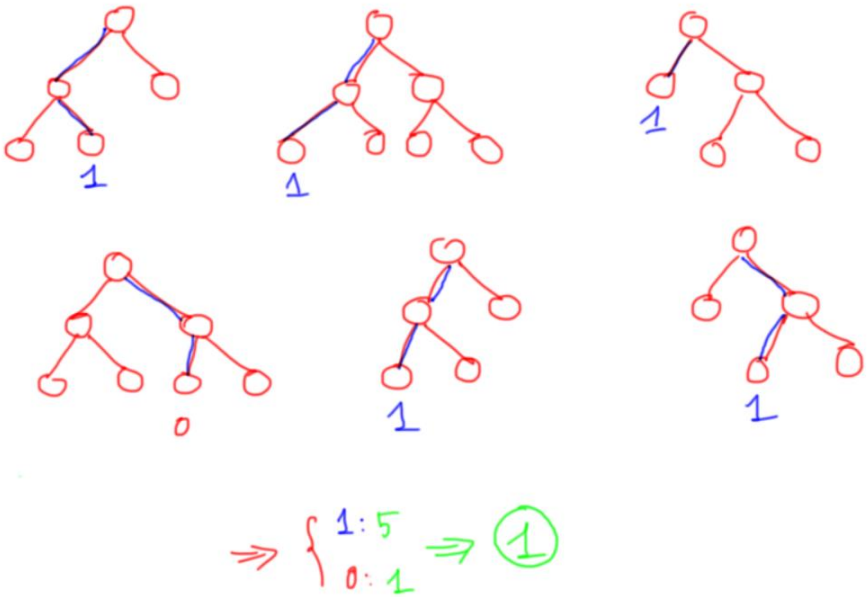
- **KMeans**: KMeans là một mô hình học máy không có giám sát được sử dụng để phân cụm các mẫu dữ liệu. Nó hoạt động bằng cách nhóm các mẫu dữ liệu gần nhau thành các cụm.
- **Hierarchical clustering**: Hierarchical clustering là một mô hình học máy không có giám sát được sử dụng để phân cụm các mẫu dữ liệu theo một cấu trúc phân cấp.
- **PCA**: PCA là một mô hình học máy không có giám sát được sử dụng để giảm chiều của dữ liệu. Nó hoạt động bằng cách tìm ra các thành phần chính của dữ liệu.

❖ *Trong đồ án này em sử dụng học máy có giám sát cụ thể là mô hình **Random forest***

## 3. Random forest

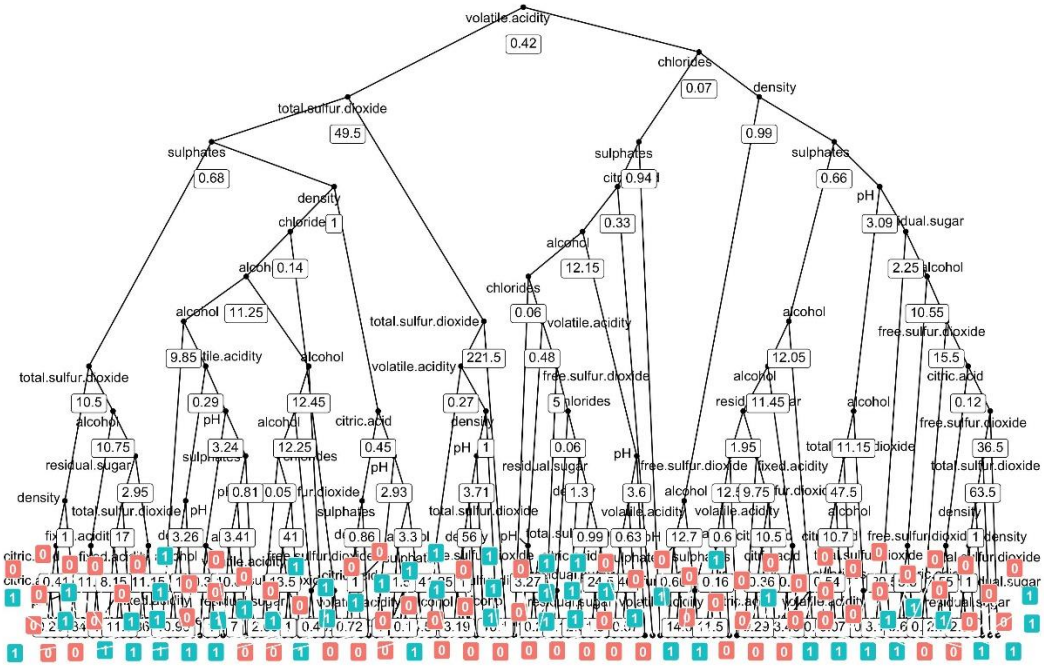
### 3.1. Ý tưởng

Mô hình rừng cây được huấn luyện dựa trên sự phối hợp giữa luật kết hợp (ensembling) và quá trình lấy mẫu tái lặp (boostrapping). Cụ thể thuật toán này tạo ra nhiều cây quyết định mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử (voting) từ toàn bộ những cây quyết định. Như vậy một kết quả dự báo được tổng hợp từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình. Điều này giúp cho mô hình khắc phục được hiện tượng quá khớp.




