**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**Đồ án chuyên ngành**

**Đề tài:**

**A study on natural language processing-based method for Windows malware detection**

Giảng viên hướng dẫn: **ThS. Đỗ Thị Thu Hiền**

Lớp: **NT114.O11.ANTT**

Sinh viên thực hiện:

**Nguyễn Quang Huy**  **20520546**

**Bùi Đức Hoàng**  **20520514**

*TP. Hồ Chí Minh, ngày 25 tháng 12 năm 2023*

**NHẬN XÉT TỪ GIẢNG VIÊN**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………

# MỤC LỤC

[MỤC LỤC i](#_Toc156655269)

[LỜI CẢM ƠN iv](#_Toc156655270)

[DANH MỤC HÌNH VẼ v](#_Toc156655271)

[DANH SÁCH BẢNG vi](#_Toc156655272)

[TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH 1](#_Toc156655273)

[CHƯƠNG 1: TỔNG QUAN 2](#_Toc156655274)

[1.1 Giới thiệu 2](#_Toc156655275)

[1.2 Thách thức và phương pháp 2](#_Toc156655276)

[1.2.1 Thách thức 2](#_Toc156655277)

[1.2.2 Phương pháp 3](#_Toc156655278)

[1.3 Mục tiêu, nội dung cụ thể 3](#_Toc156655279)

[1.3.1 Mục tiêu 3](#_Toc156655280)

[1.3.2 Nội dung cụ thể 3](#_Toc156655281)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 5](#_Toc156655282)

[2.1 State of the art của các phương pháp malware detection 5](#_Toc156655283)

[2.2 Strings trong malware 6](#_Toc156655284)

[2.3 API Sequences trong malware 7](#_Toc156655285)

[2.4 Natural Processing Language (NLP) 7](#_Toc156655286)

[CHƯƠNG 3: PHƯƠNG PHÁP ĐỀ XUẤT 10](#_Toc156655287)

[3.1 Tổng quan hệ thống 10](#_Toc156655288)

[3.2 Chi tiết các thành phần 11](#_Toc156655289)

[3.2.1 Hybrid analysis stage 11](#_Toc156655290)

[3.2.2 Preprocessing stage 11](#_Toc156655291)

[3.2.2.1 String 12](#_Toc156655292)

[3.2.2.2 API call sequence 14](#_Toc156655293)

[3.2.2.3 Tokenizing, Encoding và Embedding 16](#_Toc156655294)

[3.2.3 Classification stage 17](#_Toc156655295)

[3.2.3.1 Các mô hình học sâu 17](#_Toc156655296)

[3.2.3.2 Concatenate layer 18](#_Toc156655297)

[CHƯƠNG 4: HIỆN THỰC VÀ THỰC NGHIỆM 20](#_Toc156655298)

[4.1 Hiện thực 20](#_Toc156655299)

[4.2 Chi tiết các thành phần 20](#_Toc156655300)

[4.2.1 Phân tích file từ Cuckoo sandbox 20](#_Toc156655301)

[4.2.2 Trích xuất dữ liệu từ report của Cuckoo sandbox 21](#_Toc156655302)

[4.2.3 Cleaning 22](#_Toc156655303)

[4.2.3.1 String 22](#_Toc156655304)

[4.2.3.2 API call 23](#_Toc156655305)

[4.2.4 Tokenization 23](#_Toc156655306)

[4.2.5 Embedding và các thông số mô hình 23](#_Toc156655307)

[4.3 Thực nghiệm 25](#_Toc156655308)

[4.3.1 Giới thiệu về Dataset 25](#_Toc156655309)

[4.3.2 Thông số đánh giá (metric) 26](#_Toc156655310)

[4.3.3 Các kịch bản 27](#_Toc156655311)

[4.3.4 Kết quả thực nghiệm 27](#_Toc156655312)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 29](#_Toc156655313)

[5.1 Kết luận 29](#_Toc156655314)

[5.1.1 Kết quả đạt được 29](#_Toc156655315)

[5.1.2 Hạn chế 30](#_Toc156655316)

[5.2 Hướng phát triển 30](#_Toc156655317)

[TÀI LIỆU THAM KHẢO 31](#_Toc156655318)

# LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến cô Đỗ Thị Thu Hiền vì những hướng dẫn tận tâm và sự hỗ trợ vô cùng quý giá mà cô đã dành cho chúng em trong quá trình thực hiện Đồ án chuyên ngành.

Sự chỉ dẫn của cô không chỉ giúp chúng em nắm vững kiến thức chuyên môn mà còn giúp chúng em phát triển kỹ năng nghiên cứu và phân tích vấn đề. Cô đã dành thời gian để lắng nghe, đồng hành cùng chúng em và đảm bảo rằng chúng em hiểu rõ mục tiêu và yêu cầu của dự án.

Sự kiên nhẫn và tận tâm của cô đã truyền cảm hứng cho chúng em để vượt qua những thách thức và khó khăn trong quá trình nghiên cứu. Nhờ sự hướng dẫn của cô, chúng em đã có cơ hội thực hành và áp dụng những kiến thức lý thuyết vào thực tế.

Không chỉ là một người hướng dẫn, cô còn là một người đồng đội đáng tin cậy. Cô đã luôn sẵn sàng lắng nghe ý kiến của chúng em, đồng thời đưa ra những gợi ý xây dựng và giúp chúng em hoàn thiện dự án một cách tốt nhất.

Chúng em biết ơn tình cảm và sự quan tâm mà cô đã dành cho chúng em không chỉ trong quá trình hướng dẫn, mà còn ngoài đời hàng ngày. Cô đã truyền động lực và niềm tin cho chúng em để vươn lên và vượt qua mọi khó khăn.

Em xin chân thành cảm ơn.

*TP. Hồ Chí Minh, ngày ... tháng 01 năm 2024*

# DANH MỤC HÌNH VẼ

[Hình 3‑1. Mô hình tổng quan 10](#_Toc156561308)

[Hình 3‑2. Phương pháp cleaning string đề xuất. 12](#_Toc156561309)

[Hình 4‑1. Mô hình chính chi tiết 20](#_Toc156561310)

[Hình 4‑2. Chạy API server của Cuckoo. 21](#_Toc156561311)

[Hình 4‑3. Hàm trích xuất dữ liệu từ report của Cuckoo Sandbox 22](#_Toc156561312)

[Hình 4‑4. Top frequent word của các file benign (a) và malware (b). 22](#_Toc156561313)

# DANH SÁCH BẢNG

[Bảng 1. Thông số kết quả của từng kịch bản 28](#_Toc156654554)

# TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Nghiên cứu này tập trung vào việc áp dụng Natural Language Processing (NLP) trong phát hiện mã độc, nhằm giải quyết vấn đề phức tạp của việc phân loại và phát hiện các mẫu malware đa dạng. Phương pháp đề xuất kết hợp cả hai thuộc tính tĩnh và động của mã độc, sử dụng NLP để phân tích và hiểu các đặc điểm ngôn ngữ trong mã độc.

Trong phần thuộc tính tĩnh, NLP được áp dụng để phân tích các thành phần văn bản của mã độc, như chuỗi ký tự, hằng số và tên biến. Điều này giúp xác định các đặc điểm ngôn ngữ độc hại và tìm ra mẫu mã độc tiềm ẩn.

Trong phần thuộc tính động, NLP được sử dụng để phân tích ngôn ngữ tự nhiên liên quan đến hoạt động của mã độc trong môi trường thực thi. Điều này bao gồm việc ghi nhật ký các hoạt động, giám sát thay đổi trên hệ thống và phân tích các thông điệp mạng. Sự kết hợp của các thông tin này từ NLP giúp phát hiện hành vi độc hại và xác định các mẫu mã độc mới.

