Phát hiện mã độc ứng dụng Android thông qua chiết xuất đặc trưng

Nguyễn Lâm Vũ
Khoa Khoa học và Kĩ thuật thông tin
Đại học Công nghệ Thông tin
Thành phố Hồ Chí Minh, Việt Nam
20520867@gm.uit.edu.vn

Nguyễn Văn Gia Bảo
Khoa Khoa học và Kĩ thuật thông tin
Đại học Công nghệ Thông tin
Thành phố Hồ Chí Minh, Việt Nam
20521106@gm.uit.edu.vn

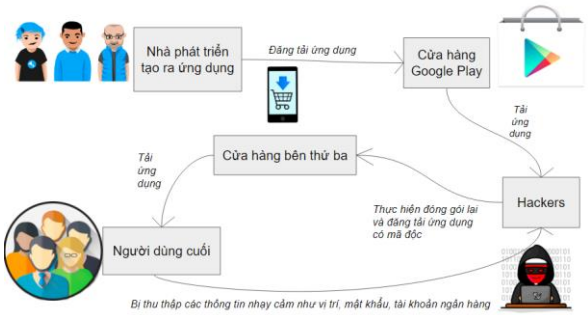
***Abstract*—Ứng dụng Android ẩn chứa nguy cơ bảo mật do sự phát triển mạnh mẽ của các thiết bị sử dụng hệ điều hành Android. Có rất nhiều thách thức trong việc phát hiện mã độc Android. Các kĩ thuật truyền thống để phát hiện mã độc như kĩ thuật tĩnh, động và kết hợp thường cần sự can thiệp rất nhiều từ con người, yêu cầu công suất xử lí cao, tiêu tốn không gian lưu trữ và có độ phức tạp cao. Bài báo cáo này đề xuất các thuật toán dựa vào Machine Learning (ML) để phát hiện mã độc ứng dụng Android thông qua chiết xuất đặc trưng. Trong phương pháp được đề xuất, chiết xuất đặc trưng của các tập tin Android Package Kit (APK) yêu cầu, bao gồm quyền hạn (permission), hoạt động (activities) và gói (package) bằng công cụ Androguard. Công cụ Androguard sẽ trích xuất thông tin tĩnh từ các tập tin AndroidManifest.xml, classes.Dex của tập tin APK.**

Keywords—mã độc Android, Machine Learning, APK, Androguard

I. GIỚI THIỆU

Hệ điều hành Android là hệ điều hành phổ biến trong hệ sinh thái thiết bị thông minh. Do đó, người dùng Android rất gần gũi với các tập tin APK. Hàng ngày, người dùng Android thường chia sẻ thông tin nhạy cảm, giao dịch ngân hàng, mua hàng trực tuyến, thông tin người dùng, vị trí, … Trong Android, bảo mật thiết bị luôn là thách thức lớn và vấn đề nghiêm trọng. Theo báo cáo thống kê của GDATA vào năm 2019 [1] cho thấy 1,852,170 mẫu mã độc Android được phát hiện vào nửa đầu năm 2019. Ở đây, theo dữ liệu mã độc Android được phát hiện mỗi 8 giây. Theo thống kê, cứ 10 thiết bị Android thì có 8 thiết bị bị nhiễm mã độc [2]. Do sự phổ biến của các thiết bị Android, ứng dụng Android trở nên bị nhắm vào nhiều hơn so với các ứng dụng khác. Theo một báo cáo sự tiến hóa của mã độc điện thoại, trong 5,321,142 ứng dụng được cài đặt trên thiết bị di động, có 151,359 ứng dụng di động được phát hiện là Trojans, 60,176 được phát hiện là phần mềm tống tiền trên điện thoại bởi Kaspersky 2018 [3].

