ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VÕ VĂN TRƯỜNG

NGHIÊN CỨU VÀ ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO BÀI TOÁN PHÁT HIỆN MÃ ĐỘC

LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM

Hà Nội - 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VÕ VĂN TRƯỜNG

NGHIÊN CỨU VÀ ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO BÀI TOÁN PHÁT HIỆN MÃ ĐỘC

Chuyên ngành: Kỹ thuật phần mềm

Mã số: 60480103

LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN VĂN VINH

Hà Nội - 2016

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả nghiên cứu trong luận văn này là sản phẩm của cá nhân tôi dưới sự hướng dẫn của thầy giáo TS. Nguyễn Văn Vinh. Các số liệu, kết quả được công bố là hoàn toàn trung thực. Những điều được trình bày trong toàn bộ luận văn này là những gì do tôi tự nghiên cứu hoặc là được tổng hợp từ nhiều nguồn tài liệu khác nhau. Các tài liệu tham khảo có xuất xứ rõ ràng và được trích dẫn đầy đủ, hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm trước lời cam đoan của mình.

Hà Nội, ngày 02 tháng 12 năm 2016

Người cam đoan

Võ Văn Trường

LỜI CẢM ƠN

Lời đầu tiên tôi xin được gửi lời biết ơn sâu sắc tới thầy giáo TS. Nguyễn Văn Vinh – Bộ môn Khoa học máy tính – Khoa Công nghệ thông tin – Đại học Công Nghệ - Đại học Quốc gia Hà Nội người thầy đã luôn tận tình chỉ bảo, giúp đỡ và hướng dẫn tôi trong suốt quá trình nghiên cứu luận văn này.

Tôi xin chân thành cám ơn các thầy, cô giáo trong Khoa Công nghệ thông tin – Đại học Công Nghệ - Đại học Quốc gia Hà Nội đã luôn tận tâm truyền dạy cho tôi những kiến thức bổ ích trong thời gian tôi tham gia học tập và nghiên cứu tại nhà trường.

Tôi cũng xin gửi lời cám ơn tới Ban lãnh đạo và các đồng nghiệp Bộ môn An toàn Hệ thống thông tin – Trường Đại học Công Nghệ thông tin và Truyền thông – Thái Nguyên nơi tôi công tác đã tạo điều kiện giúp đỡ tôi trong quá trình học tập.

Học viên

Võ Văn Trường

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
MỤC LỤC iii
DANH MỤC CÁC KÍ HIỆU VIẾT TẮT v
DANH MỤC CÁC BẢNG BIỂU vi
DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ vii
MỞ ĐẦU 1
CHƯƠNG 1 TỔNG QUAN VỀ MÃ ĐỘC 3
1.1. Giới thiệu về mã độc máy tính 3
1.2. Phân loại mã độc 3
1.2.1. Virus máy tính 4
1.2.1.1. Phân loại virus dựa vào các hình thức lây nhiễm: 6
1.2.1.2. Phân loại virus dựa trên các chiến lược ẩn náu: 11
1.2.2. Logic Bomb 15
1.2.3. Trojan Horse: 15
1.2.4. Back Door 16
1.2.5. Sâu máy tính (Worm): 17
1.3. Các kỹ thuật phát hiện mã độc 18
1.3.1. Các kỹ thuật phát hiện dựa trên phân tích tĩnh 18
1.3.1.1. Kỹ thuật dò quét (scanner): 18
1.3.1.2. Kỹ thuật Static Heuristics 19
1.3.1.3. Kỹ thuật kiểm tra sự toàn vẹn (Integrity Checkers) 20
1.3.2. Các kỹ thuật phát hiện dựa trên phân tích động 21
1.3.2.1. Kỹ thuật Behavior Monitors/Blockers 21
1.3.2.2. Kỹ thuật Emulation 22
CHƯƠNG 2 MỘT SỐ THUẬT TOÁN PHÂN LỚP DỮ LIỆU ĐIỂN HÌNH TRONG KỸ THUẬT HỌC MÁY GIÁM SÁT 24
2.1. Thuật toán cây quyết định 24
2.1.1. Giới thiệu thuật toán 24
2.1.2. Xây dựng cây quyết định dựa trên thuật toán ID3 24
2.1.3. Ví dụ minh họa: 27
2.1.4. Nhận xét: 30
2.2. Thuật toán SVM 31
2.2.1. Giới thiệu thuật toán 31
2.2.2. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và không có nhiễu: 32
2.2.3. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và có xảy ra nhiễu: 33
2.2.4. Bài toán tìm siêu phẳng tối ưu cho dữ liệu không tuyến tính: 34
2.2.5. Hàm nhân (Kernel) 35
CHƯƠNG 3 GIẢI PHÁP ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO PHÁT HIỆN MÃ ĐỘC 37
3.1. Tổng quan về phương pháp thực hiện 37
3.2. Tiền xử lý dữ liệu 39
3.2.1. Sử dụng các kỹ thuật phân tích mã độc 39
3.2.2. Phương pháp n-gram 40

3.2.3. Tính tần số xuất hiện (Term Frequency) 40
3.3. Đề xuất giải pháp chọn đặc trưng cho thuật toán phân lớp 41
3.3.1. Mô tả giải pháp 41
3.3.2. Ví dụ: 43
3.4. Xây dựng mô hình dự đoán dựa trên các thuật toán phân lớp 46
CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ 48
4.1. Dữ liệu thực nghiệm 48
4.2. Chương trình thực nghiệm 48
4.3. Đánh giá dựa trên phương pháp ma trận nhầm lẫn 52
4.4. Kết quả thực nghiệm 53
KẾT LUẬN 55
DANH MỤC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN VĂN 56
TÀI LIỆU THAM KHẢO 57

DANH MỤC CÁC KÍ HIỆU VIẾT TẮT

Ký hiệu	Ý nghĩa
API	Application Programming Interface – Giao diện lập trình ứng dụng
DT	Decision Tree – Cây quyết định
FN	False Negative
FP	False Positive
FPR	False positive rate
KNN	K-Nearest Neighbors – K-láng giềng gần nhất
NB	Naive Bayes
PE	Portable Executable
RF	Random Forest – Rừng ngẫu nhiên
SVM	Support Vector Machine – Máy véc tơ hỗ trợ
TF	Term Frequency – Tần số xuất hiện
TN	True Negative
TP	True Positive
TPR	True positive rate

DANH MỤC CÁC BẢNG BIỂU

Bảng 3.1 Mô tả dãy các n-gram byte.....40
Bảng 3.2 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 1.....43
Bảng 3.3 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 2.....43
Bảng 3.4 Kết quả tập dữ liệu lớp 1 sau khi thực hiện sắp xếp 43
Bảng 3.5 Kết quả tập dữ liệu lớp 2 sau khi thực hiện sắp xếp 43
Bảng 3.6 Kết quả tập dữ liệu lớp 1 sau khi thực hiện phân đoạn.....44
Bảng 3.7 Kết quả tập dữ liệu lớp 2 sau khi thực hiện phân đoạn.....44
Bảng 3.8 Minh họa với đặc trưng “BB BB” thuộc lớp 1 sau khi thực hiện phân đoạn 44
Bảng 3.9 Minh họa với đặc trưng “BB BB” thuộc lớp 2 sau khi thực hiện phân đoạn 44
Bảng 3.10 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 1 45
Bảng 3.11 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 2 45
Bảng 3.12 Kết quả độ lệch tần số xuất hiện trên từng đoạn.....46
Bảng 3.13 Kết quả độ lệch tần số xuất hiện trên toàn tập dữ liệu.....46
Bảng 4.1 Tỷ lệ số lượng từng mẫu mã độc tham gia thực nghiệm 48
Bảng 4.2 Thông tin về chương trình thực nghiệm 48
Bảng 4.3 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng tốt nhất 53
Bảng 4.4 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng kém nhất.....54

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

Hình 1.1 Khối khởi động bị lây nhiễm bởi nhiều virus [10] 6
Hình 1.2 virus lây nhiễm phần đầu tệp tin [10] 8
Hình 1.3 virus lây nhiễm phần cuối tệp tin [10] 8
Hình 1.4 Mô tả hoạt động của virus macro [10] 11
Hình 1.5 Mô tả quá trình virus được giải mã [10] 12
Hình 1.6 so sánh 2 kỹ thuật phân tích tĩnh và động [10] 22
Hình 2.1 Biểu diễn giá trị Entropy với tập mẫu nằm trong 2 lớp [18] 26
Hình 2.2. Tập dữ liệu mẫu ví dụ cho xây dựng cây quyết định [18] 27
Hình 2.3 cây được chọn với thuộc tính Outlook là nút gốc [18] 29
Hình 2.4 Cây quyết định được sinh ra trong dữ liệu mẫu ví dụ chơi tennis [18] 30
Hình 2.5 Biểu diễn phân lớp dựa trên thuật toán SVM [8] 31
Hình 2.6 Siêu phẳng phân chia dữ liệu tốt có khoảng cách lề xa nhất [8,17] 32
Hình 2.7 Tìm siêu phẳng tối ưu khi dữ liệu tuyến tính có xảy ra nhiễu 34
Hình 2.8 Mô tả dữ liệu không thể phân lớp tuyến tính [8] 35
Hình 2.9 Dữ liệu được phân lớp bởi hàm nhân (kernel) [8] 36
Hình 3.1 Tổng quan về phương pháp thực hiện 38
Hình 4.1 Biểu đồ mô tả tỷ lệ mẫu mã độc tham gia thực nghiệm 48
Hình 4.2 Chương trình đọc, phân tích cấu trúc các file PE và dịch ngược 49
Hình 4.3 Tập dữ liệu mẫu mã độc ban đầu 49
Hình 4.4 Tập dữ liệu mẫu mã độc sau khi được dịch ngược về mã hex và được gán nhãn phân lớp 50
Hình 4.5 Tập dữ liệu mẫu các file thực thi thông thường trên Windows 50
Hình 4.6 Tập dữ liệu mẫu các file thực thi thông thường sau khi được dịch ngược về mã hex và được gán nhãn phân lớp 51
Hình 4.7 Tập các tần số xuất hiện của dữ liệu huấn luyện 51
Hình 4.8 Chương trình trích chọn đặc trưng và xây dựng mô hình dự đoán mã độc 52
Hình 4.9 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng được đánh giá tốt nhất 53
Hình 4.10 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng bị đánh giá kém nhất 54

MỞ ĐẦU

Ngày nay song song với sự bùng nổ mạnh mẽ của công nghệ thông tin và sự phát triển của Internet toàn cầu là các nguy cơ mất an toàn thông tin đang trở nên trầm trọng và nguy hiểm hơn, trong đó mã độc hại đang là các hiểm họa hàng đầu bởi khả năng có thể lây lan phát tán trên các hệ thống máy tính và thực hiện các hành vi tấn công bất hợp pháp. Mã độc đang ngày càng tiến hóa với những biết thể đa dạng, với những cách thức che dấu ngày càng tinh vi hơn. Có thể nói phát hiện và ngăn chặn mã độc đang là một thách thức được đặt ra trong lĩnh vực An toàn thông tin. Các phương pháp phát hiện mã độc thông thường chủ yếu sử dụng kĩ thuật so sánh mẫu dựa trên cơ sở dữ liệu mã độc được xây dựng và định nghĩa từ trước, tuy nhiên phương pháp này bộc lộ nhiều nhược điểm đó là không có khả năng phát hiện ra các mẫu mã độc mới, số lượng dữ liệu mã độc ngày càng gia tăng làm cho cơ sở dữ liệu mẫu trở nên ngày càng lớn. Hiện nay hướng nghiên cứu dựa vào các mô hình học máy để phân loại và phát hiện mã độc đang tỏ ra là phương pháp tiềm năng và hiệu quả khi có thể cải thiện được các nhược điểm đã nêu ở trên so với phương pháp truyền thống. Tuy nhiên, một trong những vấn đề được quan tâm là làm sao để xây dựng được mô hình học máy tốt nhất đạt hiệu quả chính xác và hiệu suất cao. Trong đó một yếu tố quan trọng được xem là quyết định chính là giải pháp trích chọn đặc trưng. Trong các phần nghiên cứu của luận văn này tôi trình bày về phương pháp ứng dụng học máy vào xây dựng các mô hình phát hiện mã độc trong đó các thực nghiệm dựa trên phương pháp phân tích tĩnh mã độc, tiền xử lý dữ liệu bằng kỹ thuật dịch ngược đưa các file dữ liệu mẫu về dạng mã hex và thực hiện khai phá dữ liệu text sử dụng các mã n-gram byte là các đặc trưng ban đầu .Sau đó các dữ liệu đặc trưng này sẽ được trích chọn ra một bộ dữ liệu đặc trưng tốt nhất để xây dựng mô hình trên cơ sở giải pháp trích chọn đặc trưng mà trong luận văn này tôi đã tập trung nghiên cứu và đề xuất. Các kết quả của luận văn được thực nghiệm trên khoảng 4698 file mẫu thực thi trên nền Windows trong đó 2373 file mã thông thường và 2325 file mẫu mã độc với nhiều thể loại đa dạng như Backdoor, Virus, Trojan, Worm...

Nội dung luận văn được chia ra làm 4 phần như sau:

Chương 1: Chương này nghiên cứu tổng quan về mã độc trình bày các kiến thức chung nhất liên quan đến mã độc, phân loại mã độc cũng như các kỹ thuật phân tích và phát hiện mã độc hiện nay.

Chương 2: Nghiên cứu một số thuật toán phân lớp dữ liệu điển hình trong kỹ thuật

học máy giám sát trong đó 2 thuật toán phân lớp dữ liệu tiêu biểu được trình bày là cây quyết định (DT) và máy véc tơ hỗ trợ (SVM)

Chương 3: Chương này trình bày giải pháp ứng dụng kỹ thuật học máy vào phát hiện mã độc bao gồm quá trình tiền xử lý dữ liệu, xây dựng các mô hình học máy để phát hiện mã độc. Trong đó trọng tâm là trình bày một đề xuất giải pháp chọn đặc trưng cải thiện và nâng cao hiệu quả cho các thuật toán phân lớp đối với bài toán phát hiện mã độc.

Chương 4: Trình bày về quá trình thực nghiệm và đánh giá, các kết quả được thực nghiệm và so sánh trên các tập đặc trưng được chọn dựa trên giải pháp đã đề xuất, các kết quả cũng được so sánh giữa 2 thuật toán phân lớp đã trình bày là cây quyết định và máy véc tơ hỗ trợ.

CHƯƠNG 1 TỔNG QUAN VỀ MÃ ĐỘC

1.1. Giới thiệu về mã độc máy tính

Mã độc hay phần mềm độc hại là các chương trình máy tính có chứa bên trong nó nội dung các mã độc hại được tạo ra với mục đích thực hiện các hành vi bất hợp pháp như: truy cập trái phép, đánh cắp thông tin người dùng, lây lan thư rác, thậm chí thực hiện các hành vi tống tiền (ransomware), tấn công và gây tổn thương cho các hệ thống máy tính… nhằm chuộc lợi cá nhân , hoặc các lợi ích về kinh tế, chính trị hay đơn giản chúng có khi được tạo ra chỉ là một trò đùa ác ý nào đó. Bất kỳ một phần mềm nào là lý do làm tổn thương, phá vỡ đến tính bí mật, tính toàn vẹn và tính sẵn sàng của dữ liệu người dùng, máy tính hoặc môi trường mạng đều có thể được xem như các mã độc. Ngày nay ,cùng với sự phát triển của công nghệ thông tin các ứng dụng máy tính, các phần mềm hệ thống không ngừng thay đổi và phát triển mạnh mẽ, mã độc ban đầu từ chỗ chỉ là những chương trình nhỏ có khả năng tự sao chép đến nay mã độc và các biến thể của nó đang ngày càng trở nên đa dạng, tinh vi mức độ nguy hiểm ngày càng gia tăng đang trở thành một nguy cơ đe dọa trực tiếp đến an toàn, an ninh thông tin.

1.2. Phân loại mã độc

Ban đầu các phần mềm độc hại được tạo ra bằng cách sống kí sinh và lây nhiễm trên các vật chủ là các chương trình có chứa các nội dung thực thi. Các đoạn mã máy nhiễm độc được lây lan tồn tại trong một vài chương trình ứng dụng, các chương trình tiện ích hoặc các hệ thống máy tính thậm chí ngay chính trên các mã được dùng để khởi động máy tính. Các phần mềm độc hại này được xác định và phân loại bởi mục đích độc hại hay các hành vi của nó, có thể kể tên như: Virus, Trojan, Logic Bombs, BackDoors…Các dạng chương trình độc hại kiểu khác là các chương trình mã độc mà tự chúng có khả năng thực thi một các độc lập trên các phần mềm hệ thống mà không cần kí sinh trên các vật chủ là các chương trình ứng dụng hay tệp tin. Các mã độc dạng này thường có khả năng tự nhân bản chúng lợi dụng các lỗ hổng bảo mật để tấn công hay thực hiện các hành vi độc hại, đại diện cho các dạng mã độc này là các sâu máy tính (worm).

Một cách khác để phân biệt các loại mã độc khác nhau là dựa trên mục đích và các hành vi của chúng. Ngày nay mã độc đang ngành càng phát triển và tiến hóa không ngừng trong đó nổi lên là các dạng mã độc mà có khả năng ngụy trang và lén lút thực hiện các hành vi đánh cắp thông tin, thực hiện nghe lén hay thâm nhập như là các phần

mềm gián điệp hay các chương trình khai thác lỗ hổng bảo mật trên máy tính người dùng trong thời gian dài. Một số loại mã độc khác chủ yếu được thực thi mà hành vi của nó tập trung vào mục đích lây lan và phá hoại gây nhiễm cho hệ thống máy tính.

Như vậy có thể nói phần mềm độc hại có thể được tạm chia thành các loại khác nhau theo phương thức và hoạt động của chúng, Các phần mềm chống virus đôi khi không quan tâm đến tên của nó, có khả năng phát hiện ra các kiểu mã độc như sau:

Có 3 đặc điểm thường liên quan với các loại phần mềm độc hại là:

- Mã độc tự nhân bản và nỗ lực lây nhiễm bằng việc tạo các bản sao chép mới hoặc các thể hiện của chính nó. Mã độc cũng có thể được lây nhiễm một cách bị động thông qua thao tác vô ý của người dùng tạo ra các sự kiện kích hoạt hay sao chép nhưng điều này không được coi là tự nhân bản.
- Sự tăng trưởng về số lượng của phần mềm độc hại mô tả sự thay đổi tổng thể trong đó có số lượng lớn các trường hợp là do tự nhân bản, Những mã độc mà không tự nhân bản sẽ luôn luôn có một sự tăng trưởng bằng không nhưng mã độc với một sự tăng trưởng bằng không lại có thể là những mã độc tự nhân bản.
- Phần mềm độc hại ký sinh yêu cầu một số chương trình mã thực thi khác để tồn tại. Thực thi trong ngữ cảnh này nên được hiểu chung là bao gồm bất cứ thứ gì có thể thực thi được, như là các khối khởi động trên đĩa, mã nhị phân trong các ứng dụng và các mã biên dịch. Nó cũng bao gồm các mã nguồn như là các ngôn ngữ kịch bản, hay là các mã yêu cầu biên dịch trước khi chúng được thực thi.

1.2.1. Virus máy tính

Virus là một loại mã độc có các đặc điểm như tự nhân bản, chúng ký sinh trên các vật chủ, virus là một chương trình độc hại mà khi được thực thi nó sẽ cố gắng sao chép chính nó vào bên trong một mã thực thi khác. Khi thành công thì mã chương trình được gọi là mã bị lây nhiễm. Các mã lây nhiễm khi chạy có thể tiếp tục lây nhiễm sang các mã mới. Sự tự sao chép này tồn tại trong các mã thực thi là một đặc tính xác định quan trọng của một virus. Khi virus được lây nhiễm nó có thể thực hiện một loạt các hành vi như thay đổi, xóa, hay sao chép các tệp tin cũng như phát tán chúng trên các hệ thống máy tính.

Hiểu theo cách truyền thống thì Virus có thể lây lan bên trong một máy tính duy nhất hoặc có thể di chuyển từ một máy tính đến máy tính khác bằng việc sử dụng các phương tiện vận chuyển đa phương tiện của con người như là đĩa mềm, CD-ROM,

DVD-ROM, hoặc USB,… Nói cách khác Virus không lây lan thông qua mạng máy tính thay vào đó mạng máy tính là một miền hoạt động của các sâu máy tính. Tuy nhiên Virus hiểu theo cách truyền thống đôi khi được dùng để chỉ bất kỳ phần mềm độc hại nào có khả năng tự sao chép.

Virus có thể được bắt gặp trong các giai đoạn khác nhau của sự tự nhân bản. Một mầm mống nảy sinh là hình thức ban đầu của Virus trước khi diễn ra bất kỳ một sự tự sao chép nào, một Virus mà thất bại trong việc tự tái tạo lại được gọi là một dự định, điều này có thể xảy ra như là kết quả của lỗi trong các virus hoặc sự gặp phải một phiên bản bất ngờ của hệ điều hành. Một virus có thể trú ẩn, nơi mà nó hiện diện nhưng chưa lây nhiễm hay làm bất cứ điều gì ví dụ một cửa sổ có chứa virus có thể ở trên một máy chủ tập tin chạy trên hệ điều hành Unix và không có hiệu lực ở đó, nhưng có thể được xuất và di chuyển sang các máy Windows về cơ bản một Virus được trải qua 4 giai đoạn, giai đoạn 1 là trú ẩn như đã trình bày ở trên, giai đoạn 2 virus thực hiện lây nhiễm bằng việc việc sao chép chính nó vào các tệp tin hay chương trình máy tính khác, giai đoạn 3 virus được kích hoạt để chuẩn bị chạy các đoạn mã độc, giai đoạn 4 virus thực thi các chức năng của nó theo đúng kịch bản mà kẻ đã tạo ra nó.