Kết quả nghiên cứu cho thấy phương pháp áp dụng NLP trong phát hiện mã độc đạt được hiệu suất cao, với khả năng phân loại chính xác các mẫu độc hại. Việc kết hợp cả thuộc tính tĩnh và động cùng với sự sử dụng NLP cung cấp một cách tiếp cận toàn diện và hiệu quả hơn trong phát hiện và chống lại các mẫu malware đa dạng và biến thể.

Sự phát triển và ứng dụng của phương pháp này có tiềm năng rất lớn trong lĩnh vực bảo mật thông tin, góp phần cung cấp những giải pháp hiệu quả và tiên tiến hơn trong việc đối phó với sự gia tăng và phát triển ngày càng phức tạp của malware.

# TỔNG QUAN

Trong chương này, nhóm tác giả trình bày sơ lược về phương pháp áp dụng NLP trong việc phát hiện các loại mã độc (malware detection). Nhóm tác giả đặt mục tiêu thực hiện trên nhiều mô hình nhằm tìm ra mô hình tối ưu nhất.

## Giới thiệu

Trong thời gian gần đây, phương pháp xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đã được áp dụng trong lĩnh vực phát hiện mã độc (malware detection) để tăng cường khả năng phát hiện và phân loại các mẫu mã độc. NLP trong malware detection sử dụng các kỹ thuật và công cụ NLP để phân tích và hiểu các đặc điểm của mã độc dựa trên ngôn ngữ tự nhiên. Tuy nhiên, câu hỏi được đặt ra ở đây là liệu mô hình học máy nào khi kết hợp NLP sẽ cho ra kết quả tối ưu nhất trong việc phát hiện malware?

Trong đồ án này, nhóm tác giả sẽ tiến hành thực hiện kết hợp phương pháp NLP với các mô hình học máy phổ biến khác nhau, qua đó so sánh với các bài báo khác, cũng như để tìm ra mô hình tối ưu nhất.

## Thách thức và phương pháp

### Thách thức

Related works: [1] Han, Weijie & XUE, Jingfeng & Wang, Yong & Huang, Lu & Kong, Zixiao & Mao, Limin. (2019). MalDAE: Detecting and Explaining Malware based on Correlation and Fusion of Static and Dynamic Characteristics. Computers & Security. 83. 10.1016/j.cose.2019.02.007.

Nghiên cứu này đề xuất một phương pháp mới gọi là MalDAE (Malware Detection and Explanation), sử dụng kỹ thuật học sâu tự động mã hóa (Autoencoder) để tìm ra các đặc điểm quan trọng của phần mềm độc hại từ cả hai khía cạnh tĩnh và động.

Trong phương pháp này, các đặc điểm tĩnh của phần mềm (như tệp tin, mã độc) và các đặc điểm động (như hành vi chương trình, gọi API) được biểu diễn dưới dạng các chuỗi ngữ nghĩa. Các chuỗi ngữ nghĩa này được chuyển đổi thành các chuỗi ngữ nghĩa tĩnh và động, và sau đó được kết hợp thành một chuỗi hỗn hợp (Hybrid).

Tiếp theo, mô hình MalDAE so sánh các chuỗi ngữ nghĩa tĩnh và động và tính toán trọng số cho các chuỗi con tương ứng trong hai chuỗi này. Các trọng số này thể hiện tầm quan trọng của các chuỗi con API trong chuỗi ngữ nghĩa động và tĩnh.

Thách thức được đặt ra trong bài báo kể trên là đối với việc kết hợp thành một chuỗi hỗn hợp (Hybrid), mô hình yêu cầu một chuỗi ngữ nghĩa phải dài hơn chuỗi còn lại để xác định chuỗi cơ sở. Điều này có nghĩa rằng mô hình không thể đảo ngược vai trò của các chuỗi để tạo ra chuỗi hỗn hợp.

### Phương pháp

* Natural Processing Language: Trong đồ án này, nhóm tác giả sẽ triển khai phương pháp NLP được kết hợp với các mô hình học máy
* Purify: Trong đồ án này, nhóm tác giả đề xuất thuật toán purify, nhằm làm sạch dữ liệu đầu vào và để tối ưu hiệu suất xử lý.

## Mục tiêu, nội dung cụ thể

### Mục tiêu

Tìm hiểu và nghiên cứu các mô hình học máy phổ biến với từng loại đầu vào là strings hoặc API Sequences, sau đó áp dụng thuật toán Purify cũng như kết hợp mô hình để đánh giá độ tối ưu và hiệu quả.

### Nội dung cụ thể

**Nội dung 1:**  Khảo sát các mô hình đơn lẻ đối với Strings

Phương pháp thực hiện:

* Thực hiện chạy các mô hình training với đầu vào là Strings.
* Xác định môi trường, lựa chọn được mô hình phù hợp cho đề tài này.

Dự kiến kết quả:

Nắm vững kiến thức cơ bản của phương pháp, tìm ra được các mô hình hiệu quả nhất đối với đầu vào là Strings

**Nội dung 2**: Khảo sát các mô hình đơn lẻ đối với API Sequence

Phương pháp thực hiện:

* Thực hiện chạy các mô hình training với đầu vào là API Sequences
* Tham khảo tài liệu...

Dự kiến kết quả: Nắm vững kiến thức cơ bản của phương pháp, tìm ra được các mô hình hiệu quả nhất đối với đầu vào là API Sequences

**Nội dung 3**: Hiện thực với thuật toán Purify

Phương pháp thực hiện: <tìm hiểu thuật toán, cài đặt môi trường, công cụ...>

**Nội dung 4:** Thực nghiệm kết quả...

Phương pháp thực hiện: Xây dựng kịch bản, lựa chọn thông số đánh giá. Thực nghiệm và đánh giá kết quả dựa trên số liệu, thông tin ghi nhận được.

Dự kiến kết quả: ....

# CƠ SỞ LÝ THUYẾT

## State of the art của các phương pháp malware detection

Trong lĩnh vực malware detection, đã có rất nhiều các công trình nghiên cứu được thực hiện bởi các nhóm sinh viên, các nhóm nhà khoa học, nhà nghiên cứu, những kết quả thu được nhiều sự khả quan. Các ví dụ có thể được kể ra như trong bài báo MalDetConv, ta được một framework dynamic malware detection, hay trong bài báo MalDAE, ta có một framework hybrid, tuy nhiên có một hạn chế là phụ thuộc vào việc định hình được chuỗi tĩnh và động.