Người dùng Android gặp nguy cơ bởi nhiều chủng loại mã độc khác nhau; một số được phân phối bởi cửa hàng Google Play, một số được người dùng tải về từ các nguồn không minh bạch ở trên Internet [4]. Có rất nhiều loại mã độc, ví dụ như Adware, Ransomware, Scareware, SMS Malware, … Có thể thấy ở hình 1, các hackers thường đăng tải những ứng dụng đã được đóng gói lại lên các trang mạng, những cửa hàng bên thứ ba [5]. Với sự trợ giúp của công cụ đóng gói, họ lăp ráp lại ứng dụng gốc bằng cách thêm những đoạn mã độc vào trong mã gốc. Và đây là thử thách chính trong việc nhận định mã độc. Hầu hết các kĩ thuật hiện nay là kĩ thuật tĩnh và động; các kĩ thuật thường sẽ sử dụng hành vi và cơ sở chữ kí để phân loại mã độc Android.



Hình 1. Quá trình đánh cắp thông tin từ người dùng

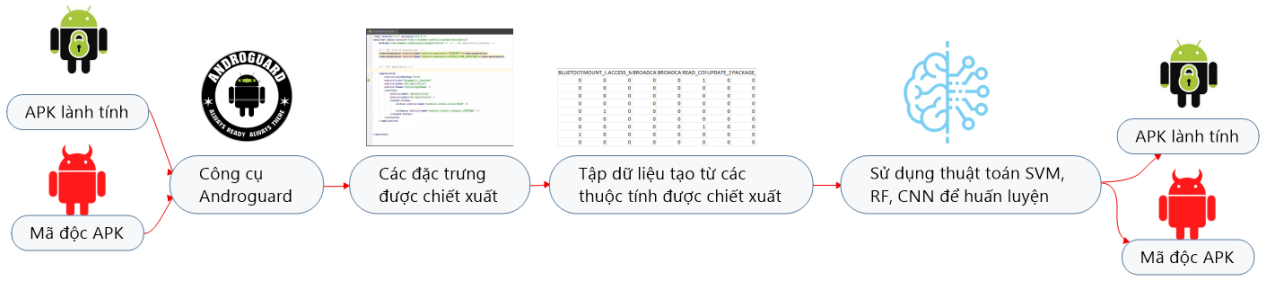
Kĩ thuật tĩnh không yêu cầu khởi chạy ứng dụng; đoạn mã sẽ được tháo rời và chiết xuất các đặc trưng của ứng dụng để phân loại. Kĩ thuật động luôn luôn cần khởi chạy ứng dụng và phân loại mã độc Android dựa trên hành vi và cơ sở chữ kí. Các kĩ thuật trên có các nhược điểm đáng kể; đòi hỏi sức mạnh xử lý tính toán, tài nguyên và dung lượng nhiều hơn [6]. Kĩ thuật động bị qua mặt bởi một vài mã độc thông minh và mạnh mẽ; một số mã độc có thể phát hiện môi trường đang thực thi là máy ảo (kĩ thuật anti-VM) và thay đổi hành vi từ mã độc thành lành tính gây khó khăn trong quá trình phân tích và theo dõi. Hơn nữa, các kĩ thuật động và tĩnh có sẵn thường cần có can thiệp thủ công của con người.

Phương pháp được đề xuất trong báo cáo này sử dụng công cụ Androguard để thu thập thông tin tĩnh từ tập tin APK. Thông tin chủ yếu được thu thập từ tập tin AndroidManifest.xml và classes.Dex. Các thông tin được chiết xuất bao gồm permission, activities và package. Sau đó, tạo tập dữ liệu dựa theo các thuộc tính đã được chiết xuất. Áp dụng ML bao gồm các mô hình là SVM, Random Forest, CNN để huấn luyện. Đối với CNN, có một bước trung gian là chuyển dữ liệu thành ảnh grayscale.

AndroidManifest.xml: là tập tin để bản kê khai trình bày những thông tin thiết yếu về ứng dụng với hệ thống Android, thông tin mà hệ thống phải có trước khi có thể chạy bất kỳ mã nào của ứng dụng.

Classes.Dex: là tập tin chứa đoạn mã dùng để tham chiếu mọi lớp hoặc phương thức được sử dụng trong một ứng dụng Android.

