Explainable Machine Learning for Cybersecurity

Vuong Dinh Thanh Ngan*†, Le Minh Nha*†
†Vietnam National University, Ho Chi Minh City, Vietnam 20521649@gm.uit.edu.vn, 20521690@gm.uit.edu.vn

***Abstract*—Nhận dạng phần mềm độc hại trên Android bằng Machine Learning là một chủ đề phổ biến trong an ninh di động. Các nghiên cứu đã chứng minh rằng học máy là phương pháp hiệu quả để phát hiện phần mềm độc hại, đạt độ chính xác lên tới 99%. Tuy nhiên, các nghiên cứu trước đây đã chỉ ra rằng các thiết kế thực nghiệm không thực tế gây ra thiên vị và kết quả không thực tế. Nghiên cứu này sử dụng phương pháp XAI để giải thích hoạt động của các mô hình Machine Learning và phát hiện rằng sự không nhất quán thời gian trong dữ liệu huấn luyện dẫn đến kết quả phân loại quá lạc quan. Kết quả cũng cho thấy rằng các mô hình Machine Learning phân loại dựa trên thời gian chứ không phải hành vi độc hại thực sự. Đánh giá cũng xác nhận rằng thiết kế thực nghiệm không thực tế không chỉ làm giảm độ chính xác mà còn làm giảm độ tin cậy, gây khó khăn cho việc áp dụng thực tế. Do đó, phương pháp XAI nên được sử dụng để hiểu rõ hơn về hoạt động của các mô hình AI/ML, không chỉ tập trung vào việc cải thiện độ chính xác.**

***Index Terms*—Machine Learning, AI, Android Malware, XAI**

I. Introduction

Phần mềm độc hại vẫn là một trong những mối đe dọa nghiêm trọng nhất trong không gian mạng, bất chấp sự cải tiến của các cơ chế an ninh mạng. Số lượng mẫu phần mềm độc hại ngày càng tăng, với khoảng 1,5 tỷ mẫu vào năm 2020, và các loại phần mềm độc hại khác nhau đang lan rộng trên không gian mạng, gây thiệt hại tài chính lớn cho cá nhân và ngành công nghiệp. Để đối phó với mối đe dọa này, phát hiện phần mềm độc hại dựa trên máy học (ML) và học sâu (DL) đã trở thành một lĩnh vực nghiên cứu quan trọng trong những năm gần đây. Nhờ huấn luyện trên các bộ dữ liệu lớn, các mô hình ML/DL có thể tự động phân biệt phần mềm độc hại và phần mềm không xác định. Các nghiên cứu đã chứng minh rằng các kỹ thuật ML/DL có thể đạt được hiệu suất phát hiện cao, với độ chính xác lên đến 99%, với ít khả năng cải thiện hơn. Tuy nhiên, các phương pháp hiệu suất cao này có vẻ ít khả thi hơn trong thực tế, vì vấn đề phòng chống phần mềm độc hại vẫn là một thách thức khó khăn và các ứng dụng độc hại tiếp tục tăng cường mối đe dọa. Các nghiên cứu trước đây đã chỉ ra rằng các mô hình phát hiện phần mềm độc hại dựa trên ML dễ bị sai lệch trong các thiết kế thử nghiệm, dẫn đến hiệu suất không thực tế. Việc loại bỏ các thành kiến kinh nghiệm trực quan cũng có thể làm giảm đáng kể hiệu suất của các mô hình này. Đánh giá của tác giả về nghiên cứu gần đây cho thấy rằng nhiều nghiên cứu không xem xét thiết kế thử nghiệm thực tế, mặc dù đạt được độ chính xác cao trong việc phát hiện phần mềm độc hại trên Android dựa trên ML. [1]

II. Related work

Nhiều nghiên cứu đã lo ngại về sự lạc quan quá mức của các kỹ thuật phát hiện phần mềm độc hại dựa trên học máy (ML). Để giải quyết vấn đề này, đã có nhiều phương pháp giải thích dự đoán của các mô hình ML phát hiện phần mềm độc hại (tính giải thích cục bộ).

