HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

ĐẬU ĐỨC SIÊU

PHÁT HIỆN MÃ ĐỘC DỰA TRÊN PHÂN TÍCH MẪU

CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN

TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. PHẠM HOÀNG DUY

HÀ NỘI – 2021

PHÁT HIỆN MÃ ĐỘC DỰA TRÊN PHÂN TÍCH MẪU

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

Người hướng dẫn khoa học: TS. Phạm Hoàng Duy

Phản biện 1: PGS TS. Ngô Quốc Tạo .....

Phản biện 2: TS. Hoàng Xuân Dậu .....

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn họp tại:
Học viện Công nghệ Bưu chính Viễn thông
Vào hồi 09 giờ 15 ngày 28 tháng 8 năm 2021

Có thể tìm hiểu luận văn tại:
Thư viện học viện Công nghệ Bưu chính Viễn thông

MỞ ĐẦU

Phần mềm độc hại đang là một mối đe dọa rất lớn về bảo mật trong thời đại kỹ thuật số ngày nay. Người dùng máy tính, các công ty và chính phủ đang chịu các cuộc tấn công sử dụng các phần mềm độc hại gia tăng theo cấp số nhân. Phân tích phần mềm độc hại trở thành một thành phần quan trọng của cơ chế bảo vệ. Phương pháp phân tích tĩnh mã độc cổ điển đã đem lại những hiệu quả cao tuy nhiên nhiều phần mềm độc hại gần đây sử dụng các kỹ thuật đa hình, biến hình và các kỹ thuật lẩn tránh khác để thay đổi hành vi của phần mềm độc hại một cách nhanh chóng và tạo ra một số lượng lớn phần mềm độc hại.

Từ nhu cầu phát triển phân tích mã độc trên, luận văn sẽ tập trung nghiên cứu phương pháp phát hiện mã độc dựa trên phân tích mẫu với nội dung được trình bày như sau:

Chương 1: Mã độc và một số phương pháp phân tích mã độc

Giới thiệu chương về các khái niệm cơ bản về mã độc và phân tích mã độc cũng như một số phương pháp, công cụ xác định mã độc hiện hành.

Chương 2: Mô hình phát hiện mã độc

Giới thiệu chương: Chương này đưa ra mô hình chung cho việc xử lý các dữ liệu độc hại, giới thiệu cách thức để trích xuất dữ liệu sang dạng phân tích được, cuối cùng là khai phá các mẫu tuần tự để tìm ra ngưỡng phù hợp để xác định mã độc và phân tích xem dữ liệu có phải là độc hại hay không.

Chương 3: Thử nghiệm và đánh giá

Giới thiệu chương: Chương này giới thiệu về tập dữ liệu, cách thức thực hiện và triển khai mô hình phân tích mã độc

Nội dung chương sẽ giới thiệu quá trình thu thập dữ liệu thử nghiệm, xử lý và trích chọn đặc trưng, huấn luyện các mô hình thuật toán học máy, đưa ra kết quả và nhận xét đánh giá.

CHƯƠNG I: TỔNG QUAN VỀ MÃ ĐỘC VÀ CÁC PHƯƠNG PHÁP PHÁT HIỆN

Chương I trình bày khái niệm về mã độc, lịch sử phát triển và các loại mã độc phổ biến. Ngoài ra, trong chương này cũng sẽ đi tìm hiểu về các phương pháp phân tích và phát hiện mã độc cùng các nghiên cứu có liên quan.

1.1 Tổng quan về mã độc

1.1.1. Định nghĩa

Mã độc hay Malware (Malicious software) là một khái niệm chung dùng để chỉ các phần mềm độc hại được viết với mục đích có thể lây lan phát tán (hoặc không lây lan, phát tán) trên hệ thống máy tính và internet.

1.1.2. Lịch sử phát triển

Lịch sử phát triển của mã độc gắn liền với lịch sử phát triển máy tình và mạng máy tính. Các virus đầu tiên là trò đùa lành tính; virus độc hại không được công khai cho đến đầu những năm 1980. Trojan bắt đầu lộ diện vào giữa năm 1980. Trong nửa sau của những năm 1990, một số thay đổi quan trọng trong máy tính tạo ra cơ hội mới cho các phần mềm độc hại: số lượng máy tính cá nhân tăng lên rất nhiều và việc sử dụng các ứng dụng Thư điện tử và phần mềm với ngôn ngữ vĩ mô. Theo đó, người viết virus bắt đầu phát triển virus hiểu và lan truyền qua thư điện tử, cũng như phát triển Sâu với khả năng tương tự

Một xu hướng khác là nhiều trường hợp của các phần mềm độc hại, bao gồm worms, trojan và mã độc di động, cung cấp các công cụ tấn công, chẳng hạn như các rootkit, keystroke logger, và backdoors, để hệ thống bị nhiễm.

Trong những năm gần đây, các cuộc tấn công tống tiền bằng mã độc bùng nổ. Nổi bật nhất có thể kể đến cuộc tấn công WannaCry (năm 2017) lây nhiễm cho 250000 máy tính. Trong tương lai không xa, mã độc được dự báo sẽ có thêm các bước biến đổi khác, nó bao gồm mọi điểm mạnh sẵn và còn kết hợp với các thủ đoạn khác của phần mềm gián điệp. Đồng thời chúng có thể tấn công vào nhiều hệ điều hành khác nhau chứ không nhất thiết nhắm vào một hệ điều hành độc nhất.

1.1.3. Các loại mã độc phổ biến

a) Vi rút khởi động

Virus khởi động, là loại virus lây vào phân vùng khởi động hoặc bản ghi gốc của ổ đĩa cứng. Loại virus này khởi động được thực thi trước khi hệ điều hành được nạp lên. Ngày nay gần như không còn thấy sự xuất hiện của Virus khởi động do đặc điểm lây lan chậm và không phù hợp với thời đại Internet.

b) Virus tác vụ (Macro virus)

Đây là loại virus đặc biệt tấn công vào chương trình trong bộ Microsoft Office của Microsoft: Word, Excel, Powerpoint. Macro là tính năng hỗ trợ trong bộ công cụ văn phòng Microsoft Office cho phép người sử dụng lưu lại các công việc cần thực hiện lại nhiều lần. Thực tế hiện nay cho thấy virus macro gần như đã "tuyệt chủng".