Các công cụ phân tích file PE trên Linux cũng đa dạng về các phương pháp phân tích một file cụ thể nhưng nhìn chung, được chia thành 2 thể loại chính:

* Các công cụ phân tích tĩnh: các công cụ này cho phép phân tích một file dựa trên mã nguồn hoặc dịch ngược mà không cần thực thi chúng. Các thông tin mà các công cụ này mang lại thường là assembly code (objdump command của Linux), readable string (strings command của Linux), PE headers (PEDA - Python Exploit Development Assistance),… Do không cần thực thi nên các công cụ này có khả năng xử lí và phân tích trong thời gian thực. Tuy nhiên, đối với các file đã sử dụng các phương pháp rối mã, mã hóa hay packed, các công cụ này sẽ không đem lại hiệu quả cao do không trích xuất được các thông tin đã bị ẩn giấu.
* Các công cụ phân tích động: Trái ngược với các công cụ phân tích tĩnh, các công cụ phân tích động cho phép thực thi một file trên một môi trường có kiểm soát và quan sát hành vi của nó. Các phương pháp phân tích động thường được sử dụng là:
  + Sandboxing: Tạo ra một môi trường ảo độc lập cho phép thực thi các file PE an toàn, đồng thời theo dõi hành vi của việc thực thi file, bao gồm lời gọi hệ thống (system call), các hoạt động mạng (network activity), sự thay đổi của file hệ thống (file system changes), các file DLL được load, và sự chỉnh sửa các registry (registry modifications). Một số công cụ sandbox cho file PE nổi tiếng có thể kể đến như Cuckoo Sandbox, FireEye, ….
  + Debugging: Cho phép thực thi một file bằng cách lần lượt chạy qua từng dòng code, thiết lập breakpoint, theo dõi các thanh ghi và memory trong quá trình thực thi. Một số debugger như OllyDbg, x64dbg, IDA Pro, …

Phương pháp phân tích động cho phép xác định được hành vi đáng ngờ, độc hại của các malware, cho dù đã thực hiện các biện pháp ẩn giấu. Tuy nhiên, phương pháp này yêu cầu thời gian và tài nguyên để thực thi và phân tích file, do đó không phù hợp với yêu cầu phát hiện thời gian thực. Đồng thời, có một số loại malware có cơ chế phát hiện hành vi phân tích động (anti-debug, anti-sandbox,…), điều này dẫn đến các malware sẽ không thực thi các hành động của chúng.

Để có thể hạn chế được khuyết điểm của cả hai phương pháp, nhóm đề xuất sử dụng phương pháp phân tích Hybrid. Phương pháp này cho phép thực hiện cả hai phương pháp phân tích động và tĩnh cùng lúc đối với file PE. Kết quả thu thập được sẽ được kết hợp lại với nhau để có thể đánh giá file PE hiệu quả nhất bằng cách tận dụng ưu điểm của 2 phương pháp, đồng thời khắc phục được nhược điểm của cả hai.

## Strings trong malware

Trong ngữ cảnh của phần mềm độc hại (malware), thuật ngữ "strings" được sử dụng để chỉ các chuỗi ký tự có ý nghĩa trong tập tin thực thi của malware.

Khi một tập tin thực thi của malware được phân tích, các chuỗi ký tự được tìm thấy bên trong có thể cung cấp thông tin quan trọng về tính chất và mục đích của chương trình độc hại. Những chuỗi này có thể bao gồm các đoạn mã, địa chỉ IP, tên miền, tên tệp tin, chuỗi lệnh, thông điệp hoặc bất kỳ thông tin nào khác mà malware sử dụng để thực hiện các hoạt động hay giao tiếp với máy chủ điều khiển hoặc hệ thống mục tiêu.

Việc phân tích chuỗi (strings) trong malware có thể giúp các nhà nghiên cứu bảo mật và chuyên gia phân tích malware nhận biết các chỉ mục điểm khởi đầu hoặc các chuỗi mã độc hại trong tập tin thực thi. Điều này có thể cung cấp thông tin quan trọng để hiểu cách hoạt động của malware, xác định các tính năng độc hại và phát hiện các mối đe dọa tiềm ẩn.

## API Sequences trong malware

Trong ngữ cảnh của phần mềm độc hại (malware), thuật ngữ "API sequence" (chuỗi API) được sử dụng để chỉ sự sử dụng liên tiếp của các gọi hàm giao diện ứng dụng (Application Programming Interface - API) trong mã độc hại.

API sequence đề cập đến một chuỗi các gọi hàm API liên tiếp mà malware sử dụng để thực hiện các hoạt động độc hại hoặc giao tiếp với hệ thống mục tiêu. Các API có thể được sử dụng để thực hiện các chức năng như ghi vào tập tin, đọc thông tin từ hệ thống, tạo tiến trình mới, gửi dữ liệu qua mạng, và nhiều hành vi khác.

Việc phân tích API sequence trong malware có thể giúp nhà nghiên cứu bảo mật và chuyên gia phân tích malware hiểu cách hoạt động của mã độc hại và xác định mục tiêu của nó. Bằng cách phân tích chuỗi API, các nhà nghiên cứu có thể tìm ra các hành vi độc hại, xác định các chức năng chính của malware và xây dựng các phương pháp phát hiện và ngăn chặn.

## Natural Processing Language (NLP)

Tất cả các phương pháp xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) trong malware detection nhắm mục tiêu là tận dụng thông tin ngôn ngữ tự nhiên trong mã độc để tăng cường khả năng phát hiện và phân loại các mẫu độc hại. Các phương pháp NLP này sử dụng các kỹ thuật và công cụ NLP để phân tích và hiểu các đặc điểm của mã độc dựa trên ngôn ngữ tự nhiên. Các loại phân tích đó gồm:

* Phân tích cú pháp (Syntax Analysis): Phương pháp này tập trung vào việc phân tích cấu trúc và quy tắc ngữ pháp của mã độc. Cú pháp được sử dụng để xác định các thành phần trong mã, bao gồm các câu lệnh, biểu thức, hàm, lớp, và các quy tắc cú pháp khác. Các công cụ như lexer và parser được sử dụng để phân tích cú pháp mã độc và xác định các đặc điểm không phù hợp hoặc đáng ngờ.
* Phân tích từ vựng (Lexical Analysis): Phương pháp này tập trung vào các từ và thuật ngữ trong mã độc. Các từ khóa đặc trưng, biểu thức chính quy, và từ điển từ khóa được sử dụng để tìm kiếm các từ và cụm từ có liên quan đến mã độc. Việc phân tích từ vựng giúp xác định các đặc điểm ngôn ngữ độc hại, như tên biến, hàm, lớp, hoặc các chuỗi ký tự đáng ngờ.
* Phân tích ngữ nghĩa (Semantic Analysis): Phương pháp này tập trung vào việc hiểu ý nghĩa của các câu và văn bản trong mã độc. Các kỹ thuật semantic parsing và semantic role labeling được sử dụng để phân tích và rút trích thông tin cần thiết từ các câu và văn bản liên quan đến mã độc. Điều này giúp xác định các hành vi, mục tiêu, và tác động tiềm năng của mã độc.
* Phân tích ngữ cảnh (Contextual Analysis): Phương pháp này đánh giá mã độc dựa trên ngữ cảnh mà nó xuất hiện. Các thuật toán học máy và mô hình ngữ cảnh được sử dụng để phân tích và xác định xem một đoạn mã có khả năng là độc hại hay không dựa trên các thông tin xung quanh nó. Ví dụ, việc phân tích ngữ cảnh có thể xác định liệu một hàm cụ thể có được gọi từ một đoạn mã độc khác hay không.
* Phân tích dựa trên mẫu (Pattern-based Analysis): Phương pháp này sử dụng các mẫu đã biết hoặc các quy tắc để phát hiện và phân loại các mẫu mã độc. Các mẫu đặc trưng và quy tắc mô tả các đặc điểm và hành vi của mã độc được sử dụng để so khớp và tìm kiếm các mẫu tương tự trong mã đích. Các kỹ thuật như regular expressions (biểu thức chính quy) và rule-based matching được sử dụng để áp dụng các mẫu này và phát hiện các mã độc. Tuy nhiên, việc áp dụng NLP trong lĩnh vực malware detection cũng đối mặt với một số thách thức. Mã độc ngày càng trở nên phức tạp và tiên tiến hơn, với sự phát triển của các kỹ thuật che dấu (obfuscation) và tấn công zero-day. Do đó, cần phải liên tục cập nhật và phát triển các kỹ thuật NLP mới nhằm đối phó với các mẫu mã độc mới.