*Hình 2: Ý tưởng mô hình Random Forest*

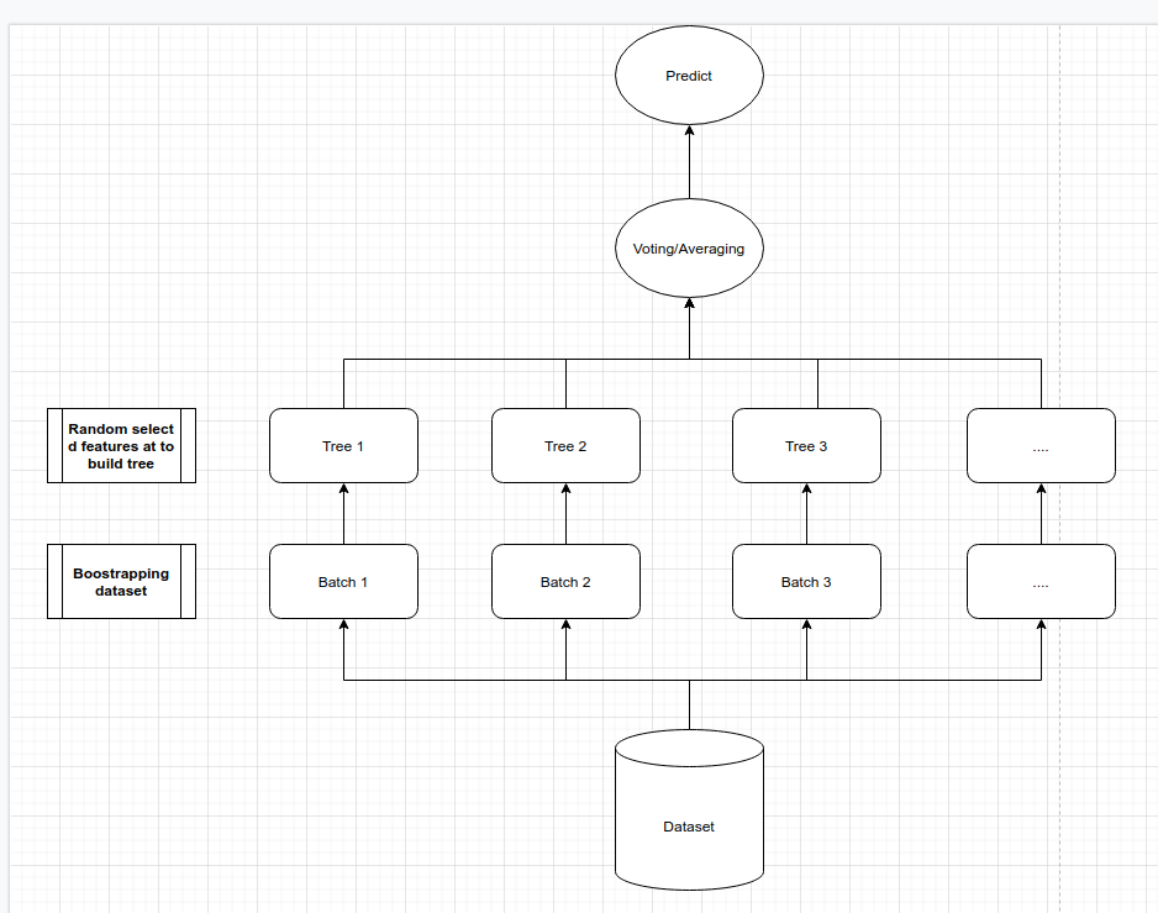
## 3.2. Cấu trúc




*Hình 3: Cấu trúc mô hình Random Forest*

Mô hình rừng cây sẽ áp dụng cả hai phương pháp học kết hợp (ensemble learning) và lấy mẫu tái lập (boostrapping). Thứ tự của quá trình tạo thành một mô hình rừng cây như sau:

1. Lấy mẫu tái lập một cách ngẫu nhiên từ tập huấn luyện để tạo thành một tập dữ liệu con.
2. Lựa chọn ra ngẫu nhiên ***d*** biến và xây dựng mô hình cây quyết định dựa trên những biến này và tập dữ liệu con ở bước 1. Chúng ta sẽ xây dựng nhiều cây quyết định nên bước 1 và 2 sẽ lặp lại nhiều lần.
3. Thực hiện bầu cử hoặc lấy trung bình giữa các cây quyết định để đưa ra dự báo.




*Hình 4: Sơ đồ khối của mô hình Random Forest*

Kết quả dự báo từ mô hình rừng cây là sự kết hợp của nhiều cây quyết định nên chúng tận dụng được trí thông minh đám đông và giúp cải thiện độ chính xác so với chỉ sử dụng một mô hình cây quyết định.

Nếu như mô hình cây quyết định thường bị nhạy cảm với dữ liệu ngoại lai (outlier) thì mô hình rừng cây được huấn luyện trên nhiều tập dữ liệu con khác nhau, trong đó có những tập được loại bỏ dữ liệu ngoại lai, điều này giúp cho mô hình ít bị nhạy cảm với dữ liệu ngoại lai hơn.

Sự kết hợp giữa các cây quyết định giúp cho kết quả ít bị chệch và phương sai giảm. Như vậy chúng ta giảm thiểu được hiện tượng quá khớp ở mô hình rừng cây, một điều mà mô hình cây quyết định thường xuyên gặp phải.

Cuối cùng các bộ dữ liệu được sử dụng từ những cây quyết định đều xuất phát từ dữ liệu huấn luyện nên quy luật học được giữa các cây quyết định sẽ gần tương tự như nhau và tổng hợp kết quả giữa chúng không có xu hướng bị chệch.

# IV. PHƯƠNG PHÁP THỰC HIỆN

## 1. Làm sạch data để huấn luyện model

Do không có nguồn file mã độc đủ để huấn luyện model nên thường chúng ta phải tải trên mạng về và làm sạch để lấy ra những đặc trưng mình có thể sử dụng.

```
dataset=pd.read_csv(r'/content/drive/MyDrive/PTMD/PE_Header/data.csv',sep='|')
dataset.groupby(dataset['legitimate']).size()
```

```
legitimate
0    46724
1    41323
dtype: int64
```