c) Virus script

Đây là loại virus được viết bằng các ngôn ngữ script (kịch bản) như VBScript, JavaScript, Batch script. Những loại virus này thường có đặc điểm dễ viết, dễ cài đặt. Chúng thường tự lây lan sang các file script khác, thay đổi nội dung cả các file html để thêm các thông tin quảng cáo, chèn banner … Đây cũng là một loại virus phát triển nhanh chóng nhờ sự phổ biến của Internet.

d) Virus thực thi

Virus này chuyên lây vào các file thực thi (ví dụ file có phần mở rộng .com, .exe, .dll) một đoạn mã để khi file được thực thi, đoạn mã virus sẽ được kích hoạt trước và tiếp tục thực hiện các hành vi phá hoại, lây nhiễm.

e) Virus gián điệp (Trojan)

Các đoạn mã của Trojan được "che giấu" trong các loại virus khác hoặc trong các phần mềm máy tính thông thường để bí mật xâm nhập vào máy nạn nhân. Khi tới thời điểm thuận lợi chúng sẽ tiến hành các hoạt động ăn cắp thông tin cá nhân, mật khẩu, điều khiển máy tính nạn nhân …

f) BackDoor

Backdoor (cửa hậu) trong phần mềm hay hệ thống máy tính thường là một cổng

không được thông báo rộng rãi có thể đến và đi tùy ý, cho phép truy cập hệ thống từ xa.

g) Adware và Spyware

Đây là loại Trojan khi xâm nhập vào máy tính với mục đích quảng cáo hoặc "gián điệp". Chúng đưa ra các quảng cáo, mở ra các trang web, thay đổi trang mặc định của trình duyệt … gây khó chịu cho người sử dụng. …

h) Worm

Vào thời điểm ban đầu, Worm được tạo ra chỉ với mục đích phát tán qua thư điện tử. Khi lây vào máy tính, chúng thực hiện tìm kiếm các sổ địa chỉ, danh sách thư điện tử trên máy nạn nhân rồi giả mạo các thư điện tử để gửi bản thân chúng tới các địa chỉ thu thập được.

Bên cạnh Worm lây lan theo cách truyền thống sử dụng thư điện tử, Worm hiện nay còn sử dụng phương pháp lân lan qua ổ USB. Thiết bị nhớ USB đã trở nên phổ biến trên toàn thế giới do lợi thế kích thước nhỏ, cơ động và trở thành phương tiện lây lan lý tưởng cho Worm.

i) Rootkit

Với sự xuất hiện của rootkit, các phần mềm độc hại như trở nên "vô hình" trước những công cụ thông thường thậm chí vô hình cả với các phần mềm diệt virus. Việc phát hiện mã độc và tiêu diệt virus trở nên khó khăn hơn rất nhiều trước sự bảo vệ của rootkit – vốn được trang bị nhiều kĩ thuật mới hiện đại.

j) Botnet

Từ "botnet" là sự kết hợp của hai từ, "robot" **và** "network"**.** Ở đây, một tên tội phạm mạng thực hiện vai trò của một "botmaster" sử dụng virus để xâm phạm bảo mật của một số máy tính và kết nối chúng vào mạng vì mục đích xấu. Mỗi máy tính trên mạng hoạt động như một "bot", và được kẻ xấu kiểm soát để lây truyền mã độc, spam hoặc nội dung độc hại nhằm khởi động cuộc tấn công.

k) Keylogger

Ghi lại tất cả các phím do người dùng nhấn và lưu trữ tất cả dữ liệu, bao gồm mật khẩu, số thẻ ngân hàng và thông tin nhạy cảm khác.

l) Ransomware

Loại phần mềm độc hại này nhằm mục đích mã hóa tất cả dữ liệu trên máy và yêu cầu nạn nhân chuyển tiền để lấy khóa giải mã. Thông thường, một máy bị nhiễm phần mềm ransomware bị "đóng băng" vì người dùng không thể mở bất kỳ tệp nào.

1.2 Các phương pháp phát hiện mã độc

1.2.1 Các kỹ thuật phân tích mã độc

Có hai phương pháp chính để thực hiện phân tích mã độc gồm: Phân tích tĩnh và Phân tích động.

a) Phân tích tĩnh

Phân tích tĩnh là kỹ thuật sử dụng các công cụ để đọc một phần hoặc toàn bộ mã nguồn của chương trình độc hại và từ đó cố gắng suy ra được đặc tính hành vi của chương trình. Chương trình độc hại có thể được viết bằng nhiều ngôn ngữ khác nhau, phổ biến nhất là assembly. Vì vậy có nhiều phương pháp phân tích tĩnh khác nhau như:

- *Kiểm tra định dạng tệp*:
- *String Extraction*: đề cập đến việc kiểm tra đầu ra phần mềm
- *Fingerprinting*:
- *AV scanning*:
- *Disassembly*:

Ưu điểm của phân tích tĩnh là có thể tìm ra tất cả kịch bản thực thi có thể có của mã độc mà không bị hạn chế về bất kỳ điều kiện gì. Hơn nữa, phân tích tĩnh an toàn hơn phân tích động bởi không cần thực thi mã độc trực tiếp, vì thế sẽ không gây nguy hiểm cho hệ thống. Tuy nhiên phân tích tĩnh lại tốn rất nhiều thời gian, vì thế phân tích tĩnh thường không được sử dụng trong thực tế mà thường dùng để nghiên cứu, ví dụ khi nghiên cứu chữ ký cho các mã độc zero-day

b) Phân tích động

Không giống với phân tích tĩnh ở chỗ, các hành vi của mã độc được giám sát trong khi nó đang thực thi, từ đó có thể tìm hiểu được thuộc tính và mục đích của mã độc. Thông thường mã độc sẽ được thực thi trong môi trường ảo. Trong quá trình phân tích sẽ phát hiện tất cả hành vi của mã độc, như mở tệp tin, tạo mutexes, ... và kiểu phân tích này sẽ nhanh hơn phân tích tĩnh rất nhiều. Tuy nhiên, phân tích động chỉ biết được hành vi

của mã độc trong hệ thống ảo dùng để kiểm tra, ví dụ kết quả thu được khi thực thi hai mã độc giống nhau trong môi trường Windows 7 và Windows 8.1 sẽ khác nhau

1.2.2 Các phương pháp hiện mã độc

Các phương pháp này gồm: Phương pháp phát hiện dựa trên chữ ký và Phương pháp phát hiện dựa trên hành vi.