Hơn nữa, việc sử dụng NLP trong malware detection cũng đòi hỏi kiến thức chuyên sâu về an ninh mạng và NLP. Để xử lý mã độc hiệu quả, các chuyên gia phải nắm vững kiến thức về ngôn ngữ tự nhiên, các kỹ thuật xử lý ngôn ngữ tự nhiên, các thuật toán học máy, và các công cụ NLP.

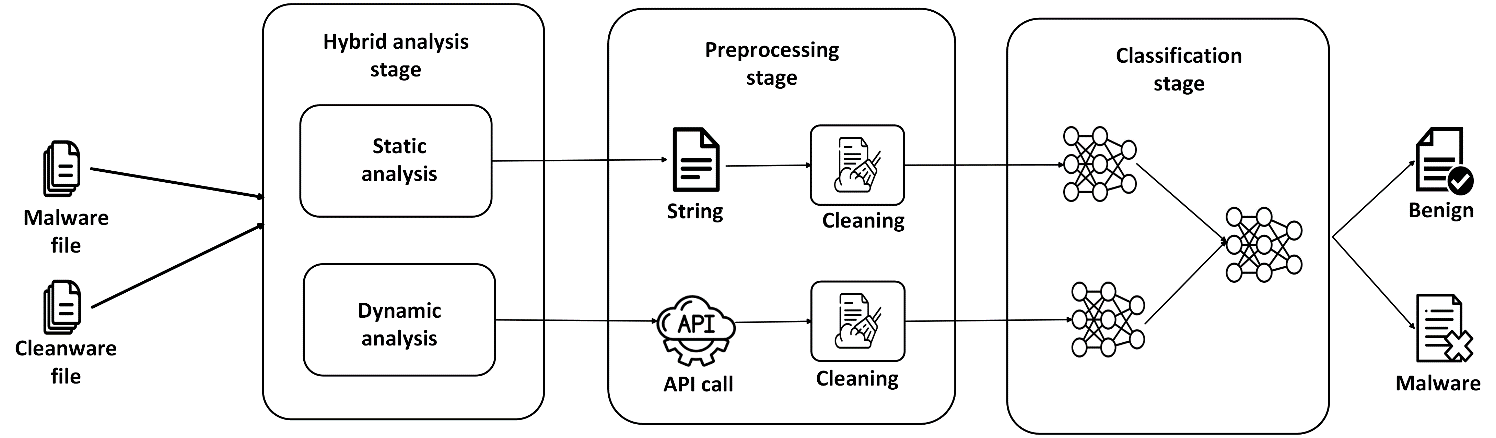
Một số ứng dụng cụ thể của NLP trong malware detection bao gồm:

* Phát hiện mã độc dựa trên phân tích ngữ nghĩa của các hàm, lớp, hoặc các đoạn mã.
* Phát hiện mã độc dựa trên phân tích từ vựng, tìm kiếm các từ khóa đặc trưng hoặc các biểu thức chính quy trong mã độc.
* Phát hiện mã độc dựa trên phân tích cú pháp, xác định các câu lệnh hoặc cấu trúc ngôn ngữ độc hại.
* Phân loại và gom nhóm các mẫu mã độc dựa trên các đặc điểm ngôn ngữ, hành vi, hoặc mục tiêu của chúng.
* Xác định các quan hệ và mối liên hệ giữa các thành phần mã độc, như các hàm gọi nhau hoặc tương tác với nhau.

Tổng quan, NLP trong malware detection đóng vai trò quan trọng trong việc nâng cao khả năng phát hiện và phân loại mã độc. Bằng cách áp dụng các phương pháp và công cụ NLP, chúng ta có thể phân tích và hiểu mã độc dựa trên thông tin ngôn ngữ tự nhiên, từ đó nâng cao hiệu suất và độ chính xác của quá trình phát hiện malware.

# PHƯƠNG PHÁP ĐỀ XUẤT

## Tổng quan hệ thống



Hình 3‑1. Mô hình tổng quan

Tổng quan, mô hình nhóm đề xuất gồm 3 giai đoạn chính:

* Hybrid analysis stage: Giai đoạn này giúp phân tích dữ liệu và hành vi của các file PE trong một môi trường riêng độc lập, an toàn. Các file PE lần lượt được trải qua các bước phân tích tĩnh và phân tích động và ghi nhận lại các dữ liệu string của file, API sequence, sau đó đưa đến giai đoạn tiếp theo.
* Preprocessing stage: Sau khi có được dữ liệu từ giai đoạn trước, giai đoạn này trích xuất các đặc trưng trong dữ liệu, loại bỏ các thành phần gây nhiễu, dư thừa để tạo thành bộ dữ liệu phù hợp cho việc huấn luyện mô hình học máy. Sau đó, xử lí chúng với các phương pháp NLP sao cho phù hợp với đầu vào giai đoạn sau.
* Classification stage: Giai đoạn này tạo và huấn luyện một mô hình học máy sử dụng đến các mạng neural trong machine learning nhằm thực hiện phân loại trên các đặc trưng dữ liệu của file đã trích xuất, từ đó đưa ra xác xuất đánh giá xem file phân tích thuộc về nhóm malware hay benign.

Chi tiết các thành phần sẽ được trình bày ở phần sau.

## Chi tiết các thành phần

### Hybrid analysis stage

Để có thể phân tích được dữ liệu và hành vi của các file, cụ thể là các file Portable Executable (PE) hoạt động trên môi trường Windows, mà không gây ảnh hưởng đến tính an toàn của hệ thống, đồng thời không cho phép các malware có khả năng hoạt động và lây nhiễm đến các máy khác, ta cần phải có một môi trường cách biệt đối với hệ thống chính. Để giải quyết vấn đề trên, nhóm đề xuất sử dụng một máy tính riêng sử dụng hệ điều hành dạng Linux làm môi trường để phân tích. Các loại hệ điều hành Linux (ex: Ubuntu, Kali Linux, Fedora, …) là một môi trường lí tưởng để có thể thao tác với các file PE bởi tích chất nhẹ, dễ dàng tương tác bằng cách sử dụng câu lệnh và cho phép lưu trữ các file PE mà không cần lo đến khả năng kích hoạt hay lây lan tự động của các loại malware nguy hiểm. Đồng thời đây là một môi trường tương thích với nhiều bộ công cụ phân tích file mạnh mẽ với khả năng cài đặt, điều chỉnh dễ dàng cũng như khả năng tự động hóa quy trình phân tích.

Có rất nhiều công cụ hỗ trợ phân tích Windows malware như Joe Sandbox, Anyrun, VMRay, FireEye Sandbox, Hybrid Analysis, Sandboxie, …. Trong đó, công cụ được nhóm sử dụng là Cuckoo sandbox. Cuckoo Sandbox là một nền tảng mã nguồn mở dựa trên Python, có thể hoạt động trên Linux. Công cụ cho phép thực hiện phân tích nhiều thể loại file như dll, pdf, docx, php, zip, jar, …. Nó cung cấp một môi trường cách ly và kiểm soát có khả năng tùy chọn hệ điều hành thông qua các nền tảng ảo hóa như VMware, Virtualbox,... để chạy các các file và quan sát hành vi của chúng trong thời gian thực và lưu lại các kết quả phân tích. Công cụ còn cung cấp khả năng tự động hóa mạnh mẽ cho phép thực hiện phân tích số lượng lớn các file, đồng thời cung cấp report ở dạng json về cả 2 kết quả phân tích tĩnh và động, trong đó có dữ liệu string và API call theo thời gian thực.