Androguard: là một trong dự án mã nguồn mở lớn nhất dành cho việc phân tích tĩnh tập tin Android.



Hình 2. Phương pháp luận được đề xuất để phát hiện mã độc Android

Ứng dụng Android được phát triển bằng cách sử dụng các tập tin java.class. Bằng công cụ DX, nhiều tập tin java.class được chuyển đổi thành tập tin DEX. Tập tin DEX và manifest là tập tin quan trọng trong tập tin APK. Phương pháp được đề xuất không yêu cầu cần phải dịch ngược, vì vậy tiết kiệm thời gian và giảm thiểu độ phức tạp. Bộ dữ liệu được nhập từ nguồn CIC (Canadian Institute for Cybersecurity) Dataset, bao gồm khoảng 5,500 file apk (426 mã độc và 5,065 lành tính), được chia làm 5 danh mục là Benign (lành tính), Adware (mã độc quảng cáo), Ransomware (mã độc tống tiền), Scareware (mã độc hù dọa), SMS Malware (mã dộc qua tin nhắn). Sau khi chiết xuất đặc trưng từ 600 ứng dụng lành tính, 104 Adware, 101 Ransomware, 109 SMS Malware, 112 Scareware, dữ liệu có 1021 dòng và 95 cột. Tiếp theo, tạo dataset dựa theo 95 thuộc tính, benign là 0, còn lại là 1. Cuối cùng, sử dụng các thuật toán ML như Support Vector Machine (SVM), Random Forest (RF), Convolutional Neural Network (CNN) để huấn luyện dữ liệu và phân loại mã độc như hình 2.

II. MÔ TẢ DỮ LIỆU

A. *Android Malware Dataset (CIC-AndMal2017)*

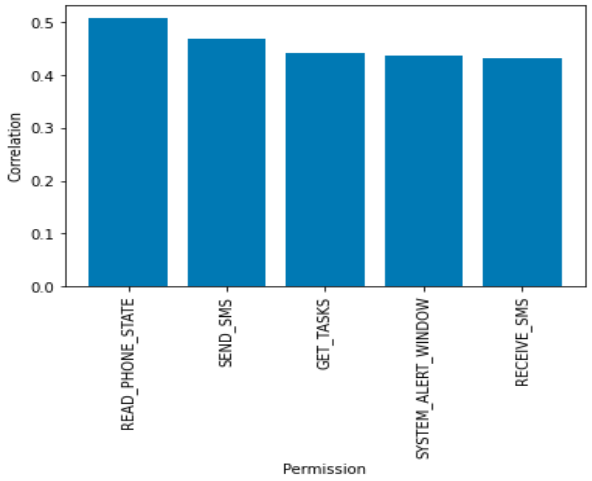
Bộ dữ liệu CIC-AndMal2017 [7] là các tập tin APK được thu thập từ nhiều nguồn. Khoảng 5500 tập tin được chạy thử nghiệm trên điện thoại thông minh thật để tránh các mã độc cấp cao phát hiện môi trường giả lập. Các mẫu mã độc được chia thành 5 loại: Adware, Ransomware, SMS Malware, Scareware và Benign.

Sau khi nhập dữ liệu về và chiết xuất đặc trưng bằng công cụ Androguard, thu được khoảng 10,000 permission. Vì dữ liệu quá lớn, nên chỉ lấy 50 permission từ 5 loại và sau khi loại bỏ các permission trùng với nhau, thu được 95 permission từ 1021 ứng dụng. Vì vậy bộ dữ liệu có 1021 dòng và 96 cột, thêm vào cột cuối cùng để phân biệt mã độc. Các permission được gắn nhãn là 0 nếu ứng dụng có permission đó, ngược lại là 1. Cột cuối cùng 0 là lành tính, 1 là mã độc.

B. *Độ tương quan Pearson*

Pearson correlation coefficient (hệ số tương quan Pearson) là một thước đo độ tương quan tuyến tính giữa hai bộ dữ liệu. Về cơ bản, hệ số tương quan Pearson là hiệp phương sai của hai biến chia cho tích độ lệch chuẩn của chúng, sao cho kết quả luôn có giá trị trong khoảng từ -1 đến 1.