a) Phương pháp phát hiện dựa trên chữ ký

Phân tích dựa trên chữ ký là một phương pháp tĩnh dựa trên các chữ ký được xác định trước. Kịch bản phát hiện, trong trường hợp này, sẽ như sau: khi một tệp đến hệ thống, nó được phân tích tĩnh bởi phần mềm chống vi-rút. Nếu bất kỳ chữ ký nào được khớp, cảnh báo được kích hoạt, cho biết tệp này là đáng ngờ. Đa phần phân tích kiểu này là đủ vì các mẫu phần mềm độc hại nổi tiếng thường có thể được phát hiện dựa trên giá trị băm.

b) Phương pháp phát hiện dựa trên hành vi

Tuy nhiên, những kẻ tấn công bắt đầu phát triển phần mềm độc hại theo cách nó có thể thay đổi chữ ký của nó. Tính năng phần mềm độc hại này được gọi là đa hình (polymorphism). Rõ ràng, phần mềm độc hại như vậy không thể được phát hiện bằng cách sử dụng chữ ký.

Ưu điểm chính của phát hiện dựa trên heuristics là nó không chỉ phát hiện các mã độc đã biết mà còn phát hiện được các cuộc tấn công zero-day và các loại virus đa hình. Tuy nhiên, một số loại mã độc có khả năng phát hiện môi trường ảo, nó sẽ không thực thi các hành vi độc hại trong môi trường sandbox.

1.3 Các nghiên cứu liên quan

Mặc dù chưa được ứng dụng rộng rãi, ý tưởng sử dụng học máy trong việc phát hiện mã độc không còn mới. Một số nghiên cứu về lĩnh vực này đã được thực hiện nhằm kiểm tra tính chính xác của các phương pháp khác nhau.

Wang và Stolfo [4] trình bày PAYL, một công cụ tính toán tải trọng dự kiến cho mỗi dịch vụ (cổng) trên hệ thống.

Boldt và Carlson [3] đưa ra khái niệm về phần mềm xâm phạm quyền riêng tư (PIS).

Alazab [5] đã đề xuất phương pháp sử dụng API để biểu diễn đặc trưng của mã độc..

Baldangombo và cộng sự đã giới thiệu phương pháp trích chọn đặc trưng dựa trên tiêu đề PE, các thư viện DLL và các hàm chức năng API [6]

Dragos Gavrilut [7] đã đề xuất hệ thống phát hiện mã độc dựa trên các thuật toán perceptron cải tiến..

Singhal và Raul đã thảo luận về phương pháp phát hiện dựa trên thuật toán Random Forest cải tiến kết hợp với Information Gain để biểu diễn đặc trưng tối ưu hơn [8]..

Kết quả đưa ra của các nghiên cứu ở trên đều không giống nhau, do chưa có một phương pháp thống nhất trong việc phát hiện cũng như biểu diễn đặc trưng. Độ chính xác của từng trường hợp còn phụ thuộc vào các loại mã độc được dùng để lấy mẫu và quá trình chạy thực tế.

1.4 Kết luận chương

Chương I đã trình bày về khái niệm mã độc, lịch sử hình thành mã độc, phân loại và giới thiệu các một số loại mã độc phổ biến. Có thể thấy, mã độc ngày càng phát triển mạnh mẽ với nhiều biến thể khác khác nhau. Chính vì vậy, chương I cũng đã đưa ra các kỹ thuật phân tích mã độc làm cơ sở cho các phương pháp phát hiện mã độc. Trong chương II, luận văn sẽ tìm hiểu sâu hơn vào học máy và kỹ thuật phân tích, trích xuất mã lệnh (lệnh vận chuyển dữ liệu) của mã độc nhằm mục đích xây dựng phương pháp phát hiện mã độc dựa trên phân tích mẫu.

CHƯƠNG II: MÔ HÌNH PHÁT HIỆN MÃ ĐỘC

Chương II giới thiệu tổng quan về học máy và các thuật toán học máy phổ biến. Tiếp đó, chương sẽ trình bày về kỹ thuật phân tích mã độc để trích xuất mã lệnh và xây dựng phương pháp phát hiện mã độc dựa trên phân tích mẫu kết hợp với học máy.

2.1 Tổng quan về học máy

2.1.1 Khái niệm học máy

Học máy (hay Machine Learning) là một công nghệ phát triển từ lĩnh vực trí tuệ nhân tạo. Các thuật toán học máy là các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ và cách cải thiện hiệu suất theo thời gian.

Học máy vẫn đòi hỏi sự đánh giá của con người trong việc tìm hiểu dữ liệu cơ sở và lựa chọn các kĩ thuật phù hợp để phân tích dữ liệu. Đồng thời, trước khi sử dụng, dữ liệu phải sạch, không có sai lệch và không có dữ liệu giả.

2.1.2 Các phương học máy phổ biến

Dự theo phương thức học, các thuật toán Machine Learning thường được chia làm 4 nhóm: Học có giám sát, Học phi giám sát, Học bán giám sát và học tăng cường. Ngoài ra có một số cách phân nhóm không có Học bán giám sát hoặc Học tăng cường.

- Supervised learning (hay học có giám sát): là phương pháp sử dụng những dữ liệu đã được gán nhãn từ trước để suy luận ra quan hệ giữa đầu vào và đầu ra. Các dữ liệu này được gọi là dữ liệu huấn luyện và chúng là cặp các đầu vào-đầu ra. Học có giám sát sẽ xem xét các tập huấn luyện này để từ đó có thể đưa ra dự đoán đầu ra cho 1 đầu vào mới chưa gặp bao giờ.
- Unsupervised learning (hay học phi giám sát): Khác với học có giám sát, học phi giám sát sử dụng những dữ liệu chưa được gán nhãn từ trước để suy luận. Phương pháp này thường được sử dụng để tìm cấu trúc của tập dữ liệu. Tuy nhiên lại không có phương pháp đánh giá được cấu trúc tìm ra được là đúng hay sai.
- Semi-supervised learning (hay học nửa giám sát): là một lớp của kỹ thuật học

máy, sử dụng cả dữ liệu đã gán nhãn và chưa gán nhãn để huấn luyện – điển hình là một lượng nhỏ dữ liệu có gán nhãn cùng với lượng lớn dữ liệu chưa gán nhãn. Học nửa giám sát đứng giữa học không giám sát (không có bất kỳ dữ liệu có nhãn nào) và có giám sát (toàn bộ dữ liệu đều được gán nhãn)..