### Preprocessing stage

Sau khi có kết quả phân tích gồm chuỗi string và API call sequences từ giai đoạn trước, giai đoạn này sẽ bắt đầu xử lí chúng.

Các dữ liệu thô về chuỗi string và API call sequences thường tồn tại với đặc điểm là rất lớn, một số chuỗi có thể có độ dài lên đến hàng ngàn, thậm chí hàng triệu do nhiều yếu tố như kích thước file, file thực hiện nhiều tác vụ, cơ chế gây nhiễu của các malware,… Trong đó các thông tin đặc trưng chính dùng để đánh giá đặc điểm và hành vi của file thường bị xen lẫn với những dữ liệu không cần thiết hoặc dữ liệu nhiễu, khiến cho việc xử lí bằng các phương pháp NLP sẽ gặp nhiều khó khăn cũng như việc huấn luyện mô hình học máy sẽ không hiệu quả. Do đó, nhóm đề xuất 2 phương pháp cleaning string và API call sequence như sau:

#### String

Dữ liệu string được phân tích từ file PE có độ dài rất lớn và tồn tại nhiều kí tự đặc biệt cũng như các “word” không có ý nghĩa cao cho việc phân loại. Do đó, phương pháp cleaning mới do nhóm đề xuất gồm hai giai đoạn sau:

A diagram of a diagram

Description automatically generated with medium confidence

Hình 3‑2. Phương pháp cleaning string đề xuất.

* **Giai đoạn 1**: Đối với mỗi string sequence, nhóm loại bỏ các word trùng lặp, tạo nên một sequence mới chỉ chứa các word duy nhất có độ dài <3 trong sequence cũ, đồng thời lowercase toàn bộ word và thay thế các kí tự đặc biệt (trừ dấu . , dấu\_ và dấu /) thành khoảng trắng . Các dấu . / \_thường có ý nghĩa trong ngữ cảnh của nội dung file, ví dụ như dấu . là kí hiệu trong tên file chỉ kiểu dữ liệu file được định nghĩa ie: \*.exe; \*.dll, \*.dat, …; hoặc kí hiệu phân cách cho địa chỉ IP, địa chỉ mail, tên miền,…; dấu / thường được sử dụng trong đường dẫn thư mục, địa chỉ trang web; dấu \_ thường được sử dụng nhiều trong việc đặt tên file. Do đó việc loại bỏ các kí tự này có khả năng ảnh hưởng đến các thông tin có ý nghĩa khi bị lượt bỏ.

A screen shot of a computer program

Description automatically generated

Hình 3‑3. Đoạn code thực hiện cleaning string

A close up of text

Description automatically generated

Hình 3‑4. Mẫu dataset khi đã qua giai đoạn 1

Mặc dù giai đoạn 1 đã loại bỏ các dữ liệu không cần thiết trong việc phân loại, tuy nhiên số lượng word trong mỗi sequence vẫn còn quá lớn, đồng thời tồn tại nhiều word không đặc trưng nhưng lại có tần suất rất lớn trong bộ dataset. Nhóm đề xuất phương pháp giai đoạn 2 như sau:

* **Giai đoạn 2:**

1. Trích xuất top n word có tần suất cao nhất từ mỗi bộ dataset benign và malware thành 2 bộ vocab benign và malware.
2. Loại bỏ các word có tần suất cao hơn p% của mỗi bộ vocab benign và malware.
3. Lần lượt thêm vào vocab chung các word có tần suất từ cao đến thấp từ 2 bộ vocab benign và malware, sao cho các word của 2 vocab xen lẫn nhau và không trùng lặp. Bộ vocab để trích xuất đã được tạo.
4. Để tạo sequence mới, thay vì trích xuất các word trong sequence cũ có trong vocab, ta thực hiện ngược lại: trích xuất từ các word từ vocab có trong file sequence. Điều này giúp cho các word được trích xuất luôn là duy nhất và tuân theo 1 trình tự của vocab với giới hạn tối đa số word là 1000 word/sequence.

A computer screen shot of text

Description automatically generated

Hình 3‑5. Code trích xuất các unique-word để train.

#### API call sequence

Chuỗi API call sequence được trích xuất độ dài rất lớn do quá trình thực thi của file có thể truy xuất nhiều chuỗi API để thực hiện các tác vụ cần thiết. Bên cạnh đó, một số malware có cơ chế gây nhiễu bằng cách gọi các API không quan trọng với số lượng rất lớn. Các dấu hiệu dễ nhận ra là các API sequence có thể tồn tại cùng một API được gọi liên tục, hoặc tồn tại các chuỗi API sub-sequence có tính chất cấp số cộng – nhiều API khác nhau lần lượt được gọi liên tục. Để loại bỏ những API sub-sequence gây nhiễu trên, đồng thời rút gọn chuỗi API sequence nhằm giữ lại những API quan trọng, nhóm đề xuất phương pháp cleaning gồm 2 giai đoạn sau:

Giai đoạn 1: Để loại bỏ các chuỗi noise sub-sequence, nhóm sử dụng phương pháp sau:

1. Loại bỏ sub-sequence có API được gọi liên tiếp: Với mỗi API sequences, tạo ra một sequence mới bằng cách duyệt qua từng API trên sequence cũ, nếu API sau trùng lặp với API trước sẽ bị loại bỏ cho đến khi gặp API khác.
2. Loại bỏ sub-sequence có chuỗi API sub-sequence có tính chất cấp số cộng:
   * B1: Lập 1 positions dictionary gồm tất cả các API có trong sequence, mỗi API được set là 0 (pos),
   * B2: Duyệt qua API trong sequence theo tuần tự, lấy pos của api hiện tại,
   * B3: Nếu pos hiện tại – pos trong dictionary > distance hoặc pos trong dictionary = 0 , và trong 3 API cuối cùng trong sequence mới không tồn tại API đang xét=> Cập nhật api vào chuỗi sequence mới, cập nhật pos vào dictionary. Ngược lại, bỏ qua api đó và duyệt tiếp.

Giai đoạn 2: Để tiếp tục giảm kích thước của mỗi API sequence, nhóm chỉ giữ lại các API trong sequence thỏa mãn điều kiện sau:

* **count(api)**: số lần xuất hiện của api trong sequence
* **lensequence:** độ dài chuỗi sequence
* **napi**: số loại api trong sequence
* **ratio**: tỉ số threshold cần thiết lập

Điều kiện trên nhằm loại bỏ các API có tần suất xuất hiện rất cao so với độ xuất hiện trung bình của một API nhân với một tỉ trọng được điều chỉnh. Từ đó ta có thể giữ lại các API có tần suất thấp hơn và giảm đi số lượng features mà mô hình cần phải học.

#### Tokenizing, Encoding và Embedding

Do mô hình học máy chỉ “hiểu” được dữ liệu ở dạng số, việc đưa dữ liệu text vào mô hình cần phải có một phương pháp xử lí riêng. Do đó, các kĩ thuật NLP là giải pháp để thực hiện công việc trên. Các giải pháp NLP đặc điểm chung gồm 3 giai đoạn:

* Tokenizing: Các dữ liệu text ở dạng chuỗi sau khi được cleaning sẽ được tách thành các mảng gồm những “word”, dựa vào các kí tự phân cách (thường là space), hoặc “char” tùy theo
* Encoding: Các “word” sau khi phân tách sẽ được thay thế bằng dữ liệu số đại diện cho chúng, khi này các dữ liệu dạng chuỗi sẽ được thay thế thành các vector số. Có nhiều phương pháp để thực hiện việc này như Index-Based, One-Hot, Bag of Words (BOW), Term Frequency — Inverse Document Frequency (TF-IDF),… Khi đó, dữ liệu số thay thế lúc này đã có thể đưa vào mô hình học máy. Tuy nhiên, dữ liệu Encoding thường có độ dài lớn, cũng như các số thay thế cho word sẽ không thể hiện được ý nghĩa về vị trí tương đối của word đó trong chuỗi. Do đó, giai đoạn tiếp theo sẽ giải quyết vấn đề trên.
* Embedding: Embedding cho phép “nén” vector thành vector mới có số chiều nhỏ hơn và dày đặc hơn, đồng thời cho phép thể hiện được sự tương đồng ngữ nghĩa giữa các word. Việc này có thể thực hiện được dựa vào việc tự học và huấn luyện mô hình dựa trên các word được cung cấp, như Word2vec, Doc2vec, LSI, … hay sử dụng các mô hình pre-train từ trước, như GloVe, BERT, GPT… Khi này, dữ liệu đã có thể đưa vào mô hình học máy để bắt đầu huấn luyện.