Sau khi tính toán độ tương quan giữa các quyền hạn và cột is_malicious (cột cuối cùng để phân loại là mã độc hay không), có thể thấy ở hình 3 rằng các quyền hạn như READ_PHONE_STATE, SEND_SMS, GET_TASKS, SYSTEM_ALERT_WINDOW và RECEIVE_SMS thường được yêu cầu bởi các ứng dụng APK chứa mã độc.



Hình 3. Độ tương quan giữa is_malicious với các quyền hạn

READ_PHONE_STATE: cho phép truy cập chỉ đọc vào trạng thái của điện thoại, bao gồm thông tin mạng di động hiện tại, trạng thái của mọi cuộc gọi đang diễn ra và danh sách mọi tài khoản điện thoại đã đăng ký trên thiết bị.

SEND_SMS: cho phép ứng dụng gửi tin nhắn.

GET_TASKS: cho phép ứng dụng thu thập thông tin tác vụ hiện hành và gần đây.

SYSTEM_ALERT_WINDOW: cho phép ứng dụng tạo ra cửa sổ có thể hiện thị đè lên trước mọi ứng dụng khác.

RECEIVE_SMS: cho phép ứng dụng thu thập tin nhắn SMS.

Quyền hạn READ_PHONE_STATE thường được yêu cầu bởi ứng dụng chứa mã độc, với 1179 trên 1260 ứng dụng yêu cầu quyền hạn này và chỉ 34% ứng dụng lành tính yêu cầu quyền này [8].

Quyền hạn READ_PHONE_STATE và quyền hạn SEND/RECEIVE_SMS có thể được sử dụng cùng nhau để cướp tài khoản. Nhiều dịch vụ cho phép người dùng đặt lại mật khẩu bằng cách gửi tin nhắn phục hồi với một mã ngẫu nhiên đến điện thoại của người dùng. Nếu kẻ tấn công có được số điện thoại và có thể đọc tin nhắn được gửi tới, họ có thể đặt lại mật khẩu và đạt được quyền truy cập. Một khi đã truy cập kẻ tấn công có thể thay đổi thông tin để phục hồi tài khoản, như là email dự bị, và người dùng khó có thể lấy lại tài khoản [8].

Quyền hạn SYSTEM_ALERT_WINDOW có thể được sử dụng trong mã độc hù dọa hoặc mã độc quảng cáo vì có thể

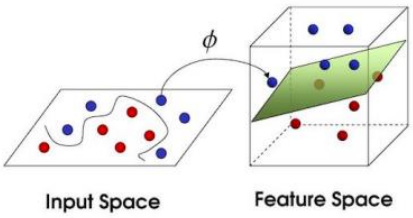
hiện lên các thông báo giả như là máy bị virus hoặc hiện quảng cáo bất cứ lúc nào, gây phiền toái cho người dùng.

III. GIỚI THIỆU PHƯƠNG PHÁP MÁY HỌC

Thuật toán được sử dụng trong phương pháp được đề xuất bao gồm 3 thuật toán là SVM, Random Forest, CNN để huấn luyện các đặc trưng đã được chiết xuất.

A. *Support Vector Machine (SVM)*

Support Vector Machine (SVM) là mô hình ML được giám sát, dùng để phân loại dữ liệu đầu vào. Mục tiêu của thuật toán là tìm ra đường thẳng phân chia tối ưu nhất hai lớp khác nhau. Đường thẳng được coi là tối ưu khi margin (biên/lề) giữa 2 lớp bằng nhau và lớn nhất có thể. Đối với dữ liệu 2 chiều: đường thẳng phân lớp được xây dựng dựa trên các điểm dữ liệu. Đối với dữ liệu nhiều chiều: siêu phẳng phân lớp được xây dựng dựa trên các vector (là các điểm dữ liệu nhiều chiều). Các điểm dữ liệu trên được gọi là support vector nên thuật toán này gọi là Support Vector Machine. Có 2 loại margin là soft margin (biên cứng) và hard margin (biên mềm). Soft margin hướng đến việc cân bằng giữa điều kiện large margin (biên lớn nhất) và giảm số lượng điểm dữ liệu vi phạm đến biên. Hard margin yêu cầu tất cả điểm dữ liệu phải làm ngoài margin, vì vậy nếu có các điểm dữ liệu nhiễu thì điều kiện của hard margin không thể đạt được. Ngoài ra, SVM có thể được hỗ trợ bằng cách chia tỉ lệ dữ liệu, giúp phân chia tốt hơn. Trong trường hợp dữ liệu không tuyến tính, thuật toán khó có thể tìm ra margin hoặc chỉ tìm thấy margin có biên độ nhỏ.