- Reinforcement learning (hay học tăng cường): Phương pháp học tăng cường tập trung vào việc làm sao để cho 1 tác tử trong môi trường có thế hành động sao cho lấy được phần thưởng nhiều nhất có thể.

2.1.3 Quy trình huấn luyện mô hình học máy

Quá trình xây dựng một mô hình học máy cơ bản cần được dựa theo các bước như sau:

- Chuẩn bị dữ liệu
- Chọn mô hình
- Tìm tham số
- Suy luận.

Để một mô hình đạt được hiệu quả cao, quá trình chuẩn bị dữ liệu gần như là quá trình quan trọng nhất, đây cũng được coi là quá trình tiên quyết. Dữ liệu sau khi thu thập được cần phải:

a) **Chuẩn hoá**: Tất cả các dữ liệu đầu vào đều cần được chuẩn hoá để máy tính có thể xử lý được. Quá trình chuẩn hoá bao gồm số hoá dữ liệu, điều chỉnh thông số cho phù hợp với bài toán. Có nhiều phương pháp chuẩn hoá dữ liệu khác nhau, phổ biến nhất có thể kể đến các phương pháp sau:

- Trung tâm hóa dữ liệu (centering data)
- Chuẩn hóa min-max (rescaling)
- Co giãn trung bình (mean normalization)
- Chính quy hóa (standardisation)

b) **Trích chọn đặc trưng:** Mục đích của quá trình này là thu được một tập dữ liệu chi tiết và không dư thừa. Các đặc trưng phải biểu diễn thông tin quan trọng và liên quan tới tập dữ liệu nếu không kết quả dự đoán sẽ không chính xác.

c) **Phân chia**: Nhằm tránh khớp quá (*Overfitting*). Vì vậy khi huấn luyện phải phân

chia dữ liệu ra thành 3 loại để có thể kiểm chứng được phần nào mức độ tổng quát của mô hình. Cụ thể 3 loại đó là:

- **Tập huấn luyện** (*Training set*): Chiếm 60%. Dùng để học khi huấn luyện.
- **Tập kiểm chứng** (*Cross validation set*)**:** Chiếm 20%. Dùng để kiểm thử mô hình khi huấn luyện.
- **Tập kiểm tra** (*Test set*): Chiếm 20%.

2.2 Một số kỹ thuật học máy phổ biến

2.2.1 Đinh lý Bayes

Naive Bayes là một thuật toán dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các phán đoán cũng như phân loại dữ liệu dựa trên các dữ liệu được quan sát và thống kê, được ứng dụng rất nhiều trong các lĩnh vực Machine learning dùng để đưa các dự đoán có độ chính xác cao, dựa trên một tập dữ liệu đã được thu thập. Ý tưởng chính của thuật toán là tính toán xác suất của mỗi đặc trưng một cách độc lập, sau đó đưa ra dự đoán dựa trên định lý Bayes.

Ưu điểm của thuật toán là đơn giản và dễ hiểu. Hơn nữa, nó phù hợp với các tập dữ liệu có nhiều đặc trưng khác nhau, bởi dự đoán phụ thuộc vào xác suất của các đặc trưng. Ngoài ra, thuật toán tiêu tốn ít tài nguyên, có hiệu năng cao, không cần tính toán các hệ số phụ như các thuật toán khác.

2.2.2 Support Vector Machine

Support Vector Machines (SVM) là một thuật toán phổ biến thường được dùng trong các bài toán phân lớp. Ý tưởng chính là tìm kiếm một siêu mặt phẳng phân chia các lớp một cách tối ưu nhất. Khoảng cách giữa support vector và siêu mặt phẳng được gọi là khoảng cách biên (margin).

Thuật toán SVM thường cho kết quả khá chính xác, đặc biệt là đối với các tập dữ liệu "sạch". Hơn nữa, nó còn phù hợp với các tập dữ liệu nhiều chiều, kể cả khi số chiều nhiều hơn số lượng mẫu. Nó cũng hiệu quả với các tập dữ liệu có nhiều nhiễu hoặc chồng chéo nhau. Tuy nhiên, thời gian huấn luyện có thể rất lâu.

2.2.3 Decision Tree

Decision Tree (Cây quyết định) là một cây phân cấp có cấu trúc được dùng để phân

lớp các đối tượng dựa vào dãy các luật. Khi cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các đối tượng chưa biết. Các quy tắc này được trích ra dựa trên bộ các đặc trưng của các dữ liệu huấn luyện. Trong cây quyết định, các lá đại diện cho các lớp (nhãn), mỗi nút con trong cây và các nhánh cây của nó biểu diễn sự kết hợp của các đặc trưng để dẫn dắt tới việc phân lớp. Như vậy, việc phân loại một đối tượng sẽ bắt đầu với việc kiểm tra giá trị của nút gốc, sau đó tiếp tục đi xuống dưới theo các nhánh cây tương ứng với các giá trị đó. Quá trình này được lặp đi lặp lại đối với từng nút, cho đến khi không thể đi tiếp được nữa và chạm đến nút lá. Mục đích của thuật toán là đạt được kết quả chính xác nhất với số lần lựa chọn ít nhất.

Cây quyết định là một thuật toán phổ biến bởi nó đơn giản và có thể xử lý tốt các tập dữ liệu lớn và có nhiều dữ liệu nhiễu. Một ưu điểm khác của cây quyết định là người ta có thể theo dõi quá trình lựa chọn một cách tường minh. Điều này khiến cho thuật toán này trở nên phổ biến cho các bài toán như chẩn đoán y tế, lọc thư rác, sàng lọc an ninh.

2.2.4 Random Forest

Random Forest là một thành viên trong chuỗi thuật toán cây quyết định. Ý tưởng của Random Forest là tạo ra một vài cây quyết định. Các cây quyết định này sẽ chạy và cho kết quả độc lập. Câu trả lời được dự đoán bởi nhiều cây quyết định nhất sẽ được Random Forest lựa chọn. Để đảm bảo các cây quyết định không giống nhau, Random Forest sẽ ngẫu nhiên chọn ra một tập con các đặc trưng ở mỗi nút (thay cho toàn bộ). Các tham số còn lại được sử dụng trong Random Forest giống như trong cây quyết định.

