Machine Learning In Android Malware Evasion

Trần Tấn Hải
21522036

Bùi Nguyên Phúc
21522469

Huỳnh Anh Nguyễn
21522388

1 GIỚI THIỆU

Hiện nay các thiết bị mobile đang càng ngày phổ biến (smartphones, tablets, ...) và phần lớn các thiết bị này chạy hệ điều hành Android. Nên việc mã độc càng phổ biến trên nền tảng này là điều hiển nhiên. Tuy nhiên các giải pháp machine learning được áp dụng vào việc tự động phát hiện mã độc trên nền tảng này cũng đang được phát triển liên tục và đã có được một vài kết quả tốt. Như hai Android malware detection schemes: MaMaDroid và Drebin có tỉ lệ phát hiện mã độc là 96%, 97%.

Mục tiêu của đồ án này là tạo một chương trình có khả n*u*ang tạo mẫu mã độc đột biến có khả n*u*ang trốn tránh các trình phát hiện mã độc trên nền tảng Android (cụ thể ở đây là MaMaDroid).

2 CƠ SỞ LÝ THUYẾT

Để tạo được mẫu mã độc có khả n*u*ang tránh né trình phát hiện mã độc dựa trên Machine Learning cụ thể ở đây là MaMaDroid thì ta cần phải hiểu rõ được cách hoạt động của đối tượng mà ta muốn tấn công.

2.1 File APK

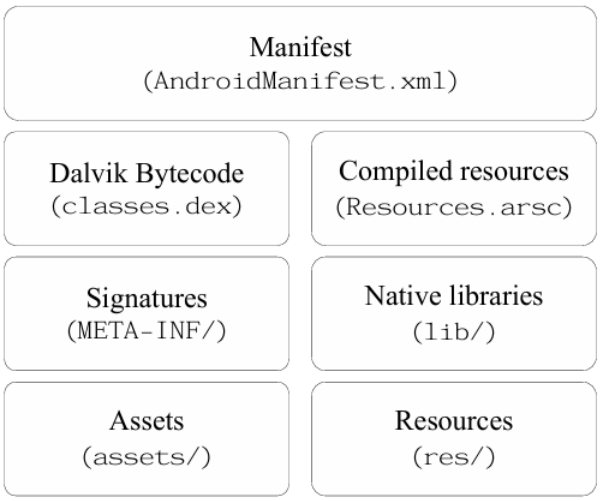

Figure 1: Cấu trúc của file APK

Ứng dụng Android được đóng gói và phân phối dưới dạng file apk. File APK là 1 dạng file giống như file jar, file APK được đóng gói với các thành phần như manifest, dexcode(.dex file), resources, signature, assets. Cấu trúc của 1 file APK được thể hiện ở Hình 1.

Cụ thể AndroidManifest.xml bao gồm các thông tin như các quyền mà ứng dụng sẽ yêu cầu được cấp, định nghĩa các thành phần như Activities, Services, BroadCast Receivers và Content Providers. Classes.dex thì được dùng để lưu trữ Dalvik Bytecode để được thực thi trên Android Runtime. Thư mục Res thì chứa các tài nguyên về hình ảnh, chuỗi ký tự, giao diện người dùng, ... Thư mục Assets thì chứ các file không được biên dịch và META-INF thì chứa các thông tin về chữ ký và các chứng chỉ.

2.2 MaMaDroid

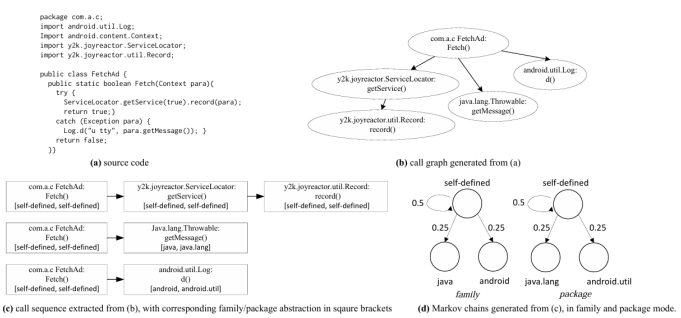

Figure 2: Quá trình MaMaDroid trích xuất đặc trưng

MaMaDroid là một công cụ trích xuất đặc trưng và tận dụng các mô hình machine learning cơ bản để giải quyết bài toán phần lớp nhị phân (binary classification) từ đó có thể phân biệt file mã độc và file bình thường.