*Hình 5: Đọc dữ liệu từ file csv*

```
dataset
```

| | Name | md5 | Machine | SizeOfOptionalHeader | Characteristics | M |
|---|---|---|---|---|---|---|
| 0 | memtest.exe | 631ea355665f28d4707448e442fbf5b8 | 332 | 224 | 258 | |
| 1 | ose.exe | 9d10f99a6712e28f8acd5641e3a7ea6b | 332 | 224 | 3330 | |
| 2 | setup.exe | 4d92f518527353c0db88a70fddcfd390 | 332 | 224 | 3330 | |
| 3 | DW20.EXE | a41e524f8d45f0074fd07805ff0c9b12 | 332 | 224 | 258 | |
| 4 | dwtrig20.exe | c87e561258f2f8650cef999bf643a731 | 332 | 224 | 258 | |
| ... | ... | ... | ... | ... | ... | |
| 88042 | VirusShare_8e292b418568d6e7b87f2a32aee7074b | 8e292b418568d6e7b87f2a32aee7074b | 332 | 224 | 258 | |
| 88043 | VirusShare_260d9e2258aed4c8a3bbd703ec895822 | 260d9e2258aed4c8a3bbd703ec895822 | 332 | 224 | 33167 | |
| 88044 | VirusShare_8d088a51b7d225c9f5d11d239791ec3f | 8d088a51b7d225c9f5d11d239791ec3f | 332 | 224 | 258 | |
| 88045 | VirusShare_4286dccf67ca220fe67635388229a9f3 | 4286dccf67ca220fe67635388229a9f3 | 332 | 224 | 33166 | |
| 88046 | VirusShare_d7648eae45f09b3adb75127f43be6d11 | d7648eae45f09b3adb75127f43be6d11 | 332 | 224 | 258 | |

88047 rows × 57 columns

*Hình 6: Chi tiết dataset chưa làm sạch*

Lựa chọn những đặc trưng có giá trị

```
pe_df = dataset[[ 'Machine',
                  'Characteristics',
                  'DllCharacteristics',
                  'Subsystem',
                  'AddressOfEntryPoint',
                  'MinorLinkerVersion',
                  'NumberOfRvaAndSizes',
                  'MinorLinkerVersion',
                  'SizeOfInitializedData',
                  'SizeOfUninitializedData',
                  'ImageBase',
                  'MajorSubsystemVersion',
                  'CheckSum',
                  'SizeOfStackReserve',
                  'SectionsMaxEntropy',
                  'SectionsNb',
                  'legitimate']]
pe_df
```

*Hình 7: Lựa chọn những đặc trưng có ích để huấn luyện mô hình*

Sau khi chọn những đặc trưng có giá trị ta loại bỏ những dòng trùng nhau và thu về bộ data đã được làm sạch

pe_df

| | Machine | Characteristics | DllCharacteristics | Subsystem | AddressOfEntryPoint | MinorLinkerVersion | NumberOfRvaAndSizes |
|---|---|---|---|---|---|---|---|
| 0 | 332.0 | 258.0 | 1024.0 | 16.0 | 6135.0 | 0.0 | 16.0 |
| 1 | 332.0 | 3330.0 | 33088.0 | 2.0 | 81778.0 | 0.0 | 16.0 |
| 2 | 332.0 | 3330.0 | 32832.0 | 2.0 | 350896.0 | 0.0 | 16.0 |
| 3 | 332.0 | 258.0 | 33088.0 | 2.0 | 451258.0 | 0.0 | 16.0 |
| 4 | 332.0 | 258.0 | 33088.0 | 2.0 | 217381.0 | 0.0 | 16.0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 81318 | 332.0 | 258.0 | 34112.0 | 2.0 | 14819.0 | 0.0 | 16.0 |
| 81319 | 332.0 | 258.0 | 33088.0 | 2.0 | 57436.0 | 0.0 | 16.0 |
| 81320 | 332.0 | 258.0 | 33088.0 | 2.0 | 24735.0 | 0.0 | 16.0 |
| 81321 | 332.0 | 258.0 | 33088.0 | 2.0 | 61658.0 | 0.0 | 16.0 |
| 81322 | 332.0 | 271.0 | 32768.0 | 2.0 | 12491.0 | 0.0 | 16.0 |