Hình 4. Hàm kernel tìm hàm số Φ(x)

Để giải quyết việc này, SVM sử dụng hàm kernel để tìm một hàm số biến đổi dữ liệu x từ không gian thuộc tính (feature) ban đầu thành dữ liệu trong một không gian mới bằng hàm số Φ(x) như hình 4. [9]

B. *Random Forest (RF)*

Random Forest (RF) là một trong những thuật toán ML có giám sát phổ biến và mạnh mẽ nhất. RF thực thi hiệu quả các tập dữ liệu lớn và dự đoán kết quả chính xác. Thuật toán RF hỗ trợ cả bài toán phân loại và hồi quy. Thuật toán sử dụng Decision Tree (cây quyết định) để gia tăng độ chính xác và tính linh hoạt. Nói chung, càng nhiều cây trong rừng thì kết càng có thể đoán được. Nhiều cây quyết định còn giúp tránh hiện tượng overfit (đúng quá nhiều trong bộ dữ liệu này nhưng sai trong bộ khác)

Decision Tree (cây quyết định) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau. Các thành phần của 1 cây quyết định:

+ Nút không phải nút lá (non-leaf node).

+ Nút con (child node).

+ Nút gốc (root node).

+ Nút lá (leaf node / terminal node).

+ Đường đi (path)

Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết. [9]

C. *K-Nearest Neighbor (KNN)*

K-Nearest Neighbor (KNN) là một thuật toán học máy có giám sát dạng lazy learning (thuộc dạng lười học). Khi có dữ liệu mới thì thuật toán mới đi thực hiện tính toán để ra kết quả dự đoán. Nhãn của 1 dữ liệu mới được dự đoán dựa vào k "láng giềng" (neighbor) gần nhất của nó (k = 1...n). — Điểm cốt lõi của KNN là việc tính khoảng cách (distance) của dữ liệu với các điểm lân cận của nó. Một số công thức tính khoảng cách thường thấy là Manhattan, Euclidean, Minkowski được liệt kê ở hình 5.

Khoảng cách	Công thức
Manhattan	$\sum_i^n \lvert x_i - y_i \rvert$
Euclidean	$\sqrt{\sum_i^n (x_i - y_i)^2}$
Minkowski	$\sqrt[p]{\sum_i^n (x_i - y_i)^p}$

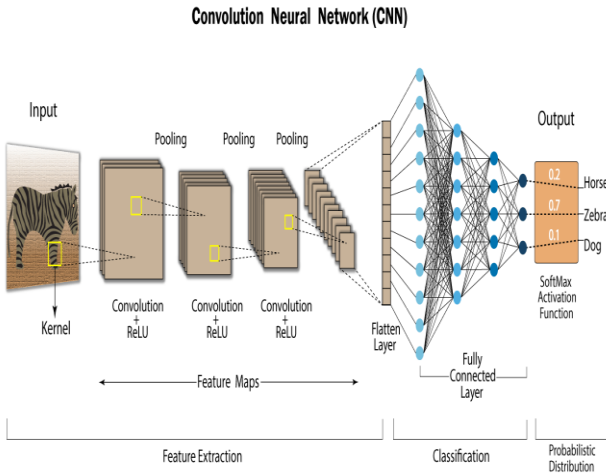
Hình 5. Một số công thức tính khoảng cách

D. *Logistic Regression (LR)*

Logistic regression (LR) là một thuật toán máy học thường được dùng trong tác vụ phân lớp (classification). Mục tiêu của logistic regression là ước lượng (estimate) xác suất của một điểm dữ liệu rơi vào một lớp cụ thể. Đầu ra của LR được đưa qua một hàm logit.