Để triển khai các kĩ thuật NLP trên, nhóm sử dụng các thư viện có sẵn bao gồm Tokenizer và Embedding layer của Keras cho cả dữ liệu string và API call. Tokenizer của Keras giúp encode các dữ liệu dưới dạng text sang dạng int. Mỗi word được thể hiện bằng một số với số này là thứ hạng của tần số word có trong sequence, nếu thứ hạng càng cao thì word đó có tần suất xuất hiện càng lớn. Layer Embedding của Keras cho phép chuyển đổi các encoding vector thành các vector có độ dài cố định và kích thước xác định với trọng số các word được cập nhật bằng cách sử dụng continuous bag-of-words model.

### Classification stage

Ở giai đoạn phân loại, nhóm nghiên cứu xây dựng và huấn luyện nhiều mô hình học máy để tìm ra mô hình tốt nhất để phân loại và phát hiện malware trong bộ dữ liệu. Mục tiêu là xây dựng một mô hình có khả năng nhận biết và phân loại chính xác các mẫu độc hại.

Sự kết hợp của multi-modal trong giai đoạn này là một lựa chọn quan trọng. Lý do sử dụng multi-modal trong giai đoạn phân loại malware là vì thông tin trong malware có thể xuất hiện ở nhiều dạng dữ liệu khác nhau. Ví dụ, một tệp tin malware có thể chứa các chuỗi ký tự độc hại, các API Sequence đáng ngờ. Bằng cách kết hợp và xử lý đồng thời các dạng dữ liệu này, mô hình có thể nhận biết và phân loại malware một cách toàn diện hơn. Các thuật toán Deep learning được sử dụng trong mô hình này bao gồm:

#### Các mô hình học sâu

##### 1D CNN

1D convolutional neural là một loại mạng nơ-ron tích chập được sử dụng phổ biến để xử lý dữ liệu tuần tự. Trong khi các mạng nơ-ron tích chập truyền thống chủ yếu được thiết kế cho dữ liệu hình ảnh, 1D CNN được tùy chỉnh đặc biệt để phân tích dữ liệu tuần tự hoặc chuỗi thời gian, chẳng hạn như tín hiệu âm thanh, xử lý ngôn ngữ tự nhiên (NLP) và dữ liệu cảm biến. Một khối cơ bản thường bao gồm các Convolutional Layer, thực hiện việc trích xuất đặc trưng cục bộ, và Pooling Layer, giảm chiều dữ liệu và trích xuất các đặc trưng quan trọng nhất. Khác với 2D CNN sử dụng bộ lọc 2D để đọc các mẫu không gian 2 chiều trong hình ảnh, 1D CNN sử dụng bộ lọc 1D trên mảng dữ liệu 1 chiều để đọc các mẫu trong dữ liệu tuần tự.

##### Bi-LSTM

Mô hình Bi-LSTM (Bidirectional Long Short-Term Memory) là một kiến trúc mạng nơ-ron sử dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Nó là một dạng mở rộng của mạng LSTM (Long Short-Term Memory) truyền thống, cho phép mô hình có khả năng xem xét ngữ cảnh từ cả hai phía của một câu hoặc một văn bản.

Điểm mạnh của mô hình Bi-LSTM là khả năng xử lý thông tin theo cả hai hướng: từ trái sang phải và từ phải sang trái. Điều này cho phép mô hình học được cấu trúc ngữ pháp và thông tin từ cả hai phía của một câu. Bằng cách kết hợp thông tin từ cả hai hướng, Bi-LSTM có khả năng hiểu được ngữ cảnh phức tạp hơn so với các mô hình mạng nơ-ron truyền thống.

Mô hình Bi-LSTM thường được sử dụng trong các tác vụ NLP như dự đoán từ loại, phân loại văn bản, dịch máy, tạo tiêu đề, và nhiều tác vụ khác. Với khả năng xử lý ngữ cảnh toàn diện, Bi-LSTM có thể học được các mối quan hệ phức tạp giữa các từ và xây dựng các đặc trưng phù hợp cho tác vụ cụ thể.

##### Transformer

Transformer là một kiến trúc mạng nơ-ron sử dụng rộng rãi trong lĩnh vực xử lý ngôn ngữ tự nhiên và dịch máy. Nó đã đạt được thành công đáng kể và trở thành một trong những tiến bộ quan trọng nhất trong lĩnh vực NLP trong vài năm qua. Kiến trúc Transformer được giới thiệu trong bài báo "Attention is All You Need" của Vaswani et al. vào năm 2017. Nó thay thế việc sử dụng kiến trúc mạng nơ-ron hồi quy và mạng nơ-ron tích chập truyền thống bằng cơ chế chú ý (attention mechanism) để tạo ra các kết nối giữa các từ hoặc phần tử trong một chuỗi. Thành phần chính của kiến trúc Transformer bao gồm:

* Encoder: Bộ mã hóa nhận đầu vào là một chuỗi từ hoặc phần tử và tạo ra các biểu diễn dữ liệu ẩn. Mỗi từ hoặc phần tử được đưa vào lớp embedding, sau đó được đưa qua một số lượng lớp biến đổi (transformer layers). Mỗi lớp biến đổi bao gồm hai phần: Multi-Head Attention và Feed-Forward Neural Networks.
* Decoder: Bộ giải mã nhận các biểu diễn ẩn từ bộ mã hóa và tạo ra dự đoán cho đầu ra mong muốn, chẳng hạn như dịch máy hoặc sinh văn bản. Tương tự như bộ mã hóa, bộ giải mã cũng bao gồm nhiều lớp biến đổi.

Trong ngữ cảnh mô hình phân loại, kiến trúc Encoder được sử dụng rộng rãi trong các nghiên cứu trước bởi độ hiệu quả của chúng.

#### Concatenate layer

Concatenate layer là một loại lớp trong mạng thần kinh thực hiện thao tác nối trên nhiều tensor đầu vào dọc theo một trục xác định. Nó lấy một danh sách các tensor đầu vào và nối chúng thành một tensor duy nhất dọc theo trục được chỉ định. Trong TensorFlow và Keras, lớp Concatenate được cung cấp như một phần của framework và có thể được sử dụng như một khối xây dựng để xây dựng các kiến trúc mạng nơ-ron phức tạp. Nó thường được sử dụng trong các tình huống kết hợp thông tin từ nhiều nguồn hoặc hợp nhất các tính năng từ các lớp khác nhau của mạng.