MaMaDroid trích xuất đặc trưng từ 2 file Manifest và Dalvik Bytecode. Nội dung của file Manifest thường rất cụ thể ví dụ android.permission.SEND_SMS có nghĩa là ứng dụng này muốn có quyền gửi SMS. Do đó đặc trưng từ file Manifest thường được biểu diễn dưới dạng vector có giá trị nhị phân tượng trưng cho sự có hoặc không của một vài đặc trưng trong file manifest. Còn Dalvik Bytecode thì chứa code thực thi của ứng dụng Android, tất cả các code java được biên dịch và tập hợp lại thành 1 file classes.dex. Các đặc trưng được trích xuất từ file này là Control-Flow-Graph (CFG) và Data-Dependency-Graph (DDG) đó là hiểu biểu đồ thể hiện cấu trúc của một ứng dụng Android. Nó thường được biểu diễn dưới 2 dạng 1) là 1 chuỗi gọi API 2) những thông tin tĩnh được thu thập biểu đồ gọi hàm. Những đặc trưng này có khả n*u*ang phân biệt mạnh mẽ giữa 1 file thông thường và 1 file mã độc (vì file mã độc sẽ gọi tới những hành vi bất thường ví dụ: ứng dụng ghi chú nhưng lại gọi đến các hàm lấy thông tin trong danh bạ). Nó sử dụng chuỗi gọi API (API call sequence) thay vì kiểm tra ứng dụng có gọi đến các API khả nghi không, việc này nhằm nắm bắt được hành vi của ứng dụng. MaMaDroid có hai chế độ là family và package. Và các API call sẽ được trích xuất tùy theo chế độ (Hình 3).

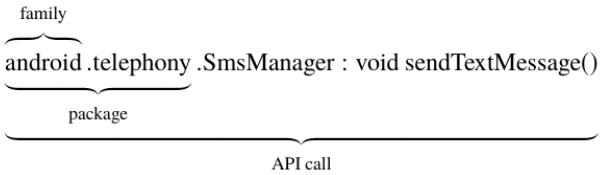

Figure 3: Cách MaMaDroid trích xuất API call

Family mode thì nhẹ hơn, trong khi đó package mode thì chi tiết hơn. Sau khi MaMaDroid lấy được API call sau đó nó sẽ xây dựng nên chuỗi Markov, và chuỗi này sẽ được sử dụng như đặc trưng của ứng dụng để huấn luyện cho mô hình machine learning để giải bài toán phân lớp nhị phân. Quá trình này được thể hiện cụ thể như hình 2.
Và với đặc trưng đã được trích xuất MaMaDroid tận dụng các thuật toán như Random Forest (RF), K-nearest neighbors(KNN), support vector machine(SVM) để huấn luyện cho mô hình machine learning để từ đó giải được bài toán phân lớp nhị phân (phân biệt file mã độc hay bình thường).

2.3 Drebin

Drebin cũng trích xuất đặc trưng từ file Manifest và Dalvik Bytecode. Nhưng thay vì xây dựng nên CFG hay DDG từ Dalvik Bytecode, Drebin sẽ thực hiện quét qua file và tìm các từ khóa là các đặc trưng được định nghĩa sẵn. Và việc quét này chỉ kiểm tra là file đó có hay không các từ khóa nguy hiểm. Sau đó Drebin tận dụng thuật toán Support Vector Machine (SVM) để xây dựng nên mô hình phát hiện mã độc.

2.4 C&W Attack

Để tấn công ta sẽ sử dụng thuật toán C&W attack. Thuật toán này vốn được dùng để tạo ra các mẫu hình ảnh đối kháng có khả năng làm sai phán đoán về nhận diện hình ảnh bằng cách làm nhiễu pixel. Tuy nhiên ở đây ta sẽ tuy chỉnh lại để cho phù hợp với mục tiêu của chúng ta, bằng cách tính toán số lượng API call và làm nhiễu nó chuỗi API call từ đó làm nhiễu vector đặc trưng để MaMaDroid không nhận diện đây là file mã độc. Mục đích chính của C&W attack là làm nhiễu vector đặc trưng từ đó khiến cho MaMaDroid phân lớp nhầm.
C&W attack phù hợp với dataset được trích xuất bằng công cụ MaMaDroid.

2.5 JSMA Attack

JSMA cũng là một thuật toán được dùng để tạo các mẫu hình ảnh đối kháng bằng cách làm nhiễu pixel. Nhưng nó sẽ có cách tính toán làm nhiễu khác với C&W attack. JSMA làm nhiễu bằng cách thử thay đổi 1 giá trị nào đó của 1 đối tượng rồi kiểm tra độ ảnh hưởng của sự thay đổi đó đối với việc các mô hình machine learning phân lớp, sau khi thử thay đổi tất cả các giá trị thì sẽ chọn ra giá trị có ảnh hưởng nhất rồi thay đổi giá trị đó, rồi lặp lại quá trình này cho tới khi có thể đánh lừa được mô hình phân lớp mà ta muốn tấn công.
JSMA attack phù hợp với dataset được trích xuất bằng công cụ Drebin