E. *Convolutional neural network (CNN)*

Convolutional Neural Network (CNN) là một trong những mô hình Deep Learning cực kỳ tiên tiến, bởi chúng cho phép bạn xây dựng những hệ thống có độ chính xác cao và thông minh. Nhờ khả năng đó, CNN có rất nhiều ứng dụng, đặc biệt là những bài toán cần nhận dạng vật thể (object) trong ảnh.



Hình 6. Quá trình thực hiện của Convolutional Network Neuron [10]

Thuật toán có các lopes như là Convolutional layer, Relu layer, Pooling layer, Fully connected layer.

Convolutional layer: Đây chính là lớp đóng vai trò mấu chốt của CNN, khi layer này đảm nhiệm việc thực hiện mọi tính toán để xử lý ảnh từ ban đầu trở thành nhiều ảnh khác theo các quy tắc khác nhau.

Relu layer: Còn có tên gọi khác là activation function, đây là một hàm được kích hoạt trong neural network. Nó có tác dụng mô phỏng các neuron có tỷ lệ truyền xung qua axon.

Pooling layer: Khi nhận phải đầu vào quá lớn, các lớp pooling layer sẽ được xếp giữa những lớp Convolutional layer nhằm mục đích giảm parameter. Pooling layer được chia thành 2 loại phổ biến là max pooling và average.

Fully connected layer: Khi 2 lớp convolutional layer và pooling layer nhận được ảnh truyền, lớp này sẽ có nhiệm vụ xuất kết quả. [11]

IV. ĐÁNH GIÁ MÔ HÌNH

Phương pháp được đề xuất sử dụng 3 thuật toán ML là SVM, RF, CNN. Phương pháp đánh giá độ chính xác cho nhiều mô hình ML dựa vào các chỉ số như là precision, recall, f1-score, accuracy được tính theo các công thức lần lượt là (1), (2), (3), (4). Kết quả được thống kê bằng bảng 1.

Precision là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive (TP + FP).

Bảng 1. Chỉ số đánh giá trung bình cộng Accuracy, Precision, Recall, F1-score của các thuật toán ML

Macro Average				
ML Methods	Accuracy	Precision	Recall	F1-score
Support Vector Machine (SVM)	92%	92%	93%	92%
Random Forest (RF)	92%	92%	93%	92%
Convolutional Neural Network (CNN)	97%	97%	97%	97%
K-Nearest Neighbor (KNN)	90%	90%	90%	90%
Logistic Regression (LR)	90%	90%	90%	90%

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (1)$$

Recall là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN).

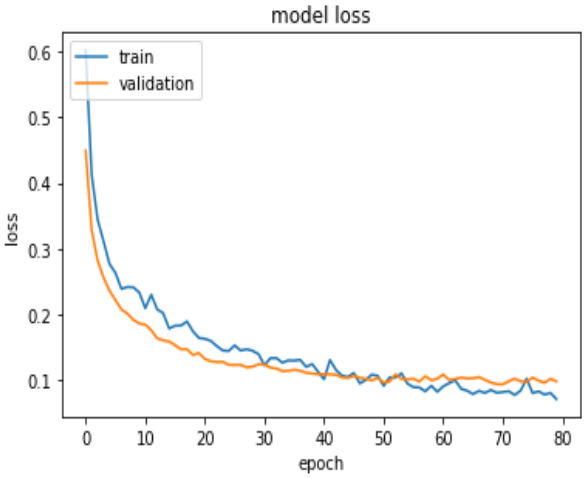
$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (2)$$

F1-score là là harmonic mean (trung vị hài hòa) của precision và recall.