81323 rows × 17 columns

*Hình 8: Data sau khi được làm sạch*

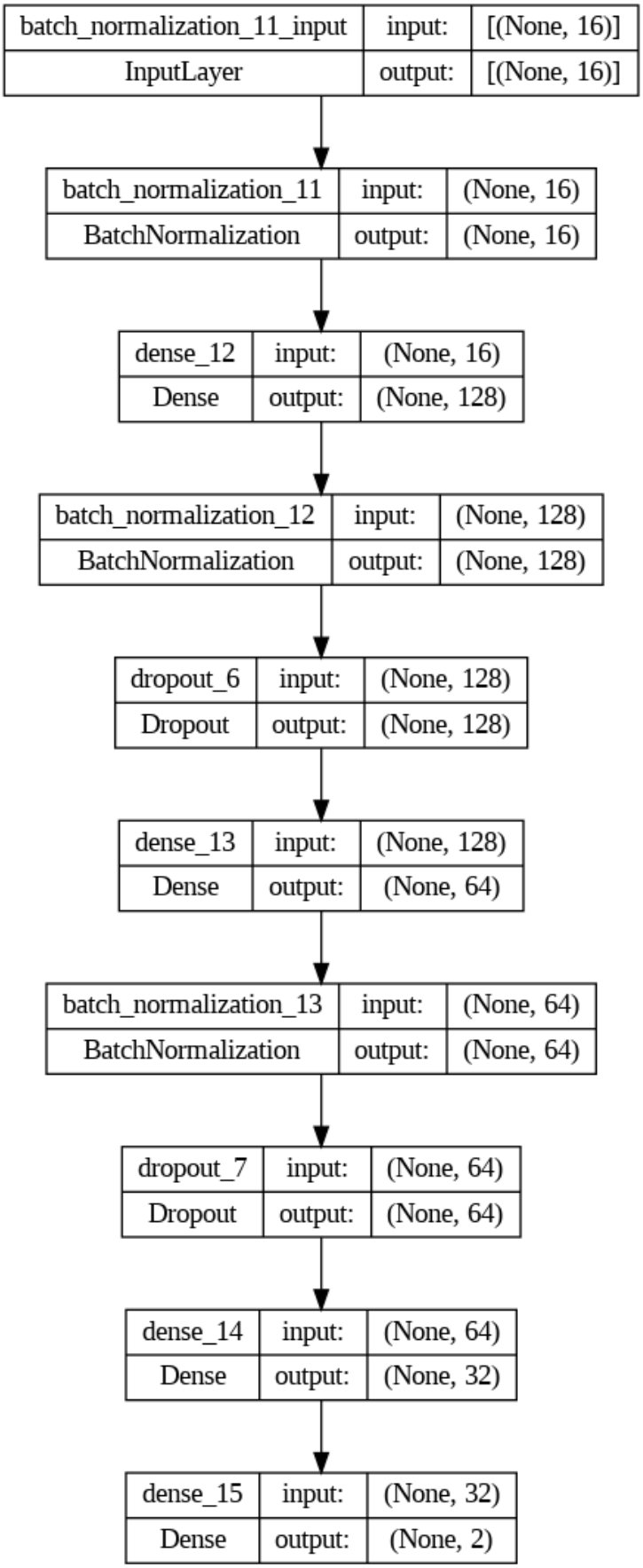
## 2. Chạy thử một số mô hình

### 2.1. Mô hình Neural Network(Mạng Nơ ron)

Xây dựng mô hình

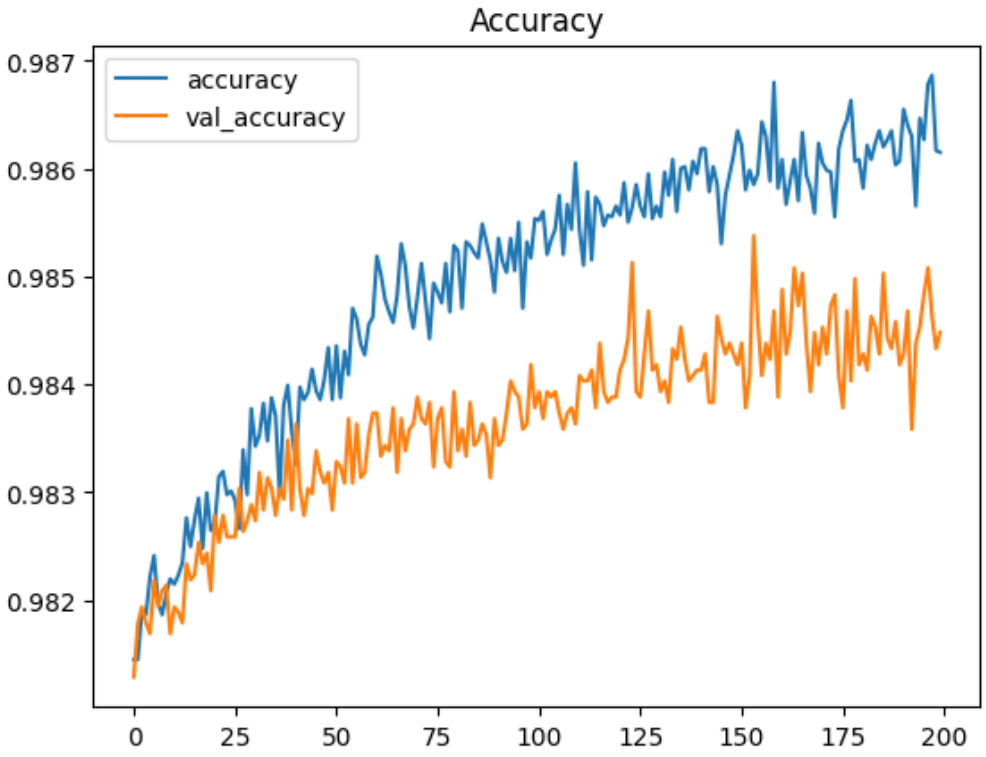
```
from tensorflow.keras import layers

model = keras.Sequential([
    layers.BatchNormalization(input_shape=input_shape),
    layers.Dense(128, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.4),
    layers.Dense(64, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.4),
    layers.Dense(32, activation='relu'),
    layers.Dense(2, activation='softmax'),
])
```