Random Forest thừa kế rất nhiều ưu điểm của thuật toán cây quyết định. Random Forest có thể dùng cho cả hai bài toán phân loại và hồi quy, bởi nó đơn giản và dễ thích nghi, kết quả đưa ra cũng chính xác hơn. Tuy nhiên, không như cây quyết định, cấu trúc của Random Forest rất phức tạp nên không thể hiểu được cơ chế hoạt động bên trong của thuật toán. Ngoài ra, Random Forest cũng ổn định hơn so với cây quyết định. Đối với cây quyết định, chỉ cần dữ liệu bị sửa đổi một chút thì cả cây cũng sẽ bị thay đổi, làm giảm độ chính xác. Còn với thuật toán Random Forest, do nó được kết hợp từ rất nhiều cây quyết

định nên nó sẽ ổn định hơn.

2.3 Một số phương pháp trích chọn đặc trưng phổ biến với bài toán phát hiện mã độc

2.3.1 Trích chọn đặc trưng dựa trên PE Header

PE Header là cấu trúc IMAGE_NT_HEADERS bao gồm các thông tin cần thiết cho quá trình loader load file lên bộ nhớ. Cấu trúc này gồm 3 phần được định nghĩa trong windows.inc:

- **Signature**
- **FileHeader**
- **OptionalHeader**

Như vậy việc trích xuất thông tin của phần PE Header có thể cho ta được cái nhìn tổng quan về định dạng, nhiệm vụ, logic thực thi của file. Đây cũng chính là các đặc trưng thường được trích xuất để phân loại một file bình thường hay file chứa mã độc. Tuy nhiên, các đặc trưng này có nhược điểm là kẻ tấn công có thể thay đổi thông tin một file mã độc sao cho giống với file bình thường.

2.3.2 Trích chọn đặc trưng dựa trên ảnh nhị phân

Các file dù là định dạng gì đều có thể được biểu diễn dưới dạng nhị phân. Vì vậy, có thể chuyển đổi dữ liệu nhị phân của một file thành ảnh nhị phân. Cụ thể hơn có thể biến đổi theo 2 cách khác nhau như sau:

- Dữ liệu nhị phân ban đầu được chuyển về dạng ma trận nhị phân có kích thước phù hợp với bài toán. Sau đó ma trận được đưa về ảnh nhị phân
- Dữ liệu nhị phân được đọc dưới dạng từng byte (tương ứng với 8 bit – có giá trị từ 0 đến 255). Tiếp đó dữ liệu ở dạng byte cũng được đưa về dạng ma trận phù hợp với bài toán. Cuối cùng là được chuyển đổi thành ảnh nhị phân xám, mỗi pixel tương ứng với 1 byte (giá trị của pixel tương ứng từ 0 đến 255 đại diện cho độ xám của pixel).

Sau khi đã thu được ảnh nhị phân từ dữ liệu nhị phân, có thể sử dụng các phương pháp trích chọn đặc trưng cho ảnh để trích xuất đặc trưng cho bài toán phát hiện mã độc.

2.3.3 Trích chọn đặc trưng dựa trên mã nguồn

Có thể sử dụng các công cụ dịch ngược để lấy được nội dung mã nguồn của mã độc.

Mã nguồn của một chương trình thực chất là một file văn bản, nội dung văn bản thể hiện logic của chương trình, trong đó khi muốn thực hiện logic nào đó, các hàm, các câu lệnh phải viết theo 1 trình tự nhất định. Vì lý do này, có thể sử dụng các phương pháp trình chọn đặc trưng cho văn bản như n-gram, tf-idf để trích xuất đặc trưng từ mã nguồn của một chương trình.

2.4 Phương pháp phát hiện mã độc dựa trên phân tích mẫu

2.4.1 Cấu trúc đoạn mã Assembly

Các chương trình mã độc thường được xây dựng từ các đoạn mã Assembly có cấu trúc 4 trường riêng biệt cách nhau bởi dấu tab hoặc dấu cách như ví dụ dưới đây:

```
1 | Label Opcode  Operands    Comment
2 | Func  MOV     R0, #100    ; đặt R0 bằng 100
3 |               BX  LR      ; hàm return
```

Hình II-4: Ví dụ cấu trúc đoạn mã Assembly

- **Label**
- **Opcode**
- **Operand**
- **Comment**

2.4.2 Công cụ trích xuất và phân tích mã lệnh của mã độc

Có nhiều công cụ giúp đọc được đoạn mã assembly của một chương trình như nasm, gdb, objdump, strace. Công cụ được sử dụng nhiều nhất là objdump. Objdump là công cụ giúp bạn xem các thông tin quan trọng trong các file object hay file executable.

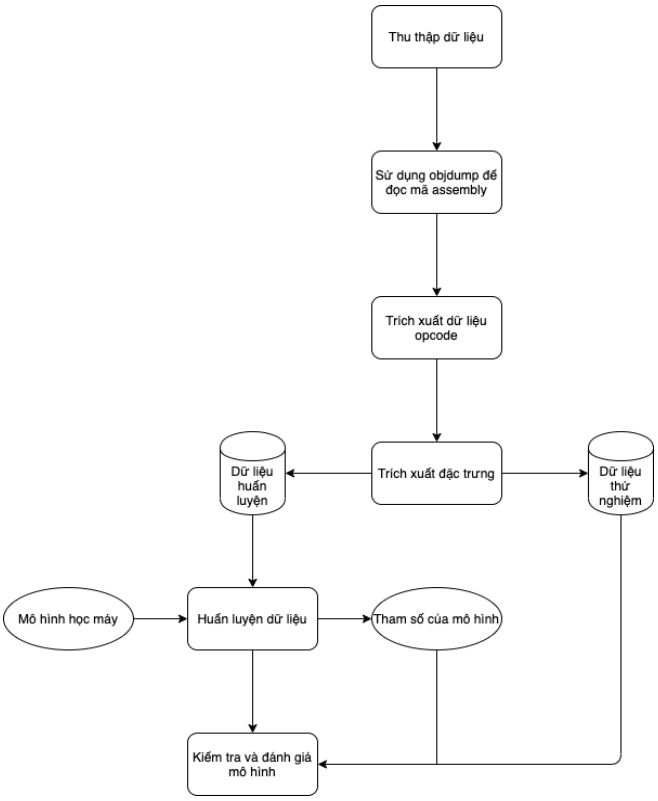
Để thực hiện đọc đoạn mã assembly của một chương trình, với công cụ objdump, người dùng có thể sử dụng lệnh:

objdump –M <mã kiến trúc bộ vi xử lý chạy chương trình> -D <đường dẫn tới chương trình> > <đường dẫn tới nơi muốn lưu trữ file báo cáo>

Sau khi đã thu được đoạn mã assembly của chương trình, hoàn toán có thể thực hiện quá trình trích xuất thông tin dữ liệu mã lệnh của chương trình để thực hiện phân tích và phát hiện xem chương trình có phải chương trình mã độc không.