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Accuracy là tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (4)$$

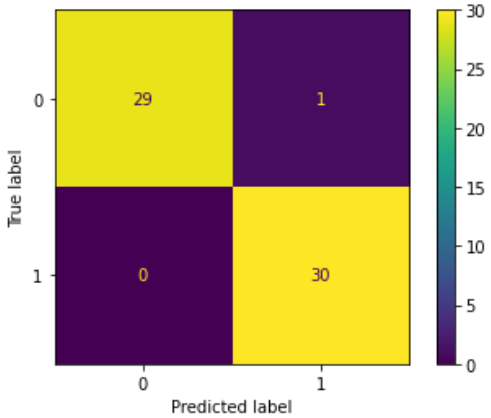


Hình 7. Biểu đồ training và validation loss của thuật toán CNN

Theo như hình 7, training loss và validation loss đều giảm và chạm nhau ở khoảng 30 epoch. Do đó, thuật toán không bị overfit hoặc underfit.

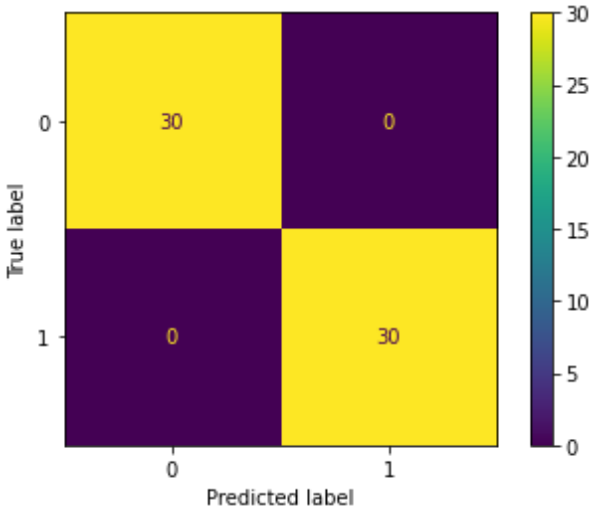
V. THỬ NGHIỆM

Tải 30 tập tin APK lành tính từ apkpure.com. Tiếp đó, tải 30 tập tin APK mã độc banking (đánh cắp thông tin ngân hàng). Chiết xuất permission (quyền hạn) từ 60 tập tin APK, xây dựng bộ dữ liệu dựa trên 95 nhãn của bộ dữ liệu gốc. Tuy nhiên chỉ sử dụng SVM và RF để huấn luyện dữ liệu. Kết quả được thể hiện qua ma trận nhầm lẫn ở hình 60.



Hình 8. Ma trận nhầm lẫn của 2 thuật toán SVM và LR

Qua hình 8, có thể thấy được 59 tập tin APK đã được phân loại đúng lành tính và mã độc, với không ứng dụng chứa mã độc bị phân nhầm vào lành tính và chỉ 1 ứng dụng lành tính bị phân nhầm vào mã độc.



Hình 9. Ma trận nhầm lẫn của 2 thuật toán KNN và RF

Qua hình 9, có thể thấy mô hình KNN và RF đã thực hiện phân loại tốt hơn, với 60 tập tin APK được phân loại đúng và không có tập tin nào bị phân nhầm.

VI. KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

Mô hình được đề xuất nhập dữ liệu từ nguồn CIC, chiết xuất đặc trưng bằng công cụ Androguard. Sau đó, sử dụng 3 thuật toán ML như là Support Vector Machine (SVM), Random Forest (RF), Convolutional Neural Network (CNN) để huấn luyện dữ liệu.

Phương pháp đề xuất hoạt động rất hiệu quả, có độ chính xác cao. Tuy nhiên, bộ dữ liệu còn khá nhỏ nên chưa có khả năng ứng dụng thực tế. Ngoài ra, các thuật toán chỉ đang chạy trên nền tảng Google Colab nên sức mạnh xử lý còn thấp.

Hơn nữa, phương pháp chỉ mới xử lý trên quyền hạn của các tập tin APK, chưa bao quát trọn bộ các trường hợp của một tập tin APK có chứa mã độc.

Cuối cùng, phương pháp dược đề xuất hoàn toàn có khả năng phát triển tiếp. Tuy có nhiều điều vẫn chưa được hoàn thành, nhưng đã có thể đưa ra phương hướng nhận diện mã độc Android cơ bản.

REFERENCES

[1] G Data, Bochum, Germany, Tech. Rep. [Online] Access date 5/5/2021. [Online], Available: https://www.gdatasoftware.com/mobile-internet-security-android. [Accessed Oct. 10, 2022].

[2] Gartner (2018) Gartner says worldwide sales of smartphones recorded firstever decline during the fourth quarter of 2017. [Online], Available: https://www.gartner.com/en/newsroom/press-releases/2018-02-22-gartner-says-worldwide-sales-of-smartphones-recorded-first-ever-decline-during-the-fourth-quarter-of-2017. [Accessed Oct. 11, 2022].

[3] SecureList (2018) Mobile malware evolution 2018. [Online], Available: https://securelist.com/mobile-malware-evolution-2018/89689/ [Accessed Oct. 11, 2022].

[4] DoctorWeb (2019) Doctor Web's overview of malware detected on mobile devices in September 2019." [Online], Available: https://news.drweb.com/show/review/?lng=en&i=13446 [Accessed Oct. 12, 2022].

[5] M.R. Khan, R.C. Tripathi, and A. Kumar, Repacked android application detection using image similarity, Nexo Revista Cient´ıfica, vol 33, no.1, pp.190-199, june, 2020. [Online] Available: https://dialnet.unirioja.es/descarga/articulo/7600309.pdf [Accessed Oct. 13, 2022].

[6] W. Wang, M. Zhao, Z. Gao, Xu, G., H. Xian, Li, Y. and X. Zhang, Constructing features for detecting android malicious applications: issues, taxonomy and directions. IEEE access, vol.7, no. 2019, pp.67602-67631, June, 2019. [Online] Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8720030 [Accessed Oct. 14, 2022].

[7] A. H. Lashkari, A. F. A. Kadir, L. Taheri and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," 2018 International Carnahan Conference on Security Technology (ICCST), 2018, pp. 1-7, doi: 10.1109/CCST.2018.8585560. [Online], Available: https://ieeexplore.ieee.org/abstract/document/8585560 [Accessed Oct. 15, 2022].

[8] Alshehri, Ali & Hewins, Anthony & McCulley, Maria & Alshahrani, Hani & Fu, Huirong & Zhu, Ye. (2017). Risks behind Device Information Permissions in Android OS. Communications and Network. 09. 219-234. 10.4236/cn.2017.94016. [Online], Available: https://www.researchgate.net/publication/320952529_Risks_behind_Device_Information_Permissions_in_Android_OS [Accessed Oct 16, 2022].

[9] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, by Aurélien Géron, released September 2019, publisher(s): O'Reilly Media, Inc. ISBN: 9781492032649 [E-book], Available: https://www.knowledgeisle.com/wp-content/uploads/2019/12/2-Aur%C3%A9lien-G%C3%A9ron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O%E2%80%99Reilly-Media-2019.pdf [Accessed Oct. 20, 2022].

[10] aditi kothiya, "Understanding "convolution" operations in CNN", June. 23, 2021 [Online]. Available: https://medium.com/analytics-vidhya/convolution-operations-in-cnn-deep-learning-compter-vision-128906ece7d3 [Accessed Nov. 10, 2022].

[11] Nguyễn Hưng, "Thuật toán CNN là gì? Tìm hiểu về Convolutional Neural Network.", July. 19, 2022 [Online]. Available: https://vietnix.vn/cnn-la-gi/#:~:text=cho%20h%C3%ACnh%20%E1%BA%A3nh%3F-,%C4%90%E1%BB%8Bnh%20ngh%C4%A9a%20CNN%20l%C3%A0%20g%C3%AC%3F,x%C3%A1c%20cao%20v%C3%A0%20th%C3%B4ng%20minh. [Accessed Nov. 15, 2022].