*Hình 9: Mô hình Neural Network*

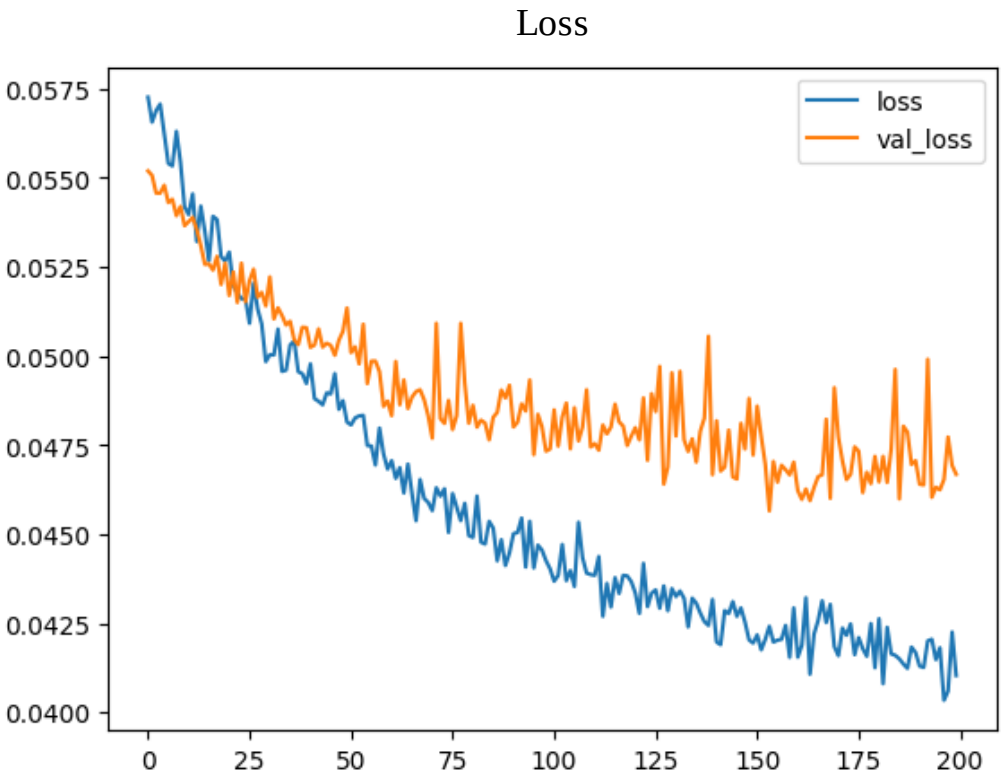



*Hình 10: Mô hình Neural Network*

Kết quả sau khi train model




*Hình 11: Accuracy của mô hình Neural Network*




*Hình 12: Loss của mô hinh Neural Network*

Kết quả thu được val_accuracy: 0.98

val_loss: 0.04

Đánh giá kết quả nhận dạng rất khả quan accuracy ở mức cao

## 2.2. Mô hình RandomForest

Chia tập dữ liệu thành 2 phần train test

```
[ ] X = pe_1.drop(['legitimate'],axis= 1 ).values
    y = pe_1['legitimate'].values
```

```
X
```

```
array([[3.32000000e+02, 6.13500000e+03, 9.00000000e+00, ...,
        1.60000000e+01, 1.60000000e+01, 7.22105073e+00],
       [3.32000000e+02, 8.17780000e+04, 9.00000000e+00, ...,
        2.00000000e+00, 1.80000000e+01, 6.56690933e+00],
       [3.32000000e+02, 3.50896000e+05, 9.00000000e+00, ...,
        2.00000000e+00, 1.80000000e+01, 7.60095678e+00],
       ...,
       [3.32000000e+02, 2.49838000e+05, 4.80000000e+01, ...,
        2.00000000e+00, 0.00000000e+00, 7.65970674e+00],
       [3.32000000e+02, 6.13472000e+05, 2.00000000e+00, ...,
        2.00000000e+00, 0.00000000e+00, 6.62239057e+00],
       [3.32000000e+02, 1.16734000e+05, 8.00000000e+00, ...,
        2.00000000e+00, 0.00000000e+00, 5.68446836e+00]])
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,random_state=42)
```

*Hình 13: Chuẩn bị data để huấn luyện mô hình Random Forest*

Khởi tạo mô hình

```
[ ] rf = RandomForestClassifier( n_estimators=200,
                                 random_state=42)
    rf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=200, random_state=42)
```

*Hình 14: Khởi tạo mô hình Random Forest*

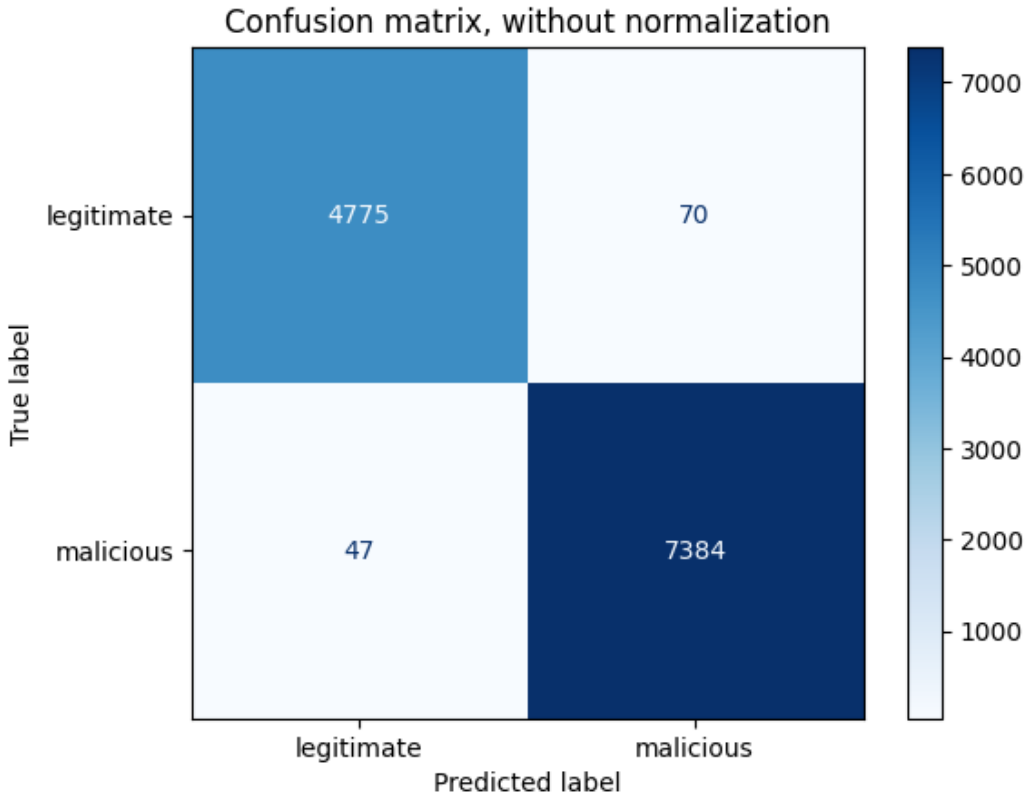
Kết quả sau khi huấn luyện

```
y_pred = rf.predict(X_test)
```