2.4.3 Phương pháp phát hiện mã độc dựa trên phân tích mẫu

Mô hình tổng quan của phương pháp phát hiện mã độc dựa trên học máy được thể hiện trên hình sau:



Sơ đồ phương pháp phát hiện mã độc dựa trên phân tích mẫu

Bước đầu tiên của bất kỳ bài toán ứng dụng học máy cũng là thực hiện quá trình thu thập dữ liệu cho bài toán. Dữ liệu cần đảm bảo đủ lớn và có tính đa dạng về dữ liệu. Ngoài ra, để mô hình đạt kết quả tốt, phân bố của các lớp phải là xấp xỉ nhau. Với bài toán phát hiện mã độc, bước đầu cần phải thu thập các chương trình mã độc cùng với đó là các chương trình bình thường. Số lượng chương trình mã độc và chương trình bình thường cần phải là tương đương.

Các mô hình học máy phổ biến sau đó được sử dụng để huấn luyện và tìm ra các ngưỡng tham số của mô hình. Chính các ngưỡng tham số này kết hợp với mô hình sẽ phân loại và phát hiện dữ liệu mới thuộc nhóm chương trình mã độc hay chương trình bình thường. Cuối cùng, để đánh giá tính hiệu quả của phương pháp này, dữ liệu thử nghiệm sẽ được sử dụng để đánh giá các mô hình học máy.

2.5 Kết luận chương

Chương II đã trình bày khái niệm về học máy và các thuật toán nổi bật trong học máy. Ngoài ra, chương đã giới thiệu kỹ thuật phân tích, trích xuất mã lệnh của một chương trình mã độc. Áp dụng kết hợp học máy, chương đã trình bày về các bước trong phương pháp phát hiện mã độc sử dụng phân tích mẫu. Trong chương tiếp theo, luận văn sẽ trình bày về quá trình thực nghiệm để đánh giá hiệu quả của phương pháp trên.

CHƯƠNG III: THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương III trình bày về quá trình thực nghiệm thông qua các bước thu thập dữ liệu, tiền xử lý trích chọn đặc trưng của dữ liệu, phân chia dữ liệu và cài đặt môi trường để tiến hành thực nghiệm. Tiếp đến, trình bày các kết quả thực nghiệm và đánh giá hiệu quả của mô hình phát hiện mã độc được trình bày trong chương 2.

3.1 Thu thập dữ liệu và tiền xử lý dữ liệu

3.1.1 Thu thập dữ liệu

Để thực nghiệm phương pháp phát hiện mã độc dựa trên phân tích mẫu, luận văn thu thập được 2736 file dữ liệu mẫu từ hai nguồn online là virustotal và virusshare. Trong 2736 file dữ liệu mẫu này có 1738 file mã độc và 998 file bình thường. Các file bình thường chủ yếu thuộc định dạng .exe, ngoài ra cũng có nhiều định dạng khác như: .doc, .xls, .pdf, .png, … nhằm làm tăng tính đa dạng cho dữ liệu và đây cũng là các định dạng file thường được sử dụng để phát tán mã độc. Các file mã độc thu thập được thuộc các dạng chính như: trojan, worm, virus, adware, spyware, … Các file mã độc sau đó sẽ được gán nhãn là "mã độc", còn các file bình thường sẽ được gán nhãn "bình thường".

3.1.2 Tiền xử lý dữ liệu

Sau khi thu thập được dữ liệu mẫu, luận văn sẽ sử dụng công cụ objdump để đọc từng tập dữ liệu mẫu. Sau đó tiến hành đếm các hàm có trong file txt sinh ra từ objdump để làm đầu vào cho quá trình huấn luyện

Quá trình tiền xử lý này diễn ra nhanh hơn nhiều so với các cách phân tích mã độc khác do chỉ việc dịch ngược để tìm mã assembly của chương trình mà không phải xây dựng môi trường ảo và đợi các file chương trình chạy hết toàn bộ như các cách phân tích động.

3.1.3 Trích chọn đặc trưng dữ liệu

Để mô hình đạt độ chính xác cao, quá trình trích chọn đặc trưng là vô cùng quan trọng. Dữ liệu mã assembly tương ứng của từng chương trình được xử lý để trích xuất ra duy nhất thông tin danh sách mã lệnh được sử dụng trong từng dòng lệnh. Sau khi có được danh sách toàn bộ mã lệnh của chương trình, luận văn tìm ra danh sách các loại mã

lệnh đã xuất hiện trong chương trình và số lần xuất hiện của từng loại mã lệnh.

Có thể dễ thấy, doanh sách các loại mã lệnh và số lần xuất hiện của từng loại mã lệnh của mỗi chương trình là khác nhau, trừ khi 2 chương trình giống nhau hoàn toán. Điều này dẫn đến việc cần có một bộ mã lệnh chuẩn để thống nhất giữa tất cả các chương trình. Bộ mã lệnh chuẩn này cũng chính là bộ đặc trưng dữ liệu cho toàn bộ dữ liệu đã thu thập được. Giá trị của từng đặc trưng chính là số lần xuất hiện của từng mã lệnh trong mỗi chương trình.

Như trình bày ở trên, đa số các chương trình bất kể là mã độc hay bình thường sẽ có 14 loại mã lệnh là phổ biến nhất chiếm hơn 90% tổng số mã lệnh của một chương trình. Để tăng độ chính xác của của mô hình phát hiện, đồ án sử dụng thêm 15 loại mã lệnh khác có tần suất xuất hiện là nhiều nhất và khác 14 loại mã lệnh ở trên để làm danh sách đặc trưng chuẩn cho toàn bộ dữ liệu. Như vậy, luận văn thu được tổng cổng 29 đặc trưng cho mỗi chương trình. Danh sách 29 đặc trưng này là: mov, push, call, pop, cmp, jz, lea, test, jmp, add, jnz, retn, xor, and, bt, fdivp, fild, fstcw, imul, int, nop, pushf, rdtsc, sbb, setb, setle, shld, std và bad.

