Hướng tiếp cận phát hiện mã độc dựa trên phân tích tĩnh kết hợp thuật toán học máy

Nguyễn Đức Việt*

Học viện Công nghệ bưu chính Viễn thông.
*Email: vietnd@ptit.edu.vn
Nhận bài: 10/7/2023; Hoàn thiện: 15/9/2023; Chấp nhận đăng: 10/10/2023; Xuất bản: 25/10/2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.90.2023.134-139

TÓM TẮT

Kỹ thuật tấn công phát tán mã độc thông qua người dùng rồi từ đó, leo thang lên hệ thống ngày càng được nhiều kẻ tấn công ưu thích sử dụng. Do đó, để phát hiện mã độc thì hướng tiếp cận phát hiện mã độc dựa trên hành vi với sự hỗ trợ của các thuật toán học máy đã mang lại nhiều hiệu quả cao. Mặt khác, trong thực tế những kẻ tấn công thường tìm nhiều cách thức và kỹ thuật khác nhau nhằm che giấu hình vi của mã độc dựa trên Portable Executable File Format (PE File) của mã độc. Điều này đã gây ra nhiều khó khăn cho quá trình phát hiện mã độc của các hệ thống giám sát. Từ những lý do trên, trong bài báo này, chúng tôi đề xuất phương pháp phát hiện mã độc dựa trên kỹ thuật phân tích tĩnh PE File sử dụng thuật toán học máy.

Từ khóa: Mã độc; Phát hiện mã độc; Phân tích tĩnh; Thuật toán học máy; Hành vi bất thường.

1. MỞ ĐẦU

Mã độc là các phần mềm được thiết kế một cách có chủ đích, dùng để gây thiệt hại tới máy tính cá nhân, máy chủ hoặc hệ thống mạng máy tính [1, 2]. Mục đích của mã độc là thực thi 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, 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 [1]. Trong nghiên cứu [1] đã liệt kê một số loại mã độc phổ biến bao gồm: Virus, Worm, Trojan Horse, Malicious Mobie Code, Tracking Cookie, Attacker Tool, Phishing, Virus Hoax. Theo thống kê tại [5] thì tình hình phát tán mã độc trong năm 2020 tăng 75% so với năm 2019. Điều này hoàn toàn hợp lý bởi vì những kẻ tấn công trước kia thường chỉ tập trung tấn công vào hệ thống thông tin nhưng ngày này chúng thường chọn cách tấn công vào người dùng là chủ yếu. Chính vì vậy, mã độc không chỉ tăng nhanh về số lượng tấn công mà còn cả về mức độ nguy hiểm của chúng. Trong nghiên cứu [2, 6-8] đã liệt kê một số hướng tiếp cận để phát hiện mã độc bao gồm phát hiện dựa trên chữ ký và phát hiện dựa trên hành vi. Đối với phương pháp phát hiện dựa trên chữ ký là phân tích tĩnh, phân tích mã nguồn mà không cần thực thi tệp tin [9]. Một số kỹ thuật dùng trong phân tích tĩnh bao gồm [9]: Kiểm tra định dạng tệp; Trích xuất chuỗi; Lưu vết;... Đối với phương pháp phát hiện dựa trên hành vi là dựa trên phân tích động. Phương pháp này sẽ đánh giá một đối tượng dựa trên hành vi của nó. Khi một đối tượng cố gắng thực thi các hành vi bất thường hoặc không được cấp quyền biểu thị đối tượng đó độc hại hoặc đáng ngờ. Có một số hành vi được coi là nguy hiểm như vô hiệu hóa các điều khiển bảo mật, cài đặt rootkits, autostart, sửa tệp tin, thiết lập các kết nối đáng ngờ,… Mỗi hành vi có thể không nguy hiểm nhưng kết hợp với nhau có thể làm tăng độ đáng ngờ của đối tượng. Có một ngưỡng được xác định sẵn, nếu bất kỳ tệp tin nào vượt qua ngưỡng này sẽ được cảnh báo là mã độc [10-13]. Phương pháp này được áp dụng để phát hiện các loại mã độc có khả năng thay đổi chữ ký (đa hình) hoặc các loại mã độc mới (zero-day). Tuy nhiên, một số loại mã độc có khả năng phát hiện môi trường ảo, nó sẽ không thực thi các hành vi độc hại trong môi trường sandbox [13]. Hơn nữa, trên thực tế, với lượng mã độc đang ngày một gia tăng, phương pháp này không thực sự hiệu quả trước các loại mã độc mới. Chính vì vậy, trong bài báo này, chúng tôi đề xuất phương pháp phát hiện mã độc dựa trên kỹ thuật phân tích PE File