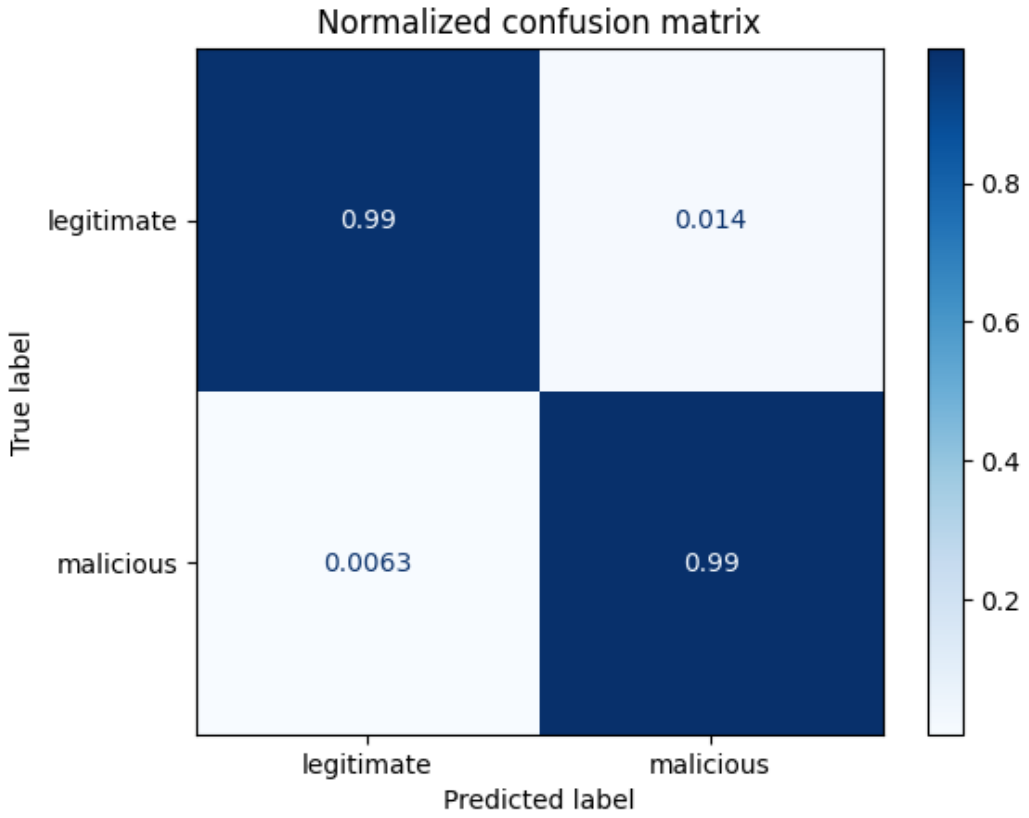
```
[ ] accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)
```

```
Accuracy: 0.9904692082111437
```

*Hình 15: Accuracy của mô hình Random Forest*




*Hình 16: Ma trận nhầm lẫn của mô hình Random Forest*

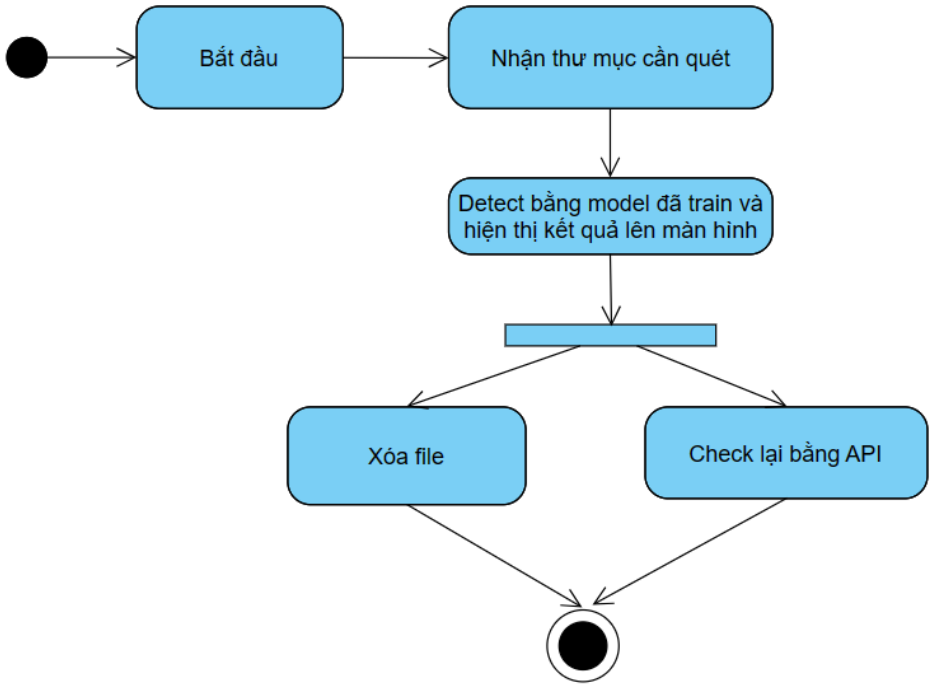



*Hình 17: Ma trận nhầm lẫn của mô hình Random Forest 2*

Đánh giá kết quả nhận dạng cao hơn mô hình Neural Network một chút (0.98 và 0.99) nhận dạng khá tốt trên tập test