Một virus máy tính gồm 3 thành phần chính như sau:

Cơ chế lây nhiễm: Làm thế nào để virus có thể lây lan bằng cách sửa đổi mã khác để chứa một sao chép của virus, các cách chính xác thông qua đó một loại virus lây lan được gọi là vector lây nhiễm của nó. Đây không phải là duy nhất, một loại virus lây nhiễm theo nhiều cách được gọi là đa phương.

Kích hoạt: Các cách quyết định xem có hay không để mở một payload

Payload: là mã lệnh hay cái mà virus thực hiện bên cạnh việc lây nhiễm

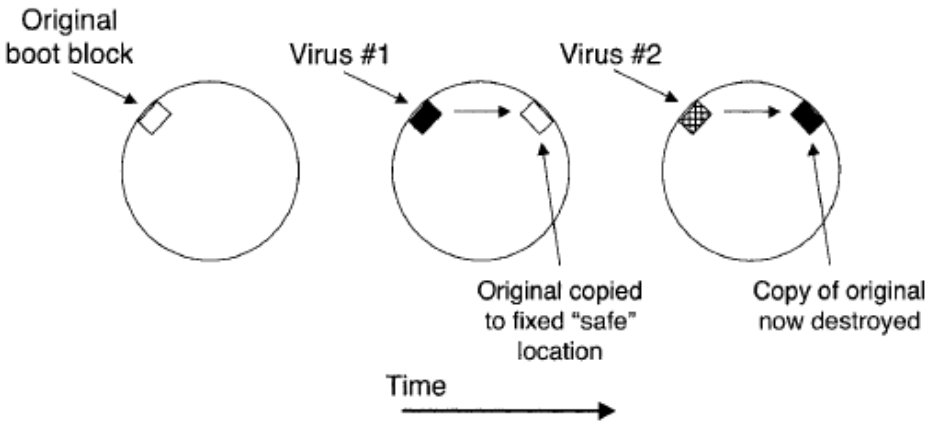
Ngoại trừ cơ chế lây nhiễm, 2 thành phần còn lại có thể tùy chỉnh do vậy lây nhiễm là một trong những chìa khóa định nghĩa các đặc điểm của một virus. Trong sự thiếu vắng tính chất lây nhiễm nếu mã độc hại chỉ chứa thành phần kích hoạt và payload trường hợp này được gọi là mã độc logic bomb.

Virus có thể được phân loại bằng nhiều cách, dựa vào mục tiêu cố gắng lây nhiễm của virus, và dựa vào các phương thức mà virus sử dụng để che giấu bản thân nó với các hệ thống phát hiện và các phần mềm chống virus.

1.2.1.1. Phân loại virus dựa vào các hình thức lây nhiễm:

1.2.1.1.1. Virus lây nhiễm Boot-Sector

Virus lây nhiễm Boot-Sector là một virus lây nhiễm bằng việc sao chép chính nó đến khối khởi động bằng cách như vậy sau khi hệ điều hành máy tính tiến hành khởi động nó sẽ thực thi các đoạn mã virus bị chèn vào chứ không phải là các đoạn mã khởi động thông thường. Virus có thể sao chép nội dung của khối khởi động cũ ở nơi khác trên đĩa đầu tiên hoặc nơi cố định an toàn trên ổ đĩa để virus có thể chuyển điều khiển đến nó sau khi hoàn thành xử lý tiến trình khởi động.



Hình 1.1 Khối khởi động bị lây nhiễm bởi nhiều virus [10]

Phương pháp này sẽ gặp vấn đề nếu như máy tính bị nhiễm nhiều lần bởi những loại virus khác nhau khi chúng cùng sử dụng một địa điểm an toàn để lưu trữ đoạn mã khởi động, do ở lần tiếp theo virus sẽ tiếp tục sao chép đoạn mã virus trước đó đến vùng an toàn vô tình ghi đè lên đoạn mã nạp chương trình thực sự do vậy bị xóa và chúng sau khi thực hiện các chức năng của mình không có cách nào có thể quay trờ lại khởi động máy tính, đây là một tổn hại không chủ ý được gây ra bởi virus, ngày nay virus boot không còn được thấy xuất hiện nhiều do tốc độ lây lan chậm và khả năng tiến hóa không cao của nó.

1.2.1.1.2. Virus lây nhiễm tập tin:

Virus lây nhiễm tập tin hệ điều hành có một khái niệm về tập tin được thực thi, trong một ý nghĩa rộng hơn các tập tin thực thi cũng có thể bao gồm các tập tin mà có thể chạy bởi các dòng lệnh người sử dụng như "shell". Một tập tin lây nhiễm là một virus lây lan là những tập tin được hệ điều hành hay "shell" xem là để thực thi nó có thể bao gồm các tập tin "batch" và mã kịch bản "shell" nhưng các mã nhị phân thực thi là những mã thực thi phổ biến nhất.

Có 2 vấn đề cần quan tâm chính trong lây nhiễm tập tin đó là:

1. Virus thì nằm ở đâu ?
2. Virus được thực thi như thế nào khi mà các tập tin lây nhiễm chạy ?

Các kỹ thuật mà một virus thực hiện lây nhiễm:

Để tiến hành lây nhiễm virus thực hiện ác công việc như sau:

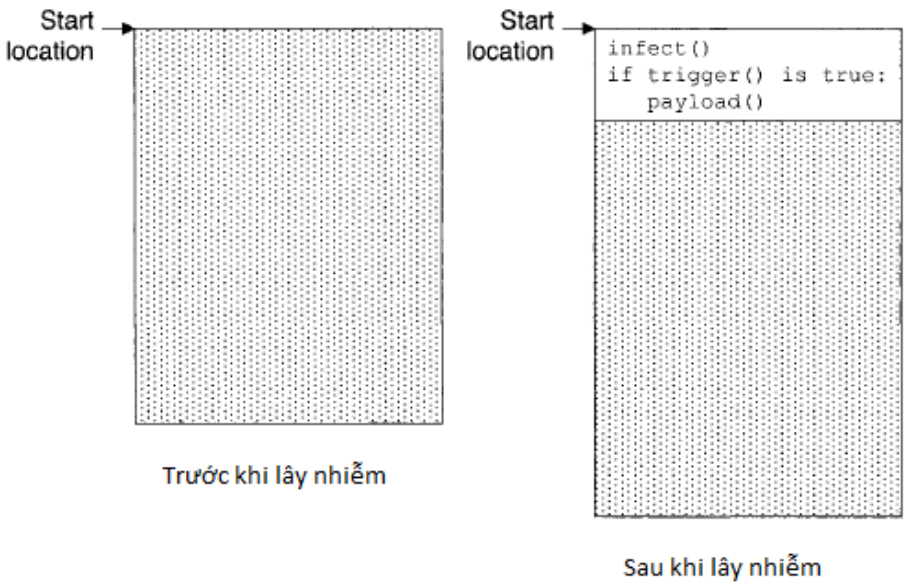
- ✓ Tìm kiếm đối tượng tệp thực thi để lây nhiễm
- ✓ Nạp các mã độc lây nhiễm vào các tập tin tìm được
- ✓ Lưu trú và đảm bảo nó tồn tại duy nhất trong tập tin bị lây nhiễm
- ✓ Tiếp tục tìm kiếm các tệp tin khác để tiến hành lẫy nhiễm mã độc

Mỗi một loại virus sẽ có những cách thức để thực hiện lây nhiễm khác nhau song chúng đều được trải qua các nguyên lý bao gồm 3 giai đoạn là:

- ✓ Lây nhiễm và chiếm quyền điều khiển,
- ✓ Chạy các chức năng của nó là các đoạn mã độc được gắn vào tệp tin lây nhiễm
- ✓ Trao trả quyền điều khiển cho tệp và đảm bảo tính toàn vẹn ban đầu của dữ liệu.

Thông thường virus được tiến hành đặt ở hai nơi, nơi bắt đầu một tệp tin và nơi kết thúc hoặc một vị trí nào đó trong tệp tin, các kĩ thuật thực hiện của virus được trình bày như sau:

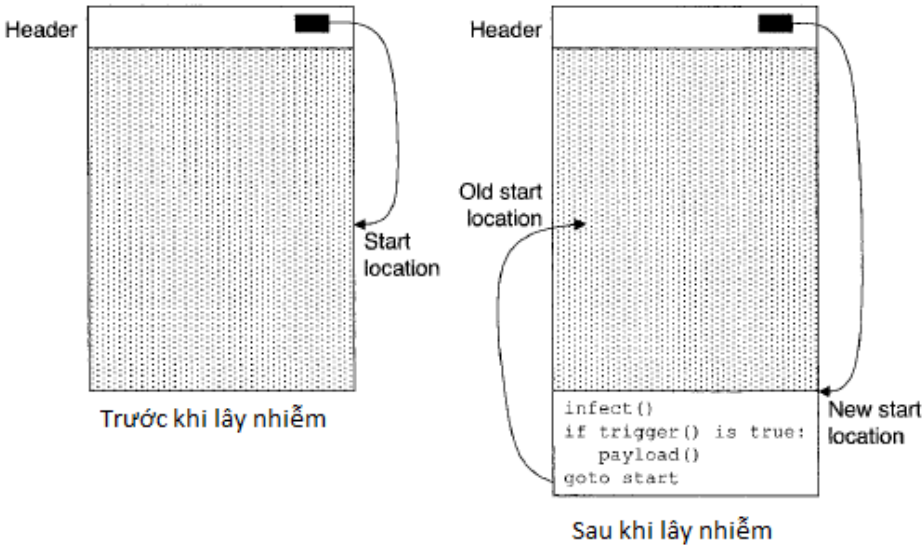
Lây nhiễm phần đầu của tập tin: các tập tin đơn giản thường có định dạng như .EXE, .COM, MS-DOS toàn bộ các tập tin được kết hợp bởi phần mã và phần dữ liệu. Khi được thực thi toàn bộ tập tin sẽ được nạp lên bộ nhớ và thực thi sẽ được bắt đầu bởi lệnh nhảy đến phần đầu tiên nạp tập tin đó. Trong trường hợp này virus được đặt ở vị trí bắt đầu của tập tin nơi mà các tập tin nhận quyền điều khiển đầu tiên khi các tập tin bị lây nhiễm chạy nó sẽ thực thi các đoạn mã độc đầu tiên.



Hình 1.2 virus lây nhiễm phần đầu tệp tin [10]

Lây nhiễm phần cuối của tập tin: gắn thêm các mã vào phần cuối của một tập tin thì cực kì đơn giản, một virus đặt ở vị trí kết thúc một tập tin bằng cách nào nó có thể chiếm đoạt quyền điều khiển, có 2 phương pháp cơ bản như sau:

Các mã lệnh chương trình gốc có thể được lưu lại, và được thay thế bởi một lệnh nhảy đến các mã của virus, sau khi thực thi xong virus sẽ sử dụng quyền điều khiển trỏ đến các mã ban đầu mà nó đã lây nhiễm. Virus có thể sẽ cố gắng chạy trực tiếp các mã lệnh chương trình gốc ngay tại khu vực nó đã lưu hoặc virus có thể phục hồi các mã lệnh gốc này về trạng thái ban đầu và chạy nó.



Hình 1.3 virus lây nhiễm phần cuối tệp tin [10]

Nhiều các tập tin thực thi có định dạng chỉ rõ vị trí khởi động trong phần đầu của tập

tin. Virus có thể thay đổi vị trí khởi động này trỏ đến phần mã của nó và thực thi sau đó lệnh nhảy trở lại vị trí khởi động gốc khi chúng hoàn thành.

Ghi đè vào tập tin: nhằm tránh sự thay đổi về kích thước của một tập tin một vài virus thực hiện ghi đè các phần mã của nó lên các phần mã của tập tin gốc. Việc ghi đè một cách không đúng đắn sẽ gây ra sự thay đổi cũng như phá vỡ cấu trúc của tệp tin ban đầu điều này vô tình làm cho các chương trình chống virus dễ dàng phát hiện ra chúng, có một vài cách mà mức độ rủi ro và phức tạp khác nhau: phương pháp đầu tiên virus hoạt động giống như Boot-virus nó thực hiện lưu các phần nội dung của mã chương trình ban đầu ở một nơi nào đó và sau đó thực hiện ghi đè lên một phần của tập tin, cách thứ 2 là virus có thể thực hiện thao tác nén các dữ liệu và mã chương trình ban đầu sau đó tiến hành ghi đè lên vị trí trống được tạo ra và khi virus thực hiện xong chức năng của nó thì sẽ thực hiện giải nén phần mã gốc để phục hồi lại tập tin ban đầu.

Chèn vào tập tin: dựa vào cấu trúc của các tệp tin khác nhau virus sử dụng kỹ thuật chèn mã của nó vào các vùng còn trống không được sử dụng, các vị trí này thường là các vị trí bên trong của tệp tin, về cơ bản nó cũng có thể chiếm quyền điều khiển để khi tệp tin khởi động sẽ trỏ về đúng vị trí các đoạn mã virus đã chèn, sau khi thực hiện xong các công việc của mình, virus sẽ trỏ lại lệnh thực thi đúng như vị trí ban đầu của tệp tin.

Không nằm trong tập tin: các virus lây nhiễm dạng này hay còn gọi là các virus đồng dạng không giống như các cách lây nhiễm như bên trên loại virus này không làm thay đổi mã của chương trình gốc chúng chủ yếu tồn tại dưới hình thức ngụy trang để có những đặc điểm giống như các tệp tin thực thi là mục tiêu của chúng, chúng tỏ ra là một tệp tin bình thường nào đó nằm trên hệ điều hành và bằng cách chiếm quyền thực thi trước các chương trình gốc dựa trên các cơ chế về quyền ưu tiên trong hệ điều hành hay các hoạt động tìm kiếm shell:

Thông thường các chương trình thực thi sẽ nằm trên các đường dẫn tìm kiếm của hệ điều hành, lợi dụng các cơ chế tìm kiếm ưu tiên virus có thể sao chép chính nó vào các đường dẫn này và tạo tên giống với tên của chương trình mà nó muốn lây nhiễm, sau đó chiếm quyền ưu tiên để hệ điều hành thực thi nó trước khi chương trình gốc thực sự được thực thi.

Các tệp tin mục tiêu cũng có thể bị thay đổi tên và các virus đồng dạng có thể sẽ lấy lại tên gốc của tệp tin mục tiêu đó.

Các định dạng tập tin ELF thường được sử dụng trên các hệ thống Unix có một trình

biên dịch quy định trong mỗi phần đầu của tập tin thực thi thông thường được trỏ đến các liên kết trong thời gian chạy của hệ thống. Một virus đồng dạng có thể thay thế các liên kết thời gian chạy này, đó là lý do mà tất cả các tập tin thực thi có thể bị lây nhiễm cùng một lúc.

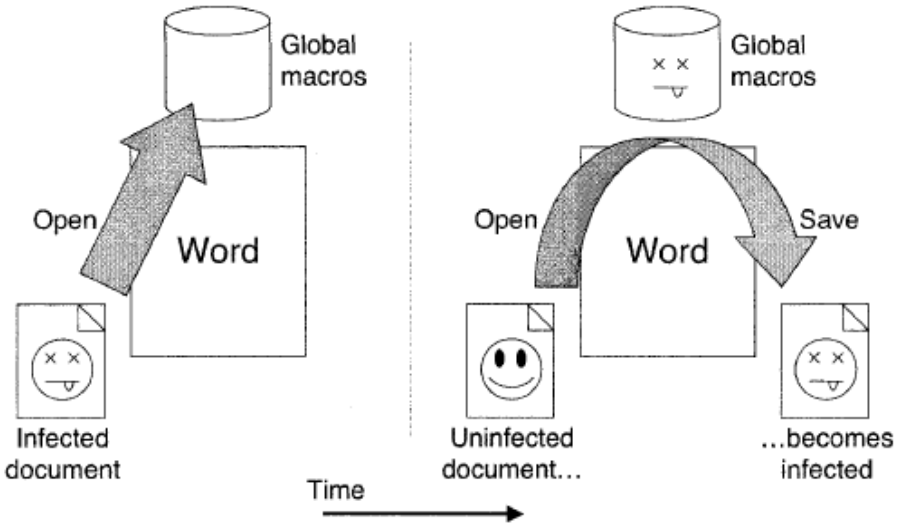
Virus đồng dạng thậm chí cũng có khả năng hoạt động dựa trên các môi trường giao diện đồ họa, một biểu tượng của ứng dụng mục tiêu có thể bị che dấu bởi một biểu tượng của virus đồng dạng. Khi một người sử dụng click chọn vào biểu tượng mà họ nghĩ là biểu tượng của ứng dụng nhưng thay vào đó một virus đồng dạng sẽ thực thi.

Đối với hệ điều hành Windows virus sẽ tự ghi các nội dung của nó vào một số địa chỉ trong Registry để những lần khởi động tiếp theo nó sẽ được gọi đến và kích hoạt trở lại. Đây là một hình thức sử dụng tìm kiếm shell đối với các tập tin thực thi sau khi chúng được lây nhiễm vào mày tính cớ chế này cho phép virus được thực thi và lây lan các hoạt động của chúng mỗi lần máy tính bật hay khởi động lại.

Kiểm tra tính duy nhất: để đảm bảo tăng tốc độ lây nhiễm thì virus cần phải tránh việc lây nhiễm lặp lại trên các tập tin hay vùng nhớ đã chứa nó vì thế chúng cần phải có cơ chế đảm bảo sự tồn tại duy nhất trên bộ nhớ cũng như trong các tệp tin lây nhiễm, một số phương pháp để thực hiện điều này như sau: trước khi tiến hành lây nhiễm nó thực hiện kiểm tra một đoạn mã trên vùng nhớ và so sánh với đoạn mã virus nếu có sự khác biệt thì nó sẽ tiến hành lây nhiễm lên vùng nhớ này. Hoặc virus có thể xây dựng thêm chức năng cho phép kiểm tra để trước khi lây nhiễm chức năng này sẽ được gọi quyết định có hay không sự tồn tại của nó trong bộ nhớ được dựa vào các giá trị trả về trong thanh ghi hoặc các biến hệ thống sau khi chức năng được gọi.

1.2.1.1.3. Virus Macro:

Một số ứng dụng cho phép chứa các tập tin dữ liệu, như các trình xử lý văn bản cho phép nhúng các hàm macro bên trong chúng, Macro là một đoạn mã nhỏ được viết bởi một ngôn ngữ chuẩn được biên dịch bởi ứng dụng, ngôn ngữ này cung cấp đầy đủ các hàm để viết một virus. Do vậy virus Macro chiếm được nhiều lợi thế hơn là virus lây nhiễm tập tin.



Hình 1.4 Mô tả hoạt động của virus macro [10]

Khi một tài liệu chứa một macro được nạp bởi ứng dụng, macro có thể là lý do gây ra việc chạy tự động, cho phép kiểm soát điều khiển đến các virus macro. Một vài ứng dụng có thể tạo ra các cảnh báo người dùng về sự xuất hiện của marco trong văn bản tuy nhiên những cảnh báo này thường bị bỏ qua một cách dễ dàng. Trong hệ điều hành Windows thì Virus macro chủ yếu là các đoạn mã được viết bởi ngôn ngữ lập trình Visual Basic đây là một ngôn ngữ được sử dụng trong phần mềm thuộc bộ Microsoft Office như Word, PowerPoint, Excel.

Một tài liệu bị lây nhiễm thường bởi 2 kiểu macro với các tính chất:

AutoOpen: là các đoạn mã macro sẽ tự động chạy khi mà một tập tin được mở các macro này sẽ chiếm được quyền kiểm soát tập tin bị lây nhiễm.

FileSaveAs: là các đoạn mã lưu tập tin trong macro sẽ chạy khi mà tác vụ "File save as" trong trình đơn được lựa chọn. Nói cách khác đoạn mã này có thể được sử dụng để lây nhiễm đến bất kì tài liệu nào chưa bị nhiễm virus mà được lưu bởi người dùng.

Đứng trên quan điểm kỹ thuật, ngôn ngữ cho macro thì dễ dàng được sử dụng hơn các ngôn ngữ lập trình bậc thấp vì vậy lợi thế của macro làm giảm các rào cản tạo ra một virus.

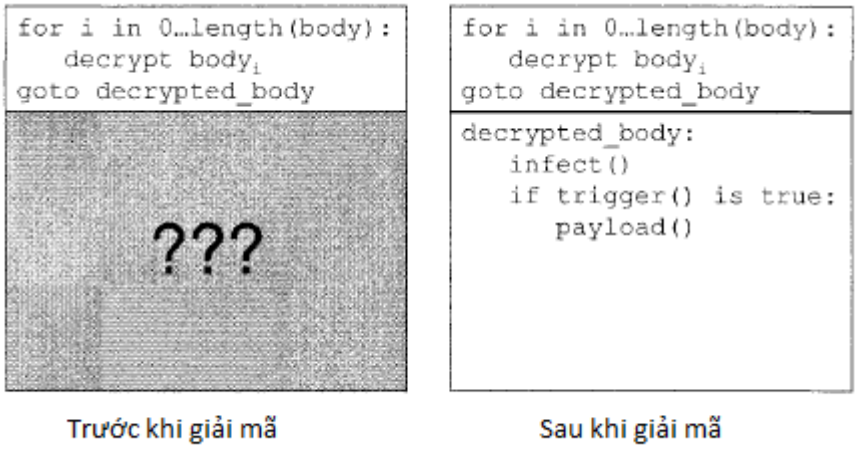
1.2.1.2. Phân loại virus dựa trên các chiến lược ẩn náu:

Một cách khác để phân loại virus là dựa vào các cách làm thế nào để chúng có thể ẩn náu và qua mặt được người dùng cũng như các phầm mềm diệt virus.