Ở trường hợp này, nhóm sử dụng layer để kết hợp 2 output của 2 model string và API riêng biệt thành 1 kết quả duy nhất, sau đó tiếp tục sử dụng các lớp mạng neuron và lớp classifier để có thể cho ra kết quả phân loại cuối cùng

# HIỆN THỰC VÀ THỰC NGHIỆM

## Hiện thực

A diagram of a computer network

Description automatically generated

Hình 4‑1. Mô hình chính chi tiết

## Chi tiết các thành phần

### Phân tích file từ Cuckoo sandbox

Để cài đặt một môi trường riêng dành cho việc phân tích file và cài đặt cuckoo sandbox, nhóm sử dụng một máy tính riêng sử dụng hệ điều hành Ubuntu 18.04. Trên máy đó, sử dụng VirtualBox để tạo một máy ảo Analysis Guests chạy hệ điều hành Windows 7 Professional 64-bit SP1. Máy ảo này đã được cài đặt một số phần mềm thường được sử dụng trên Windows như Adobe reader 9.0, Java runtime environment (JRE 7), .NET Framework 4.0, Python 2.7 (32 bit) Microsoft Office... nhằm tạo điều kiện cho các file malware hoạt động.

Để có thể phân tích các file PE một cách tự động, nhóm tận dụng các tính năng REST API của Cuckoo sandbox. REST API của Cuckoo Sandbox là một giao diện lập trình ứng dụng (API) dựa trên kiến trúc RESTful, được cung cấp bởi Cuckoo Sandbox để tương tác với sandbox và thực hiện các hoạt động phân tích và quản lý thông qua 2 phương thức HTTP là POST và GET. Các tác vụ quan trọng liên quan đến việc phân tích bao gồm:

* Gửi tệp tin để phân tích: Bằng cách gửi tệp tin dưới dạng multipart form-data hoặc gửi URL của tệp tin để phân tích.
* Truy vấn kết quả phân tích: Truy vấn kết quả phân tích dựa trên ID của tác vụ phân tích. Nếu không có đặt loại file của kết quả phân tích, mặc định file sẽ ở dạng JSON
* Xóa kết quả phân tích: Xóa kết quả phân tích dựa trên ID của tác vụ phân tích.

Nhóm tận dụng khả năng tự động gửi tệp tin để phân tích bằng cách tạo một HTTP POST đến http://[địa chỉ cuckoo api server]/tasks/create/submit kèm theo file PE cần phân tích.

Để có thể active được api server, ta có thể thực hiện như sau

**A screenshot of a computer

Description automatically generated**

Hình 4‑2. Chạy API server của Cuckoo.

### Trích xuất dữ liệu từ report của Cuckoo sandbox

Sau khi file đã phân tích xong, nhóm tiếp tục sử dụng tính năng REST API để có thể lấy được kết quả phân tích từ sandbox từ một máy tính cá nhân khác sử dụng hệ điều hành Windows 11, bằng sử dụng một Python script để tạo một HTTP GET đến http://[địa chỉ cuckoo api server]/tasks/report/[task id của file]. Response trả về có thể được đọc ở định dạng JSON để có thể có thể trích xuất dữ liệu từ report của Sandbox.

**A screen shot of a computer program

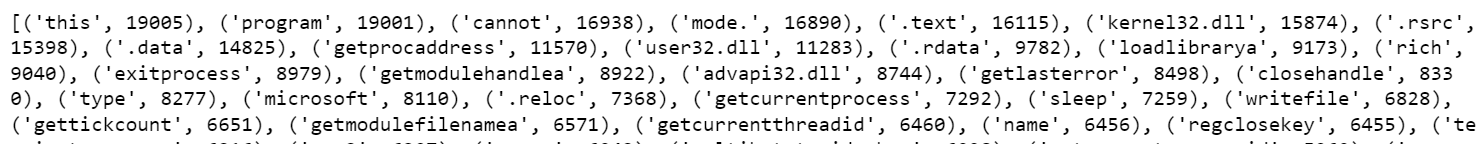
Description automatically generated**

Hình 4‑3. Hàm trích xuất dữ liệu từ report của Cuckoo Sandbox

### Cleaning

#### String

Về tỉ lệ loại bỏ frequent word:



(a)

A close-up of numbers

Description automatically generated

(b)

Hình 4‑4. Top frequent word của các file benign (a) và malware (b).

Dựa vào word count đã được trích xuất từ hai bộ data của riêng file malware và benign (mỗi loại 20000 file), ta nhận thấy rằng một số word chung của hai loại file lại có tần suất xuất hiện khác nhau và không thể đạt đến 100%, do đó nếu tỉ lệ loại bỏ thấp ( < 20 % ), một số word có thể vẫn xuất hiện lại. Từ đó, nhóm chọn tỉ lệ loại bỏ là 30% ( ~ > 14000/20000)

#### API call

Nhóm sẽ thử nghiệm cả hai trường hợp có sử dụng thuật toán cleaning giai đoạn 1a và sử dụng thuật toán cleaning tất cả các giai đoạn. Tỉ lệ ratio cho giai đoạn 2 được chọn là 3 cho kết quả hiệu quả nhất về mặt phân loại. Kết quả phân loại sẽ được trình bày ở phần 4.3.

### Tokenization

Tokenizer của Keras được sử dụng cho cả 2 mô hình string và API. Sau khi thực thi hàm Tokenizer, nhóm sử dụng thêm phương pháp padding cho các sequence nhằm đảm bảo độ dài của các sequence là bằng nhau. Đối với dữ liệu string, post padding được áp dụng. Còn trường hợp dữ liệu API call sequence, kĩ thuật post padding cũng được áp dụng cùng lúc với pre truncating, và giới hạn độ dài tối đa của chuỗi sequence là 512.

### Embedding và các thông số mô hình

Đối với Embedding, tất cả các mô hình đều có output size là 256.

Đối với mô hình CNN:

A close-up of a computer screen

Description automatically generated

Đối với mô hình Bi-LSTM:

A diagram of a computer

Description automatically generated with medium confidence

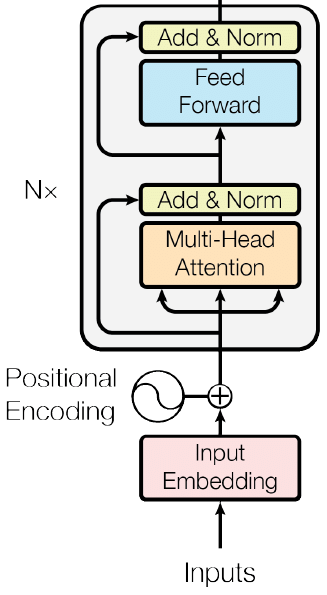
Đối với mô hình CNN + Bi-LSTM:

A diagram of a system

Description automatically generated

Đối với mô hình Transformer

Các khối Encoder được thiết kế theo như kiến trúc trong bài báo “Attention is All You Need”:



Mô hình Transformer tổng quan được sử dụng:

A diagram of a number of steps

Description automatically generated with medium confidence

## Thực nghiệm

### Giới thiệu về Dataset

Nguồn Dataset được nhóm lấy từ trang Practical Security Analytics LLC (<https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/>). Đây là nguồn dataset công khai về các file raw PE đã gán nhãn được tập hợp từ nhiều nguồn nổi tiếng như MalShare, VirusShare và TheZoo.

Thông tin chi tiết về bộ dataset như sau:

A screenshot of a computer

Description automatically generated

Do số lượng sample rất lớn, do đó nhóm trích xuất từ bộ dataset trên gồm 10,000 file malicious và 10,000 file benign để đưa vào phân tích cũng như làm bộ dataset mới.

### Thông số đánh giá (metric)

Để đánh giá hiệu suất của mô hình, nhóm tác giả sử dụng k-fold cross validation trong thực nghiệm. Đối với quá trình đánh giá, nhóm tác giả tận dụng các tiêu chuẩn sau:

1. True Positive (TP)

Số lượng mẫu tệp được phân loại là độc hại đúng.

1. True Negative (TN)

Số lượng mẫu tệp được phân loại là không độc hại đúng.

1. False Positive (FP)

Số lượng mẫu tệp được phân loại là độc hại nhưng sai.

1. False Negative (FN)

Số lượng mẫu tệp được phân loại là không độc hại nhưng sai.

1. Accuracy

Tỉ lệ số lượng mẫu được phân loại đúng (cả độc hại và không độc hại) trên tổng số mẫu.

1. Precision

Tỉ lệ số lượng mẫu malware phát hiện được đúng trên tổng số mẫu được phân loại là malware.

1. Recall

Tỉ lệ số lượng mẫu malware phát hiện được đúng trên tổng số mẫu malware thực tế.

1. F1-Score

Trung bình có trọng số của Precision và Recall, cung cấp sự cân đối giữa hai độ đo trên. Những công thức để tính được định nghĩa như dưới đây:

Accuracy = Precision =

Recall = F1-Score = 2 x

### Các kịch bản

Bao gồm 4 kịch bản chính:

* Đối với thử nghiệm độ hiệu quả của thuật toán cleaning của API
  + Sử dụng mô hình CNN kết hợp với Bi-LSTM, có thuật toán cleaning, trên 20% dataset
  + Sử dụng mô hình CNN kết hợp với Bi-LSTM, không sử dụng thuật toán cleaning giai đoạn 4a, trên 20% dataset
* Đối với dữ liệu API Sequence
  + Sử dụng mô hình Bi-LSTM, có thuật toán cleaning
  + Sử dụng mô hình CNN, có thuật toán cleaning
  + Sử dụng mô hình CNN kết hợp với Bi-LSTM, có thuật toán cleaning
* Đối với dữ liệu string
  + Sử dụng mô hình Bi-LSTM
  + Sử dụng mô hình CNN
  + Sử dụng mô hình Transformer
* Đối với kết hợp cả hai dữ liệu
  + Sử dụng mô hình CNN trên string và CNN+Bi-LSTM trên API sequence
  + Sử dụng mô hình CNN trên string và CNN trên API sequence

### Kết quả thực nghiệm

Các kịch bản được thực hiện train và test sử dụng stratified k fold với k = 5, do đó có 80% dữ liệu ngẫu nhiên được train và 20% dữ liệu còn lại được sử dụng để test. Các kịch bản được thực hiện bằng Google Colab phiên bản free sử dụng trình tăng tốc phần cứng T4 GPU.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Metric | | | | |
| Input | Model | Avg Accuracy | Precision | Recall | F1-Score |
| API Sequence | Bi-LSTM | 0.9264 | 0.922 | 0.177 | 0.9175 |
| CNN | 0.933 | 0.933 | 0.931 | 0.93 |
| CNN - Bi-LSTM | 0.9289 | 0.9321 | 0.9289 | 0.9288 |
| String | Bi-LSTM | 0.97 | 0.966 | 0.965 | 0.965 |
| CNN | 0.983 | 0.9828 | 0.9827 | 0.9827 |
| Transformer | 0.985 | 0.9865 | 0.9865 | 0.9865 |
| API Sequence + String | CNN - Bi-LSTM + CNN | 0.9874 | 0.9847 | 0.9847 | 0.9847 |
| 2 CNN | 0.9789 | 0.988 | 0.988 | 0.988 |

Bảng 1. Thông số kết quả của từng kịch bản

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Việc áp dụng NLP vào việc phát hiện malware đã đem lại nhiều tiềm năng và phát triển trong những năm gần đây. Các phương pháp hybrid, tức là sự kết hợp giữa các phương pháp static và dynamic analysis, đã được đề xuất để khắc phục nhược điểm của cả hai phương pháp.

Phương pháp static analysis tập trung vào việc phân tích các đặc điểm tĩnh của tệp tin, như mã nguồn, cấu trúc file và chuỗi ký tự. Điều này giúp phát hiện các đặc điểm đáng ngờ như các hàm gọi hệ thống nguy hiểm hoặc chuỗi mã độc. Tuy nhiên, phương pháp này có thể bị đánh lừa bởi các kỹ thuật ẩn danh mã hóa và biến thể mã độc.

Phương pháp dynamic analysis tập trung vào việc thực thi malware trong một môi trường cô lập để quan sát hành vi hoạt động của nó. Điều này có thể bao gồm việc giám sát các hoạt động mạng, ghi lại các hành vi ghi nhật ký hoặc giám sát các thay đổi trên hệ thống. Tuy nhiên, phương pháp này có thể mắc phải các kỹ thuật phòng vệ như việc phát hiện môi trường giả lập hoặc các kỹ thuật ẩn danh.

Bằng cách kết hợp cả hai phương pháp, các phương pháp hybrid tận dụng sự mạnh mẽ của cả static và dynamic analysis. Sử dụng NLP, các phương pháp này có thể phân tích các thành phần văn bản của mã độc, như các chuỗi ký tự, hằng số, tên biến, và các phép toán. Đồng thời, chúng cũng có thể quan sát và phân tích hành vi của malware trong một môi trường thực thi.

Sự kết hợp giữa NLP và phương pháp hybrid cung cấp những ưu điểm đáng kể. Đầu tiên, nó tăng cường khả năng phát hiện malware bằng cách kết hợp cả những đặc trưng tĩnh và động. Thứ hai, việc sử dụng NLP giúp hiểu và phân tích ngôn ngữ tự nhiên liên quan đến malware, từ đó tìm ra các đặc điểm ngôn ngữ độc hại. Cuối cùng, phương pháp hybrid cũng có thể giảm tỷ lệ phát hiện nhầm và tăng độ chính xác của hệ thống phát hiện.

### Kết quả đạt được

Nhóm thành công kết hợp được việc phân tích string và API Sequence vào cùng một mô hình, kết quả thu được là vô cùng khả quan theo các thông số đã kể ở trên.

### Hạn chế

Thuật toán cleaning đối với dữ liệu API call chưa đạt được hiệu quả như mong muốn.

Mô hình được đề xuất chưa có các kịch bản thử nghiệm cụ thể đối với các loại malware có cơ chế ẩn mình như anti-debugger, anti-sandbox và anti-vm cũng như packing.

Bộ dataset đang sử dụng chưa được dùng trong các nghiên cứu khác, do đó không thể so sánh tính hiệu quả của mô hình đối với các mô hình khác đã được nghiên cứu trước đó.

## Hướng phát triển

Trong tương lai, nhóm sẽ tìm phương pháp cải thiện hoặc thay đổi thuật toán cleaning dựa trên các nghiên cứu khác.

Thực hiện thử nghiệm trên các kịch bản mới nhằm chứng minh được tính hiệu quả của mô hình.

Tạo bộ dataset mới dựa vào nguồn mới, cụ thể là VirusShare

# TÀI LIỆU THAM KHẢO

Han, Weijie & XUE, Jingfeng & Wang, Yong & Huang, Lu & Kong, Zixiao & Mao, Limin. (2019). MalDAE: Detecting and Explaining Malware based on Correlation and Fusion of Static and Dynamic Characteristics. Computers & Security. 83. 10.1016/j.cose.2019.02.007.

Maniriho, P., Mahmood, A.N., & Chowdhury, M.J. (2022). MalDetConv: Automated Behaviour-based Malware Detection Framework Based on Natural Language Processing and Deep Learning Techniques. *ArXiv, abs/2209.03547*.