Tương ứng với mỗi chương trình, luận văn có được nhãn của chương trình là "mã độc" hay "bình thường". Kết hợp với danh sách đặc trưng tương ứng của mỗi chương trình, luận văn thu được file dữ liệu với mỗi dòng tương ứng với một chương trình, 29 giá trị đầu tiên trên mỗi dòng là 29 đặc trưng của chương trình, giá trị cuối cùng là nhãn của chương trình.

3.2 Cài đặt và thử nghiệm

3.2.1 Cài đặt môi trường thực nghiệm

a) Môi trường thực nghiệm

Để thực hiện quá trình thực nghiệm, luận văn sử dụng máy tính có cấu hình tiêu chuẩn như sau để thực hiện quá trình huấn luyện các giải thuật:

- Vi xử lý: Intel i5
- Dung lượng Ram: 4GB
- Hệ điều hành: Ubuntu Desktop 18.04 phiên bản x64 (64 bits)

Ngoài ra, luận văn sử dụng ngôn ngữ lập trình Python 3 và các thư viện Numpy,

Pandas và scikit-learn để cài đặt và huấn luyện các môi.

b) Cài đặt thực nghiệm

Với toàn bộ dữ liệu đã thu thập và trích xuất đặc trưng ở trên, luận văn chia dữ liệu thành 2 tập con là: tập dữ liệu huấn luyện và tập dữ liệu thử nghiệm cho mục đích huấn luyện và đánh giá mô hình theo tỉ lệ tương ứng là 80% và 20%. Cụ thể dữ liệu trong hai tập được phân bố như bảng sau:

Dữ liệu	Mã độc	Bình thường	Tổng
Huấn luyện	1382	806	2188
Thử nghiệm	356	192	548

Bảng III-1: Phân bố 2 tập dữ liệu huấn luyện và thử nghiệm

Để tăng tốc độ huấn luyện và độ chính xác của mô hình, luận văn sử dụng phương pháp Chính quy hóa (standardisation) như đã trình bày ở chương 2 để chuẩn hoá dữ liệu.

Luận văn sẽ sử dụng nhiều thuật toán khác nhau cho bài toán phát hiện mã độc như: Navie-Baye, SVM, Decision Tree, Random Forest. Với mỗi thuật toán, sẽ cho những kết quả khác nhau. Do đó, mô hình sẽ chạy thực nghiệm với toàn bộ các thuật toán, đánh giá và lựa chọn ra thuật toán có độ chính xác cao nhất phù hợp với bài toán phát hiện mã độc.

3.2.2 Phương pháp đánh giá

Để đánh giá độ chính xác của từng thuật toán học máy, luận văn sử dụng các phương pháp sau:

a) Accuracy

Accuracy là cách đánh giá đơn giản nhất và hay được sử dụng nhất. Phương pháp đánh giá này dựa trên công thức đơn giản là tỉ lệ số mẫu được dự đoán đúng so với tổng số mẫu có trong tập dữ liệu thử nghiệm. Công thức cụ thể như sau:

$$acc = \frac{total(correctly\ samples)}{total(samples)}$$

Phương pháp này không quan tâm đến độ chính xác của từng nhãn mà chỉ quan tâm số mẫu được dự đoán đúng nhãn. Các phương pháp tiếp theo sẽ đánh giá chi tiết hơn

dựa trên kết quả dự đoán của từng nhãn.

b) Precision và Recall

Một cách toán học, Precision và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Trong đó:

- TP: số lượng các bản ghi gán nhãn "bình thường" được phân loại đúng.
- TN: số lượng các bản ghi gán nhãn "mã độc" được phân loại đúng.
- FP: số lượng các bản ghi gán nhãn "mã độc" bị phân loại sai thành "bình thường".
- FN: số lượng các bản ghi gán nhãn "bình thường" bị phân loại sai thành "mã độc".

Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. Recall cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các điểm thực sự positive là thấp.

c) F1 score

F1 Score là trung bình điều hòa giữa precision và recall. Do đó nó đại diện hơn trong việc đánh giá độ chính xác trên đồng thời precision và recall. Công thức tính:

$$\frac{2}{F_1} = \frac{1}{Precision} + \frac{1}{Recall}$$

Hay:

$$F_1 = 2\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2\frac{Recision * Recall}{Precison + Recall}$$

F1 – score có giá trị nằm trong nửa khoảng (0; 1]. F1 càng cao, bộ phân lớp càng tốt.

d) Ma trận nhầm lẫn

Ma trận nhầm lẫn là một ma trận thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một lớp, và được dự đoán là rơi vào một lớp. Ma trận có kích thước mỗi chiều bằng số lượng lớp dữ liệu.

Một ma trận nhầm lẫn gồm 4 chỉ số sau đối với mỗi lớp phân loại:

Để phù hợp, ta sẽ sử dụng lại bài toán phát hiện mã độc để giải thích 4 chỉ số này. Trong bài toán ta có 2 lớp: lớp mã độc được đoán Positive và lớp bình thường được đoán là Negative:

- **TP (True Positive)**: dự đoán đúng là file bình thường
- **TN (True Negative)**: dựa đoán đúng là file mã độc.
- **FP (False Positive)**: Dự đoán mã độc và trên thực tế là bình thường.
- **FN (False Negative)**: Dự đoán bình thường và thực tế mã độc.

3.3 Kết quả đánh giá

3.3.1 Naive Bayes

Thuật toán Naive Bayes cho kết quả phân loại mã độc ở mức khá thấp, đạt 58,76%. Kết quả chi tiết như sau:

Precision	Recall	F1 score	Accuracy
98,96%	45,89%	62,71%	58,76%

Kết quả thực nghiệm của thuật toán Naive Bayes

Nhìn vào thông tin ma trận nhầm lẫn, có thể thấy với nhãn dữ liệu bình thường, thuật toán cho kết quả nhận diện rất tốt khi nhận đúng đến 190 chương trình là chương trình bình thường. Tuy nhiên, với nhãn mã độc, thuật toán lại nhận nhầm tới hơn 60% các chương trình mã độc thành chương trình bình thường và chỉ có 132 chương trình là được nhận đúng.