1.2.1.2.1. Virus mã hóa

Đối với một virus được mã hóa ý tưởng là phần thân của virus bao gồm phần lây

nhiễm, kích hoạt và payload sẽ được mã hóa thông qua một vài cách thức, mục đích là gây ra sự khó khăn để có thể phát hiện ra chúng, khi phần thân của virus ở dạng mã hóa nó sẽ không thực thi cho đến khi được giải mã, sau một vòng giải mã sẽ giải mã phần thân của virus và chuyển điều khiển trỏ đến nó, bằng cách này virus giải mã sẽ cung cấp ít thông tin về chúng hơn với các phần mềm phát hiện mã độc. Một quá trình giải mã có thể giải mã phần thân virus tại chỗ hoặc ở một nơi khác, sự lựa chọn này có thể được quyết định bởi các ràng buộc bên ngoài như là khả năng viết mã của chương trình bị lây nhiễm.



Hình 1.5 Mô tả quá trình virus được giải mã [10]

Một điểm yếu lớn nhất của phương pháp mã hóa ở trên là phần thân virus sẽ được mã hóa giống nhau ở một lần lây nhiễm kế tiếp, sự không thay đổi này làm cho virus dễ dàng bị phát hiện giống như là một virus không hề che dấu. Điều này sẽ được khắc phục với một khóa mã sinh ngẫu nhiên : Khóa được sử dụng cho việc mã hóa sẽ thay đổi ngẫu nhiên với mỗi một lây nhiễm mới. Rõ ràng, quá trình giải mã virus cần phải được cập nhật cho mỗi lây nhiễm để kết hợp chặt chẽ với khóa mới.

1.2.1.2.2. Virus tàng hình

Virus tàng hình là một virus nỗ lực tiến hành các bước để che dấu sự lây nhiễm của chính nó, không chỉ có phần thân virus, một virus tàng hình luôn cố gắng ẩn mọi thứ không chỉ với các phần mềm chống virus. Một vài kỹ thuật virus tàng hình thực hiện được trình bày bên dưới:

Một nhãn thời gian của tập tin gốc bị lây nhiễm có thể được khôi phục sau khi lây nhiễm do đó tập tin không thấy sự thay đổi mới.

Virus có thể được lưu dữ hoặc có khả năng tái sinh tất cả những thông tin về tập tin trước

khi lây nhiễm bao gồm nhãn thời gian của nó, kích thước tập tin, và nội dung của tập tin, sau đó các cuộc gọi vào/ra hệ thống có thể bị ngăn chặn, và virus sẽ phát lại các thông tin ban đầu đề đáp ứng bất kì hoạt động vào/ra nào trên các tập tin bị lây nhiễm, làm cho nó xuất hiện như là không bị lây nhiễm. Kỹ thuật này cũng được ứng dụng đến các khối khởi động vào ra.

Phương pháp chính xác của việc ngăn chặn các cuộc gọi vào/ra thì phụ thuộc vào hệ điều hành. Với MS-DOS là một ví dụ những yêu cầu vào/ra được tạo ra từ các cuộc gọi ngắt, toàn bộ xử lý được đặt thông qua véc tơ ngắt, virus chỉ cần sửa véc tơ ngắt để chèn chính nó vào chuỗi xử lý ngắt. Trên các hệ thống khác thì biểu diễn vào/ra sử dụng các thư viện chia sẻ, vì vậy một virus có thể đặt chính nó thành một khóa vào trong thư viện chia sẻ những công việc thường lệ để ngăn các cuộc gọi vào ra với hầu hết các ứng dụng.

Một số hệ thống lưu trữ các bộ nạp khởi động thứ cấp như các khối đĩa liên tiếp, để làm nhiệm vụ nạp các khởi động chính đơn giản hơn .Trên các hệ thống này, có 2 cách xem các bộ nạp khởi động thứ cấp như là một chuỗi các khối và như là một tập tin trong hệ thống tập tin. Một virus có thể chèn chính nó vào khối khởi động thứ cấp của khối và di chuyển các khối ban đầu đến nơi khác trong hệ thống tập tin. Kết quả là mọi thứ tỏ ra bình thường, hệ thống tập tin cho thấy không có sự thay đổi nào rõ ràng, nhưng trên thực tế virus đã bị ẩn và chạy một cách tự do của bộ nạp khởi động chính.

Kỹ thuật tàng hình chồng chéo với các kỹ thuật được sử dụng bởi rootkit, Rootkits là một bộ công cụ cho những người đã đột nhập vào máy tính, chúng sử dụng những bộ công cụ toolkit để ẩn và tránh bị phát hiện, mã độc ngày nay cũng sử dụng rootkit ví dụ như Ryknos Trojan horse cố gắng che dấu chính nó bằng việc sử dụng một rootkit dành cho quản lý bản quyền số.

1.2.1.2.3. Virus Oligomorphism:

Giả sử một virus mã hóa với khóa mã hóa ngẫu nhiên được thay đổi với mỗi lần lây nhiễm mới, chỉ có những phần không thay đổi của virus là các mã nằm trong vòng lặp giải mã. Các phần mềm chống virus sẽ khai thác điều này để phát hiện, vì vậy sự phát triển tiếp theo hợp lý là thay đổi mã trong vòng lặp giải mã với mỗi lây nhiễm.

Một virus oligomorphic hay là virus bán đa hình là một virus được mã hóa trong đó có một hữu hạn nhỏ số lượng các vòng lặp giải mã khác nhau theo ý của nó. Vì vậy virus này nó có thể sinh ra được hữu hạn các biến thể khác nhau.

Virus này chọn một vòng lặp giải mã mới từ vùng chứa cho mỗi lần lây nhiễm mới ví

dụ: Whale đã có 30 biến thể giải mã khác nhau và Memorial có 96 biến thể giải mã.

Trong khả năng phát hiện, oligomorphic chỉ tạo ra một virus hơi khó khăn để spot. Thay vì tìm kiếm một vòng lặp giải mã cho virus, phần mềm chống virus có thể đơn giản thực hiện liệt kê ra các vòng giải mã mà virut có khả năng và tìm ra kiếm tất cả biến thể của chúng.

1.2.1.2.4. Virus đa hình:

Một virus đa hình bề ngoài thì giống như một virus Oligomorphism, cả hai cùng mã hóa virus, cả hai cùng thay đổi vòng lặp giải mã của chúng cho mỗi lần lây nhiễm, tuy nhiên một virus đa hình có một số lượng vô hạn các biến thể của vòng lặp giải mã. Tremor là một ví dụ nó có tới gần sáu tỷ khả năng lặp giải mã tức là có thể sinh ra 6 tỷ biến thể khác nhau. Có thể nói virus đa hình rõ ràng không thể bị phát hiện bởi liệt kê tất cả các kết hợp có thể của chúng.

Như vậy có thể thấy một virus đa hình là một virus thực hiện các phương pháp che dấu dựa vào việc mã hóa các thân mã chương trình của nó bằng việc thay đổi các cách thức mã hóa hay thay đổi phương thức giải mã nhiều lần, chúng có khả năng tinh vi lẩn trốn trước sự tìm kiếm của các phần mềm diệt virus bởi các biến thể sinh ra ngẫu nhiên theo thời gian và các đối tượng lây nhiễm.

1.2.1.2.5. Virus siêu đa hình:

Virus siêu đa hình là virus mà đa hình phần thân mã bên trong của nó. Chúng không thực hiện mã hóa do đó không cần các vòng giải mã tuy nhiên để tránh bị phát hiện bởi sự thay đổi, một phiên bản mới hay biến thể của phần thân virus được thực hiện cho mỗi lần lây nhiễm mới.

Tất cả các kỹ thuật sửa đổi mã được sử dụng bởi virus đa hình đều được áp dụng vào virus siêu đa hình, cả 2 đều sử dụng một “mutation engine” (“mutation engine” là một chương trình máy tính có thể được sử dụng để chuyển đổi một chương trình thành phiên bản tiếp theo bao gồm các mã lệnh khác nhau nhưng hoạt động với các chức năng tương tự nhau) , ngoại trừ việc một virus đa hình thì không cần thay đổi “engine” cho mỗi lần lây nhiễm bởi vì nó có thể cư trú bên trong phần mã hóa của virus. Ngược lại “mutation engine” của một virus siêu đa hình sẽ tự biến đổi chính nó cho mỗi lần lây nhiễm.

Có thể nói virus siêu đa hình rất tinh vi với khả năng lai tạo và kết hợp đa hình chúng tao ra vô số các biến thể chỉ từ một mã virus ban đầu làm cho các phần mềm tìm và diệt virus vô cùng khó khăn trong việc phát hiện ra chúng và bị qua mặt dễ dàng.

1.2.2. Logic Bomb

Logic bomb là loại mã độc không tự nhân bản chúng có thể ký sinh trên các vật chủ, một mã độc logic bomb bao gồm 2 phần chính là :

Một payload là các mã nạp có thể là bất cứ thứ gì nhưng có ý đồ xấu hay là các mục đích của mã độc như là phá hoại hay phá hủy các chức năng an toàn của hệ thống.

Một trigger (kích hoạt) là một điều kiện đúng hoặc sai được đánh giá và điều khiển khi một payload được thực thi. Một điều kiện kích hoạt chính xác thì được giới hạn bởi khả năng sáng tạo và có thể dựa trên các điều kiện địa phương như là thời gian, ngày, tháng , năm để thực hiện kích hoạt, dựa vào một đăng nhập người dùng, hay một phiên bản hệ điều hành, hoặc bất kỳ một sự kiện nào đó mà kẻ viết mã độc nghĩ ra . Kích hoạt cũng có thể được thiết kế để kích "nổ" từ xa.

Đặc tính của mã độc logic bomb là chúng sẽ cư trú bằng cách ký sinh trên các phần mềm thông thường hoặc đứng một cách độc lập bằng cách nào đó ẩn nấp khỏi sự phát hiện của người dùng hay các phần mềm diệt virus giai đoạn này hầu như chúng không thực hiện hành vi gì cho đến khi điều kiện kích hoạt xảy ra hay một thông điệp tấn công được gửi các mã độc hại bên trong sẽ được thực thi.

1.2.3. Trojan Horse:

Mã độc Trojan Horse là một cái tên được xuất phát từ một điển tích "Con ngựa thành Troy" trong thần thoại Hy Lạp. Trojan là một loại mã độc mặc dù không có khả năng tự nhân bản tuy nhiên nó thường lây nhiễm vào hệ thống với những thể hiện rất bình thường dưới hình thức đóng giả như là các chương trình phần mềm hữu ích thực hiện những nhiệm vụ thông thường và hợp pháp, nhưng thực chất bên trong lại bí mật thực hiện một số nhiệm vụ nguy hiểm, ác tính mà kẻ có ý đồ xấu đã cài đặt vào nó. Ví dụ như một mã độc Trojan có thể là một chương trình đánh cắp thông tin tài khoản và mật khẩu người dùng bằng cách khi người dùng thực hiện chức năng đăng nhập mặc dù các thông tin được gõ vào là chính xác tuy nhiên chúng sẽ tạo ra các nhắc nhở thông báo rằng tên tài khoản và mật khẩu không đúng và thông báo yêu cầu người dùng nhập lại vào đó để xác thực trước khi chạy một chương trình đăng nhập thực sự. Người dùng không nghi ngờ rằng họ đã thực hiện một lỗi đánh máy và nhập lại các thông tin mà không hề hay biết. Đôi khi các Trojan cũng có thể được cài đặt các chức năng gián điệp được điều khiển và thực hiện các hành vi theo yêu cầu của máy chủ từ xa,

Chúng có thể ký sinh trên các vật chủ là các phần mềm thông thường và thực hiện

các chức năng của vật chủ một cách bình thường nhưng lại thực hiện các chức năng độc hại một cách riêng biệt hoặc làm những công việc của chương trình chủ nhưng chỉnh sửa các chức năng để thực hiện ý đồ ác ý, núp dưới danh một phần mềm hữu ích để thực hiện các hành vi gây hại. Trên các máy tính chạy hệ điều hành Windows các Trojan thường được xuất hiện đính kèm thư điện tử bằng cách đặt tên dưới danh nghĩa là các chương trình lương thiện như các chương trình văn bản, tập tin .txt hay các tập tin đa phương tiện tuy nhiên ẩn bên trong lại là các tập tin thực thi như là .exe, .com, .scr, .bat vì một số phần mềm được cấu hình trong Windows mặc định không hiển thị thông tin này vì vậy chúng đánh lừa người dùng dụ dỗ họ mở các chương trình này và vô tình kích hoạt chúng. Trojan đôi khi cũng xuất hiện dưới dạng các biểu tượng thông thường của các phần mềm hữu ích thậm chí chúng có thể đóng giả chính các chương trình diệt virus và dễ dàng qua mặt người dùng thông thường.

1.2.4. Back Door

Back door là một mã độc không tự tái tạo và chúng thường kí sinh trên các vật chủ một back door là bất kì một cơ chế nào cái mà cho phép vượt qua các kiểm tra an ninh thông thường. Các lập trình viên đôi khi tạo ra back door là một chức năng bí mật trong một chương trình cho những lý do chính đáng như là bỏ qua một quá trình xác thực tốn thời gian khi gỡ lỗi một máy chủ mạng hay kiểm tra chương trình và khắc phục lỗi. Giống như là logic bomb, back door có thể được cài đặt trên các mã hay chương trình hợp pháp hoặc như một chương trình độc lập.

Một loại đặc biệt của back door là RAT được viết tắt của công cụ quản trị từ xa (Remote Administration Tool) hay là truy cập Trojan từ xa (Remote Access Trojan) điều này phụ thuộc vào người được hỏi. Các chương trình này cho phép một máy tính có thể giám sát và điều khiển từ xa, Người sử dụng có thể chủ ý cài đặt để truy cập vào một máy tính để làm việt ở nhà hoặc cho phép những nhân viên trợ giúp chẩn đoán và sửa chữa máy tính từ xa. Tuy nhiên nếu mã độc lén lút cài đặt một RAT trên một máy tính, sau đó nó sẽ mở ra một cửa hậu (back door) trở lại trong máy tính đó. Back door khi chạy trên các máy tính bị nhiễm sẽ lưu trú trong bộ nhớ đợi lệnh điều khiển từ các cổng dịch vụ, chúng mở ra các cổng cho phép kẻ tấn công truy cập vào máy nạn nhân từ các cuộc tấn công mạng từ đó kẻ tấn công sẽ chiếm quyền giám soát và điều khiển máy nạn nhân. Back door tỏ ra rất nguy hiểm bởi khả năng lẩn trốn của nó đôi khi chúng được hẹn trước thời gian để kết nối ra ngoài vì thế trong thời gian lưu trú chúng không để lộ bất kỳ

hành vi hay thông tin gì cho phép các phần mềm diệt mã độc phát hiện ra.

1.2.5. Sâu máy tính (Worm):

Sâu máy tính là loại mã độc có khả năng tự nhân bản tuy nhiên chúng thường xuất hiện như những chương trình độc lập mà không cần tập tin chủ để mang nó. Do đó nó không ký sinh vào các tệp tin hoặc một khu vực nào trên đĩa cứng bản thân nó đã là một chương trình hoàn chỉnh vì vậy không thể dùng các chương trình dò quét tệp tin để phát hiện và loại bỏ sâu máy tính. Môi trường hoạt động của sâu máy tính là mạng Internet chúng đặc biệt mạnh trong khả năng tự nhân bản nếu như virus cần có một sự kiện từ người dùng để kích hoạt tự sao chép và lây nhiễm như là : Mở tập tin, khởi động máy tính, sao chép tập tin….Sâu máy tính hoàn toàn có khả năng sao chép và nhân bản chính nó mà không cần bất kỳ thao tác nào của người sử dụng đây là một đặc điểm giúp chúng lan rộng với tốc độ chóng mặt. Mục đích của sâu máy tính là làm tiêu tốn tài nguyên băng thông của mạng máy tính và phá hoại các hoạt động của hệ thống như là phá hủy tệp tin, tạo ra cửa hậu cho phép tin tặc xâm nhập và điều khiển trái phép hệ thống. Chính nhờ khả năng về tốc độ lan truyền và khả năng lẫy nhiễm xuyên các hệ thống trong khoảng thời gian ngắn làm cho các tin tặc ngày càng ưu thích sâu máy tính hơn là các virus thông thường, ngày nay mục tiêu thường thấy của sâu máy tính là thực hiện các cuộc tấn công từ chối dịch vụ Ddos hoặc thực hiện xâm nhập cài đặt các back door và tiến hành các công việc phá hoại vô cùng nguy hiểm.

Sâu máy tính bao gồm 2 loại chính là sâu dịch vụ mạng và sâu gửi điện thư hàng loạt:

Sâu dịch vụ mạng (Network Service Worm): chúng phát tán và lây lan thông qua việc khai thác các lỗ hổng bảo mật của các dịch vụ mạng, các ứng dụng hay hệ điều hành. Sau khi lây nhiễm vào hệ thống chúng sẽ dò quét tất cả các ứng dụng hay hệ thống chạy cùng dịch vụ với các lỗ hổng đang bị khai thác và tiếp tục tấn công trên diện rộng với tốc độ chóng mặt chúng có thể làm tê liệt mạng máy tính và các hệ thống an ninh, một số sâu dịch vụ mạng như là Sasser được phát hiện lây lan trên mạng ngày 1/5/2004 bằng việc khai thác lỗ hổng bảo mật của dịch vụ LSASS (Local Security Authority Subsystem Service) tấn công vào các máy tính cài Windows 2000/ XP/ Server 2003. Hay một sâu dịch vụ nổi tiếng khác là Witty.

Sâu gửi điện thư hàng loạt (Mass Mailing Worm): Chúng là các sâu tấn công vào dịch vụ thư điện tử cách thức hoạt động của chúng là lây lan trên các hệ thống thư điện tử

một khi hệ thống bị lây nhiễm nó sẽ dò quét danh sách các địa chỉ email và tự động gửi bản sao của chính nó tới những địa chỉ mà nó tìm được bằng cách sử dụng các email máy khách thuộc hệ thống bị lây nhiễm hoặc email thuộc hệ thống do nó tạo ra, nó có thể gửi đồng loạt các thư này vào mạng lưới của hệ thống làm cho các máy chủ trở nên quá tải và làm cho hệ thống chịu một lượng lớn "bomb mail". Loại sâu này hiện nay chủ yếu được dùng với mục đích quảng cáo như phát tán các thư rác hay những chương trình mã độc nguy hiểm khác một số ví dụ về sâu máy tính gửi điện thư hàng loạt là : Beagle, Mydoom và Netsky.

1.3. Các kỹ thuật phát hiện mã độc

Mục đích của việc phát hiện hiện mã độc là đưa ra những cảnh báo sớm để có cơ chế ngăn chặn kịp thời trước khi các mã độc thực hiện các hành vi hay chức năng của chúng chính vì vậy vai trò của phát hiện mã độc là hết sức quan trọng, chúng ta luôn luôn phải tìm kiếm và phát hiện sự tồn tại của mã độc ngay cả khi chúng không thực thi hay làm bất cứ điều gì ,thậm chí nếu một mã độc không hoạt động trên một hệ thống việc phát hiện ra nó vẫn rất cần thiết bởi vì điều đó đảm bảo rằng nó sẽ không ảnh hưởng đến hệ thống khác. Lấy hệ thống email là một ví dụ nơi mà các máy nhận có thể chạy trên hệ thống hay chứa những ứng dụng, dịch vụ hoàn toàn khác với máy chủ vì vậy phải đảm bảo rằng mã độc không thể được thực thi trên bất kỳ hệ thống nào.

Các kỹ thuật phát hiện mã độc là một quá trình tìm kiếm và thẩm định xem một chương trình phần mềm có thể đã bị lây nhiễm mã độc hay bên trong có chứa các đoạn mã được xem là mã độc hay không, thêm vào đó các hành vi của chúng cũng được phân tích và xem xét là nhóm các hành vi thông thường hay các hành vi thuộc về mã độc, dựa vào các kết quả đó để có thể chứng minh và phát hiện sự tồn tại của mã độc trên các hệ thống.

1.3.1. Các kỹ thuật phát hiện dựa trên phân tích tĩnh

Kỹ thuật phát hiện mã độc dựa trên phương pháp phân tích tĩnh có đặc điểm là phát hiện mã độc mà không cần phải chạy hay thực thi bất kỳ đoạn mã nào của nó gồm có 3 phương pháp chính là kỹ thuật dò quét, chẩn đoán dựa trên kinh nghiệm và kiểm tra tính toàn vẹn.

1.3.1.1. Kỹ thuật dò quét (scanner):

Thông thường mỗi một mã độc được biểu diễn bởi một hay nhiều mẫu, hoặc là các dấu hiệu (signatures), chuỗi tuần tự các byte là những cái được coi là đặc trưng duy nhất

của mã độc. Các dấu hiệu này đôi khi còn được gọi là các chuỗi (scan strings) và chúng không cần bất kỳ một ràng buộc về chuỗi nào. Một vài phần mềm chống mã độc có thể hỗ trợ việc sử dụng các ký tự đại diện (wildcards) cho mỗi một byte tùy ý, một phần của một byte, hoặc không hay nhiều byte. Quá trình phát hiện mã độc bằng cách tìm kiếm thông qua một tập tin với các dấu hiệu của nó thì được gọi là scanning và các mã được tìm thấy được gọi là một scanner. Cụ thể hơn nữa, quá trình phát hiện được thực hiện thông qua một dòng các mã byte chúng có thể là toàn bộ nội dung của một khối khởi động, toàn bộ nội dung của tập tin, hoặc là một phần của tập tin được đọc hoặc ghi, hay cũng có thể là các gói tin mạng.