[14] sử dụng thuật toán học máy có giám sát bao gồm rừng ngẫu nhiên (Random forest -RF), và máy học(Support Vector Machine- SVM).

2. CÁC NGHIÊN CỨU LIÊN QUAN

Dragos Gavrilut [10] đã đề xuất hệ thống phát hiện mã độc dựa trên các thuật toán perceptron cải tiến. Với các thuật toán khác nhau, độ chính xác dao động trong khoảng 69.90% - 96.18%. Tuy nhiên, thuật toán có độ chính xác cao nhất cũng có nhiều kết quả dương tính sai nhất. Thuật toán cân đối nhất có tỉ lệ dương tính sai thấp và có độ chính xác là 93.01%.

Singhal và Raul đã thảo luận về phương pháp phát hiện dựa trên thuật toán RF cải tiến kết hợp với độ lợi thông tin (Information Gain) để biểu diễn đặc trưng tối ưu hơn [11]. Tập dữ liệu được tác giả sử dụng chỉ bao gồm tệp tin thực thi, vì thế việc trích chọn đặc trưng đơn giản hơn. Tỉ lệ phát hiện là 97% và tỉ lệ dương tính sai là 0.03.

Baldangombo và cộng sự đã giới thiệu phương pháp trích chọn đặc trưng dựa trên tiêu đề PE, các thư viện DLL và các hàm chức năng API [12]. Các thuật toán được sử dụng bao gồm Naïve Bayes, Cây quyết định J48, và SVM. Thuật toán có kết quả tốt nhất là J48, với tỉ lệ chính xác lên tới 99%.

Alazab [13] đã đề xuất phương pháp sử dụng API để biểu diễn đặc trưng của mã độc. Thuật toán SVM cho kết quả tốt nhất với tỉ lệ 97.6%, tỉ lệ dương tính sai là 0.025.

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

3. PHƯƠNG PHÁP PHÁT HIỆN MÃ ĐỘC DỰA TRÊN PHÂN TÍCH TĨNH

3.1. Danh sách đặc trưng bất thường của mã độc dựa trên phân tích tĩnh

Trong nghiên cứu [14] đã trình bày chi tiết về PE File. Trong bài báo này, chúng tôi sẽ xem xét và trích xuất ra một số thuộc tính thể hiện hành vi của mã độc trong PE header sử dụng thư viện LEIF. Bảng 1 dưới đây liệt kê các 75 hành vi của mã độc được trích xuất dựa trên các thành phần khác nhau trong PE header.

Bảng 1. *Danh sách thuộc tính hành vi mã độc trong PE header.*

STT	Tên	Mô tả
1	pe.has_configuration	Chứa địa chỉ và kích thước cấu hình tải
2	pe.has_debug	Địa chỉ và kích thước điểm bắt đầu debug
3	pe.has_exceptions	tính năng xử lý ngoại lệ
4	pe.has_exports	Xuất ký tự đặc biệt
5	pe.has_imports	Nhập ký tự đặc biệt
6	pe.has_nx	Vùng bộ nhớ để sử dụng bằng cách lưu trữ các lệnh của bộ xử lý
7	pe.has_relocations	Địa chỉ và kích thước bảng di dời cơ sở
8	pe.has_resources	Các tài nguyên được lập chỉ mục
9	pe.has_rich_header	Cấu trúc ngay sau MZ DOS header
10	pe.has_signature	Chữ ký số
11	pe.has_tls	một lớp lưu trữ đặc biệt mà Windows hỗ trợ
12	64 bit đầu tiên của Entry point	Hàm này nằm trong IMAGE_OPTIONAL_HEADER và chứa địa chỉ ảnh cơ sở
…		
75		