3 PHƯƠNG PHÁP THỰC HIỆN

‘

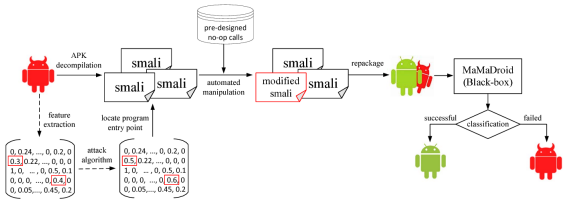

Figure 4: Quá trình thử nghiệm

Để thực hiện được đồ án chúng ta sẽ cần những thứ sau đây:

3.1 Dataset

Dataset ta sẽ dùng dataset đã được trích xuất đặc trưng để huấn luyện cho mô hình machine learning từ đó giải bài toán phân lớp nhị phân.
Dataset:drebin215dataset5560malware9476benign (https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning).

Tên Dataset	drebin215dataset5560malware9476benign
Số lượng đặc trưng	215
Số lượng dòng	15036
Lớp	Benign, Malicious
Số lượng file Benign	9476
Số lượng file Malicious	5560

Ngoài ra ta cũng sẽ tải vài file APK malware về để cho quá trình thực nghiệm chỉnh sửa file malware sao cho có thể tránh né trình phát hiện.

3.2 Xây dựng MaMaDroid

Như đã nói ở trên MaMaDroid là có thể xem như một chương trình để trích xuất đặc trưng từ file APK và tận dụng các thuật toán machine learning cơ bản để xây dựng mô hình machine learning để phát hiện mã độc
Ta sẽ sử dụng code được cung cấp bởi tác giả MaMaDroid để xây dựng nên MaMaDroid
(https://bitbucket.org/gianluca_students/mamadroid_code/src/master/).

3.3 Xây dựng Drebin

Ngoài MaMaDroid ta cũng có thể xây dựng Drebin để thay thế
(https://github.com/annamalai-nr/drebin)

3.4 Xây dựng C&W attack model

Như đã nói ở trên C&W attack là một thuật toán giúp tạo mẫu mã độc đối kháng bằng cách làm nhiễu vector đặc trưng nhờ việc thêm các API call vào trong file APK malware. Việc thêm bao nhiêu và thêm API call nào sẽ được tính toán bằng cách sử dụng AdaGrad để tối ưu số lượng các API call.
Nguồn tham khảo:
https://github.com/zacharykzhao/AndroidHIV/tree/main

3.5 Xây dựng JSMA attack model

Ngoài ra chúng ta cũng có thể dùng JSMA attack để tấn công https://github.com/hai2036/NT522.O21.ATCL-Machine-Learning-in-Android-Malware-Evasion

4 THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Kịch bản thử nghiệm

Do chúng ta không thành công trong việc xây dựng công cụ để trích xuất đặc trưng là MaMaDroid và Drebin, nhưng lại có dataset đã được drebin trích xuất đặc trưng sẵn nên chúng ta sẽ thực hiện theo kịch bản sau.

Đầu tiên Dataset đã được trích xuất đặc trưng sẽ được tải lên và xử lý nó để đưa nó về chuẩn để huấn luyện các mô hình. Sau đó ta sẽ chia ra thành 2 tập train và test. Ta sẽ đưa tập train vào huấn luyện mô hình SVM binary classifier sau đó đưa tập test vào để kiểm tra mô hình. Sau đó ta sẽ đưa tập test (chỉ những dòng được đánh lớp là 1 - malicious) qua JSMA attack để tạo ra các mẫu data đã được biến đổi. Sau đó ta sẽ kiểm tra những mẫu data này có né được hay không.

4.2 Thực nghiệm và đánh giá

Mô hình SVM binary classifier của chúng ta có kết quả cao hơn bài báo 1% vì dataset của chúng ta nhỏ hơn nhiều so với bài báo.

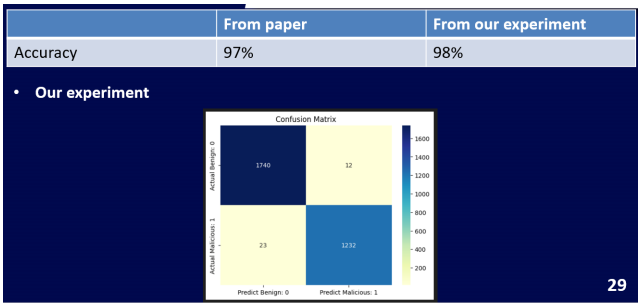

Figure 5: Kết quả của mô hình SVM

Kết quả né tránh của chúng ta trong kịch bản là ta có truy cập black-box vào công cụ trích xuất đặc trưng và có được dataset mà mô hình phát hiện mã độc được huấn luyện.