Với hàng trăm ngàn dấu hiệu để phát hiện việc tìm kiếm chúng tại một thời điểm thì không khả thi chút nào. Một trong thách thức lớn nhất của kỹ thuật này là tìm ra các thuật toán có khả năng tìm được nhiều mẫu một cách hiệu quả và đồng thời đánh giá được chúng.

1.3.1.2. Kỹ thuật Static Heuristics

Kỹ thuật này được áp dụng để nhân lên khả năng chuyên gia trong các phần mềm chống virus, chẩn đoán dựa trên kinh nghiệm trong phương pháp phân tích tĩnh có thể tìm thấy các mã độc đã biết hoặc chưa biết bằng cách tìm kiếm một mẩu mã mà có những đặc điểm chung giống như là một mã độc thay vì scanning các dấu hiệu đặc biệt của mã độc. Đây là một kỹ thuật phân tích tĩnh có nghĩa là các mã sẽ được phân tích mà không thực thi và không có gì đảm bảo về bất kỳ một mã nghi ngờ được tìm thấy sẽ thực thi khi nào. Kỹ thuật này được thực hiện thông qua 2 bước:

Thu thập dữ liệu: Dữ liệu thu thập có thể được sử dụng từ bất kỳ một kỹ thuật dựa trên kinh nghiệm nào, có hay không một kỹ thuật kinh nghiệm phân loại chính xác các đầu vào điều đó không thực sự quan trọng bởi vì các kết quả của nhiều kinh nghiệm sẽ được kết hợp và được phân tích sau đó. Một scanner có thể được sử dụng để xác định vị trí một dấu hiệu là các biểu hiện được chỉ ra của những mã đáng nghi được gọi là các booster. Sự hiện diện của các booster này làm tăng khả năng các mã sẽ được phân tích là các mã độc như là: các mã Junk, tự thay đổi mã, sử dụng các cuộc gọi hàm API không được cung cấp, điều khiển véc tơ ngắt, sử dụng các lệnh bất thường đặc biệt là cái không được sinh ra bởi trình biên dịch, chuỗi chứa những từ ngữ khiêu dâm, hoặc các tín hiệu rõ ràng như là một từ “virus”. Một việc cũng không kém phần quan trọng là tìm kiếm những thứ xuất hiện trong một mã bình thường đây là những cái mà mã độc không

thường làm. Ví dụ virus thì không thường tạo một hộp thoại pop-up cho người sử dụng. Điều này sẽ được xem xét là một heuristic phủ định hay một stopper.

Các kinh nghiệm heuristic khác có thể được tính toán mà không dựa trên scanning như là:

Sự khác nhau giữa một điểm vào và kết thúc của một tập tin có thể được tính toán, các giá trị cực nhỏ khi thực hiện so sánh với những giá trị giống như vậy ở một tập tin không bị lây nhiễm.

Phân tích phổ của mã lệnh có thể được thực hiện, tính toán các tần số của các byte hoặc là các lệnh được sử dụng trong mã. Các mã được mã hóa sẽ có dấu hiệu phổ khác với mã không được mã hóa.

Phân tích: phân tích dữ liệu tĩnh heuristic có thể xác định đơn giản như là đánh các trọng số cho mỗi giá trị heuristic và tính tổng các kết quả. Nếu tổng số vượt qua một giá trị ngưỡng được sử dụng như là một căn cứ phát hiện thì dữ liệu đầu vào có thể đã bị lây nhiễm.

Một số phương pháp phức tạp trong các phân tích dữ liệu có thể sử dụng trí tuệ nhân tạo như là mạng neural, hệ chuyên gia, hay các kỹ thuật khai phá dữ liệu.

Kỹ thuật Static heuristics có thể được xem như là một cách để giảm các yêu cầu tài nguyên của kỹ thuật scanner. Toàn bộ dấu hiệu của mã độc trong một cơ sở dữ liệu mã độc có thể được chắt lọc giảm xuống một tập nhỏ hơn, tổng quát hơn. Một kỹ thuật quét mã độc có thể tìm kiếm những dấu hiệu đặc trưng ngắn này và nạp vào trong bộ dữ liệu chứa toàn bộ các dấu hiệu nếu phù hợp với những gì đã tìm thấy. Điều này làm giảm bớt yêu cầu lưu dữ toàn bộ các dấu hiệu đặc trưng trong bộ nhớ.

1.3.1.3. Kỹ thuật kiểm tra sự toàn vẹn (Integrity Checkers)

Ngoại trừ một số virus đồng dạng các mã độc chủ yếu hoạt động bằng cách thay đổi các tập tin. Kỹ thuật kiểm tra tính toàn vẹn nhằm khai thác các hành vi này để tìm ra mã độc bằng cách xem các thay đổi trái phép vào các tập tin. Một kiểm tra toàn vẹn được khởi đầu bằng việc tính và lưu dữ một checksum cho mỗi tập tin trong hệ thống được xem xét. Sau đó một checksum của tập tin cần kiểm tra sẽ được tính lại và so sánh với giá trị checksum gốc của nó. Nếu checksum khác nhau có nghĩa là đã có một sự thay đổi diễn ra. Kỹ thuật này được sử dụng rất phổ biến trong các phần mềm chống mã độc ban đầu các mã độc được tìm thấy sẽ được sử dụng một hàm băm như là MD5,SHA,CRC… để tính toán ra giá trị băm duy nhất của nó sau đó các giá trị này được lưu trong cơ sở dữ

liệu như là các dữ liệu mẫu định danh mã độc đó. Khi muốn kiểm tra một tập tin có phải là mã độc hay không chúng ta chỉ cần tính toán lại hàm băm và so sánh giá trị kết quả với cơ sở dữ liệu đã có. Nếu tồn tại một mã băm như vậy trong cơ sở dữ liệu mã độc thì có thể biết chính xác nó là một mã độc.

1.3.2. Các kỹ thuật phát hiện dựa trên phân tích động

Kỹ thuật phát hiện mã độc dựa trên phân tích động là kỹ thuật quyết định một tập tin có bị lây nhiễm hay không thông qua việc thực thi các mã chương trình và quan sát các hành vi của nó. Để thực hiện được phương pháp này cần phải thiết lập môi trường để mã độc thực thi và các công cụ cho phép quan sát các hành động của chúng như: theo dõi các tiến trình, theo dõi các thông tin về registry, sự thay đổi của các tệp tin, thư mục, theo dõi sự thay đổi của lưu lượng mạng và các kết nối TCP/IP… Tất cả các hành vi của mã độc sẽ được lưu dưới dạng các nhật ký phục vụ cho công việc phân tích sau này.

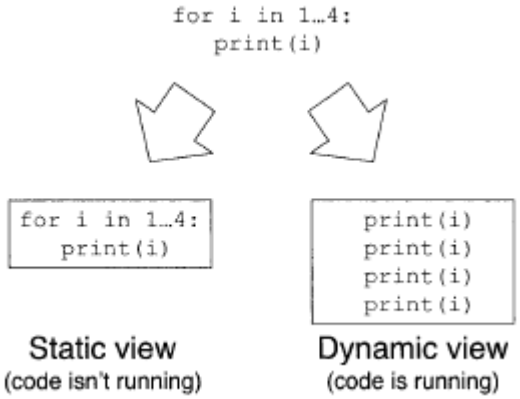
1.3.2.1. Kỹ thuật Behavior Monitors/Blockers

Một behavior blocker là một kỹ thuật chống mã độc giám sát các hành vi thực thi của một chương trình trong thời gian thực, theo dõi các hành động, các khối lệnh khả nghi của nó, nếu những hành động này được tìm thấy behavior blocker có thể ngăn chặn những hành động khả nghi thành công và chấm dứt các tiến trình của một chương trình hay là có thể hỏi người dùng cho cho các hành động phù hợp để xử lý. Một behavior blocker thì tìm kiếm điều gì? Về cơ bản một behavior blocker xem xét một chương trình thuộc vào các hành vi được xem xét là bị chặn hay là các hành vi thông thường. Chúng có thể là những hành vi liên quan đến tập tin như mở, sửa đổi hay xóa, hay những thao tác fomat ổ đĩa phân vùng, thay đổi registry, kết nối mạng…Một hành vi thông thường được mô hình hóa trong 3 cách được miêu tả như sau:

a. Một hành động được cho phép điều này được gọi là một phát hiện positive
b. Một hành động không được cho phép điều này được gọi là một phát hiện negative
c. Một cách kết hợp cả 2 giống như kỹ thuật static heuristics bao gồm có boosters và stoppers.

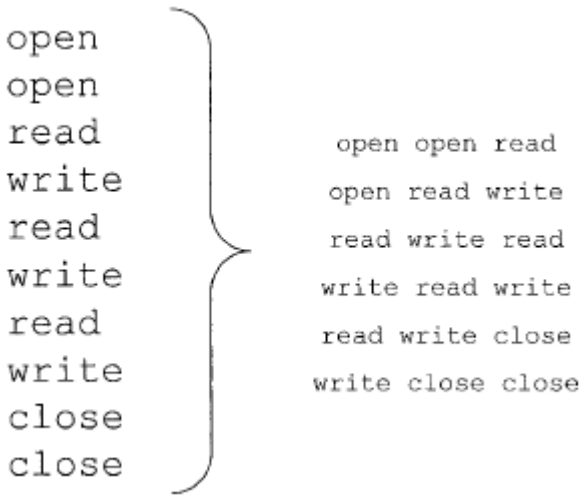
Một điều tương tự được rút ra từ các hệ thống miễn dịch trong tự nhiên, các behavior blocker cố gắng để nhận thức chính nó từ những thứ không phải là nó, hay là các hành vi bình thường từ những hành vi bất bình thường. Cách tư duy này là cái mà hệ thống miễn dịch cần làm để phân biệt các tế bào bình thường xâm nhập từ bên ngoài.

Tuy nhiên điều này cần phải được xem xét cẩn thận vì đôi khi các hành vi bất thường không có nghĩa chắc rằng nó là một hành vi thuộc về mã độc.



Hình 1.6 so sánh 2 kỹ thuật phân tích tĩnh và động [10]

Kỹ thuật behavior blocker có thể tìm kiếm những dấu hiệu theo phương pháp phân tích động mà những biểu hiện là một hành vi thuộc về mã độc. Ví dụ nhìn vào cách hoạt động vào/ra được thêm vào của một mã độc cũng thể hiện các dấu hiệu của nó.



Các dấu hiệu đôi khi là chuỗi các hành vi như hình minh họa ở trên là chuỗi các hành vi liên quan đến các thao tác xử lý tập tin với độ dài là 3. Dựa vào các dấu hiệu này chúng ta hoàn toàn có thể tìm ra được những bất thường trong các hành vi của một mã độc.

1.3.2.2. Kỹ thuật Emulation

Giám sát các hành vi kỹ thuật Behavior blocking cho phép mã chương trình chạy trên máy thực. Ngược lại kỹ thuật phát hiện mã độc sử dụng mô phỏng giả lập (emulation) cho phép các mã được chạy và phân tích trong một môi trường mô phỏng giả lập. Với kỳ vọng rằng dưới môi trường mô phỏng một mã độc sẽ tiết lộ chính nó. Bởi

vì bất kỳ một mã độc nào được tìm thấy sẽ không được chạy trên máy tính thực sự vì vậy không có một hiểm họa nào xảy ra.

Mô phỏng có thể được áp dụng qua 2 cách mặc dù ranh giới phân biệt giữa chúng là mập mờ:

Dynamic heuristic: đây là kỹ thuật có độ chính xác tương tự như kỹ thuật static heuristic chỉ có một sự khác nhau là các dữ liệu được thu thập như thế nào: dynamic heuristic phân tích thu thập các dữ liệu của nó từ môi trường giả lập về các mã chương trình sẽ được phân tích. Các phân tích này được thực hiện giống như là cách mà kỹ thuật static heuristic đã thực hiện.

Dynamic heuristic cũng có thể tìm kiếm các đặc trưng tương tự như là behavior blockers, như là các lời gọi hệ thống. Môi trường giả lập là một môi trường ảo an toàn trong đó các mã chạy được giám sát. Dynamic heuristics có thể được sử dụng hiệu quả để phát hiện các dấu hiệu hành vi của virus siêu đa hình.

Generic decryption: với những virus đa hình vòng giải mã có thể là một yếu tố gây ra khó khăn cho các phần mềm diệt mã độc để phát hiện được chúng. Kỹ thuật generic decryption có thể là những cứu cánh cho vấn đề này bởi vì dựa vào vòng lặp giải mã riêng của mã độc để giải mã phần thân của nó. Có một nhận xét là các mã độc cho dù mã hóa thế nào thì chúng đều có chung những hành vi để thực hiện cùng một mục đích. Một khi được giải mã, phần thân của mã độc có thể được phát hiện bằng cách sử dụng phương pháp dò quét thông thường, điều này xác định chính xác khả năng nhận biết các mã độc đa hình, kỹ thuật này sử dụng các kinh nghiệm để xác minh khi một mã độc mã hóa chính nó. Ví dụ như một virus đang cố gắng thực thi một lệnh nằm thường chú trong vùng nhớ ở lần sửa đổi trước (ví dụ giải mã). Một cái dấu hiệu khác là sự thay đổi về kích thước của một quá trình giải mã mặc dù khoảng thay đổi này sẽ tùy vào kiến trúc. Trên nền tảng Intel x86, 24 byte hoặc nhiều hơn bị sửa đổi trên bộ nhớ là một dấu hiệu tiềm năng của một quá trình giải mã đã điễn ra. Bên cạnh kinh nghiệm kỹ thuật dự trên mô phỏng có thể quét bộ nhớ định kỳ trong khoảng thời gian mô phỏng để phát hiện những dấu hiệu bất thường.

CHƯƠNG 2 MỘT SỐ THUẬT TOÁN PHÂN LỚP DỮ LIỆU ĐIỂN HÌNH TRONG KỸ THUẬT HỌC MÁY GIÁM SÁT

2.1. Thuật toán cây quyết định

2.1.1. Giới thiệu thuật toán

Thuật toán cây quyết định là một phương pháp được ứng dụng rộng rãi trong thực tế đặc biệt là các lĩnh vực khai phá dữ liệu hay như giải quyết các bài toán phân lớp trong kỹ thuật học máy. Học máy dựa trên cây quyết định là một phương pháp xấp xỉ với hàm mục tiêu có giá trị rời rạc trong đó các hàm học được biểu diễn bởi một cây quyết định. Cây quyết định cũng có thể được biểu diễn bởi một tập các luật if-then các luật này thì khá là gần gũi với tư duy con người. Một trong những ưu điểm khác là cây quyết định có thể thực hiện ngay cả với các dữ liệu có chứa nhiễu.

Một cây quyết định trong học máy được mô tả bởi:

- Mỗi nút trong biểu diễn một thuộc tính hay một đặc trưng cần kiểm tra của dữ liệu mẫu
- Mỗi nhánh đi ra từ một nút thuộc tính biểu diễn giá trị có thể có của biến thuộc tính đó
- Mỗi nút lá biểu diễn cho một giá trị phân lớp (là giá trị các nhãn trong tập huấn luyện)

Cây quyết định được dùng để dự đoán các dữ liệu chưa biết bằng cách duyệt cây từ nút gốc đến một nút lá được gán nhãn.

Một cây quyết định được phát triển theo phương pháp đệ quy bắt đầu với mỗi nút gốc được xây dựng bằng cách chọn thuộc tính tốt nhất có thể, một vòng lặp các cây con dẫn xuất được tạo mới theo phương pháp như vậy cho đến khi gặp các điều kiện dừng. Trong cây quyết định mỗi thuộc tính hay đặc trưng chỉ được xuất hiện duy nhất một lần với bất kỳ đường đi nào trong cây.

2.1.2. Xây dựng cây quyết định dựa trên thuật toán ID3

Ta xét một thuật toán đơn giản và phổ biến nhất là thuật toán ID3, cây quyết định được xây dựng từ các tập đặc trưng của dữ liệu mẫu đã được gán nhãn từ trước theo phương pháp top-down:

Giải thuật ID3:

Function Build_Tree(tập_ví_dụ, thuộc_tính_mục_tiêu, tập_thuộc_tính)

Trong đó tập_ví_dụ là tập các trường hợp hay dữ liệu mẫu huấn luyện, thuộc_tính_mục_tiêu là thuộc tính mà giá trị của nó là giá trị đích được dự đoán bởi cây quyết định , tập_thuộc_tính là một danh sách thuộc tính khác có thể được kiểm tra để xây dựng cây quyết định được học .

Begin

If tất cả các ví dụ trong tập_ví_dụ thuộc vào lớp dương **then return** một nút gốc được gán nhãn là (+)

Else If tất cả các ví dụ trong tập_ví_dụ thuộc vào lớp âm **then return** một nút gốc được gán nhãn là (–)

Else If tập_thuộc_tính mà rỗng **then return** một nút gốc với nhãn được gán bằng giá trị phổ biến nhất của thuộc_tính_mục_tiêu trong tập_ví_dụ

Else

Chọn ra A là thuộc tính tốt nhất để phân lớp từ tập_thuộc_tính

Đặt A là nút gốc cho cây quyết định hiện tại

For each giá trị v_i có thể có của A

Begin

Thêm một nhánh cây mới bên dưới nút gốc, tương đương kiểm tra A=v_i

Đặt tập_ví_du$_{vi}$ là một tập con của tập_ví_dụ là các ví dụ có giá trị v_i cho A

If tập_ví_du$_{vi}$ là rỗng **then** thêm vào bên dưới nhánh mới này một nút lá với nhãn được gán bằng giá trị phổ biến nhất của thuộc_tính_mục_tiêu trong tập_ví_dụ

Else thêm vào bên dưới nhánh mới này một cây con và gọi hàm đệ quy với tập_thuộc_tính không chứa thuộc tính A.

Function Build_Tree(tập_ví_dụ, thuộc_tính_mục tiêu, tập_thuộc_tính - {A})

End

Return ***Nút_gốc***

End

Đối với thuật toán cơ bản ID3 cây quyết định được xây dựng dựa trên chiến lược top-down theo đó mỗi cây sẽ được xây dựng một cách đệ quy bắt đầu bằng việc lựa chọn

nút gốc, chúng ta sẽ bắt đầu với câu hỏi rằng thuộc tính nào nên được kiểm tra ở nút gốc của một cây ? để giải quyết điều này mỗi thuộc tính có trong dữ liệu mẫu sẽ được đánh giá thông qua việc thống kê các độ lợi về thông tin, độ lợi về thông tin của một thuộc tính được hiểu là khả năng phân tách dữ liệu huấn luyện của thuộc tính đó theo các lớp.

Độ đo Entropy: Độ đo entropy là một khái niệm được dùng trong lý thuyết thông tin nhằm đánh giá độ hỗn loạn của thông tin. Trong đó Entropy của một tập S đối với bộ phân lớp gồm C lớp được định nghĩa như sau:

$$Entropy(S) = \sum_{i=1}^{c} -p_i.\log_2 p_i$$

Trong đó: p_i là tỷ lệ các mẫu trong tập S thuộc vào lớp i

Trong công thức này nếu p_i=0 thì ta có thể coi $0.\log_2 0 = 0$

Với tập S chỉ được phân lớp thành 2 lớp khác nhau thì:

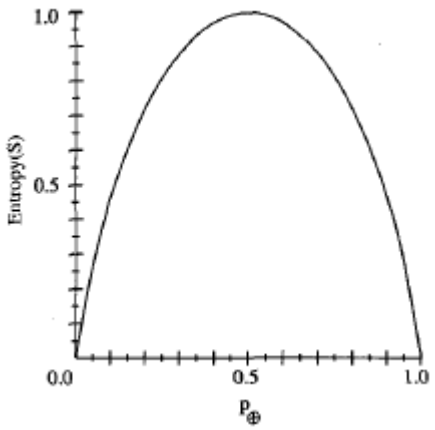
$$Entropy(S) = -p_1.\log_2 p_1 - p_2.\log_2 p_2$$

Để làm sáng tỏ ta xét một ví dụ như sau: Giả sử một tập S gồm 14 mẫu, trong đó 9 mẫu thuộc về lớp (+) và 5 lớp thuộc vào lớp (-) thì :

$$Entropy(S) = -(9/14).\log_2(9/14) - (5/14).\log_2(5/14) \approx 0.94$$

Đối với trường hợp các tập mẫu S được phân thành 2 lớp ta có nhận xét rằng:

- Entropy sẽ bằng 0 nếu tất cả các mẫu của S đều thuộc cùng một lớp
- Entropy sẽ bằng 1 nếu số các mẫu thuộc lớp (+) bằng số các mẫu thuộc lớp (-)
- Entropy sẽ nằm trong khoảng (0,1) nếu số các mẫu thuộc lớp (+) khác với số các mẫu thuộc lớp (-)



Hình 2.1 Biểu diễn giá trị Entropy với tập mẫu nằm trong 2 lớp [18]

Độ lợi về thông tin (Information Gain): trong học bằng cây quyết định độ lợi về thông tin của một thuộc tính là độ giảm về Entropy khi phân tách các mẫu theo giá trị của thuộc tính đó, tức là một thuộc tính mà có độ lợi về thông tin càng cao đồng nghĩa rằng độ hỗn loạn về thông tin khi chọn thuộc tính đó càng giảm, đối với cây quyết định một nút gốc được chọn dựa vào độ lợi thông tin của các thuộc tính có thể.