3.2. Thuật toán phân loại mã độc

Trong bài báo này, chúng tôi sẽ sử dụng một số thuật toán học máy và học sâu nhằm phân loại các tệp tin bình thường và tệp tin độc hại. Theo đó, các thuật toán chúng tôi lựa chọn sử dụng gồm: RF, SVM. Trong các tài liệu [15-18] đã mô tả chi tiết cơ sở toán học và nguyên tắc hoạt động của các thuật toán này. Trong bài báo này, chúng tôi sẽ tiến hành áp dụng các thuật toán trong nhiệm vụ phát hiện mã độc. Dựa trên các kết quả thực nghiệm, chúng tôi sẽ có cơ sở để đánh giá sự hiệu quả của từng thuật toán trong nhiệm vụ phát hiện mã độc.

4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Bộ dữ liệu và kịch bản thực nghiệm

4.1.1. Bộ dữ liệu thực nghiệm

Trong bài báo này, chúng tôi sử dụng bộ dữ liệu về mã độc và file bình thường được cung cấp tại [19]. Cụ thể, bộ dữ liệu gồm 49128 bản ghi, trong đó có 24528 bản ghi mã độc và 24602 bản ghi bình thường. Các bản ghi mã độc và bản ghi sạch đều được lựa chọn và trích xuất thành các trường và các thành phần đã được liệt kê trong bảng 1.

4.1.2. Kịch bản thử nghiệm

a) Đối với bộ dữ liệu thực nghiệm

Dựa trên bộ dữ liệu thực nghiệm đã được thu thập và mô tả như trong mục 4.1.1, chúng tôi sẽ tiến hành trộn đều và chia ngẫu nhiên, trong đó 80% số lượng bản ghi trong bộ dữ liệu sẽ được sử dụng trong quá trình training và 20% còn lại của bộ dữ liệu sẽ được dùng trong quá trình kiểm thử.

b) Đối với thuật toán phân loại

Chúng tôi sẽ sử dụng 5 thuật toán khác nhau để tiến hành thực nghiệm trên bộ dữ liệu đã được trình bày ở trên. Để đánh giá hiệu quả của từng thuật toán, chúng tôi sẽ tiến hành thực nghiệm trên từng thuật toán với sự thay đổi trong tham số của chúng. Mục đích của chúng tôi là muốn đánh giá và tìm ra được thuật toán hiệu quả nhất cũng như tham số nào tối ưu nhất trong thuật toán đó. Cụ thể, chúng tôi tiến hành tinh chỉnh các tham số của thuật toán như sau:

- Đối với thuật toán RF, chúng tôi sẽ tiến hành thực nghiệm và đánh giá thuật toán dựa trên sự thay đổi số lượng cây quyết định lần lượt là: 20, 30, 50, 70, 100.
- Đối với thuật toán SVM các tham số về Kernel được lựa chọn bao gồm: rbf, linear, sigmoid, polynomial.

4.2. Các phương pháp đánh giá một hệ thống

- Accuracy: là tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử**.** Có thể được tính bằng công thức sau:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

- Recall: Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là *positive*. Được tính bằng công thức:

$$\text{Recall} = TP / (TP + FN) \quad (2)$$

- Precision: Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là *positive*. Được tính bằng công thức:

$$\text{Precision} = TP / (TP + FP) \quad (3)$$

- F1 Score: F1-score, là *harmonic mean* của precision và recall (giả sử rằng hai đại lượng này khác không):

Trong đó:

+ True Positive (TP): Cả giá trị thực tế và dự đoán đều là Dương.
+ True Negative (TN): Cả giá trị thực tế và giá trị dự đoán đều là Âm.