A. Các phương pháp của Drebin, Xmal, Fan và cộng sự

1) Drebin với Linear Support Vector Machines: Phương pháp Drebin [2]đã sử dụng Máy Vector Hỗ trợ tuyến tính (SVM - Support Vector Machines) để phân loại ứng dụng không xác định là độc hại hay không. Để huấn luyện mô hình phát hiện phần mềm độc hại dựa trên SVM, kỹ thuật SVM tuyến tính tạo ra một siêu mặt phẳng để phân tách phần mềm độc hại và không độc hại trong dữ liệu huấn luyện. Để phát hiện hoạt động độc hại của một ứng dụng không xác định, Drebin sử dụng một biểu diễn toàn diện và nhẹ của ứng dụng di động. Drebin trích xuất các đặc trưng từ hai nguồn chính: tệp manifest (AndroidManifest.xml) và mã dex đã được tháo rời. Sau đó, thông tin này được mã hóa dưới dạng vector sử dụng kỹ thuật mã hóa one-hot. Mô hình phát hiện phần mềm độc hại dựa trên SVM được áp dụng để phân loại xem một ứng dụng không xác định trong dữ liệu kiểm tra có phải là phần mềm độc hại hay không. Drebin cũng tạo ra lời giải thích cho mỗi dự đoán bằng cách nhân trọng số đặc trưng của bộ phân loại tuyến tính với giá trị đặc trưng thực tế của ứng dụng kiểm tra.

Drebin tạo ra lời giải thích cho mỗi dự đoán bằng cách sử dụng $w_i = w * v_i$, một phép nhân giữa trọng số đặc trưng (w) của bộ phân loại tuyến tính và giá trị thực tế của đặc trưng (v) của trường hợp kiểm tra đó (i). Điều này có nghĩa là mỗi đặc trưng được định danh và có một trọng số ứng với nó trong mô hình phân loại. Khi áp dụng mô hình cho một ứng dụng kiểm tra, Drebin tính toán tích của trọng số và giá trị đặc trưng tương ứng để xác định đóng góp của đặc trưng đó đối với kết quả phân loại.

Việc này giúp giải thích lý do tại sao một ứng dụng được phân loại là độc hại hoặc không độc hại bằng cách xem xét đóng góp của các đặc trưng cụ thể trong quá trình phân loại.

2) XMal với Attention Mechanism: Phương pháp XMal [3] đã sử dụng một mạng perceptron đa tầng (MLP - Multilayers Perceptron) với cơ chế chú ý để phân loại phần mềm độc hại và có khả năng giải thích dự đoán. Tương tự như phương pháp Drebin [1], phương pháp XMal sử dụng các tập đặc trưng liên quan đến cuộc gọi API và quyền hạn. Vì có rất nhiều đặc

trưng có thể có (tức là hơn 20.000), phương pháp XMal chỉ chọn ra 154 đặc trưng hiệu quả nhất (bao gồm 94 cuộc gọi API và 60 quyền hạn) để huấn luyện mô hình. Phương pháp XMal bao gồm hai tầng: tầng chú ý và mạng perceptron đa tầng (MLP). Đầu tiên, một vectơ đặc trưng được tạo ra bằng cách sử dụng kỹ thuật mã hóa one-hot với kích thước 158. Sau đó, vectơ đặc trưng được đưa vào tầng chú ý. Tầng chú ý sử dụng cơ chế chú ý được đề xuất bởi Bahdanau et al [4], được sử dụng để bắt lấy mối quan hệ giữa các đặc trưng trong dãy đầu vào và các đặc trưng đầu ra tiếp theo, cho phép mô hình giữ lại tất cả thông tin của dãy đầu vào. Để chính xác, vector chú ý:

$$\alpha_i = (\alpha^{(1)}, \alpha^{(2)}, ..., \alpha^{(j)})$$

biểu thị trọng số chú ý của đặc trưng thứ j cho trường hợp kiểm tra thứ i được tính thông qua hàm softmax:

$$\alpha_i^{(j)} = \frac{\exp(e_i^{(j)})}{\sum_{k=1}^{n} \exp(e_i^{(k)})}$$

trong đó,

$$\alpha_i^{(j)}$$

biểu thị trọng số chú ý của đặc trưng thứ j cho trường hợp kiểm tra thứ i,

$$e_i^{(k)}$$

là vectơ đặc trưng của trường hợp kiểm tra thứ i. Sau đó, vectơ chú ý được đưa vào tầng Multi-layer Perceptron (MLP) để ánh xạ các trọng số đặc trưng thành phân loại nhị phân. Cuối cùng, lời giải thích cho mỗi dự đoán được tạo ra dựa trên trọng số chú ý để chỉ ra các đặc trưng đóng góp nhiều nhất cho dự đoán.