Độ lợi về thông tin của thuộc tính A đối với tập mẫu S được tính như sau:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Trong đó Values(A) là tập các giá trị có thể của thuộc tính A và

$$S_v = \{x \mid x \in S, x_A = v\}$$

2.1.3. Ví dụ minh họa:

Xét tập dữ liệu mẫu những ngày mà một người đi chơi hay không đi chơi tennis dựa trên các điều kiện thời tiết như sau:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Hình 2.2. Tập dữ liệu mẫu ví dụ cho xây dựng cây quyết định [18]

Ta sẽ xem xét cách xây dựng một cây quyết định ờ lần chọn nút gốc đầu tiên tập thuộc tính được xét bao gồm: thông tin về outlook(quang cảnh), Temperature(Nhiệt độ), Humidity (độ ẩm), Wind (gió) đầu tiên ta cần tính toán độ lợi về thông tin hay khả năng phân tách của các thuộc tính này:

Đối với thuộc tính Windy: biến này có 2 giá trị là Weak và Strong

S= (9 mẫu lớp Yes và 5 mẫu lớp No)

S_{Weak} (6 mẫu thuộc lớp Yes và 2 mẫu thuộc lớp No)

S_{Strong} (3 mẫu thuộc lớp Yes và 3 mẫu thuộc lớp No)

Ta có:

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - (8/14) Entropy(S_{Weak}) - (6/14) Entropy(S_{Strong})$$

$$= 0.94 - (8/14).(0.81) - (6/14).(1) = 0.048$$

Đối với thuộc tính Outlook: thuộc tính này có 3 giá trị là Sunny, Overcast, Rainy

S_{Sunny} (2 mẫu thuộc lớp Yes và 3 mẫu thuộc lớp No)

$S_{Overcast}$ (4 mẫu thuộc lớp Yes và 0 mẫu thuộc lớp No)

S_{Rainy} (3 mẫu thuộc lớp Yes và 2 mẫu thuộc lớp No)

Ta có :

$$Entropy(S_{Sunny}) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5) = 0.971$$

$$Entropy(S_{Overcast}) = -4/4 \log_2(4/4) - 0 \log_2(0) = 0$$

Chú ý: ở đây ta quy ước $0.\log_2 0 = 0$.

$$Entropy(S_{Rainy}) = -3/5 \log_2(3/5) - 2/5 \log_2(2/5) = 0.971$$

Như vậy:

$$\sum_{v \in \{Sunny, Overcast, Rainy\}} \frac{|S_v|}{|S|} Entropy(S_v) = (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971$$

$$= 0.693$$

$$Gain(S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rainy\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= 0.940 - 0.693$$
$$= 0.247$$

Tương tự ta tính các độ lợi thông tin cho các thuộc tính còn lại trên tập dữ liệu kết quả như sau:

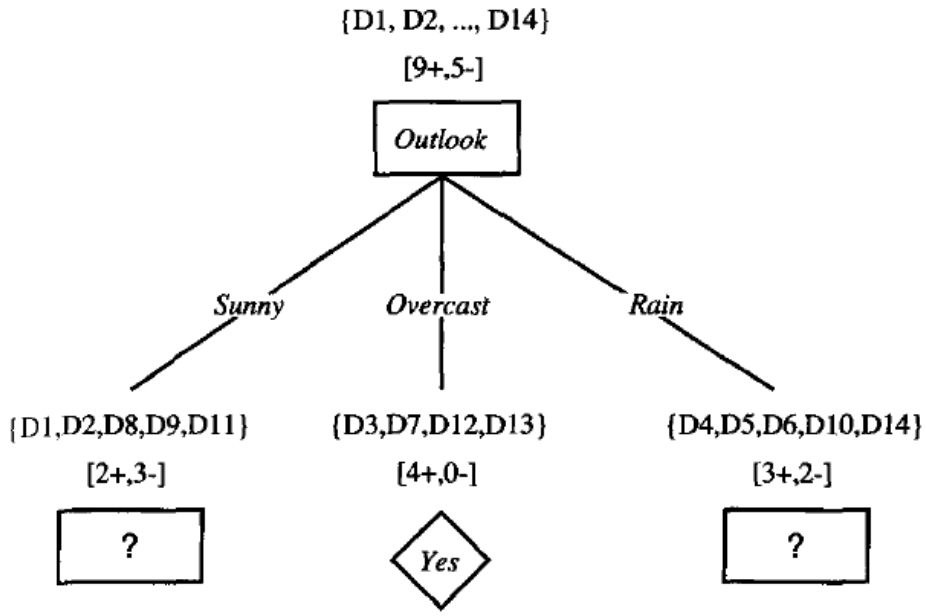
$Gain(S, Outlook) = 0.247$

$Gain(S, Temperature) = 0.029$

$Gain(S, Humidity) = 0.152$

$Gain(S, \text{W}indy) = 0.048$

Như vậy dựa vào các kết quả có được Outlook có độ lợi thông tin cao nhất nên là thuộc tính được chọn để kiểm tra cho nút gốc:



Hình 2.3 cây được chọn với thuộc tính Outlook là nút gốc [18]

Bằng phương pháp tương tự ta có thể tiếp tục tách các nút tiếp theo với các thuộc tính được lựa chọn lúc này phải loại trừ thuộc tính đã được chọn ở những bước trên ví dụ như nút với điều kiện Outlook=Sunny thì các thuộc tính còn lại được xét là Humidity, Temperature, Wind ta có:

$S_{Sunny} = \{D1, D2, D8, D9, D11\}$

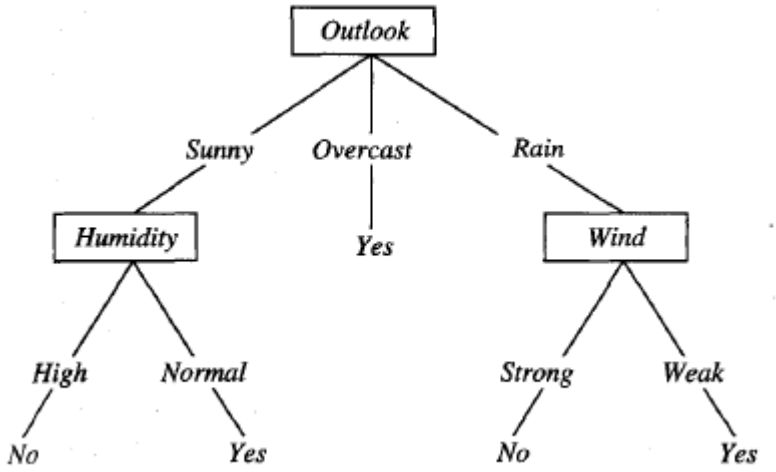
$Gain(S_{Sunny}, Humidity) = 0.97$

$Gain(S_{Sunny}, Temperature) = 0.57$

$Gain(S_{Sunny}, Wind) = 0.019$

→ *trong nhánh thứ nhất nút gốc được chọn tiếp theo là Humidity*

Tương tự với các cây dẫn xuất khác cuối cùng ta thu được cây quyết định như sau:



Hình 2.4 Cây quyết định được sinh ra trong dữ liệu mẫu ví dụ chơi tennis [18]

Dựa vào cây quyết định này ta hoàn toàn có thể dự đoán được các dữ liệu mới giả sử dữ liệu cần dự đoán với các điều kiện thời tiết như sau:

{Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong}

Thực hiện duyệt cây ta dễ dàng kết luận được kết quả dự đoán PlayTennis=No điều này có nghĩa rằng với các điều kiện thời tiết như vậy người này sẽ không đi chơi Tennis.

2.1.4. Nhận xét:

Ta có nhận định rằng một cây quyết định thực chất là biểu diễn một phép tuyển của các phép hội xét cây ở ví dụ trên thì một quyết định đi đánh tennis hoàn toàn có thể được biểu diễn bằng mệnh đề lôgic như sau:

$$\text{Yes} = (Outlook = Sunny \wedge Humidity = Normal) \vee (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$$

Và một quyết định không đi đánh tennis có thể được biểu diễn như sau:

$$No = (Outlook = Sunny \wedge Humidity = High) \vee (Outlook = Rain \wedge Wind = Strong)$$

Từ một cây quyết định ta hoàn toàn có thể sinh ra được các luật If-then xét cây ví dụ ở trên thì một luật có thể được sinh ra như :

If Outlook=Sunny and Humidity=Normal then PlayTennis = Yes

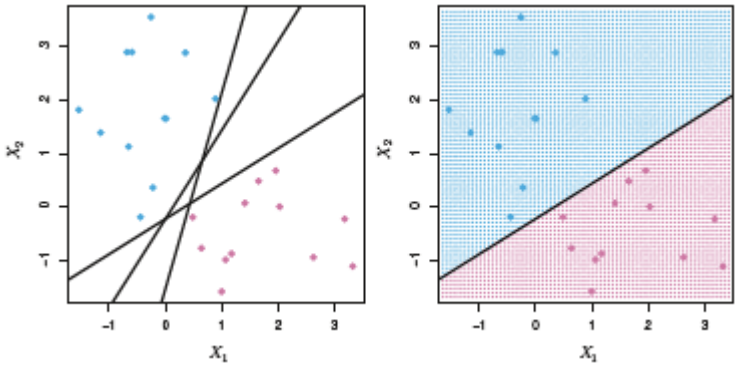
→ Như vậy ta có nhận xét về phương pháp học cây quyết định như sau: Cây quyết định là một phương pháp học máy mà dữ liệu mẫu được mô tả thông qua cấu trúc cây nó tương đối dễ hiểu bởi khả năng sinh ra các luật gần gũi với tư duy con người phương pháp này không đòi hỏi cao trong chuẩn hóa dữ liệu đầu vào cũng như xử lý tốt lượng dữ liệu lớn trong thời gian ngắn.

Tuy nhiên phương pháp này có nhược điểm là không thể giải quyết được khi nhãn phân lớp là liên tục, nó chỉ có thể thực hiện tốt các bài toán phân lớp nhị phân và đôi khi cắt tỉa một cây để tối ưu mô hình cũng là một vấn đề phức tạp.

2.2. Thuật toán SVM

2.2.1. Giới thiệu thuật toán

Trong các thuật toán phân lớp trong kỹ thuật học máy giám sát thì máy véc tơ hỗ trợ là một thuật toán hiệu quả đặc biệt trong các bài toán phân lớp dữ liệu. Thuật toán này được đề xuất bởi Vladimir N. Vapnik [20] ý tưởng chính của nó là coi các dữ liệu đầu vào như là các điểm trong một không gian n chiều và từ các dữ liệu huấn luyện ban đầu được gán nhãn sẽ tìm ra được một siêu phẳng phân lớp chính xác các dữ liệu này, siêu phẳng sau đó được dùng để phân lớp các dữ liệu chưa biết cần tiên đoán.



Hình 2.5 Biểu diễn phân lớp dựa trên thuật toán SVM [8]

Xét một tập dữ liệu mẫu:

$$\mathcal{D} = \{(x1, y1), ..., (xl, yl)\} \ , \ x \in \mathbb{R}^n \ , \ y \in \{-1,1\}$$

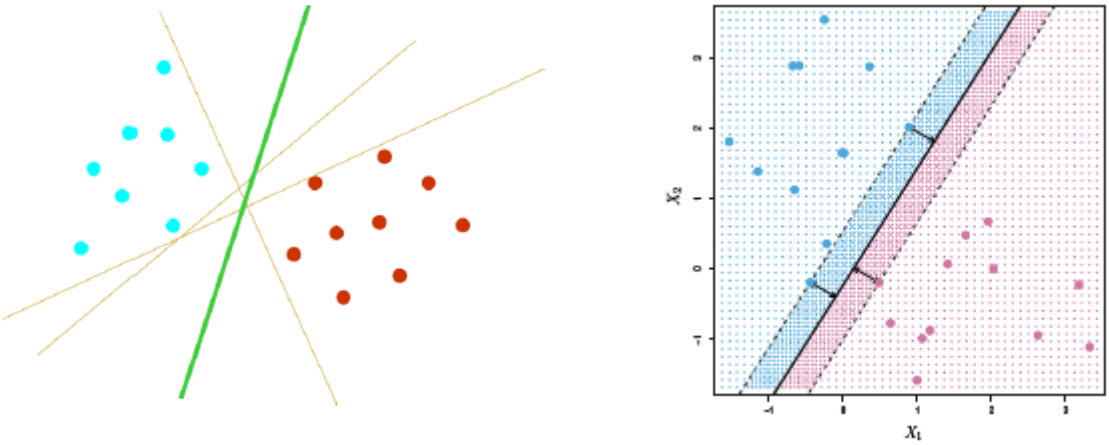
Trong đó x_i là một véc tơ đặc trưng hay một điểm (trong không gian n chiều $x_i \in \mathbb{R}^n$) biểu diễn tập mẫu d_i cặp (x_i, y_i) biểu diễn rằng với một véc tơ đặc trưng x_i thì được gán nhãn là y_i tương ứng trong đó $y \in \{-1,1\}$ hay nói cách khác với tập mẫu d_i sẽ được gán nhãn cho trước là y_i. Ta có phương trình một siêu phẳng được cho như sau :

$$f(x) = \mathrm{w} \cdot \mathrm{x} + \mathrm{b} = 0 \quad [17]$$

Trong đó $\mathrm{w} \cdot \mathrm{x}$ là tích vô hướng giữa véc tơ x và véc tơ pháp tuyến $\mathrm{w} \in \mathbb{R}^n$ được biểu diễn trong không gian n chiều, và $\mathrm{b} \in \mathbb{R}$ là hệ số tự do.

Trên thực tế với các dữ liệu ban đầu có thể sinh ra vô số các siêu phẳng khác nhau để phân lớp dữ liệu tuy nhiên bài toán đặt ra là trong một không gian n chiều với các tập dữ liệu mẫu như vậy làm sao tìm được một siêu phẳng để luôn đảm bảo rằng nó phân chia

các dữ liệu một cách tốt nhất, ta có thể hiểu rằng một siêu phẳng tốt là một siêu phẳng mà khoảng cách từ các điểm dữ liệu được phân lớp gần nhất với siêu phẳng đó là lớn nhất. Phương trình chứa các điểm dữ liệu này được gọi là các lề, như vậy nói cách khác siêu phẳng tốt là siêu phẳng mà khoảng cách giữa nó và lề càng xa càng tốt.



Hình 2.6 Siêu phẳng phân chia dữ liệu tốt có khoảng cách lề xa nhất [8,17]

2.2.2. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và không có nhiễu:

Xét các dữ liệu được phân lớp tuyến tính và không có nhiễu trong mô hình giả sử siêu phẳng phân chia các dữ liệu thành 2 lớp điều này có nghĩa rằng tất cả các điểm dữ liệu thuộc lớp dương (được gán nhãn y = 1) sẽ nằm về một phía của siêu phẳng và tất cả các điểm thuộc lớp âm (được gán nhãn y = -1) sẽ nằm về phía còn lại của siêu phẳng đó. Điều này được mô tả như sau:

Ta có phương trình siêu phẳng cần tìm có dạng: $w \cdot x + b = 0$

Với: $y = \text{sign}\{ w \cdot x_i + b \} = \begin{cases} 1 & y_i = 1 \\ -1 & y_i = -1 \end{cases} \quad \forall (x_i, y_i) \in D$

Xét các điều kiện ràng buộc với các tham số w và b như sau:

$$\min_i \left| (w \cdot x_i) + b \right| = 1 \text{ với } i = 1,...,l$$

Các ràng buộc này thì giúp đơn giản hóa vấn đề hơn nói cách khác chuẩn của một véc tơ trọng số nên được tính bằng nghịch đảo khoảng cách của điểm gần nhất trong dữ liệu đến siêu phẳng.

Một siêu phẳng phân chia cần phải thỏa mãn điều kiện ràng buộc sau đây:

$$y_i \left[w \cdot x_i + b \right] \geq 1 \text{ với } i = 1,...,l$$

Khoảng cách từ một điểm x_i đến siêu phẳng được tính như sau:

$$d(\mathrm{w},b;x_i)=\frac{|\mathrm{w}\cdot x_i+b|}{\|\mathrm{w}\|}$$

Trong đó $|\mathrm{w}\cdot x_i+b|$: là giá trị tuyệt đối của biểu thức $\mathrm{w}\cdot x_i+b$

$\|\mathrm{w}\|$: là độ dài Ơcơlit của véc tơ w

Một siêu phẳng tối ưu phân lớp tốt nhất là siêu phẳng có giá trị lề lớn nhất gọi $p(\mathrm{w},b)$ là khoảng cách của các lề giữa 2 lớp đến siêu phẳng ta có [17]:

$$\begin{aligned}p(\mathrm{w},b)&=\min_{x_i:y_i=-1}d(\mathrm{w},b;x_i)+\min_{x_i:y_i=1}d(\mathrm{w},b;x_i)\\&=\min_{x_i:y_i=-1}\frac{|\mathrm{w}\cdot x_i+b|}{\|\mathrm{w}\|}+\min_{x_i:y_i=1}\frac{|\mathrm{w}\cdot x_i+b|}{\|\mathrm{w}\|}\\&=\frac{1}{\|\mathrm{w}\|}\left(\min_{x_i:y_i=-1}|\mathrm{w}\cdot x_i+b|+\min_{x_i:y_i=1}|\mathrm{w}\cdot x_i+b|\right)\\&=\frac{2}{\|\mathrm{w}\|}\end{aligned}$$

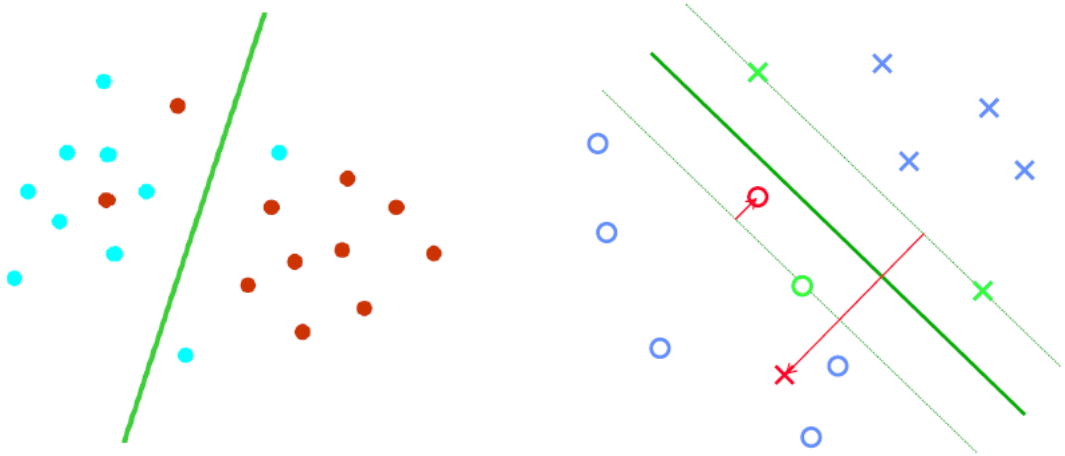
Từ kết quả trên ta có thể thấy một siêu phẳng tối ưu cần tìm là siêu phẳng có $p(\mathrm{w},b)=\frac{2}{\|\mathrm{w}\|}$ đạt giá trị lớn nhất điều này có nghĩa là độ dài $\|\mathrm{w}\|$ phải nhỏ nhất.

Như vậy để tìm được một siêu phẳng tốt nhất trong trường hợp này chúng ta cần phải đi giải quyết bài toán tối ưu với các hàm mục tiêu và ràng buộc như sau:

$$\begin{cases}\underset{\mathrm{w}}{Min}\ \Phi(\mathrm{w})=\frac{1}{2}\|\mathrm{w}\|^2\\ y_i(\mathrm{w}\cdot x_i+b)\geq 1\ \ i=1,..,l\end{cases}$$

2.2.3. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và có xảy ra nhiễu:

Điều này có nghĩa rằng không phải tất cả các điểm dữ liệu thuộc lớp dương (được gán nhãn y = 1) sẽ nằm về một phía của siêu phẳng và tất cả các điểm thuộc lớp âm (được gán nhãn y = -1) sẽ nằm về phía còn lại của siêu phẳng đó. Đôi khi có một vài điểm nhiễm thuộc lớp dương nhưng lại bị siêu phẳng phân loại nhầm vào lớp âm tức là nó nằm về phía bên âm của siêu phẳng đó. Tương tự trong trường hợp dữ liệu âm nằm vào vùng dương của siêu phẳng.



Hình 2.7 Tìm siêu phẳng tối ưu khi dữ liệu tuyến tính có xảy ra nhiễu

Trong trường hợp này các ràng buộc được linh động và thay đổi như sau:

$$y_i\left(\mathrm{w}\cdot x_i+b\right)\geq 1-\xi_i \quad \text{với } i=1,...,l$$

$\xi_i \geq 0$ được gọi là biến số nới lỏng cho các điểm dữ liệu huấn luyện.

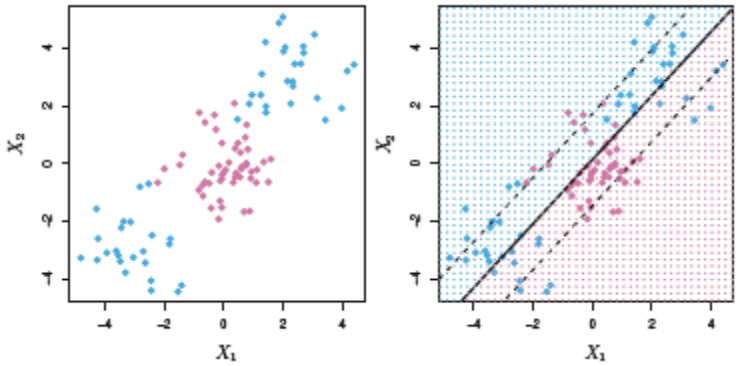
Như vậy lúc này để tìm siêu phẳng tốt nhất chính là đi giải quyết bài toán ưu với các hàm mục tiêu và ràng buộc như sau [4]:

$$\begin{cases} Min\ \Phi(\mathrm{w},\xi)=\frac{1}{2}\|\mathrm{w}\|^2+C\sum_{i=1}^{l}\xi_i \\ y_i(\mathrm{w}\cdot x_i+b) \geq 1-\xi_i \qquad i=1,..,l \\ \xi_i\geq 0 \quad i=1,..,l \end{cases}$$

Trong đó C là tham số được thêm vào với mục đích cân bằng lỗi và để điều chỉnh mức độ vi phạm đối với lỗi trong thực nghiệm.