+ False Positive (FP): Giá trị thực tế là âm nhưng dự đoán là giá trị dương.
+ False Negative (FN): Giá trị thực tế là dương dự đoán là âm.

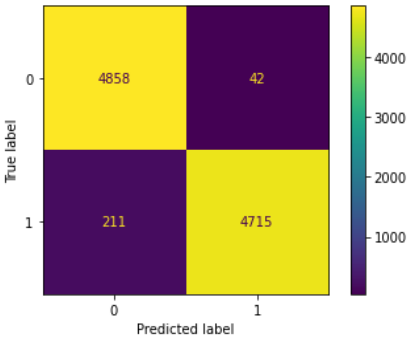
4.3. Kết quả thực nghiệm

a) Thuật toán Random Forest

Bảng 2. *Kết quả thực nghiệm thuật toán RF.*

N_estimator	Accuracy	Precision	Recall	F1_score
20	97.02	98.82	95.19	96.97
30	97.07	98.78	95.33	97.02
50	97.16	98.95	95.35	97.12
70	97.25	98.89	95.59	97.21
100	**97.62**	**99.10**	**96.123**	**97.59**

Từ kết quả thử nghiệm trên bảng 2, chúng tôi nhận thấy rằng, độ chính xác của thuật toán RF tăng dần khi số lượng cây quyết định tăng lên. Thuật toán cho kết quả phân loại tốt nhất với tất cả các độ với số lượng cây quyết định là 100. Quá trình phân loại đạt kết quả tốt nhất với Accuracy, Precision, Recall, F1_score lần lượt là 97.62; 99.10; 96.123; 97.59 tại cây quyết định là 100. Bên cạnh đó, kết quả phân loại file bình thường của thuật toán đạt tương đối cao từ 98.82% đến 99.10% còn kết quả khi phân loại mã độc chỉ đạt từ 95.19% đến 96.123%. Kết quả này tương đối tốt vì bộ dữ liệu thực nghiệm cân bằng về số lượng mã độc và file bình thường. Hình 1 dưới đây thể hiện kết quả khi kiểm thử mô hình phát hiện mã độc khi sử dụng thuật toán RF với số lượng cây quyết định là 100.



Hình 1. *Confusion Matrix Random Forest.*

Từ hình 1 có thể thấy thuật toán đã dự đoán sai 211 mã độc và 42 file bình thường. Kết quả này có thể chấp nhận được khi mà bộ dữ liệu có số lượng lớn các file độc và bình thường.

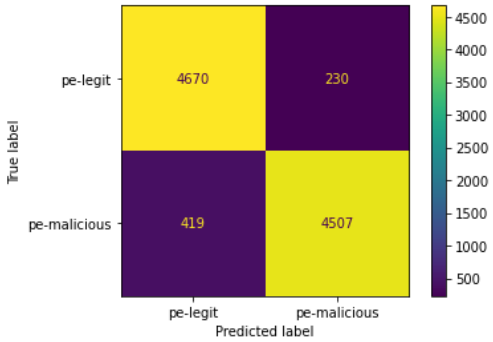
b) Thuật toán SVM

Bảng 3 dưới đây thể hiện kết quả phát hiện mã độc sử dụng thuật toán SVM.

Bảng 3. *Kết quả thực nghiệm phát hiện mã độc sử dụng thuật toán SVM.*

Kernel	C	Accuracy	F1_Score	Recall	Precision
RBF	1	93.40	93.40	93.40	93.46
	10	95.47	95.47	95.48	95.51
	100	95.77	95.78	95.78	95.78
LINEAR	1	87.06	87.06	87.07	87.14
POLYNOMIAL	1	92.45	92.45	92.45	92.45
	10	95.13	95.12	95.13	95.15
SIGMOID	1	49.56	49.56	49.56	49.56
	10	49.31	49.31	49.31	49.31