3) Fan và cộng sự với Model-Agnostic Explainable Approaches: Phương pháp của Fan và cộng sự [4] đã đánh giá năm phương pháp giải thích cục bộ và không phụ thuộc vào mô hình (gọi tắt là LIME [5], Anchor [6], LORE [7], SHAP [8] và LEMNA [9]) cho phân tích phần mềm độc hại trên Android. Khác với các phương pháp giải thích cụ thể cho từng mô hình, các phương pháp giải thích không phụ thuộc vào mô hình có thể giải thích bất kỳ mô hình học máy nào.Trong đó, LIME (Local Interpretable Model-Agnostic Explanations) là một phương pháp giải thích cục bộ và không phụ thuộc vào mô hình. Nó được sử dụng để giải thích các dự đoán của các mô hình học máy hộp đen (black-box) một cách dễ hiểu. Phương pháp này xấp xỉ ranh giới quyết định của mô hình bằng một mô hình tuyến tính đơn giản và sử dụng trọng số để chỉ ra mức độ quan trọng của các đặc trưng đối với dự đoán. LIME cho phép xem xét các đặc trưng cụ thể và giải thích tại sao một quyết định đã được đưa ra. Điều này giúp tăng khả năng giải thích và đáng tin cậy của các mô hình học máy hộp đen, đặc biệt là trong lĩnh vực phân tích phần mềm độc hại trên Android. Do đó, Fan và cộng sự đã đánh giá tính ổn định, độ mạnh mẽ và hiệu quả của các phương pháp giải thích không phụ thuộc vào mô hình trên một số bộ phân loại phần mềm độc hại khác nhau (ví dụ: Multilayer Perceptron (MLP), Random Forest (RF) và Support Vector Machines (SVM)).

B. Mục tiêu nghiên cứu và đặt câu hỏi

Bài báo này nhằm mục tiêu thực hiện một phân tích kiểm tra chi tiết về các giải thích được tạo ra bởi ba kỹ thuật phát hiện phần mềm độc hại dựa trên ML có khả năng giải thích: Drebin, XMal và Fan cùng cộng sự. Mục tiêu là giúp các chuyên gia an ninh lựa chọn mô hình phù hợp khi triển khai và giúp các nhà nghiên cứu hiểu rõ về các rủi ro tiềm tàng liên quan đến thiết lập thực nghiệm không thực tế. Ba câu hỏi nghiên cứu sẽ được giải quyết trong bài báo này:

(Câu hỏi 1) Tác động của không nhất quán thời gian đến hiệu suất của các phương pháp phát hiện phần mềm độc hại dựa trên học máy là gì?

Nghiên cứu này đánh giá hiệu suất phân loại phần mềm độc hại bằng cách sử dụng các tập dữ liệu không nhất quán thời gian và xác định tác động của không nhất quán thời gian đến hiệu suất của các kỹ thuật phát hiện phần mềm độc hại dựa trên học máy. Kết quả nghiên cứu cho thấy không nhất quán thời gian có thể làm giảm hiệu suất phát hiện phần mềm độc hại.

(Câu hỏi 2) Tại sao không nhất quán thời gian làm cho các phương pháp phát hiện phần mềm độc hại dựa trên học máy hoạt động rất tốt?

Nghiên cứu này chỉ ra rằng các mô hình phát hiện phần mềm độc hại dựa trên học máy hiện tại vẫn chưa có một phương pháp đặc thù để đưa ra dự đoán chính xác. Các phương pháp như Drebin, XMal và Fan cùng cộng sự. đã được phát triển để đạt độ chính xác cao và cung cấp tính giải thích cho các chuyên gia an ninh. Tuy nhiên, các nghiên cứu này chưa nghiên cứu đầy đủ về việc kiểm tra các giải thích tạo ra bởi mô hình, dẫn đến khả năng chọn một mô hình không thích hợp trong việc triển khai thực tế. Nghiên cứu này tập trung vào việc phân tích các giải thích tạo ra bởi các phương pháp này để hiểu rõ hơn vì sao chúng đạt được độ chính xác cao dưới sự không nhất quán thời gian.

(Câu hỏi 3) Tính nhạy cảm của tác động của không nhất quán thời gian đến độ chính xác và giải thích của các phương pháp phát hiện phần mềm độc hại dựa trên học máy là như thế nào?

Các nghiên cứu trước đã đề cập đến lo ngại về tác động của các yếu tố thực nghiệm đến độ chính xác và giải thích của các mô hình dự đoán lỗi. Các yếu tố này bao gồm chất lượng dữ liệu, mất cân bằng lớp, cài đặt tham số và kỹ thuật xác thực mô hình. Tương tự, các nghiên cứu phát hiện phần mềm độc hại dựa trên học máy cũng sử dụng các yếu tố thực nghiệm khác nhau. Tuy nhiên, hiện vẫn còn nhiều điều chưa được biết về tác động của các yếu tố thực nghiệm đến độ chính xác và giải thích của các phương pháp phát hiện phần mềm độc hại có tính giải thích. Do đó, nghiên cứu này tập trung vào việc hiểu rõ hơn về tác động của không nhất quán thời gian trong các cài đặt thực nghiệm khác nhau.