2.2.4. Bài toán tìm siêu phẳng tối ưu cho dữ liệu không tuyến tính:

Đối với các dữ liệu không thể phân lớp tuyến tính ta có thể thấy rằng nếu áp dụng các phương pháp như đã trình bày ở trên sẽ không thể tìm ra được một siêu phẳng nào có thể phân chia các dữ liệu này tốt nhất.



Hình 2.8 Mô tả dữ liệu không thể phân lớp tuyến tính [8]

Trong thuật toán máy véc tơ hỗ trợ để giải quyết những bài toán với các dữ liệu không thể phân chia tuyến tính giải pháp được đưa ra là ánh xạ các điểm dữ liệu ban đầu sang một không gian mới nhiều chiều hơn, trong không gian mới này chúng ta sẽ dễ dàng tìm được các siêu phẳng để phân tách các dữ liệu này một cách tuyến tính. Việc ánh xạ các điểm dữ liệu x trong không gian n chiều vào một không gian m chiều lớn hơn , được mô tả với một ánh xạ phi tuyến tính ϕ như sau:

$$\phi : R^n \rightarrow R^m$$

Trong không gian mới việc tìm ra các siêu phẳng sẽ dựa trên các điểm ánh xạ $\phi(x)$ các siêu phẳng trong trường hợp này được xác định thông qua việc giải bài toán tối ưu với hàm mục tiêu và các ràng buộc như sau:

$$\begin{cases} Min\ \Phi(\mathrm{w},\xi)=\frac{1}{2}\|\mathrm{w}\|^2+C\sum_{i=1}^{l}\xi_i \\ y_i(\mathrm{w}\cdot\phi(x_i)+b) \geq 1-\xi_i \qquad i=1,..,l \\ \xi_i \geq 0 \quad i=1,..,l \end{cases}$$

2.2.5. Hàm nhân (Kernel)

Trên thực tế đối với thuật toán svm chúng ta không cần phải biết một cách quá rõ ràng các giá trị của một véc tơ dữ liệu trong không gian mới như thế nào, để tìm ra các siêu phẳng tối ưu giá trị cần quan tâm nhất chính là tích vô hướng giữa các véc tơ đặc trưng là ảnh của véc tơ dữ liệu ban đầu trong không gian mới, một hàm cho phép tìm ra các tích vô hướng của các dữ liệu ảnh này được gọi hàm nhân Kernel đây là một hàm hết sức quan trọng trong thuật toán svm:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

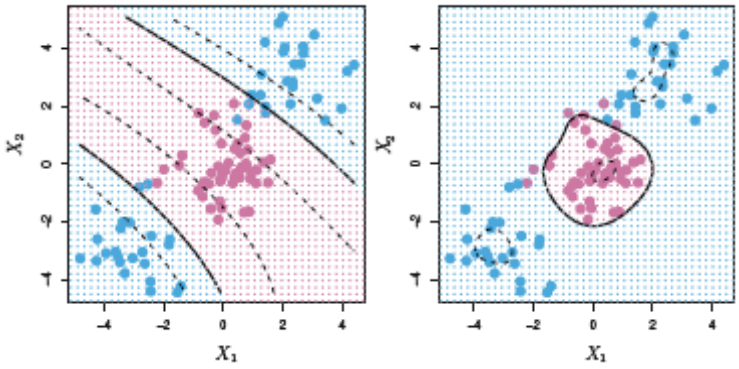
Một số hàm nhân (Kernel) cơ bản [4]:

Hàm nhân tuyến tính (linear): $K(x_i, x_j) = x_i \cdot x_j$

Hàm nhân đa thức (polynomial): $K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d, \gamma > 0$

Hàm RBF (radial basis funciont): $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \ \gamma > 0$

Hàm sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i \cdot x_j + r)$



Hình 2.9 Dữ liệu được phân lớp bởi hàm nhân (kernel) [8]

Trong hình 2.9 Bên trái là một SVM với polynominal kernel bậc 3 được áp dụng để phân lớp dữ liệu phi tuyến tính. Hình bên phải là một SVM với radial kernel được áp dụng. Trong ví dụ này cả hai Kernel đều tỏ ra tương đối thích hợp với việc ra quyết định phân lớp.

CHƯƠNG 3 GIẢI PHÁP ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO PHÁT HIỆN MÃ ĐỘC

3.1. Tổng quan về phương pháp thực hiện

Hiện nay học máy đang là một hướng nghiên cứu được ứng dụng trong nhiều lĩnh vực đặc biệt là trong các bài toán phân lớp dữ liệu. Với mục tiêu xây dựng các hệ thống có khả năng tự động nhận diện mã độc. Trong luận văn này tôi trình bày một hướng nghiên cứu trong đó ứng dụng kỹ thuật học máy vào việc phân lớp và phát hiện mã độc. Luận văn tập trung nghiên cứu và đưa ra một đề xuất cũng như xây dựng một giải pháp trích chọn đặc trưng nâng cao hiệu quả và phù hợp cho các bài toán phân lớp dữ liệu. Quá trình được thực nghiệm và phân tích trên các bộ dữ liệu mã độc chỉ ra rằng phương pháp đề xuất cho kết quả phân lớp chính xác và hiệu suất tương đối tốt.

Các nghiên cứu về hướng tiếp cận học máy trong phát hiện và nhận diện mã độc đã được nhiều tác giả tham gia và nghiên cứu [16] trong đó nổi bật lên là sử dụng các phương pháp phân tích mã độc và trích rút các đặc trưng là các mã byte hay các mã opcode [1] hay sử dụng các hàm gọi đến API [11] Mỗi phương pháp tiếp cận có những ưu điểm và nhược điểm khác nhau, phương pháp dựa vào mã opcode cho kết quả khá tốt tuy nhiên việc dịch ngược ra các mã opcode này đôi khi là khó chính xác cũng như việc tiền xử lý dữ liệu trở nên phức tạp, các phương pháp dựa trên các hàm API lại không bao phủ được đầy đủ các thông tin, phương pháp dựa vào mã byte là một phương pháp hiệu quả dễ dàng trong tiền xử lý tuy nhiên khó khăn trong việc trích chọn các đặc trưng thực sự có hiệu quả. Nhìn chung các phương pháp ở trên đều dựa vào các nguyên lý trong kỹ thuật học máy và khai phá dữ liệu mà đối tượng là các mã độc [15] các nghiên cứu chủ yếu sử dụng phương pháp n-gram [6] [9] và quá trình trích rút đặc trưng dựa vào việc tính tần số xuất hiện của các đặc trưng, tuy nhiên các tập đặc trưng này thường lớn và gây khó khăn cũng như giảm hiệu quả trong quá trình xây dựng mô hình, một số phương pháp chọn đặc trưng như nén các đặc trưng hay giảm số chiều các đặc trưng khá hiệu quả nhưng đôi khi không lọc được các đặc trưng gây nhiễu. Trong nghiên cứu này tôi sẽ đề xuất một giải pháp cho phép lựa chọn các đặc trưng tốt nhất và loại bỏ các đặc trưng có thể gây nhiễu cho mô hình, các đặc trưng này được đánh giá từ chính dữ liệu đầu vào phù hợp với mô hình. Để đơn giản các thực nghiệm và so sánh tôi sử dụng phương pháp n-gram byte trong việc trích xuất các đặc trưng đầu vào.

Trong nghiên cứu này tôi tập trung nghiên cứu các loại mã độc thực thi trên nền

Windows các file mã độc được thu thập từ trang chủ "Vxheaven" [21] và các file mã thông thường là các file hệ thống thuộc kiểu file "PE" chủ yếu là các file ".exe" chạy trên hệ điều hành Windows. Tổng quan về phương pháp được mô tả qua 6 bước sau:

Hình 3.1 Tổng quan về phương pháp thực hiện

Bước 1: Thu thập dữ liệu các file mã độc và các file PE thông thường.

Bước 2: Các file này sẽ được dịch ngược về mã hex thông qua một chương trình được viết bằng ngôn ngữ Python.

Bước 3: Sau khi dịch ngược thì các dữ liệu huấn luyện sẽ được trích rút ra các đặc trưng là các mã hex dựa vào phương pháp n-gram. Trong khuôn khổ nghiên cứu này kích thước sử dụng là 2-gram.

Bước 4: Từ các đặc trưng là các mã n-gram byte thực hiện tính tần số xuất hiện (Term Frequency) của các mã n-gram này trên mỗi tập dữ liệu. Sau đó áp dụng thuật toán được tôi đề xuất (được trình bày ở phần sau) để chọn ra được một bộ đặc trưng tốt nhất.

Bước 5: Từ bộ đặc trưng có được ta đưa chúng vào xây dựng các mô hình học máy ở đây tôi sử dụng hai giải thuật để thực nghiệm và so sánh là cây quyết định và svm.

Bước 6: Sau khi xây dựng xong mô hình thì đưa các dữ liệu test vào để đánh giá kết quả.

3.2. Tiền xử lý dữ liệu

3.2.1. Sử dụng các kỹ thuật phân tích mã độc

Như đã trình bày ở chương 1 hiện nay phân tích mã độc chủ yếu dựa vào 2 phương pháp chính là phân tích động và phân tích tĩnh. Mỗi phương pháp lại đều có những ưu và nhược điểm khác nhau. Phân tích động xác định chính xác mã độc bằng việc quan sát và phân tích các hành vi của nó thông qua việc tạo ra các môi trường thực thi phù hợp cho mã độc, phương pháp này có ưu thế là quá trình phân tích nhanh , dễ dàng ngay cả với những mã độc bị mã hóa phức tạp tuy nhiên lại không hiệu quả khi gặp phải những mã độc chạy theo thời gian hoặc các mã độc có khả năng nhận ra sự theo dõi và sẽ không thực thi các chức năng của nó. Phương pháp phân tích tĩnh dựa vào việc thực hiện dịch ngược mã độc về dạng assembly hay dạng mã hex và phân tích các dấu hiệu của mã độc, phương pháp này tuy gặp khó khăn với những mã độc bị mã hóa hay đóng gói phức tạp nhưng có ưu điểm là có thể phát hiện ra mã độc ngay cả khi không cần thực thi nó và các đặc trưng của mã độc cũng như cấu trúc của nó cho phép phát hiện mã độc một cách chính xác.

Hiện nay đa số quá trình phân tích mã độc còn thủ công tốn nhiều thời gian vì vậy việc ứng dụng học máy để xây dựng các hệ thống có khả năng tự động phát hiện mã độc là điều hết sức cần thiết. Trong nội dung nghiên cứu và thử nghiệm này tôi sử dụng phương pháp phân tích tĩnh, ở giai đoạn tiền xử lý này các file thông thường và các file mã độc là các file dạng PE sẽ được dịch ngược về các mã hex có rất nhiều công cụ cho phép thực hiện điều này có thể kể ra như: IDA, OLLYDBG… Tuy nhiên để đồng bộ và phát triển các hệ thống tự động sau này, tôi đã xây dựng một chương trình dựa trên cấu trúc của một file PE cho phép dịch ngược các file này về mã hex. Chương trình được viết bằng ngôn ngữ Python và sử dụng thư viện Pefile. Các file dữ liệu mẫu sau khi được dịch ngược sẽ được xử lý để lấy các mã hex quan trọng chủ yếu chúng nằm ở các phần PE header và Section nơi chứa các mã chương trình (Executable Code Section), các dữ liệu (Data Section), các tài nguyên (Resources Section), các thư viện (Import Data ,Export Data) …

Các file thông thường và mã độc sau khi được dịch ngược sẽ lấy nội dung các mã hex của từng file này và lưu lại thành các file text tương đương phục vụ cho quá trình trích chọn đặc trưng và xây dựng mô hình dự đoán.

3.2.2. Phương pháp n-gram

Trong nghiên cứu này tôi sử dụng chuỗi các byte là các đặc trưng đầu vào trong đó ở giai đoạn tiền xử lý các file dữ liệu mẫu được trích xuất dựa vào việc tính tần số xuất hiện của các n-gram byte. N-gram là một dãy các byte liên tiếp có độ dài N được mô tả như sau:

Với một dãy các mã hex sau khi dịch ngược giả sử là "AB C0 EF 12" thì dãy các n-gram byte thu được là:

Bảng 3.1 Mô tả dãy các n-gram byte

1-gram	2-gram	3-gram	4-gram
AB	AB C0	AB C0 EF	AB C0 EF 12
C0	C0 EF	C0 EF 12	
EF	EF 12		
12			

Có thể nhận thấy rằng với độ dài n càng cao thì kích thước đặc trưng càng lớn. Đối với mã hex có 16 giá trị khác nhau như vậy không gian đặc trưng của 1-gram sẽ là 16^2=256 với 2-gram là 16^4=65536. Trong nội dung luận văn này tôi chủ yếu tập trung vào phương pháp chọn đặc trưng vì vậy các kết quả được thử nghiệm trên dãy 2-gram. Như vậy ở giai đoạn tiền xử lý các file dữ liệu mẫu được dịch ngược sang các mã hex và sau đó tiếp tục được trích rút ra các đặc trưng dựa vào phương pháp n-gram.

3.2.3. Tính tần số xuất hiện (Term Frequency)

Như chúng ta đã phân tích ở các phần trên, dữ liệu mẫu sau khi thực hiện tiền xử lý sẽ được lưu dưới dạng các file text và được trích rút các dãy n-gram, một trong những phương pháp được biết đến trong các bài toán khai phá và phân lớp dữ liệu text đó là phương pháp tính tần số xuất hiện (TF). Dựa vào phương pháp như vậy ở bước này tôi thực hiện tính toán tần số xuất hiện của mỗi n-gram byte khác nhau trên từng file dữ liệu mẫu. Các kết quả này được lưu như mỗi véc tơ đặc trưng và trước khi được đưa vào mô hình học sẽ được xử lý để trích ra một bộ đặc trưng tốt nhất.

Công thức để tính các giá trị TF được cho như sau:

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

Tần số xuất hiện của một mã n-gram byte trong một tập mẫu (file dữ liệu mẫu đã

được đưa về dạng text) được tính bằng thương của số lần xuất hiện n-gram byte đó trong tập mẫu và số lần xuất hiện nhiều nhất của một n-gram byte bất kỳ trong tập mẫu đó (giá trị sẽ thuộc khoảng [0, 1])

- **f(t,d)** - số lần xuất hiện của một n-gram byte t trong tập mẫu d.
- **max{f(w,d):w∈d}** - số lần xuất hiện nhiều nhất của một n-gram byte bất kỳ trong tập mẫu.

3.3. Đề xuất giải pháp chọn đặc trưng cho thuật toán phân lớp

Trong phương pháp học máy có thể thấy rằng khi số lượng đặc trưng lớn sẽ làm giảm hiệu suất và đôi khi là chất lượng của mô hình học. Lượng đặc trưng quá nhiều sẽ khiến cho quá trình huấn luyện và phân lớp dữ liệu tốn kém về mặt tài nguyên cũng như thời gian xử lý thậm chí nếu nhiều đặc trưng phổ biến sẽ đẫn đến dư thừa gây nhiễu và ảnh hướng đến chất lượng khi xây dựng mô hình, chính vì vậy bài toán đặt ra và cần thiết là làm sao loại bỏ được các đặc trưng gây nhiễu và chọn được một tập đặc trưng đại diện tốt nhất mà vẫn đảm bảo độ chính xác hiệu quả của mô hình dự đoán. Trong phương pháp n-gram nêu trên ta thấy sẽ có các đặc trưng mà tần số xuất hiện của chúng tương tự nhau trên 2 lớp vì vậy khi đưa vào mô hình học máy sẽ không đạt kết quả cao.

Phương pháp trích chọn đặc trưng mà tôi đề xuất mục đích là tìm ra tập các đặc trưng mà giá trị tần số xuất hiện trung bình của chúng trên 2 lớp cần phân chia có độ lệch lớn nhất. Cụ thể là các mã n-gram byte mà có tần số xuất hiện trên các tập của lớp này khác nhất với chính nó trên các tập của lớp còn lại.

3.3.1. Mô tả giải pháp

Gọi D là tập các đặc trưng có độ dài "d" phần tử là các mã n-gram byte. 2 lớp cần phân chia lớp thứ 1 có độ dài là "n" tập mẫu. Lớp thứ 2 có độ dài là "m" tập mẫu.

Gọi TF1[i] là tập chứa các giá trị tần số xuất hiện của đặc trưng D[i] ∈ D; i ϵ [0,d] trên các mẫu dữ liệu thuộc lớp thứ 1. Mỗi TF1[i] với i ϵ [0,d] là một mảng chứa 'n' phần tử .

Gọi TF2[i] là tập chứa các giá trị tần số xuất hiện của đặc trưng D[i] ∈ D; i ϵ [0,d] trên các mẫu dữ liệu thuộc lớp thứ 2. Mỗi TF2[i] với i ϵ [0,d] là một mảng chứa 'm' phần tử.

Bước 1: Với mỗi giá trị i ϵ [0,d] sắp xếp các phần tử trong TF1[i] và TF2[i] theo chiều giảm hoặc tăng dần

Bước 2: Sau khi thực hiện sắp xếp ở bước 1:

Với mỗi i ∈ [0,d] ta thực hiện chia các phần tử trong TF1[i] tương ứng thành "k" đoạn liên tiếp bắt đầu từ phần tử đầu tiên, mỗi đoạn chứa C1 phần tử (số phần tử trong các đoạn có thể không bằng nhau).

Tương tự với mỗi i ∈ [0,d] ta cũng thực hiện chia các phần tử trong TF2[i] tương ứng thành "k" đoạn liên tiếp bắt đầu từ phần tử đầu tiên, mỗi đoạn chứa C2 phần tử (số phần tử trong các đoạn có thể không bằng nhau)

Bước 3: Tính trung bình cộng tần số xuất hiện trên từng khoảng đã chia trên mỗi TF1[i]; i ∈ [0,d] với mỗi TF1[i] ta thu được "k" giá trị TB1[i][j] với i ϵ [0,d]; j∈[0,k] là giá trị tần số xuất hiện trung bình của D[i] trên đoạn j của TF1[i]

Tương tự thực hiện tính trung bình cộng tần số xuất hiện trên từng khoảng đã chia trên mỗi TF2[i]; i ∈ [0,d] với mỗi TF2[i] ta thu được "k" giá trị TB2[i][j] với i ϵ [0,d]; j∈[0,k] là giá trị tần số xuất hiện trung bình của D[i] trên đoạn j của TF2[i]

Bước 4: Tính độ lệch tần số xuất hiện trung bình trên "k" đoạn đã chia của mỗi TF1[i] và TF2[i].

Gọi TB[i][j] là độ lệch trung bình của đặc trưng D[i] trên đoạn j của 2 tập TF1[i] và TF2[i] thì TB[i][j] được tính bằng trị tuyệt đối của phép trừ giữa TB1[i][j] và TB2[i][j] ta có:

$$TB[i][j] = | TB1[i][j] - TB2[i][j] |$$

$$(i \epsilon [0,d]; j \epsilon [0,k])$$

Với mỗi i trện đoạn [0,d] thực hiện tính các TB[i][j] của nó với mỗi j ∈ [0,k] trên 2 tập TF1[i] và TF2[i] tương ứng.

Bước 5: Với mỗi i ϵ [0,d] ta thực hiện tính độ lệch trung bình giá trị tần số của đặc trưng D[i] tương ứng trên toàn tập bằng cách tính sau:

$$DL[i] = \frac{\sum_{j=0}^{k} TB[i][j]}{k}$$

Trong đó DL[i] với i ∈[0,d] là độ lệch trung bình của đặc trưng D[i] trên 2 lớp cần phân chia.

Bước 6: Kết thúc bước 5 ta sẽ thu được kết quả độ lệch trung bình tần số xuất hiện của "d" đặc trưng ban đầu từ "d" đặc trưng này thực hiện chọn ra một bộ đặc trưng có giá trị độ lệch cao nhất để đưa vào xây dựng mô hình.

3.3.2. Ví dụ:

Bài toán: Giả sử cho 2 lớp với các đặc trưng là các mã hex 2-gram có số tập mẫu khác nhau và tần số xuất hiện như sau:

Bảng 3.2 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 1

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Tập 1	0.5	0.7	1.0	0.2
	Tập 2	0.5	0.9	0	0.1
	Tập 3	0.6	0.5	1.0	0.2
	Tập 4	0.6	0.6	0	0.1

Bảng 3.3 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 2

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Tập 1	0.6	0.2	0.5	0.1
	Tập 2	0.5	0.3	0.5	0.2
	Tập 3	0.6	0.3	0.5	0.7
	Tập 4	0.5	0.2	0.5	0.8
	Tập 5	0.6	0.3	0.5	0.3
	Tập 6	0.5	0.2	0.5	0.2

Bước 1: sau khi đã tính tần số xuất hiện của các đặc trưng trên mỗi tập mẫu thuộc 2 lớp thực hiện sắp xếp các giá trị theo chiều giảm dần ở mỗi đặc trưng trong cả 2 lớp như sau:

Bảng 3.4 Kết quả tập dữ liệu lớp 1 sau khi thực hiện sắp xếp

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
		0.6	0.9	1.0	0.2
		0.6	0.7	1.0	0.2
		0.5	0.6	0	0.1
		0.5	0.5	0	0.1

Bảng 3.5 Kết quả tập dữ liệu lớp 2 sau khi thực hiện sắp xếp

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
		0.6	0.3	0.5	0.8
		0.6	0.3	0.5	0.7
		0.6	0.3	0.5	0.3
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.1

Bước 2: Tiến hành chia lớp 1 và lớp 2 mỗi lớp thành k đoạn. Giả sử ta chọn k=2 như vậy với lớp 1 mỗi đoạn sẽ có 2 giá trị tần số liên tiếp cạnh nhau, với lớp 2 mỗi đoạn có 3 giá trị tần số liên tiếp cạnh nhau.