Các kết quả thực nghiệm trong bảng 3 cho thấy với 486 thuộc tính của file PE sử dụng thuật toán SVM chúng tôi thu được kết quả với độ chính xác là 95,77%. Dễ thấy với kernel mặc định của thuật toán – RBF (C=100.0) cho ra độ chính xác cao nhất so với các kernel còn lại. Đối với kernel Sigmoid kết quả cho ra là khá thấp chỉ ~50%. Với kết quả này thì thuật toán SVM không thực sự phù hợp với bộ dữ liệu phát hiện mã độc dựa trên file PF này. Bên cạnh đó, với kernel mặc định của thuật toán – RBF (C=100.0) cho ra độ chính xác cao nhất so với các kernel còn lại. Hình 2 dưới đây thể hiện kết quả đánh giá quá trình kiểm thử mô hình với thuật toán SVM với tham số RBF (C = 100.0).



Hình 2. *Confusion Matrix SVM với kernel RBF/C=100.0.*

Từ hình 2 chúng ta có thể thấy kết quả với bộ dữ liệu kiểm thử như sau: thuật toán dự đoán được 4507 file mã độc, dự đoán sai 230 file bình thường thành file mã độc và dự đoán thiếu 419 file mã độc. Dựa trên kết quả thực nghiệm trong bảng 3, chúng tôi nhận thấy thuật toán RF mang lại hiệu quả cao hơn so với thuật toán SVM.

5. KẾT LUẬN

Trong bài báo này, dựa trên kỹ thuật phân tích PE File chúng tôi đã đề xuất một số thuộc tính thể hiện hành vi bất thường của mã độc. Các kết quả thực nghiệm trong các bảng 2 và 3 đã chứng minh một số thuộc tính được trích xuất trên PE File do chúng tôi lựa chọn và đề xuất đã mang lại hiệu quả tốt không chỉ với quá phân loại chính xác các file bình thường mà còn với file mã độc. Bên cạnh đó, dựa trên các kết quả thực nghiệm các thuật toán RF và SVM với các tham số khác nhau chúng tôi đã chứng minh được thuật toán RF mang lại hiệu quả tốt hơn về mọi mặt so với thuật toán SVM. Đặc biệt, trong bộ dữ liệu mã độc này, số lượng thuộc tính tương đối nhiều (75 thuộc tính) thì thuật toán RF với số lượng cây quyết định là 100 đã mang lại hiệu quả tốt nhất. Trong tương lai, để nâng cao hiệu quả quá trình phát hiện mã độc dựa trên kỹ thuật phân tích PE File chúng tôi cần cải thiện 2 vấn đề chính bao gồm: i) trích xuất thêm thuộc tính của mã độc dựa trên PE File. Chúng tôi nhận thấy rằng, PE File gồm nhiều thành phần khác nhau và có nhiều thành phần quan trọng được những kẻ phát triển mã độc lợi dụng để che giấu thông tin về hành vi mã độc. Chính vì vậy, việc phân tích chi tiết và tổng quát hóa các thuộc tính này sẽ cải thiện đáng kể hiệu quả quá trình phát hiện mã độc trong bối cảnh mã độc ngày càng phát hiện về cả số lượng và hình thức phát tán. ii) sử dụng các thuật toán học máy tiên tiến khác: rõ ràng các thuật toán học máy cổ điển đã mang lại hiệu quả tốt cho quá trình phân loại nhưng trước tình hình thực tế về sự tăng nhanh về số lượng hành vi mã độc cũng như số lượng dữ liệu thực nghiệm thì cần phải có những thuật toán phân loại tiên tiến khác thì mới có thể đảm bảo được hiệu quả của quá trình giám sát và phát hiện.

TÀI LIỆU THAM KHẢO

[1]. Daniel Gibert, Carles Mateu, Jordi Planes. *"The rise of machine learning for detection and classification of malware: Research developments, trends and challenges"*. Journal of Network and Computer Applications. vol. 153. pp 1-22, (2020).