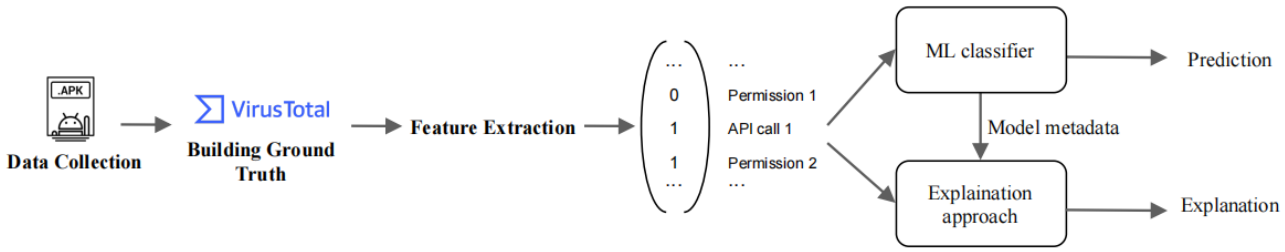

Figure 1: Tổng quan về thí nghiệm về ứng dụng được thu thập và gán nhãn, đặc trưng được trích xuất bằng công cụ đảo ngược, sau đó sử dụng bộ phân loại ML để dự đoán và giải thích.

III. Proposed Methodology

Đồ án dựa trên các thực hiện phân tích kiểm tra mô hình chi tiết được tác giả đề cập trong bài bao gồm Drebin, XMal, Fan và cộng sự. Nhóm lựa chọn thực hiện phân tích của tác giả theo hướng Fan và cộng sự. Phân tích kiểm tra mô hình chi tiết như vậy có thể giúp các nhà nghiên cứu hiểu rõ hơn về những rủi ro tiềm ẩn liên quan đến các thiết lập thử nghiệm phi thực tế. Để đạt được mục tiêu này, nhóm cũng sẽ thực hiện giải quyết ba câu hỏi ở trên.

Câu hỏi 1: Nhóm thực hiện phân tích trên dữ liệu trích xuất từ bộ dữ liệu ứng dụng lành tính Androzoo và phần mềm độc hại của CIC. Chia thành 2 bộ dữ liệu(1 bộ thu thập cùng thời gian, 1 bộ khác thời gian giữa phần mềm độc hại và lành tính). Thử nghiệm với 3 mô hình: CNN, SVM, RF.

Câu hỏi 2: Sử dụng LIME để phân tích và đánh giá trên ba mô hình ở câu hỏi số 1

Câu hỏi 3: Nhóm thực hiện trên bộ dữ liệu androzoo theo các tỉ lệ 1:1, 1:4, 4:1 với hai mô hình CNN, MLP để kiểm tra hiệu xuất. Cuối cùng là dùng LIME để phân tích trên mô hình MLP.

IV. Experiment

A. Set up

Để giải quyết các câu hỏi nghiên cứu của tác giả, thí nghiệm của nhóm bao gồm các bước sau đây: (1) thu thập dữ liệu; (2) trích xuất đặc trưng; (3) huấn luyện mô hình và đánh giá mô hình; và (4) giải thích mô hình. Hình 1, mô tả tổng quan về thí nghiệm của nhóm. Nhóm mô tả từng bước dưới đây:

(1) Thu thập dữ liệu Bộ dữ liệu Androzoo bao gồm một bộ sưu tập hơn 15 triệu ứng dụng Android được xuất bản từ năm 2010 đến năm 2021, cùng với các nhãn sự thật do phần mềm VirusTotal cung cấp. Họ sử dụng tập dữ liệu tổng cộng 165.000 ứng dụng Android (tức là 33.000 mẫu độc hại và 132.000 mẫu lành tính) trải dài trong khoảng thời gian 10 năm (2010-2020) (giữ nguyên tỷ lệ: 17.3% malware)

Kết hợp với bộ dữ liệu CIC (Canadian Institute for Cybersecurity) là một tập hợp các bộ dữ liệu công khai được phát triển bởi Viện Công nghệ thông tin và An ninh mạng Canada (Canadian Institute for Cybersecurity) thuộc Đại học New Brunswick

(2) Trích xuất tính năng

Toàn bộ dữ liệu nhóm đã tải file APK về và tự thu thông qua công cụ APKtool.

(3) Đào tạo mô hình và đánh giá mô hình

Nhóm thực hiện theo hướng phân tích của tác giả dựa trên Fan và cộng sự: đào tạo bốn mô hình ML khác nhau (ví dụ: MLP, CNN, RF và SVM) với các cài đặt giống như các cài đặt được sử dụng trong bài báo gốc.