Bảng 3.6 Kết quả tập dữ liệu lớp 1 sau khi thực hiện phân đoạn

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0.6	0.9	1.0	0.2
		0.6	0.7	1.0	0.2
	Đoạn 2	0.5	0.6	0	0.1
		0.5	0.5	0	0.1

Bảng 3.7 Kết quả tập dữ liệu lớp 2 sau khi thực hiện phân đoạn

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0.6	0.3	0.5	0.8
		0.6	0.3	0.5	0.7
		0.6	0.3	0.5	0.3
	Đoạn 2	0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.1

Minh họa với đặc trưng "BB BB" ta có:

Bảng 3.8 Minh họa với đặc trưng "BB BB" thuộc lớp 1 sau khi thực hiện phân đoạn

Lớp 1	Đặc Trưng	BB BB
	Đoạn 1 (chứa 2 phần tử)	0.9
		0.7
	Đoạn 2 (chứa 2 phần tử)	0.6
		0.5

Bảng 3.9 Minh họa với đặc trưng "BB BB" thuộc lớp 2 sau khi thực hiện phân đoạn

Lớp 2	Đặc Trưng	BB BB
	Đoạn 1 (chứa 3 phần tử)	0.3
		0.3
		0.3
	Đoạn 2 (chứa 3 phần tử)	0.2
		0.2
		0.2

Bước 3: Trên mỗi đoạn thực hiện tính ra giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng :

Minh họa với đặc trưng “BB BB”:

- Giá trị tần số xuất hiện trung bình của nó trên đoạn 1 của lớp 1 sẽ là (0.9+0.7)/2=0.8
- Giá trị tần số xuất hiện trung bình của nó trên đoạn 2 của lớp 1 sẽ là : (0.6+0.5)/2 = 0.55
- Giá trị tần số xuất hiện trung bình của nó trên đoạn 1 của lớp 2 sẽ là: (0.3+0.3+0.3)/3=0.3
- Giá trị tần số xuất hiện trung bình của nó trên đoạn 2 của lớp 2 sẽ là: (0.2+0.2+0.2)/3=0.2

Tương tự cách tính như vậy với các đặc trưng còn lại thu được bảng sau:

Bảng 3.10 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 1

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
TF trung bình trên từng đoạn	Đoạn 1	0.6	0.8	1.0	0.2
	Đoạn 2	0.5	0.55	0	0.1

Bảng 3.11 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 2

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
TF trung bình trên từng đoạn	Đoạn 1	0.6	0.3	0.5	0.6
	Đoạn 2	0.5	0.2	0.5	0.16

Bước 4: Thực hiện tính độ lệch của tần số xuất hiện giữa các đoạn của từng đặc trưng trên 2 lớp, bằng cách trừ lấy trị tuyệt đối 2 tần số trên cùng một đoạn của cùng 1 đặc trưng trên 2 lớp dữ liệu.

Minh họa với đặc trưng “ BB BB” thì:

- Độ lệch tần số giữa 2 lớp của đặc trưng “ BB BB” trên đoạn 1 là: |0.8 – 0.3|=0.5
- Độ lệch tần số giữa 2 lớp của đặc trưng “ BB BB” trên đoạn 2 là: |0.55-0.2|=0.35

Thực hiện tương tự với các đặc trưng khác thu được bảng sau:

Bảng 3.12 Kết quả độ lệch tần số xuất hiện trên từng đoạn

	Đặc trưng	AA AA	BB BB	CC CC	DD DD
Độ lệch TF trên từng đoạn	Đoạn 1	0	0.5	0.5	0.4
	Đoạn 2	0	0.35	0.5	0.06

Bước 5: Từ kết quả độ lệch tần số xuất hiện có được trên các đoạn ở bước 4 tiến hành tính trung bình độ lệch tần số xuất hiện của mỗi đặc trưng trên toàn tập, bằng cách tính trung bình cộng độ lệch tần số xuất hiện trên các đoạn của mỗi đặc trưng để ra kết quả cuối cùng.

Minh họa với đặc trưng "BB BB" thì độ lệch tần số trung bình trên cả toàn dữ liệu là: (0.5+0.35)/2=0.425

Thực hiện tương tự với các đặc trưng còn lại thu được bảng kết quả sau:

Bảng 3.13 Kết quả độ lệch tần số xuất hiện trên toàn tập dữ liệu

	Đặc trưng	AA AA	BB BB	CC CC	DD DD
Độ lệch TF trên toàn dữ liệu	Đoạn 1	0	0.425	0.5	0.23

Như vậy dễ thấy các đặc trưng có độ lệch cao nhất là "CC CC" và "BB BB" sau đó là "DD DD" ta có thể xác định được các đặc trưng này có sự sai khác lớn về tần số xuất hiện của 2 lớp đúng theo thứ tự.

Bước 6: Từ các kết quả có được thực hiện chọn ra T đặc trưng có độ lệch về giá trị tần số xuất hiện lớn nhất.

3.4. Xây dựng mô hình dự đoán dựa trên các thuật toán phân lớp

Sau khi chọn được tập đặc trưng dựa vào phương pháp đã trình bày phía trên, bước tiếp theo ở giai đoạn này là tiến hành đưa các giá trị tần số xuất hiện của các đặc trưng n-gram byte đã được chọn trên các tập mẫu ban đầu vào các thuật toán trong học máy để xây dựng mô hình dự đoán., các dữ liệu mẫu được gán nhãn thành 2 lớp mã độc và file bình thường. Giai đoạn này sử dụng các véc tơ đặc trưng như là dữ liệu huấn luyện đầu vào cho các thuật toán phân lớp dữ liệu [13,19] có thể áp dụng như: k láng giềng (KNN), Naive Bayes (NB), cây quyết định (DT), Rừng ngẫu nhiên (RF), máy véc tơ hỗ trợ (SVM)… Trong phương pháp học máy giám sát khó có thể nói là thuật toán phân lớp nào tốt nhất vì mỗi thuật toán có những ưu, nhược điểm riêng phù hợp cho mỗi loại dữ

liệu khác nhau. Riêng đối với các bài toán phân lớp dữ liệu văn bản thì cây quyết định và SVM là 2 thuật toán được cho là có độ chính xác cao và được sử dụng khá rộng rãi hiện nay. Trong đó thuật toán máy véc tơ hỗ trợ (SVM) được đánh giá là rất phù hợp và hiệu quả trong các bài toán phân lớp dữ liệu văn bản, trong khi đó cây quyết định (DT) là thuật toán dễ hiểu, dựa vào việc sinh luật khá gần gũi tư duy của con người .Trong nghiên cứu này tôi lựa chọn 2 thuật toán tiêu biểu là cây quyết định và máy véc tơ hỗ trợ để xây dựng mô hình đự đoán phát hiện mã độc đồng thời các kết quả của 2 phương pháp sẽ được so sánh và đánh giá thông qua quá trình thực nghiệm.

CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ

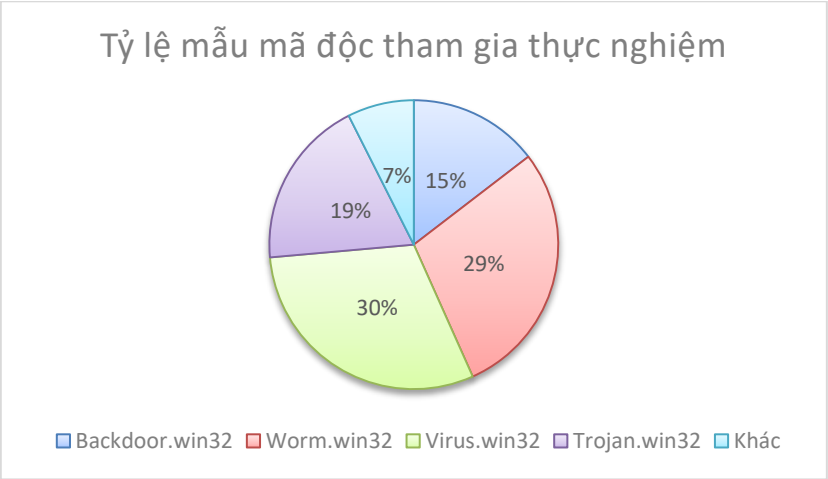
4.1. Dữ liệu thực nghiệm

Các kết quả của luận văn được thực nghiệm với đầu vào là 4698 file dữ liệu mẫu trong đó 2373 file mã thông thường và 2325 file mã độc, file mã độc được thu thập từ trang chủ "Vxheaven" [21] và các file mã thông thường là các file hệ thống thuộc kiểu file "PE" chủ yếu là các file ".exe" chạy trên trên hệ điều hành Windows số lượng các mẫu mã độc được mô tả như sau:

Bảng 4.1 Tỷ lệ số lượng từng mẫu mã độc tham gia thực nghiệm

Kiểu mã độc	Backdoor.win32	Worm.win32	Virus.win32	Trojan.win32	Khác
Số lượng mẫu	339	668	704	441	173

Trong đó tỷ lệ các mẫu mã độc được biểu diễn trực quan trong biểu đồ sau:



Hình 4.1 Biểu đồ mô tả tỷ lệ mẫu mã độc tham gia thực nghiệm

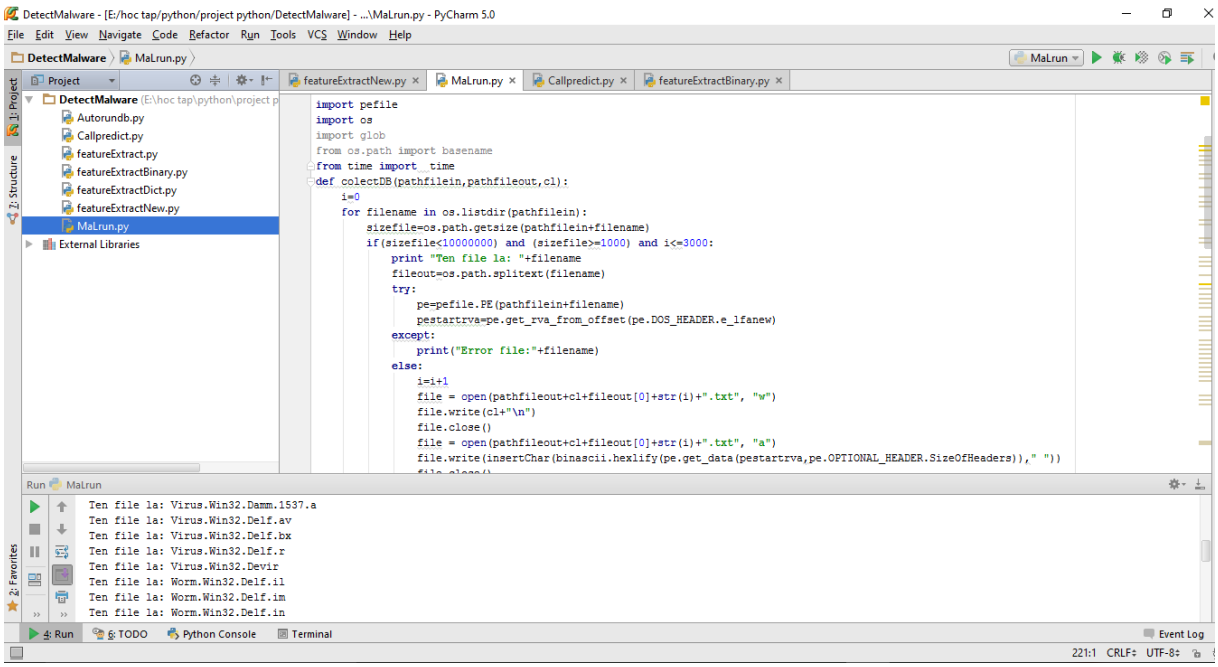
4.2. Chương trình thực nghiệm

Bảng 4.2 Thông tin về chương trình thực nghiệm

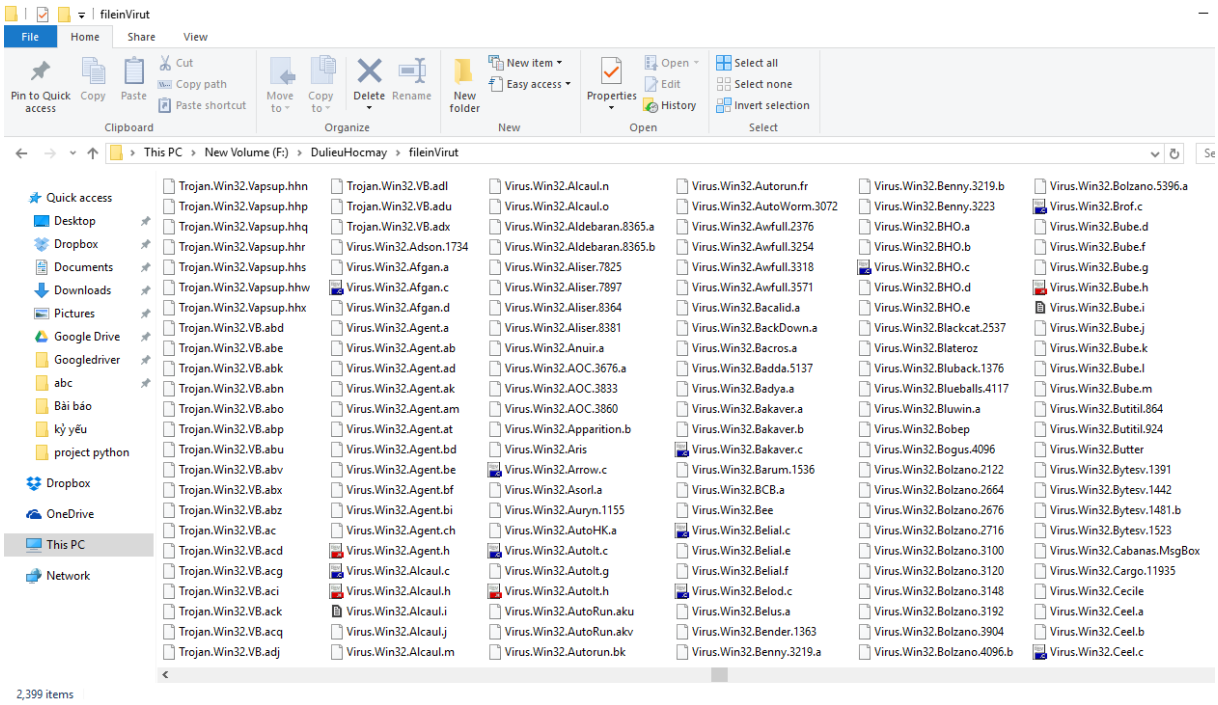
Môi trường thực nghiệm	- Processor: Intel(R) Core i5 – 2450M CPU @ 2.50GHz - Memory (Ram): 8.00 GB - System type: 64-bit Operating System, x64-based processor - Windows 10 Pro -2016
Ngôn ngữ sử dụng	Python
Công cụ lập trình (IDE)	PyCharm – JetBrains 5.0 Pro

Thư viện và các gói cài đặt chính	Scikit-learn, scipy, numpy,matplotlib,ipython, Pefile

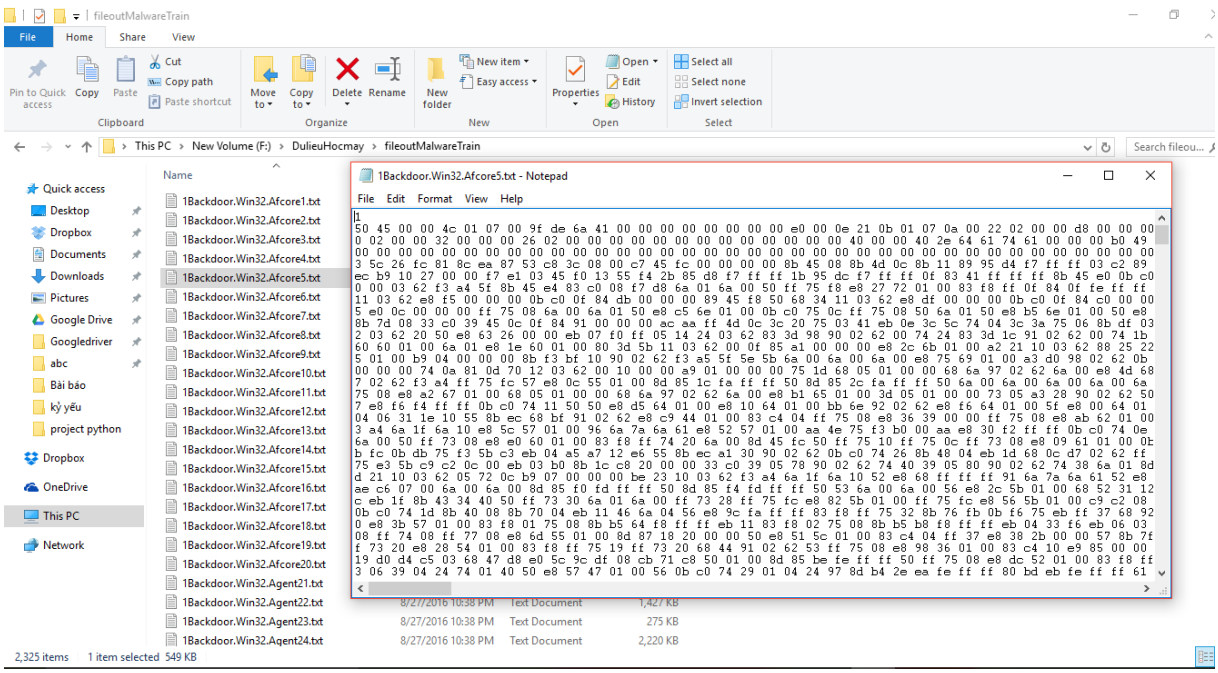
Chương trình thực nghiệm được lập trình trên ngôn ngữ Python trong đó 2 mã nguồn chính là chương trình thực hiện đọc, phân tích cấu trúc một tập tin PE và tiến hành dịch ngược về mã hex, chương trình thứ 2 là chương trình đọc các dữ liệu mẫu, chạy các thuật toán trích chọn đặc trưng và chạy các thuật toán học máy:



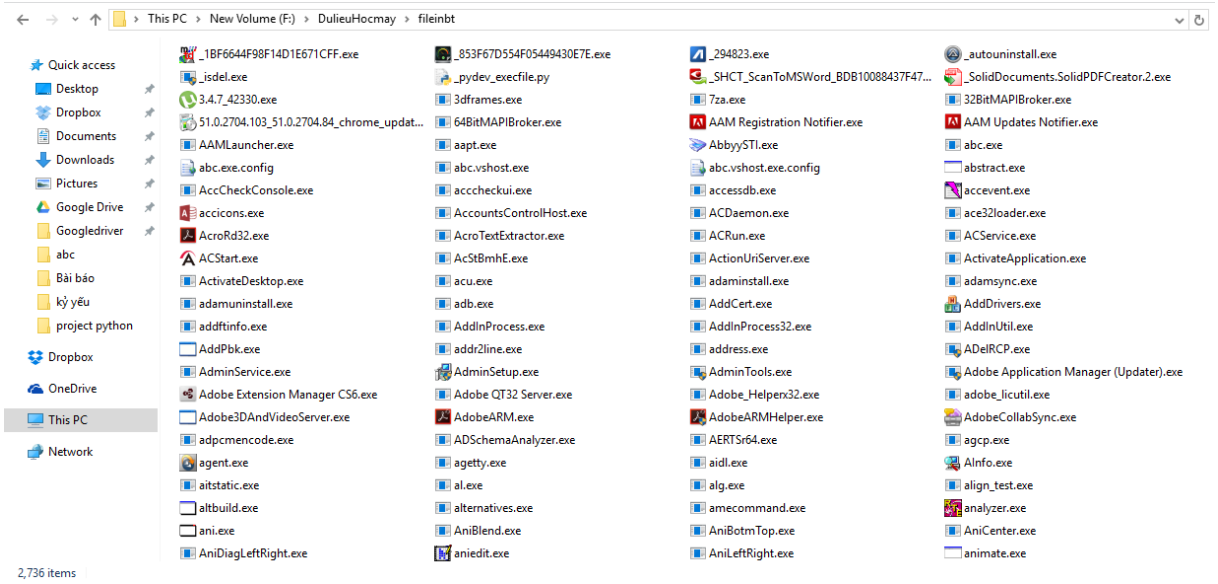
Hình 4.2 Chương trình đọc, phân tích cấu trúc các file PE và dịch ngược