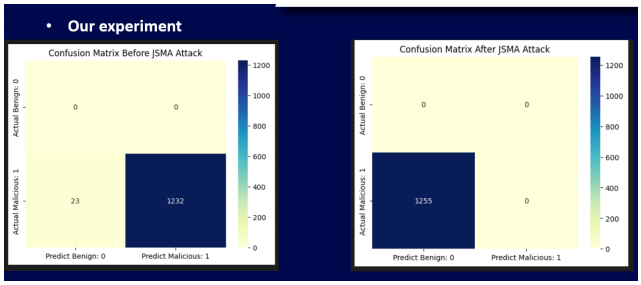

Figure 6: Trước và sau khi JSMA attack

Độ nhiễu trung bình là số lượng giá trị đặc trưng trung bình phải thay đổi ở mẫu tấn công để có thể né được. Kết quả của thực nghiệm là 8.9 còn bài báo là 3.5. Chúng ta có kết quả tệ hơn nhiều.

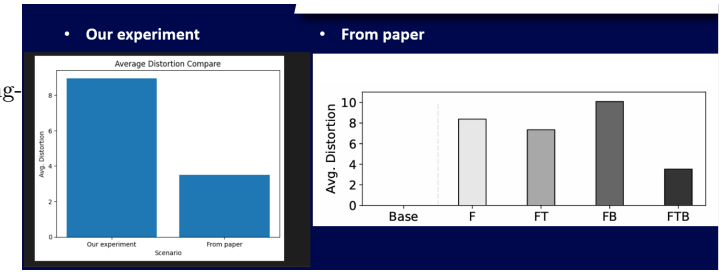

Figure 7: Độ nhiễu trung bình

Tỉ lệ né tránh được tính toán bằng tổng mẫu tấn công né được trên tổng mẫu tấn công. Tỉ lệ né tránh của chúng ta bằng với bài báo 100

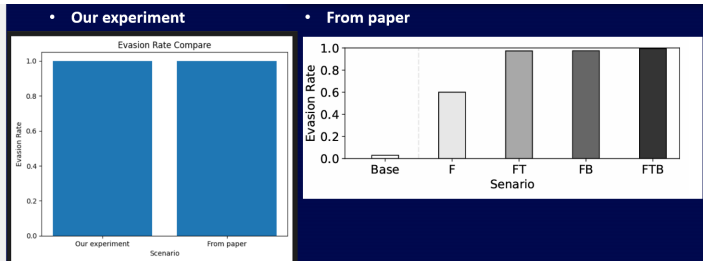

Figure 8: Tỉ lệ né

5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Qua đồ án này ta có thể thấy được các trình phát hiện mã độc dựa trên các mô hình machine learning đơn giản vẫn có điểm yếu trước các mẫu mã độc đối kháng nói chung, trên nền tảng Android nói riêng. Điểm yếu này là một điểm yếu chí mạng đối với các hệ thống phụ thuộc vào nó, nên ta cần có các giải pháp hợp lý như sử dụng các mẫu mã độc đối kháng để huấn luyện cho mô hình machine learning từ đó nâng cao khả nuang phân biệt các file độc hại hay bình thường dù cho các file đó đã được chỉnh sửa để đối kháng với mô hình.

Đồ án này đã thể hiện rất rõ sự hiểu quả của mẫu đối kháng có hiệu quả như thế nào trong việc đánh lừa hệ thống phát hiện mã độc trên nền tảng Android.

5.2 Hướng phát triển

Có thể xây dựng thành một ứng dụng web giống như trang virustotal.com ta chỉ cần chọn detection file APK malicious hay benign thì bỏ vào web sẽ cho ta kết quả. Nếu chọn evasion thì chỉ cần bỏ file APK malicious vào thì web sẽ tự động trả về 1 file APK malicious đã được chỉnh sửa để có thể né được phần detection của web

6 PHÂN CÔNG CÔNG VIỆC

Trần Tấn Hải - 21522036: Chịu trách nhiệm phân công công việc, viết report, ngoài ra còn hỗ trợ làm slide, hỗ trợ phần thực nghiệm và đánh giá và hỗ trợ các phần còn lại trong đồ án.
Bùi Nguyên Phúc - 21522469: Chịu trách nhiệm phần thực nghiệm và đánh giá cụ thể như chạy code, xây dựng môi trừng để chạy thực nghiệm, quay video demo.
Huỳnh Anh Nguyễn - 21522388: Chịu trách nhiệm phần làm slide, poster và các phần về giao diện của đồ án ngoài ra cũng hỗ trợ phần thực nghiệm và đánh giá.
Link github: https://github.com/hai2036/NT522.O21.ATCL-Machine-Learning-in-Android-Malware-Evasion

7 TÀI LIỆU THAM KHẢO

X. Chen et al., "Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 987-1001, 2020, doi: 10.1109/TIFS.2019.2932228.