(4) Giải thích mô hình

Tạo ra các giải thích mô hình từ các phương pháp đã để cập (Fan và cộng sự). Áp dụng phương pháp LIME để tạo ra giải thích cho Fan và cộng sự.

B. Result between group and author

Trong phần này, nhóm trình bày phương pháp và kết quả của việc nghiên cứu ba câu hỏi như đã đề cập ở trên. *Câu hỏi 1* Trong nghiên cứu này, nhóm thực hiện nghiên cứu trước đó của Fan cùng cộng sự. Thực hiện đánh giá dữa trên bộ dữ liệu kết hợp giữa Androzoo và CIC đồng bộ, đồng bộ. Các chỉ số được sử dụng để đánh giá bao gồm độ Accuracy, F1-score, Recall, PPrecision.

Data	Metric / Model	Accuracy	Recall	F1_score	Precision
Đồng bộ	CNN	0.941	0.956	0.942	0.928
	SVM	0.947	0.952	0.950	0.949
	RF	0.943	0.960	0.946	0.932
Bất đồng bộ	CNN	1.0	1.0	1.0	1.0
	SVM	0.947	0.954	0.948	0.952
	RF	0.944	0.959	0.947	0.935

Figure 2: So sánh số liệu giữa mô hình chạy trên bộ dữ liệu đồng vộ và bất đồng bộ của nhóm

Setup	Time periods of experimental samples (Year 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020)	Drebin Accuracy	F1	Precision	Recall
Baseline (Temporal consistent)	Malware: Benign:	0.9239	0.9243	0.9203	0.9283
Variant1 (Temporal consistent)	Malware: Benign:	0.9263	0.9259	0.9311	0.9208
Variant2 (Temporal consistent)	Malware: Benign:	0.9535	0.9538	0.9470	0.9608
Variant3 (Temporal inconsistent)	Malware: Benign:	0.9911	0.9911	0.9941	0.9881
Variant4 (Temporal inconsistent)	Malware: Benign:	0.9927	0.9927	0.9907	0.9948

Figure 3: So sánh số liệu giữa mô hình chạy trên bộ dữ liệu đồng vộ và bất đồng bộ của tác giả

Từ bảng số liệu của nhóm và tác giả đưa ra ta có thể thấy được kết quả của các mô hình khi chạy trên bộ dữ liệu bất đồng bọ sẽ cao hơn đồng bộ. Do đó, ta trả lời cho Câu hỏi số 1 với phát hiện như sau: Sự khác biệt tạm thời giữa mẫu độc hại và mẫu lành tính trong bộ dữ liệu thử nghiệm có thể ảnh hưởng đáng kể hiệu suất

Câu hỏi 2 Nhóm thực hiện thí nghiệm thức hai dựa trên phân tích từ các mô hình đã chạy ở Câu hỏi 1. Sử dụng công cụ LIME để thực hiện phân tích mô hình.

Feature Name	Feature Type	Updated at version	Updated at Year	Feature importance (Drebin) Variant 3 Malware	Variant 3 Benign	Variant 4 Malware	Variant 4 Benign
com.google.android.gms.ads.adactivity	App components	Added at API level 19	2013	0.59	0.00	0.00	0.92
android.permission.read_external_storage	Requested permissions	Added at API level 16	2012	1.13	-0.03	-0.18	0.54
android.permission.foreground_service	Requested permissions	Added at API level 28	2017	0.09	0.00	0.00	0.05
landroid/telephony/telephonymanager;->getdeviceid	Suspicious API calls	Removed at API level 26	2017	-0.03	0.05	0.44	-0.16
lorg/apache/http/client/methods/httppost	Suspicious API calls	Removed at API level 22	2015	-0.03	0.10	0.38	-0.36
android.permission.get_tasks	Used permissions	Removed at API level 21	2014	0.00	0.00	0.08	-0.02

Figure 4: Bảng so sánh số liệu thực nghiệm được của tác giả với dữ liệu đồng bộ và bất động bộ

Đối với tác giả ta có thể nhận thấy các đặc tính sẽ có sự thay đổi qua từng năm. Ví dụ như với đặc tính cuối cùng thì ta nhận thấy ở Variant 3 thì nó hầu như không có tác dụng trong việc phân biệt giữa mẫu độc hại và lành tính, nhưng qua tới Variant 4 thì lại được sử dụng để phân biệt.