[2]. Ucci, Daniele & Aniello, Leonardo. *"Survey on the Usage of Machine Learning Techniques for Malware Analysis"*. Computers & Security. 81. 10.1016/j.cose.2018.11.001, (2017).
[3]. Sanjay Sharma, C. Rama Krishna, Sanjay K. Sahay. *"Detection of Advanced Malware by Machine Learning Techniques"*. arXiv:1903.02966. (2019).
[4]. Alireza Souri, Rahil Hosseini. *"A state-of-the-art survey of malware detection approaches using data mining techniques"*. Human-centric Computing and Information Sciences 8(1):1-22.
[5]. Kaspersky-Lab. *"Machine Learning Methods for Malware Detection"*. (2020).
[6]. R. Islam, R. Tian, L. M. Batten, S. Versteeg, *"Classification of malware based on integrated static and dynamic features"*, Journal of Network and Computer Applications 36 (2) 646–656, (2013).
[7]. C.-T. Lin, N.-J. Wang, H. Xiao, C. Eckert, *"Feature selection and extraction for malware classification"*, Journal of Information Science and Engineering 31 (3) 965–992, (2015).
[8]. A. Mohaisen, O. Alrawi, M. Mohaisen, Amal: *"High-fidelity, behavior-based automated malware analysis and classification"*, Computers & Security 52, 251–266, (2015).
[9]. S. Palahan, D. Babi´c, S. Chaudhuri, D. Kifer, *"Extraction of statistically significant malware behaviors"*, in: Computer Security Applications Conference, ACM, pp. 69–78, (2013).
[10]. Gavrilut, Dragos, M. Cimpoesu, D. Anton, L. Ciortuz. *"Malware Detection Using Machine Learning"*. The International Multiconference on Computer Science and Information Technology, (2009).
[11]. Priyank Singhal, Nataasha Raul. *"Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks"*, (2015).
[12]. Baldangombo Usukhbayar, Nyamjav Jambaljav, Shi-Jinn Horng. *"A Static Malware Detection System Using Data Mining Methods"*. Cornell University, (2013).
[13]. Alazab, Mamoun, Sitalakshmi Venkatraman, Paul Watters, and Moutaz Alazab. *"Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures"*. Proceedings of the 9-th Australasian Data Mining Conference, 171-181, (2011).
[14]. Nakajima, Tatsuo & Ishikawa, Hiroo & Kinebuchi, Yuki & Sugaya, Midori & Lei, Sun & Courbot, Alexandre & Zee, Andrej & Aalto, Aleksi & Duk, Kwon. *"An Operating System Architecture for Future Information Appliances"*. 292-303. (2008). 10.1007/978-3-540-87785-1_26.
[15]. C. Corinna, V. Vladimir. *"Support-vector networks. Machine Learning"*. Vol 20, pp. 273-297, (1995).
[16]. S.S. Shai, B.D. Shai. *"Understanding Machine Learning: From Theory to Algorithms"*. Cambridge University Press, (2014).
[17]. JohnShawe-Taylor, ShiliangSun. *"Kernel Methods and Support Vector Machines"*. Academic Press Library in Signal Processing Vol 1, pp. 857-881, (2014).
[18]. Leo Breiman. *"Random Forests"*. Machine Learning. vol. 45, Issue 1, pp 5–32. (2001).
[19]. How to create a malware detection system with machine learning. https://www.evilsocket.net/2019/05/22/How-to-create-a-Malware-detection-system-with-MachineLearning/?fbclid=IwAR1vuaOJA3UryaQATPsqKErktLft2RtzzAB5kDvgOTo4U3dF4J-Op9teokQ

ABSTRACT

Detecting malicious code based on static analysis combined with machine learning algorithms

The technique of spreading malicious code through users and then escalating it into the system is increasingly favored by many attackers. Therefore, to detect malicious code, the approach of behavior-based malware detection with the support of machine learning algorithms has proven to be highly effective. On the other hand, in practice, attackers often employ various methods and techniques to conceal the characteristics of malicious code based on the Portable Executable File Format (PE File). This has posed significant challenges for the detection of malware by monitoring systems. For these reasons, in this article, we propose a method for detecting malicious code based on static analysis of PE Files using machine learning algorithms.

Keywords: Malware; Malware detection; Static analysis; Machine learning algorithms; Abnormal behavior.