Điều này có thể được lý giải bởi trong thuật toán Naive Bayes, kết quả được dự đoán từ xác suất xuất hiện độc lập của từng đặc trưng, điều này trong nhiều trường hợp là không đúng vì có nhiều đặc trưng, phân bố trong dữ liệu bình thường và dữ liệu mã độc là gần giống nhau.

3.3.2 Support Vector Machine

Đối với thuật toán SVM, kết quả thu được tuy không cao nhưng vẫn tương đối tốt với độ chính xác 94,89%.

Precision	Recall	F1 score	Accuracy
97,92%	88,68%	93,07%	94,89%

Kết quả thực nghiệm của thuật toán SVM

3.3.3 Decision Tree

Kết quả thu được khi sử dụng thuật toán Decision Tree cũng đạt kết quả xấp xỉ 98%. Kết quả cụ thể như bảng dưới đây:

Precision	Recall	F1 score	Accuracy
98,44%	95,94%	97,17%	97,99%

Kết quả thực nghiệm của thuật toán Decision Tree

Cụ thể hơn, thuật toán nhận chính xác 189 chương trình bình thường và 348 chương trình mã độc. Số chương trình bị nhận nhầm chỉ chiếm khoảng 1-2% ở mỗi nhãn.

3.3.4 Random Forest

Thuật toán Random Forest cho kết quả thực nghiệm là cao nhất với độ chính xác lên đến 99,64% cao hơn cả thuật toán Decision Tree.

Precision	Recall	F1 score	Accuracy
98,96%	100%	99,48%	99,64%

Kết quả thực nghiệm của thuật toán Random Forest

Có thể thấy, thuật toán này không nhận nhầm bất kì chương trình mã độc nào thành chương trình bình thường. Ngược lại, chỉ có 2 chương trình bình thường bị nhận nhầm thành chương trình mã độc. Điều này hoàn toàn dễ hiểu vì Random Forest có thể được coi là một thuật toán nâng cấp của thuật toán Decision Tree, trong khi thuật toán Decision Tree cho kết quả đã rất cao.

3.4 Nhận xét

Phương pháp	F1 score	Accuracy
Naive Bayes	62,71%	58,76%
SVM	93,07%	94,89%

Decision Tree	97,17%	97,99%
Random Forest	99,48%	99,64%

Hình III-12: Tổng kết kết quả của các thuật toán

Qua tất cả các thực nghiệm có thể đưa ra một số nhận xét như sau:

- Thuật toán Naive Bayes cho kết quả tệ nhất, có quá nửa chương trình độc hại bị nhận nhầm thành chương trình bình thường. Điều này là hoàn toàn không thể chấp nhận được đối với một phương pháp phát hiện mã độc. Vì vậy không thể áp dụng phương pháp này vào bài toàn phát hiện mã độc.
- Thuật toán SVM cho kết quả khá tốt với độ chính xác xấp xỉ 95%. Tuy nhiên so với các phương pháp khác, độ chính xác này chưa đạt kỳ vọng phát hiện mã độc khi còn nhiều chương trình độc hại bị nhận nhầm thành chương trình bình thường. Thêm vào đó thuật toán SVM có thời gian chạy lâu hơn các phương pháp khác.
- Thuật toán Decision Tree cho độ chính xác vào khoảng 98%, số lượng chương trình mã độc bị nhận nhầm thành chương trình bình thường cũng rất ít chỉ khoảng 1-2%.
- Được coi như một thuật toán nâng cấp hơn Decision Tree, Random Forest đã thể hiện được ưu điểm của mình khi cho kết quả là tốt nhất với hơn 99%. Với thuật toán này, trong thực nghiệm không có một chương trình độc hại nào bị nhận nhầm thành chương trình bình thường. Chỉ có khoáng 1% chương trình bình thường bị nhận nhầm thành chương trình mã độc, nhưng đây là điều chấp nhận được. Đây cũng chính là thuật toán luận văn khuyến nghị xử dụng trong bài toán phát hiện mã độc.

3.5 Kết luận chương

Như vậy, chương III đã trình bày quá trình thử nghiệm, bao gồm các bước thực hiện, kết quả thực nghiệm, đánh giá và nhận xét về các kỹ thuật học máy sử dụng trong phát hiện mã độc. Luận văn cũng dừng lại ở việc thực hiện với các file thực thi. Trong 5 thuật toán đã đề cập ở trên, luận văn kiến nghị sử dụng Random Forest vào bài toán phát hiện mã độc thực tế, với độ chính xác 99.64%.

KẾT LUẬN

Kết quả đạt được:

Luận văn đã nghiên cứu về phương pháp phát hiện mã độc dựa trên phân tích mẫu và đã đạt được một số kết quả sau:

Trình bày các kiến thức khái quát về mã độc như khái niệm, các dạng mã độc, lịch sử phát triển của mã độc. Ngoài ra, đồ án cũng giới thiệu một số kỹ thuật phát hiện mã độc dưa trên chữ kí và dựa trên hành vi.

Đưa ra khái niệm về học máy, các phương pháp học máy cũng như một số kỹ thuật học máy phổ biến hiện nay là Navie Bayes, Support Vector Machine, Decision Tree và Random Forest. Cũng với đó là giới thiệu một số phương pháp trích chọn đặc trưng cho bài toán phát hiện mã độc.

Trình bày kỹ thuật trích xuất và phân tích mã lệnh của mã độc. Từ đó xây dựng mô hình phát hiện mã độc dựa trên phân tích mẫu.

· Thu thập dữ liệu và xây dựng môi trường thực nghiệm, đánh giá kết quả phương pháp phát hiện mã đọc dựa trên phân tích mẫu.

Hướng phát triển trong tương lai:

Trên cơ sở các kiến thức tìm hiểu được và các kinh nghiệm cũng như kết quả trong quá trình thử nghiệm, luận văn có thể được cải thiện và nâng cao theo hướng:

· Dữ liệu hiện tại thu được cho quá trình thực nghiệm còn nhỏ. Luận văn sẽ thu thập thêm dữ liệu với kích thước lớn hơn, đa dạng về loại mã độc và các định dạng file chạy trên nhiều môi trường khác nhau.

· Chuyển từ mô hình phân loại 2 lớp cho bài toán phát hiện mã độc sang mô hình nhận nhiện nhiều nhãn mã độc khác nhau. Thử nghiệm một số mô hình học sâu để đạt kết quả cao hơn.

· Nghiên cứu phương pháp trích chọn đặc trưng mới dựa trên mã lệnh của mã độc.