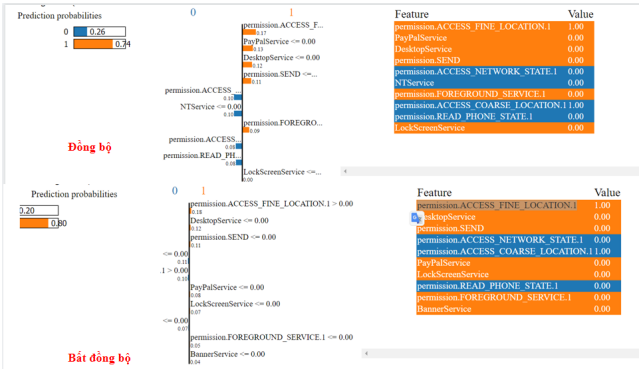

Figure 5: So sánh phân tích dữ liệu bằng Lime dữ trên dữ liệu đồng bộ và bất đồng bộ với mô hình là SVM

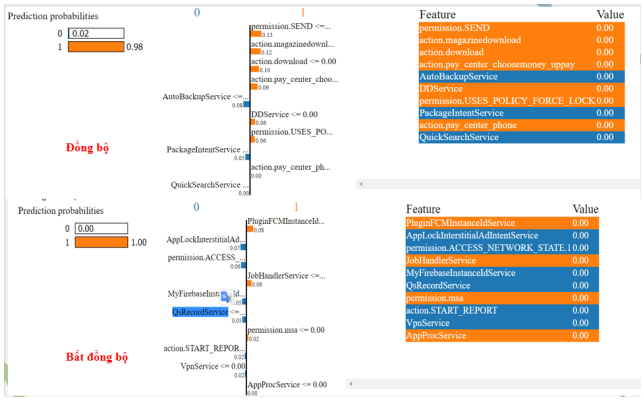

Figure 6: So sánh phân tích dữ liệu bằng Lime dữ trên dữ liệu đồng bộ và bất đồng bộ với mô hình là RF

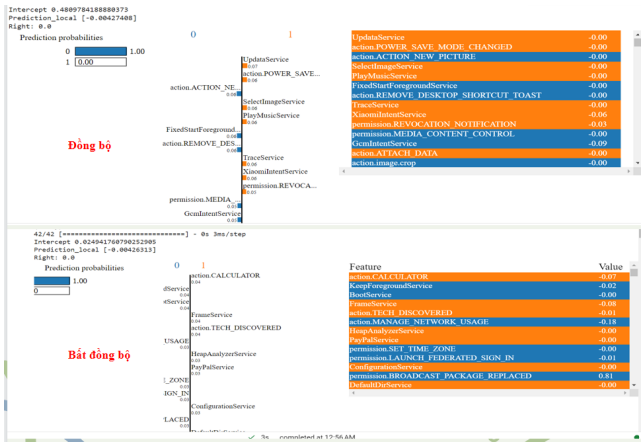

Figure 7: So sánh phân tích dữ liệu bằng Lime dữ trên dữ liệu đồng bộ và bất đồng bộ với mô hình là CNN

Tương tự như tác giả, nhóm phân tích dựa trên 3 mô hình và cũng đưa ra một bảng đánh giá các đặc tính. Ở cả SVM, CNN và RF đều có sự thay đổi tác động của các feature và trong đó nhóm có chỉ ra các đặc tính đặc biệt như QsRecordService - đây là một đặc tính vô cùng nguy hiểm nhưng lại được liệt vào là feature lành tính, lý do là vì bộ dữ liệu có sự bất động bộ giữa mẫu độc hại và lành tính, đặc tính QsRecordService trước đó không tồn tại nên đã gây ra sự phán đoán sai cho mô hình. Tương tự như vậy với các đặc tính ở mô hình CNN.

Feature	Yes	No	Feature	Yes	No	
permission.ACCESS_FINE_LOCATION.1	x		QsRecordService	x		RF
permission.ACCESS_NETWORK_STATE.1		x	permission.SEND	x		
permission.FOREGROUND_SERVICE.1	x		permission.READ_PHONE_STATE.1	x		
permission.READ_PHONE_STATE.1		x	action.UMS_CONNECTED		x	
permission.ACCESS_COARSE_LOCATION.1		x	action.main	x		
LockScreenService	x		action.SHARED_LOGIN		x	
permission.MANAGE_DEVICE_ADMINS	x		action.MANAGE_NETWORK_USAGE		x	
action.SENT_SMS_ACTION	x		action.CALCULATOR		x	CNN
action.appdetail		x	BackgroundActionsService	x		CNN
MemClearService	x		action.TECH_DISCOVERED	x		
permission.USE_SIP	x		BootService		x	
permission.WRITE_SYNC_SETTINGS		x	desktopservice		x	
action.REMOVE_DESKTOP_SHORTCUT_TOAST	x		DownloadFileService	x		
permission.PEERS_MAC_ADDRESS		x	permission.USE_SIP	x		