## 3. Áp dụng model đã train để xấy dựng phần mềm

### 3.1. Sơ đồ khối




*Hình 18: Sơ đồ khối của phần mềm*

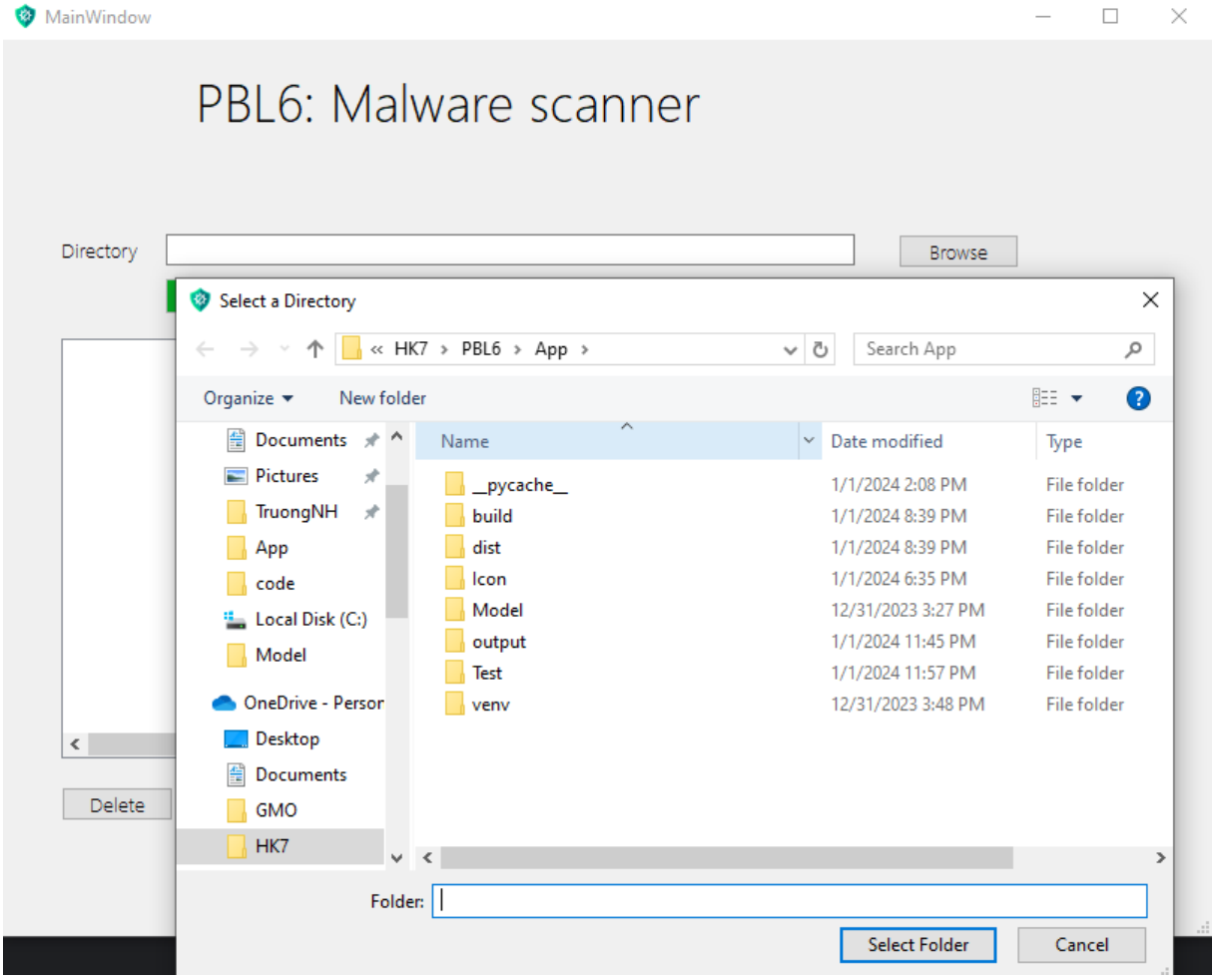
### 3.2. Công cụ sử dụng

- Ngôn ngữ lập trình Python
- Thư viện PyQt6, sklearn, pefile, os,…
- API sử dụng: https://www.virustotal.com/api

## 3.3. Mô tả các chức năng

### 3.3.1. Quét tất cả các file trong thư mục

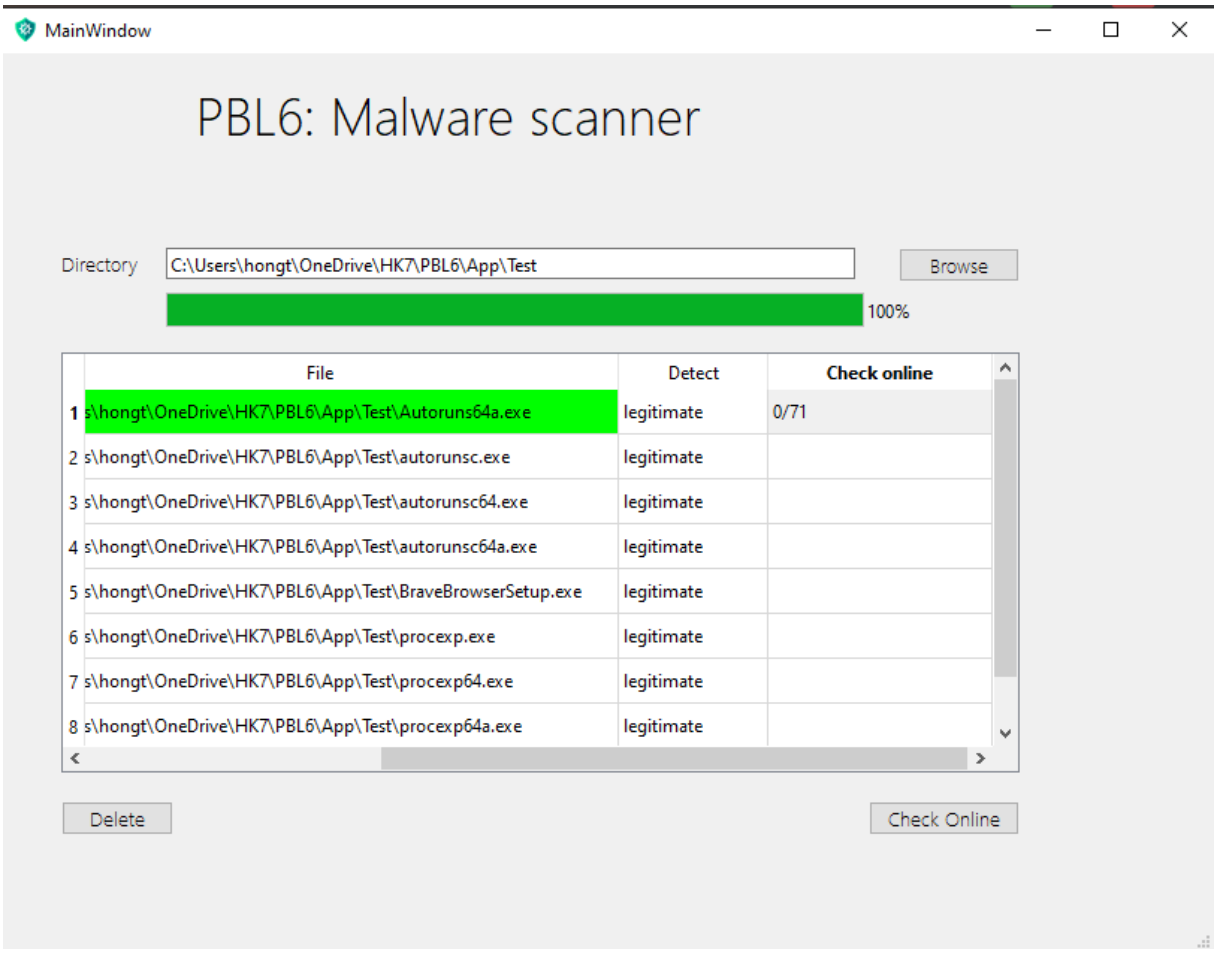
Sau khi nhấn nút 'Browse' phần mềm sẽ hiện thị hộp thoại lựa chọn thư mục. Sau khi chọn thư mục và nhấn Select Folder phần mềm sẽ bắt đầu quét tất cả các file có trong thư mục đó rồi hiện thị kết quả lên màn hình.