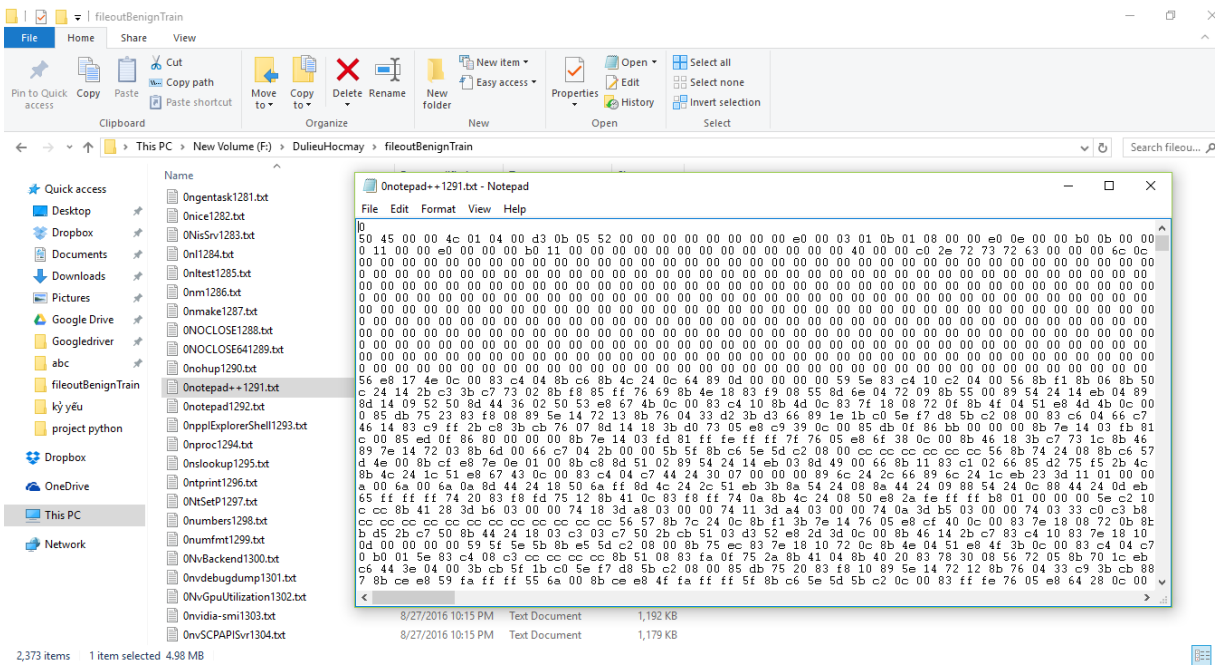
Hình 4.3 Tập dữ liệu mẫu mã độc ban đầu



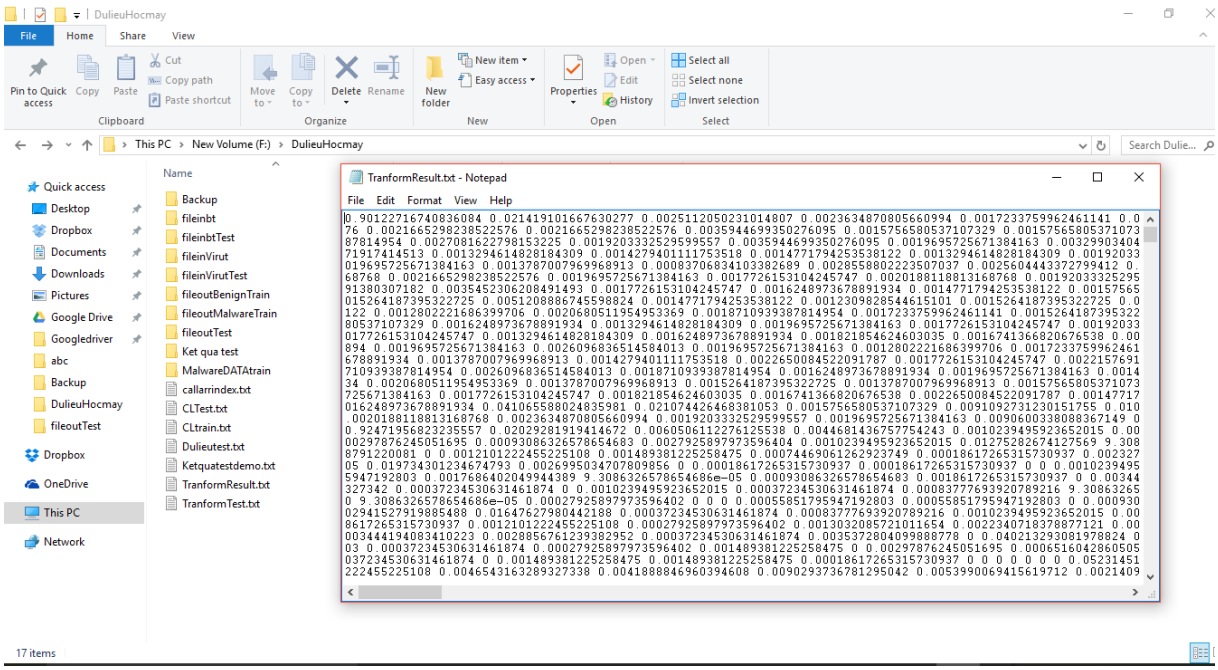
Hình 4.4 Tập dữ liệu mẫu mã độc sau khi được dịch ngược về mã hex và được gán nhãn phân lớp



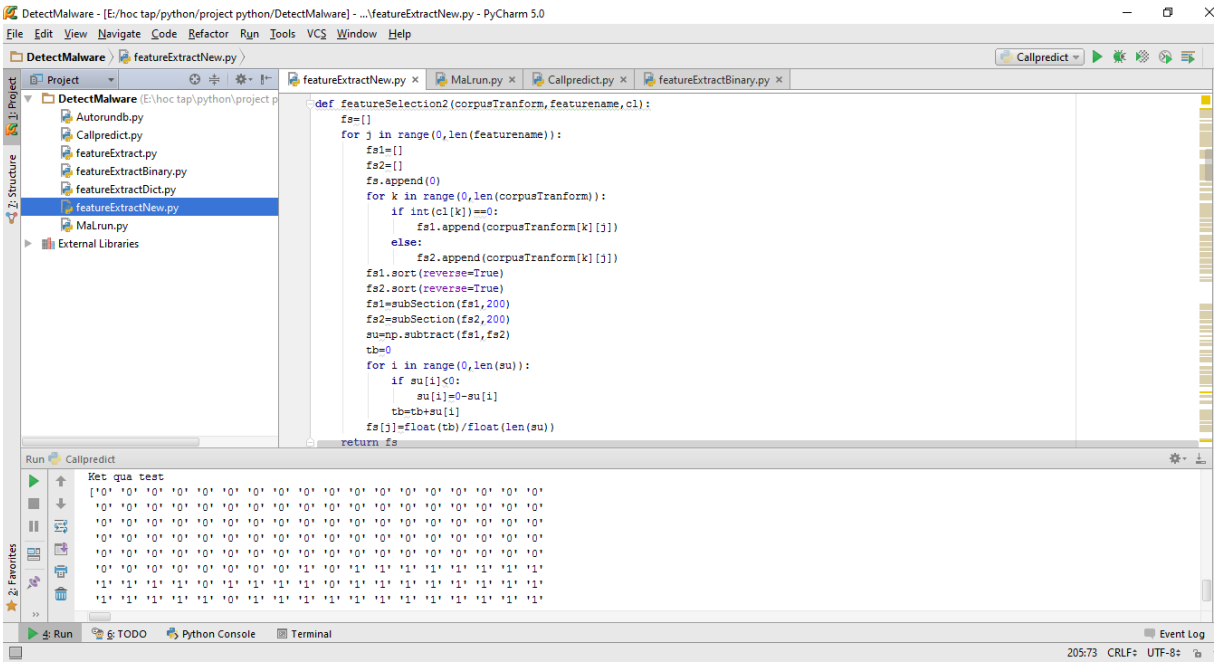
Hình 4.5 Tập dữ liệu mẫu các file thực thi thông thường trên Windows



Hình 4.6 Tập dữ liệu mẫu các file thực thi thông thường sau khi được dịch ngược về mã hex và được gán nhãn phân lớp



Hình 4.7 Tập các tần số xuất hiện của dữ liệu huấn luyện



Hình 4.8 Chương trình trích chọn đặc trưng và xây dựng mô hình dự đoán mã độc

4.3. Đánh giá dựa trên phương pháp ma trận nhầm lẫn

Một mô hình được đánh giá tốt nếu như tỉ lệ: TP (True Positive) và TN (True Negative) lớn đồng thời tỉ lệ FP (False Positive) và FN (False Negative) là nhỏ trong đó:

TP: Số lượng các mẫu thuộc lớp dương được phân loại chính xác vào lớp dương.

FP: Số lượng các mẫu không thuộc lớp dương bị phân loại nhầm vào lớp dương

TN: Số lượng các mẫu không thuộc lớp dương được phân loại đúng

FN: Số lượng các mẫu thuộc lớp dương bị phân loại nhầm vào các lớp không phải lớp dương.

		Lớp dự đoán (Predicted class) +	Lớp dự đoán (Predicted class) −
Lớp thực (True class)	+	True Positive-TP	False Negative-FN Type II error
	−	(False Positive-FP) Type I error	True Negative-TN

Độ chính xác của mô hình được tính bằng công thức sau:

$$Acc=(TP+TN)/(TP+TN+FP+FN)$$

TPR (True positive rate) được tính bằng công thức sau:

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

FPR (False positive rate) được tính bằng công thức sau:

$$FPR = \frac{FP}{FP+TN}$$

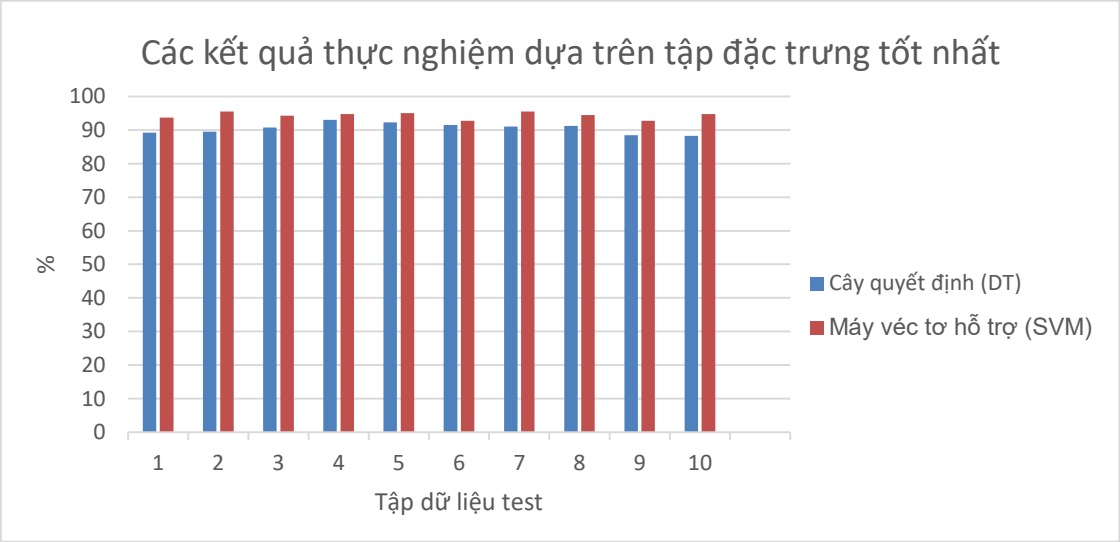
4.4. Kết quả thực nghiệm

Tiến hành trích rút các đặc trưng 2-gram byte như vậy toàn bộ không gian đặc trưng sẽ là 16^4=65536 đặc trưng, sau đó áp dụng thuật toán đã đề xuất tôi thực hiện thu gọn và trích chọn ra 800 đặc trưng tốt nhất. Trong tổng số 4698 file thực hiện chọn ra 400 file test (trong đó 200 file mã độc và 200 file thông thường) các file còn lại độc lập với dữ liệu test được dùng làm dữ liệu huấn luyện. Tiến hành kiểm tra chéo trên các tập dữ liệu bằng cách chọn ngẫu nhiên 400 file test khác nhau với các file huấn luyện độc lập còn lại và thực hiện kiểm tra nhiều lần, các kết quả được đáng giá dựa trên phương pháp ma trận nhầm lẫn trong đó coi lớp dương (Positive) là mã độc , lớp âm (Negative) là file thông thường, các kết quả thu được mô tả trong bảng sau:

Bảng 4.3 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng tốt nhất

n-gram	Số đặc trưng	Mô hình	Độ chính xác	TPR	FPR
2	800	DT	90.75%	0.90	0.085
2	800	SVM	94.50%	0.945	0.055

Trong đó các kết quả chi tiết thực nghiệm trên các tập dữ liệu test khác nhau như sau:



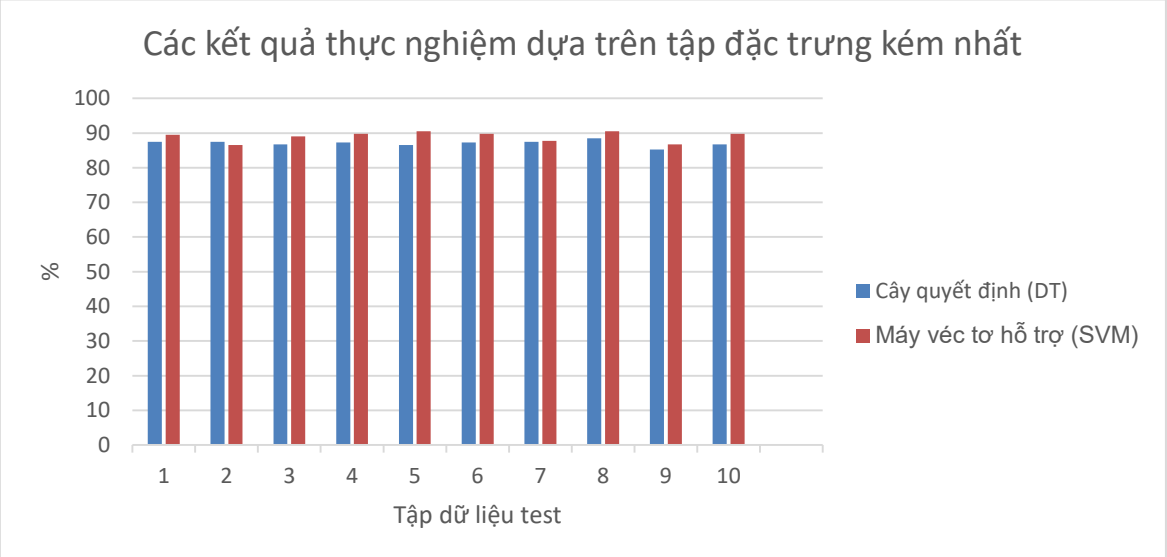
Hình 4.9 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng được đánh giá tốt nhất

Để có thể so sánh được hiệu quả của giải pháp chọn đặc trưng đã đề xuất. Tôi tiến hành kiểm tra trên cùng tập dữ liệu và sử dụng 800 đặc trưng khác bị đánh giá kém nhất (có độ lệch tần số xuất hiện thấp nhất) để xây dựng mô hình học, các kết quả kiểm tra thu được như sau:

Bảng 4.4 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng kém nhất

n-gram	Số đặc trưng	Mô hình	Độ chính xác	TPR	FPR
2	800	DT	87.3%	0.855	0.11
2	800	SVM	88.5%	0.93	0.16

Trong đó các kết quả chi tiết thực nghiệm trên các tập dữ liệu test khác nhau như sau:



Hình 4.10 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng bị đánh giá kém nhất

Nhận xét: Như vậy dựa vào các kết quả thực nghiệm có thể thấy tập đặc trưng được đánh giá cao mà thuật toán đã tìm ra cho kết quả tốt hơn rất nhiều so với tập đặc trưng bị đánh giá thấp.

KẾT LUẬN

Luận văn đã trình bày một hướng tiếp cận có hiệu quả trong đó ứng dụng kỹ thuật học máy vào việc nhận diện và phát hiện mã độc. Đồng thời đề xuất một giải pháp chọn đặc trưng để có thể gia tăng hiệu suất mà vẫn đảm bảo độ chính xác và hiệu quả của mô hình dự đoán, các kết quả được tiến hành thực nghiệm và đánh giá trên các tệp dữ liệu chạy trên nền Windows cho thấy với kỹ thuật chọn đặc trưng tôi đã đề xuất thì các đặc trưng tìm được cho kết quả mô hình dự đoán tốt hơn, đồng thời trong thực nghiệm này thuật toán học máy véc tơ hỗ trợ (SVM) tỏ ra hiệu quả hơn so với thuật toán học cây quyết định (DT). Với kết quả độ chính xác là 94.5% khi sử dụng thuật toán học SVM đây là một kết quả khá khả quan đối với yêu cầu cần giải quyết các bài toán phát hiện mã độc hiện nay.

❖ *Các kết quả chính đạt được của luận văn:*

- Tìm hiểu về mã độc các phương pháp phân tích, phân loại và phát hiện mã độc
- Trình bày cơ sở lý thuyết và thực nghiệm các thuật toán phân lớp trong kỹ thuật học máy giám sát bao gồm 2 thuật toán là cây quyết định và SVM
- Nghiên cứu và thực hiện việc ứng dụng kỹ thuật học máy vào giải quyết bài toán phát hiện mã độc.
- Đề xuất được giải pháp lựa chọn đặc trưng tốt nhất đảm bảo hiệu quả, hiệu suất của mô hình học máy trong bài toán phát hiện mã độc.
- Tiến hành thực nghiệm và đánh giá, so sánh các kết quả.

❖ *Hướng phát triển tiếp theo của nghiên cứu:*

Mở rộng phương pháp với nhiều hướng tiếp cận phân tích mã độc hơn như phân tích động, phân tích dựa trên kinh nghiệm từ đó phối hợp với các thuật toán học máy vào quá trình xây dựng các hệ thống có khả năng tự động nhận diện với đa dạng mã độc một cách chính xác, đồng thời tiếp tục nghiên cứu, phối hợp phát triển và áp dụng nhiều hơn các kỹ thuật trích chọn đặc trưng nhằm nâng cao hiệu quả hiệu suất của mô hình dự đoán.

DANH MỤC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN VĂN

1. Văn Trường Võ, Văn Vinh Nguyễn, Minh Đức Trịnh and Khánh Dương Lê. "*Đề xuất giải pháp trích chọn đặc trưng cho các thuật toán phân lớp dữ liệu trong kỹ thuật học máy giám sát và ứng dụng hiệu quả vào bài toán phát hiện mã độc".* Hội thảo lần thứ I, Một số vấn đề chọn lọc về an toàn an ninh thông tin (1st Symposium on Information Security - SoIS 2016). Đã được chấp nhận đăng trong kỷ yếu và trình bày tại hội thảo, pp. 35 -42

TÀI LIỆU THAM KHẢO

[1] Asaf Shabtai, Robert Moskovitch, Clint Feher, Shlomi Dolev and Yuval Elovici "Detecting unknown malicious code by applying classification techniques on OpCode patterns," Security Informatics 2012 1:1. doi:10.1186/2190-8532-1-1

[2] A. Shabtai, R. Moskovitch, Y. Elovici, C.Glezer,: "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," Information Security Technical Report 2009.

[3] Cai DM, Gokhale M, Theiler J "Comparison of feature selection and classification algorithms in identifying malicious executables,". Computational Statistics and Data Analysis 2007.

[4] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin "A Practical Guide to Support Vector Classification", Department of Computer Science National Taiwan University- Taipei 106- Taiwan, Last updated: May 19, 2016

[5] Digit Oktavianto, Iqbal Muhardianto,"Cuckoo Malware Analysis", Packt Publishing, 2013.

[6] D Krishna Sandeep Reddy - Arun K Pujari "N-gram analysis for computer virus detection," Springer-Verlag France 2006, doi 10.1007/s11416-006-0027-8

[7] Eldad Eilam, "Reversing-Secrets of Reverse Engineering", Wiley; 1 edition (April15, 2005)

[8] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani "An Introduction to Statistical Learning with Applications in R (Springer Texts in Statistics) ", by Springer Publishing, 2013

[9] Igor Santos, Yoseba K. Penya, Jaime Devesa and Pablo G. Bringas "n-grams-based file signatures for malware detection," Deusto Technological Foundation, Bilbao, Basque Country

[10] John Aycock, "Computer viruses and Malware (Advances in Information Security)", by Springer Publishing, 2006

[11] Madhu K. Shankarapani - Subbu Ramamoorthy - Ram S. Movva - Srinivas Mukkamala "Malware detection using assembly and API call sequences," Springer-Verlag France 2010, doi 10.1007/s11416-010-0141-5

[12] Michael Sikorski and Andrew Honig, “Practical Malware Analysis”, No Starch Press, 1 edition (March 3, 2012)

[13] Peter Harrington, “Machine Learning in Action,” in Part 1 Classification, by Manning Publications, 2012 , pp. 1–129.

[14] Peter Szor, “The Art of Computer Virus Reseach and Defense”, Addison Wesley Professional, 2005.

[15] Schultz M, Eskin E, Zadok E, Stolfo S “Data mining methods for detection of new malicious executables,” Proc of the IEEE Symposium on Security and Privacy, IEEE Computer Society 2001.

[16] Smita Ranveer, Swapnaja Hiray, “Comparative Analysis of Feature Extraction Methods of Malware Detection,” International Journal of Computer Applications (0975 8887), Volume 120 - No. 5, June 2015

[17] Steve R. Gunn ,“Support Vector Machines for Classification and Regression”, Technical Report, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, 1998

[18] Tom M. Mitchell, “Machine Learning” , by McGraw-Hill Science, 1997

[19] Trent Hauck, “scikit-learn Cookbook”, in Chapter 4 Classifying Data with scikit-learn, by Packt Publishing, 2014, pp. 119-157

[20] V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, NY, 1995.

[21] VXheavens Website:url:http://vx.netlux.org.