Figure 8: Bảng liệt kê một số feature đã phân tích được sau quá trình sử dụng Lime

Rút ra được kết luận từ các thực nghiệm để trả lời Câu hỏi 2 là tính năng chỉ hữu dụng riêng trong thời gian nhất định hoặc các phiên bản ứng dụng. Với phát hiện một số đặc điểm mà mô hình dựa vào để phân loại không hề có giá trị phân loại mẫu độc hại và lành tính cao. Và trải qua một thời gian thì sẽ có các thuộc tính được thêm vào hoặc loại bỏ (tính khả dụng của feature tại một thời điểm) nhưng mô hình của chúng ta vẫn dựa vào những đặc điểm trước đó được học gây ra phán đoán sai.

Câu hỏi 3 Đến với thí nghiệm số 3 nhóm thực hiện hoàn toàn trên bộ Androzoo, chia thành 3 tập dữ liệu với tỉ lệ benign:malware lần lượt là 1:1, 1:4. 4:1.

	Mal_fea	Benign_fea	Sample
1:1	6000	4846	8400
1:4	4122	4290	7467
4:1	8322	15053	10322

Figure 9: Bảng số lương đặc tính và mẫu của ba bộ dữ liệu được trích xuất

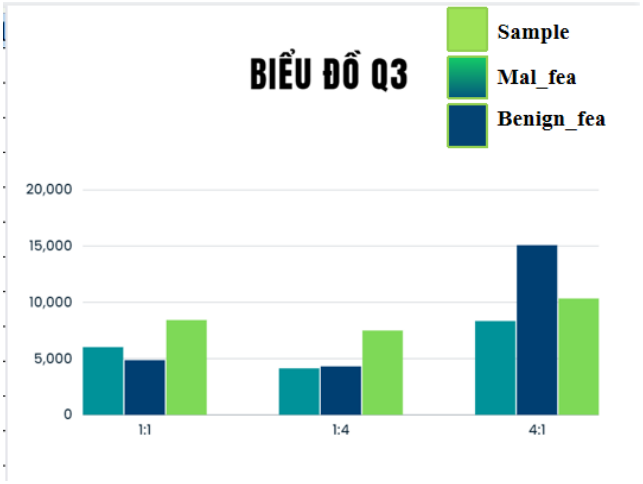

Figure 10: Biểu đồ thể hiện chênh lệch giữa số lượng đặc tính và số lượng mẫu

Trước tiên nhóm thực hiện chạy 3 bộ dữ liệu với các mô hình CNN, MLP và so sánh kết quả chênh lệch giữa 3 bộ

Data		Accuracy	Precision	Recall	F1_score
1:1	CNN	0.988	0.988	0.987	0.988
	MLP	1.0	1.0	1.0	1.0
4:1	CNN	0.972	0.95	0.982	0.956
	MLP	0.993	0.984	0.988	0.985
1:4	CNN	0.844	0.917	0.629	0.660
	MLP	0.999	0.998	0.999	0.999

Figure 11: Bảng so sánh kết quả chạy CNN, MLP trên 3 bộ dữ liệu được trích xuất của nhóm

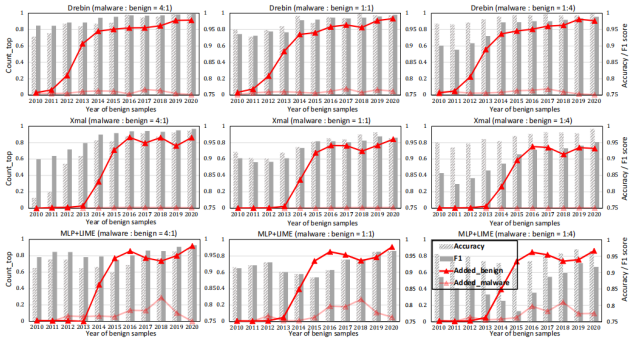

Figure 12: Hình ảnh kết quả của tác giả đưa ra so sánh số lượng đặc tính, hiệu suất của mô hình chạy trên 3 bộ dữ liệu khác nhau của tác giả

Tiếp tục nhóm sử dụng mô hình MLP theo hướng của tác giả để thực hiện phân tích quyết định của mô hình và nhận được kết quả:

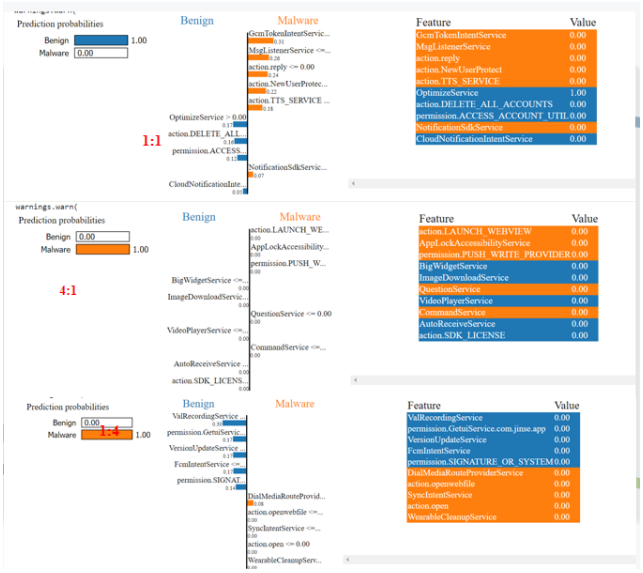

Figure 13: Hình ảnh số liệu sử dụng LIME để phân biệt trên ba bộ dữ liệu với mô hình MLP

Từ những phân tích dữ trên LIME, nhóm đưa ra được một bảng sơ lược về các đặc tính được thêm vào

Feature	Add_Mal	Add_Benign
action.LAUNCH_WEBVIEW	x	
AppLockAccessibilityService	x	
permission.PUSH_WRITE_PROVIDER	x	
BigWidgetService		x
ImageDownloadService		x
QuestionService	x	
VideoPlayerService		x
CommandService	x	
AutoReceiveService		x
action.SDK_LICENSE		x
ValRecordingService		x
permission.GetuiService.com.jinse.app		x
VersionUpdateService		x
FcmIntentService		x
permission.SIGNATURE_OR_SYSTEM		x
DialMediaRouteProviderService	x	
action.openwebfile	x	
SyncIntentService	x	
action.open	x	
WearableCleanupService	x	

Figure 14: Bảng các đặc tính lành tính và độc hại được thêm vô sau khi dữ liệu xảy ra thay đổi về tỉ lệ nhãn

Sau thí nghiệm 3, nhóm đưa ra được chứng minh cho câu trả lời của tác giả: Dù tồn tại sự không nhất quán về dữ liệu nhưng các mô hình hầu như đều đạt hiệu suất cao. Tức là dù có sự chênh lệch về các nhãn trong dữ liệu thì mô hình vẫn hoạt động tốt vì khi mà giữa mẫu độc hại và lành tính có khoảng cách về dữ liệu theo thời gian càng lớn thì mô hình có xu hướng học tốt hơn vì các đặc trưng được thêm vào trở nên ngày càng quan trọng trong việc phân biệt các mẫu không độc hại.

V. Conclusion

Bài báo tác giả đã sử dụng các mô hình phát hiện phần mềm độc hại có thể giải thích được để điều tra lý do tại sao các công trình nghiên cứu hiện tại lại thể hiện hiệu suất chính xác cao. Khi đánh giá kết quả giải thích của các mô hình ML, tác giả nhận thấy rằng hầu hết các kết quả đều không thực tế vì các mô hình ML chưa tìm ra sự khác biệt thực sự giữa phần mềm độc hại và lành tính.

Nhóm đã thực hiện một phần của bài báo với đầy đủ thực nghiệm riêng cho ba vấn đề chính tác giả đưa ra. Và nhóm cũng chứng minh lại rằng nhận định của tác giả là đúng, hiệu suất mô hình trong thực tế sẽ bị ảnh hưởng bởi tính đồng bộ của dữ liệu.

Nhóm nhận định đây là một bài báo hay và tiềm năng, tạo tiền đề để thúc đẩy việc kiểm tra tính thực tế của mô hình, nhằm có thể đưa ra các quyết định đúng đắn nhất. Và cũng giúp phát triển các mô hình có thể giải thích giúp con người hiểu được các quyết định của mô hình.

References

[1] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Explainable ai for android malware detection: Towards understanding why the models perform so well?" in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022, pp. 169–180.

[2] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[3] B. Wu, S. Chen, C. Gao, L. Fan, Y. Liu, W. Wen, and M. R. Lyu, "Why an android app is classified as malware: Toward malware classification interpretation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–29, 2021.

[4] M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan, and T. Liu, "Can we trust your explanations? sanity checks for interpreters in android malware analysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 838–853, 2020.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[7] ——, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[8] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," *arXiv preprint arXiv:1805.10820*, 2018.

[9] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 364–379.