*Hình 19: Chức năng chọn thư mục cần quét*

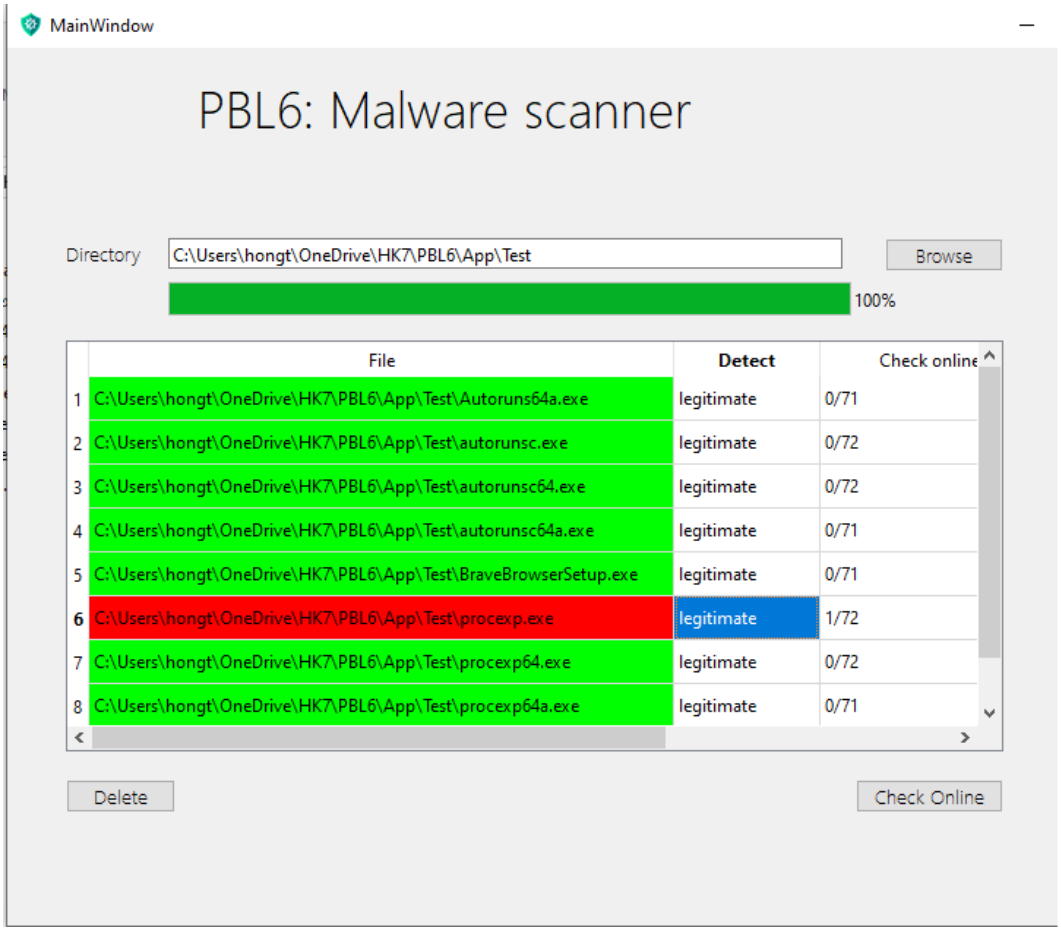
### 3.3.2. Kiểm tra online bằng API của Virus Total

Chọn 1 file và nhấn Check Online để phầm mềm submit file lên web virustotal và hiện thị kết quả ở cột check online.




*Hình 20: Chức năng kiểm tra bằng Api*

### 3.4. Kết quả




*Hình 21: Kết quả của phần mềm*

# V. ĐÁNH GIÁ TỔNG KẾT

## 1. Kết quả đạt được

- Hiểu về các loại mã độc và các phương pháp phát hiện mã độc
- Hiểu cách cách hoạt động và huấn luyện một mô hình học máy
- Xây dựng một phần mềm đơn giản áp dụng model đã huấn luyện và có các chức năng phụ bên ngoài

## 2. Ưu nhược điểm

- **Ưu điểm:**
  - Phần mềm chạy ổn định
  - Mô hình có độ chính xác cao
  - Giao diện đơn giản dễ sử dụng
- **Nhược điểm:**
  - Còn ít chức năng phụ trợ

- Chỉ quét được những file thực thi (.exe, .dll)
- Khi cập nhật dữ liệu cần phải huấn luyện lại từ đầu không thể áp dụng học tăng cường

**3. Phương hướng phát triển**

- Cập nhật mô hình thường xuyên hơn để tăng khả năng dự đoán của model
- Bổ sung các tính năng giám sát mới để tăng khả năng dự đoán của phần mềm
- Bổ sung thêm phân tích động bằng học máy
- Cập nhật giao diện đẹp mắt hơn

# VI. TÀI LIỆU THAM KHẢO

**[1] Malware Analysic(PDF),** tác giả Trần Phương Nam( Thỉnh giảng khoa công nghệ thông tin Đại học Bách Khoa-Đại học Đà Nẵng).

**[2] Phát hiện mã độc dựa vào học máy và thông tin PE Header**, tác giả Trần Ngọc Anh (Bộ tư lệnh 86), Võ Khương Lĩnh (Đại học Nguyễn Huệ)

[https://antoanthongtin.vn/giai-phap-khac/phat-hien-ma-doc-dua-vao-hoc-may-va-thong-tin-pe-header-phan-i-107476].

**[3] Qt for Python**

[https://doc.qt.io/qtforpython-6/].

**[4] 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking**
[https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles].