

HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO NGHIÊN CỨU KHOA HỌC

XÂY DỰNG HỆ THỐNG PHÁT HIỆN PHẦN MỀM ĐỘC HẠI THÔNG MINH DỰA TRÊN MẠNG THÔNG TIN KHÔNG ĐỒNG NHẤT

Giảng viên hướng dẫn: **Ths. Thái Thị Thanh Vân**

Sinh viên thực hiện: Trần Gia Lương

Kiều Duy Khánh

Trương Quốc Quân

Nguyễn Thị Vân Anh

Hà Nội, 2021

LỜI CẢM ƠN

Trong thời gian làm nghiên cứu khoa học, chúng em đã nhận được nhiều sự giúp đỡ, đóng góp nhiệt tình của thầy cô, gia đình và bạn bè. Đầu tiên, chúng em xin được gửi lời cảm ơn chân thành và sâu sắc tới cô Ths. Thái Thị Thanh Vân, người đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt quá trình nghiên cứu khoa học, giúp chúng em có thêm kiến thức chuyên môn cũng như học hỏi được tinh thần trách nhiệm, thái độ làm việc nghiêm túc, hiệu quả từ cô.

Chúng em cũng xin chân thành cảm ơn các thầy cô giáo trong Học viện Kỹ thuật Mật Mã nói chung cùng các thầy cô trong hệ quản lý sinh viên và khoa Công nghệ thông tin nói riêng đã tận tình giảng dạy, truyền đạt cho chúng em những kiến thức và kinh nghiệm quý báu trong suốt thời gian học tập, giúp chúng em có được cơ sở lý thuyết vững vàng cũng như sự quan tâm và tạo mọi điều kiện thuận lợi cho chúng em trong quá trình thực hiện nghiên cứu khoa học. Cuối cùng, chúng em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và hoàn thành nghiên cứu khoa học.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế nên không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để chúng em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

LỜI NÓI ĐẦU	1
CHƯƠNG 1. TỔNG QUAN VỀ PHẦN MỀM ĐỘC HẠI VÀ PHƯƠNG PHÁP PHÁT HIỆN PHẦN MỀM ĐỘC HẠI ANDROID	4
1.1. Khái niệm	4
1.2. Tác hại của phần mềm độc hại Android	4
1.3. Phân loại phần mềm độc hại Android	4
1.4. Các phương pháp phát hiện phần mềm độc hại trên Android	8
1.4.1. Phân tích tĩnh	8
1.4.2. Phân tích động	11
1.4.3. Phân tích lai	11
CHƯƠNG 2. TỔNG QUAN VỀ MẠNG THÔNG TIN KHÔNG ĐỒNG NHẤT.....	13
2.1. Giới thiệu về mạng thông tin không đồng nhất	13
2.2. Một số khái niệm cơ bản trên mạng thông tin không đồng nhất.....	14
2.2.1. Mạng thông tin không đồng nhất	14
2.2.2. Lược đồ.....	15
2.2.3. Phân loại mạng thông tin không đồng nhất.....	15
2.3. Siêu đường đi.....	16
2.4. Độ đo tương đồng.....	18
2.5. Ứng dụng thực tế của mạng thông tin không đồng nhất trong bài toán phát hiện phần mềm độc hại Android	19
CHƯƠNG 3. ỨNG DỤNG MẠNG KHÔNG ĐỒNG NHẤT CHO BÀI TOÁN PHÁT HIỆN PHẦN MỀM ĐỘC HẠI ANDROID	21
3.1. Mô hình đề xuất.....	21
3.2. Xây dựng cơ sở dữ liệu.....	22
3.2.1. Thu thập dữ liệu.....	22
3.2.2. Dịch ngược và trích xuất lời gọi API	24
3.2.3. Xây dựng ma trận độ đo tương đồng dựa trên mạng thông tin không đồng nhất.	28
3.2.4. Đề xuất công thức trích xuất vector đặc trưng	30

3.3. Huấn luyện học máy và đánh giá mô hình	31
3.3.1. Phương pháp đánh giá	31
3.3.2. Thực nghiệm và đánh giá	32
3.4. Xây dựng hệ thống	38
3.4.1. Tổng quan về hệ thống	38
3.4.2. Các yêu cầu chức năng và phi chức năng của hệ thống	38
3.4.3. Phân tích hệ thống	38
3.4.4. Thiết kế	40
CHƯƠNG 4. KẾT LUẬN	42
4.1. Các vấn đề đã làm được	42
4.2. Các vấn đề còn tồn đọng của hệ thống	42
4.3. Hướng nghiên cứu tiếp theo	42
THAM KHẢO	43

Danh Mục Hình Vẽ

Hình 1.1 Banking Trojan tên Rotexy tìm cách truy cập và lấy cắp thông tin ngân hàng của người dùng sau khi được cài đặt.	5
Hình 1.2 Điện thoại bị dính adware tại màn hình chủ.....	6
Hình 1.3 Các ứng dụng lừa đảo trên android	6
Hình 1.4 Kiến trúc của công cụ ANDRUBIS	12
Hình 2.1 Mô hình mạng thông tin thư mục DBLP[3]	14
Hình 2.2 Mô hình mạng Twitter.....	14
Hình 2.3 Ví dụ một mạng thông tin không đồng nhất [3]	15
Hình 2.4 Phân loại các mô hình mạng thông tin không đồng nhất	16
Hình 2.5 Minh họa các siêu đường đi trong mạng DBLP.....	17
Hình 3.1 Minh họa mô hình đề xuất.....	21
Hình 3.2 Bộ dữ liệu Drebin	22
Hình 3.3 Bộ dữ liệu CIC Dataset.....	23
Hình 3.4 Bộ dữ liệu VirusShare	23
Hình 3.5 Cấu trúc một file apk	24
Hình 3.6 Quá trình dịch ngược file apk.....	24
Hình 3.7 Các file smali thu được sau khi dịch ngược	25
Hình 3.8 Các lời gọi API trong file smali.....	26
Hình 3.9 Danh sách các lời gọi API phần mềm độc hại Android hay dùng	27
Hình 3.10 Mạng thông tin không đồng nhất	28
Hình 3.12 Stratified Shuffle Split.....	32
Hình 3.13 Trước và sau khi khử nhiễu dữ liệu, v16.....	33
Hình 3.14 Trước và sau khi khử nhiễu dữ liệu, v32.....	33
Hình 3.16 Mô hình ca sử dụng của hệ thống.....	39
Hình 3.17 Đặc tả ca sử dụng Đoán nhận ứng dụng.....	39
Hình 3.18 Biểu đồ tuần tự ca sử dụng Đoán nhận ứng dụng	40
Hình 3.19 Giao diện hệ thống phát hiện phần mềm độc hại android.....	40
Hình 3.20 Quá trình chọn file.....	41

Hình 3.21 Kết quả đoán nhận	41
-----------------------------------	----

Danh Mục Bảng Biểu

Bảng 1.1 Top 10 phần mềm độc hại Android phổ biến nhất năm 2017.....	7
Bảng 1.2 Các kỹ thuật phân tích động.....	11
Bảng 2.1 Mô tả ý nghĩa của các siêu đường đi.....	17
Bảng 3.1 Mười sáu siêu đường đi được sử dụng trong mô hình.....	30
Bảng 3.2 Kết quả của quá trình học máy sử dụng mô hình Decision Tree	34
Bảng 3.3 Kết quả của quá trình học máy sử dụng mô hình Random Forest	35
Bảng 3.4 Kết quả của quá trình học máy sử dụng mô hình Support Vector Machine..	36
Bảng 3.5 Kết quả của Neural Network.....	37
Bảng 3.6 Bảng so sánh các mô hình.....	38

LỜI NÓI ĐẦU

Theo các số liệu thống kê mới nhất, Android hiện nay vẫn là hệ điều hành chiếm lĩnh phần lớn thị phần di động trên toàn cầu (87,7% - theo IDC 2018) và tỷ lệ này vẫn giữ nguyên cho đến năm 2021. Vì vậy, các thiết bị sử dụng hệ điều hành Android đã trở thành đối tượng bị tấn công của các ứng dụng độc hại android, gây ra mối đe dọa nghiêm trọng đối với việc rò rỉ dữ liệu cá nhân như vị trí người dùng, thông tin liên lạc, tài khoản, ảnh, vv... Thực trạng trên khiến việc nghiên cứu bài toán phát hiện phần mềm độc hại Android ngày càng trở nên cấp thiết.

Trước đây, hướng tiếp cận để giải quyết bài toán phát hiện phần mềm độc hại chủ yếu dựa vào kỹ thuật đối sánh mẫu truyền thống. Tuy nhiên, với sự phát triển nhanh chóng của các phương pháp học máy và trí tuệ nhân tạo, hướng nghiên cứu về việc phát triển các hệ thống tự động phát hiện phần mềm độc hại bằng cách sử dụng các kỹ thuật khai phá dữ liệu và học máy đang thu hút sự quan tâm của các nhà nghiên cứu trong lĩnh vực an ninh mạng. Vấn đề lớn nhất của bài toán phát hiện phần mềm độc hại sử dụng học máy đó là giải pháp phân tích, biểu diễn dữ liệu và trích chọn đặc trưng. Có hai cách tiếp cận để giải quyết vấn đề này, một là sử dụng kỹ thuật phân tích hành vi và hai là kỹ thuật phân tích chữ ký [1].

Kỹ thuật phân tích chữ ký dựa trên các dấu hiệu số (Digital Footprint) để đoán nhận mã độc. Tất cả các chương trình, dù là lành hay độc, đều có những dấu hiệu số đặc trưng của riêng mình. Các phần mềm diệt virus thường kết hợp với sử dụng một cơ sở dữ liệu lưu trữ các dấu hiệu số của các mã độc. Các phần mềm ấy sẽ đối sánh dấu hiệu số của file đang được quét với các dữ liệu trong cơ sở dữ liệu, Nếu tìm thấy dấu hiệu đó trong cơ sở dữ liệu, file đó sẽ được coi là độc. Khi một loại mã độc mới được phát hiện, dấu hiệu số của nó sẽ được thêm vào cơ sở dữ liệu của công ty viết ra phần mềm virus đó, và sẽ được chia sẻ cho cơ sở dữ liệu từ phía người dùng.

Hướng nghiên cứu phân tích hành vi dựa vào việc phân tích và đánh giá mã nguồn của ứng dụng được nghi ngờ. Trong bài toán đoán nhận mã độc Android, hướng nghiên cứu này dựa vào việc phân tích lời gọi API, quyền truy cập, lời gọi hệ thống. Các thông tin trên được biểu diễn bằng các mô hình mạng thông tin đồng nhất, nghĩa là nếu biểu diễn các dữ liệu này thành đồ thị thì các đỉnh trong đồ thị có chung một kiểu, và các cạnh của đồ thị đều biểu diễn một loại quan hệ. Trên mô hình mạng đã xây dựng, các nhóm sẽ đề xuất các thuật toán trích chọn đặc trưng phù hợp với mô hình học máy sử dụng. Hiệu quả của các hệ thống tự động phát hiện mã độc sử dụng mạng thông tin đồng nhất để biểu diễn dữ liệu và trích chọn đặc trưng cho kết quả dự đoán chính xác từ 85-92%. Nguyên nhân là do dữ liệu trong các hệ thống thực tế - đặc biệt là dữ liệu về mã

độc thường phong phú, đa dạng và chứa đựng nhiều ý nghĩa tiềm ẩn khác. Nếu sử dụng mạng đồng nhất để biểu diễn thì có thể sẽ làm mất đi các thông tin ngữ nghĩa quan trọng trong mạng và việc trích chọn đặc trưng trên mô hình mạng đồng nhất đã không tận dụng tối đa các thông tin thu được để phát hiện mã độc, đặc biệt đối với các mã độc thế hệ mới. Vì vậy hướng nghiên cứu ứng dụng các kỹ thuật phân tích dữ liệu phức tạp để giải quyết thách thức trên đang được rất nhiều nhà nghiên cứu quan tâm. Một trong các công cụ thường được sử dụng để phân tích, biểu diễn dữ liệu phức tạp hiện nay là sử dụng mô hình mạng thông tin không đồng nhất (Heterogeneous Information Network-HIN).

Mạng thông tin không đồng nhất gồm nhiều loại đối tượng khác nhau và các liên kết (cạnh) trong mạng thông tin không đồng nhất có thể mang nhiều ý nghĩa khác nhau. Bởi vậy, việc khai phá các mạng thông tin không đồng nhất sẽ cho chúng ta biết thêm các tri thức còn tiềm ẩn trong các cấu trúc mạng, từ đó có thể ứng dụng trong nhiều lĩnh vực khác nhau. Cho đến hiện nay, HIN đã được ứng dụng trong nhiều lĩnh vực khác nhau, như khai phá dữ liệu văn bản, khai phá dữ liệu sinh học, và đặc biệt ứng dụng trong bài toán phát hiện mã độc. Điển hình như nhóm nghiên cứu của Yanfang Ye [2] và đồng nghiệp đã sử dụng mạng thông tin không đồng nhất để biểu diễn dữ liệu mã độc Android, mạng bao gồm 5 loại đỉnh (ứng dụng, API, IMEI, nhà sản xuất, chữ ký) và 5 loại cạnh (ứng dụng – API, ứng dụng-IMEI, IMEI-Nhà sản xuất, ứng dụng-nhà sản xuất, ứng dụng – chữ ký). Kết hợp với mạng neural học sâu, hệ thống tự động phát hiện mã độc của nhóm đã cho kết quả dự đoán mã độc với độ chính xác đến 96%.

Trong đề tài nghiên cứu này, chúng em xây dựng hệ thống phát hiện phần mềm độc hại Android dựa trên mô hình mạng thông tin không đồng nhất. Cụ thể, trước hết chúng em xây dựng mạng thông tin không đồng nhất dựa trên việc phân tích lời gọi API và các mối liên quan giữa các lời gọi API. Sau đó, chúng em đề xuất công thức trích xuất đặc trưng để biểu diễn dữ liệu sang dạng vector và thực hiện việc huấn luyện với các mô hình học máy. Dựa trên các kết quả thực nghiệm, chúng em lựa chọn ba mô hình hiệu quả nhất để xây dựng hệ thống phát hiện phần mềm độc hại Android trên server. Các kết quả thực nghiệm cũng chứng minh tính hiệu quả của mô hình biểu diễn và công thức trích chọn đặc trưng mà nhóm đã đề xuất.

Ngoài phần mở đầu, kết luận và tài liệu tham khảo, nội dung bản báo cáo được chia làm 3 chương như sau :

Chương 1 : Nghiên cứu tổng quan về phần mềm độc hại Android và các kỹ thuật phân tích phần mềm độc hại.

Chương 2 : Nghiên cứu tổng quan về mạng thông tin không đồng nhất (HIN – Heterogeneous information Network) bao gồm các định nghĩa trên mạng không đồng nhất, lược đồ, siêu đường đi, ứng dụng mạng không đồng nhất cho bài toán phát hiện phần mềm độc hại.

Chương 3 : Trình bày về quá trình thực nghiệm từ tiền xử lý dữ liệu, trích xuất các lời gọi API cùng các mối quan hệ giữa chúng, tính toán độ đo AvgSim theo các meta-path, xây dựng mạng thông tin không đồng nhất và trích xuất các đặc trưng của các mẫu dữ liệu để đưa vào huấn luyện các mô hình học máy phổ biến, đánh giá các kết quả thực nghiệm và xây dựng thành phẩm hệ thống phần mềm chạy trên máy chủ.

Chương 4: Kết luận các vấn đề đã làm được, các vấn đề còn tồn đọng của hệ thống và hướng nghiên cứu tiếp theo.

Đây là một hướng nghiên cứu khó và mới ở trong nước, tuy nhiên qua quá trình làm việc nghiêm túc, đề tài cũng đã hoàn thiện và có hai kết quả đóng góp rất quan trọng:

Thứ nhất, chúng em đã đề xuất công thức trích chọn đặc trưng hoàn toàn khác so với các nghiên cứu đã có trước đây trên mạng thông tin không đồng nhất. Tính hiệu quả của công thức đã được chứng minh qua các thực nghiệm mà chúng em đã thực hiện.

Thứ hai, chúng em đã xây dựng thành công hệ thống phát hiện mã độc Android, bước đầu thực hiện trên server, dự kiến sau này sẽ tiếp tục phát triển thành ứng dụng hoàn chỉnh và có thể triển khai trên các thiết bị sử dụng hệ điều hành Android.

Mặc dù nhóm đã có rất nhiều cố gắng trong quá trình thực hiện đề tài nghiên cứu, tuy nhiên đây là một lĩnh vực khó và còn mới tại Việt Nam nên chắc chắn báo cáo vẫn còn rất nhiều thiếu sót, chúng em rất mong nhận được sự góp ý của các thầy cô để đề tài được hoàn thiện hơn.

CHƯƠNG 1. TỔNG QUAN VỀ PHẦN MỀM ĐỘC HẠI VÀ PHƯƠNG PHÁP PHÁT HIỆN PHẦN MỀM ĐỘC HẠI ANDROID

1.1. Khái niệm

Phần mềm độc hại là một dạng chương trình hoặc một đoạn chương trình tự cài đặt hoặc được cài đặt trên bất cứ thiết bị nào mà không cần sự cho phép của người dùng và thực hiện các chức năng mà người dùng không hề biết. Động cơ bản của phần mềm độc hại là đánh cắp thông tin bảo mật từ điện thoại thông minh, khóa điện thoại thông minh, gửi tin nhắn SMS/MMS, thực hiện cuộc gọi đến đầu số cước phí đặc biệt, chia sẻ thông tin thông qua định vị GPS.

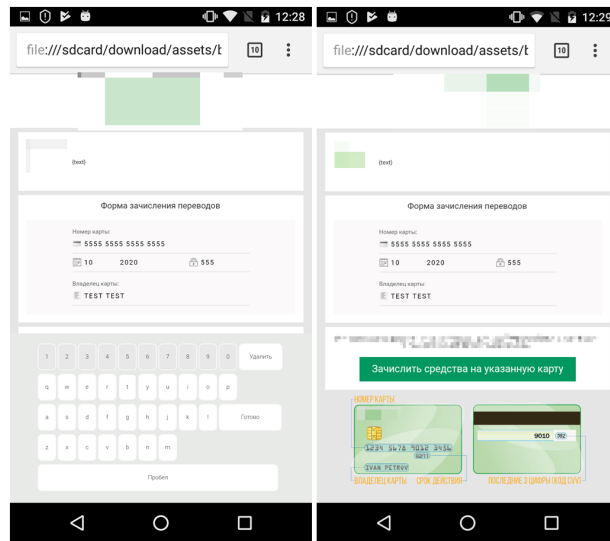
1.2. Tác hại của phần mềm độc hại Android

Tất cả những người dùng thiết bị di động chạy hệ điều hành Android có kết nối internet đều có thể trở thành nạn nhân của phần mềm độc hại. Phần mềm độc hại trên di động chủ yếu lấy cắp các dữ liệu nhạy cảm (như thẻ tín dụng, mật khẩu, nhật ký cuộc gọi, SMS) và theo dõi khách hàng. Ngoài ra, phần mềm độc hại có thể làm giảm tuổi thọ của pin hoặc giảm khả năng xử lý, chiếm quyền điều khiển trình duyệt, gửi tin nhắn trái phép và có thể vô hiệu hóa toàn bộ thiết bị. Một số trường hợp khách hàng mất tiền cước do phần mềm độc hại đăng ký dịch vụ game, thực hiện cuộc gọi quốc tế, ... Tốc độ gia tăng nhanh chóng của các ứng dụng được phát triển trên nền tảng hệ điều hành này chính là môi trường thuận lợi cho các tin tặc phát triển các loại phần mềm độc hại mới với chiêu thức hoạt động tinh vi hơn nhằm trục lợi từ thông tin cá nhân người dùng hoặc thậm chí là tống tiền.

1.3. Phân loại phần mềm độc hại Android

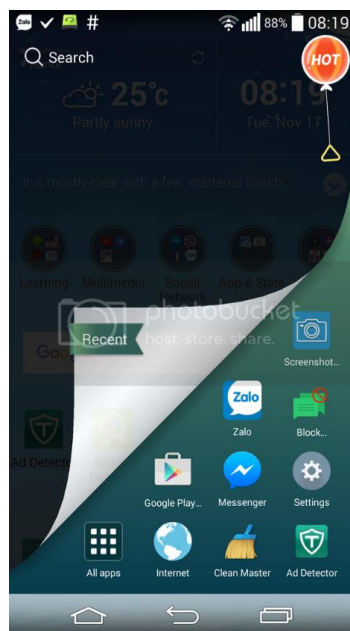
Dựa theo chức năng, phần mềm độc hại Android có thể được phân loại thành nhiều loại khác nhau như Spyware, Trojans, Virus, Phishing Apps, Bot Process, Rootkits.

- Trojan và Virus: Trojan là chương trình hợp pháp được cài đặt cùng với ứng dụng và sau đó thực hiện các hành động phá hoại như kiểm soát trình duyệt, tự động gửi tin nhắn đến các số điện thoại cước phí đặc biệt, chụp lại thông tin đăng nhập từ các ứng dụng khác như ứng dụng ngân hàng di động. Trojan có liên quan chặt chẽ tới virus di động bởi các tin tặc thường sử dụng virus di động để chiếm quyền quản trị cao nhất của thiết bị, truy cập đến các file và bộ nhớ flash để thực hiện các hành vi trái phép.



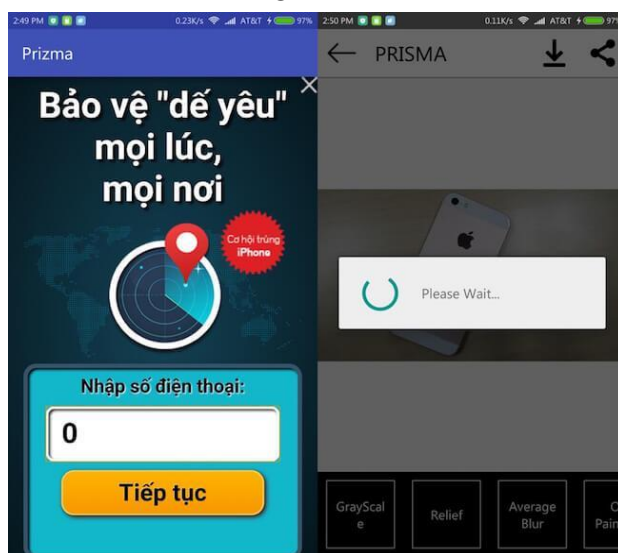
Hình 1.1 Banking Trojan tên Rotexy tìm cách truy cập và lấy cắp thông tin ngân hàng của người dùng sau khi được cài đặt.

- Phần mềm gián điệp và phần mềm quảng cáo (Spyware and Adware):
Spyware là phần mềm độc hại xâm nhập vào thiết bị di động để ngầm thu thập thông tin bảo mật của người dùng và chuyển tiếp đến cho bên thứ ba. Spyware thường chạy ngầm trong hệ thống và âm thầm giám sát, thu thập thông tin nhằm phá hoại thiết bị di động cũng như quá trình truy cập Internet bình thường của người dùng.
- Adware là phần mềm được thiết kế để hiển thị quảng cáo lên màn hình thiết bị của người dùng, thường nguy trang dưới dạng một chương trình hợp pháp hoặc ẩn mình trong một chương trình khác để lừa người dùng cài đặt nó. Khi xâm nhập thành công vào thiết bị, adware có thể thực hiện các tác vụ không mong muốn như: tự động mở tab mới, thay đổi trang chủ tìm kiếm, thu thập thông tin người dùng nhằm mục đích quảng cáo, chuyển hướng người dùng đến trang web khác.



Hình 1.2 Điện thoại bị dính adware tại màn hình chủ

- Các ứng dụng lừa đảo (Phishing Apps): Cũng giống như máy tính, kẻ tấn công tạo ra các ứng dụng lừa đảo trên thiết bị di động giống như một dịch vụ hợp pháp nhưng có thể đánh cắp thông tin nhạy cảm và thông tin xác thực để thực hiện các hành vi gian lận tài chính.



Hình 1.3 Các ứng dụng lừa đảo trên android

- Các hoạt động ảo: Phần mềm độc hại di động ngày càng tinh vi hơn với các chương trình hoạt động ngầm trên các thiết bị, tự ẩn mình và nằm đợi thời cơ để hành động, ví dụ như một phiên giao dịch ngân hàng trực tuyến để tấn công. Các hoạt động ảo có thể được thực hiện hoàn toàn vô hình đối với người dùng, thực thi nhiệm vụ hoặc kết nối với máy chủ điều khiển mạng máy tính mà để nhận các hướng dẫn mới cho hoạt động tiếp theo.

Dựa trên thống kê của Trend Micro, bảng dưới đây liệt kê 10 phần mềm độc hại Android với mô tả và chức năng của chúng.

STT	Tên	Mô tả	Chức năng
1	FakeInst	Gửi tin nhắn SMS tới các đầu số dịch vụ	Gửi tin nhắn SMS
2	OpFake	Gửi tin nhắn SMS tới các đầu số dịch vụ	Gửi tin nhắn SMS
3	SNDApps	Đánh cắp hàng loạt thông tin như ID thiết bị, email, địa chỉ, số điện thoại từ thiết bị và đăng tải thông tin lên một máy chủ từ xa	Gửi tin nhắn SMS
4	Boxer	Gửi tin nhắn SMS tới các đầu số dịch vụ đặc biệt	Gửi tin nhắn SMS
5	GinMaster	Đánh cắp thông tin bảo mật từ thiết bị và gửi tới một máy chủ từ xa	Đánh cắp thông tin
6	VDLoader	Đánh cắp thông tin cá nhân	Quyền truy cập gốc và đánh cắp thông tin
7	FakeDolphin	Cung cấp một trình duyệt thay thế và đăng ký cho người dùng các dịch vụ mà họ không biết	Đánh cắp thông tin
8	KungFu	Gửi thông tin bảo mật như IMEI, SMS, IMSI tới một máy chủ từ xa	Botnet và đánh cắp thông tin
9	BaseBridge	Gửi thông tin bảo mật như IMEI, SMS, IMSI tới một máy chủ từ xa	Botnet và đánh cắp thông tin
10	JIFake	Gửi tin nhắn SMS tới các đầu số dịch vụ	Gửi tin nhắn SMS

Bảng 1.1 Top 10 phần mềm độc hại Android phổ biến nhất năm 2017

1.4. Các phương pháp phát hiện phần mềm độc hại trên Android

1.4.1. Phân tích tĩnh

Đối tượng chính của phân tích tĩnh trong phần mềm độc hại Android là kiểm tra các quyền, mã nguồn ứng dụng, bộ phận cấu thành, tài nguyên và mã nhận diện. Tất cả thông tin liên quan đến ứng dụng nằm trong tệp APK. Quyền truy cập, tài nguyên, mã, dịch vụ và tất cả thông tin khác được trích xuất từ tệp APK và được phân tích chính xác. Có nhiều công cụ có sẵn khác nhau cho phân tích tĩnh như apktool, aapt, dex2jar, jd-gui. Đối với việc áp dụng phân tích tĩnh vào ứng dụng Android, ta có thể tiếp cận theo hướng như sau: ứng dụng Android hầu như được xây dựng trên bốn thành phần bao gồm Activity biểu diễn các giao diện có thể tương tác với người dùng, Service thực thi các tác vụ ngầm, Content Provider truy cập cấu trúc dữ liệu cho ứng dụng và Broadcast Receiver xử lý các sự kiện hệ thống hoặc của người dùng. Các thành phần này tương tác với nhau thông qua các phương thức, ví dụ như startActivity() để kích hoạt giao tiếp liên thành phần (Inter-Component Communication – ICC). Phương thức của ICC nhận các tham số Intent chứa dữ liệu thành phần đích mà thành phần gốc muốn giao tiếp cùng. Có hai loại tương tác ICC là ICC rõ ràng, nơi Intent chứa tên thành phần đích và ICC không rõ ràng, nơi Intent chỉ chứa các mục tiêu tiềm năng, có khả năng xử lý, các Intent này cần được khai báo một Intent Filter trong tệp cấu hình AndroidManifest. Các phương pháp phân tích tĩnh có thể kể đến như sau:

- Phân tích tĩnh dựa trên chữ ký/mẫu: Tệp APK chứa chữ ký cụ thể. Mỗi tệp APK có chữ ký khác nhau. Chữ ký chứa thông điệp rút gọn của tệp APK. Nếu có bất kỳ sự thay đổi nào trong tệp APK thì thông điệp rút gọn của chữ ký cũng sẽ thay đổi và người ta có thể nhanh chóng phân tích được ứng dụng là độc hại. Chữ ký của phần mềm độc hại cũng có thể được thu thập để xác định phần mềm độc hại một cách nhanh chóng. Điểm hạn chế của phương pháp này chính là nó chỉ có thể phát hiện được các phần mềm độc hại hiện có, đã được biết tới; trong khi phần mềm độc hại luôn được phát triển và ngày càng tinh vi hơn.
- Phân tích tĩnh dựa trên tài nguyên: Tệp AndroidManifest.xml là một tệp tài nguyên chứa tất cả thông tin về các tài nguyên được sử dụng trong ứng dụng [15]. Tài nguyên của một ứng dụng chứa các module giao diện người dùng như widget, menu, layout, ...Tệp AndroidManifest.xml có trong tệp APK. Phần mềm độc hại chạy ngầm cần sự tương tác của người dùng từ

các giao diện người dùng này. Vì vậy, giao diện người dùng cũng một phần quan trọng cần được phân tích chính xác.

- Phân tích tĩnh dựa trên thành phần: Ứng dụng Android được chia thành nhiều thành phần như nhà cung cấp nội dung, dịch vụ, đối tượng, hoạt động và thu nhận tín hiệu. Thông tin của tất cả các thành phần có sẵn trong tệp AndroidManifest.xml. Hầu hết phần mềm độc hại chạy dưới dạng dịch vụ ngầm và thu thập thông tin về đối tượng, hoạt động, đối tượng thu nhận. Vì vậy việc phân tích các thành phần này cũng rất quan trọng trong việc nhận ra các hành vi độc hại.
- Phân tích tĩnh dựa trên quyền truy cập: Android cung cấp mô hình dựa trên quyền truy cập cho việc triển khai bảo mật sẵn có. Tất cả các quyền truy cập được định nghĩa trong tệp AndroidManifest.xml. Các ứng dụng phải yêu cầu quyền truy cập như danh bạ, tin nhắn, internet, GPS, máy ảnh, ...Quyền truy cập đóng vai trò quan trọng trong bất kỳ ứng dụng Android nào. Một ứng dụng chỉnh ảnh đơn giản yêu cầu quyền 'READ_SMS', sau đó thực hiện một số hành vi độc hại. Vì vậy, việc phân tích các quyền cũng rất quan trọng để nhận ra các hành vi độc hại.
- Kỹ thuật phân tích sử dụng công nghệ máy ảo: Kỹ thuật phân tích này kiểm tra hành vi ứng dụng, phân tích quyền kiểm soát và luồng dữ liệu để dò tìm những chức năng nguy hiểm. Nó có thể kiểm tra bytecode của ứng dụng và theo dõi các lời gọi API nhạy cảm. Tuy nhiên với kỹ thuật này, phân tích được thực hiện bởi các lệnh, nó cũng làm tốn pin và dung lượng lưu trữ hơn.

Phương pháp phân tích tĩnh trên Android đôi khi gặp phải những vấn đề cố hữu, mà việc khắc phục rất khó khăn và gần như bất khả thi, bao gồm:

- Mã biên dịch: Như đã biết, ứng dụng Android phần lớn được xây dựng từ ngôn ngữ Java nhưng được biên dịch thành Dex bytecode và thực thi trên máy ảo Dalvik hoặc ART, tuy nhiên các chương trình thu thập dữ liệu phục vụ phân tích tĩnh lại không có khả năng giải quyết Dex bytecode hoặc các định dạng hỗ trợ. Do đó, các chương trình này tuy hoạt động được với mã nguồn Java nhưng lại trở nên vô tác dụng trong hệ sinh thái Android. Ví dụ như phần mềm FindBug 5 được phát triển và hoạt động tốt trong việc phát hiện phần mềm độc hại Java, nhưng lại không thể đưa vào khai thác đối với Android.

- **Điểm đầu vào:** Không như các chương trình Java hay C, Android không có hàm main(), thay vào đó là các điểm đầu vào được gọi bởi Framework trong quá trình chạy. Đối lập với điều này, các ứng dụng phân tích phần mềm độc hại cần phải xác định điểm vào và sơ đồ hoá các liên kết. Với nhiều điểm đầu vào, việc xác định này rất khó khăn và không thể đảm bảo được sự liên kết giữa chúng.
- **Vòng đời ứng dụng:** Một vấn đề nữa đối với hệ điều hành Android đó chính là vòng đời của các thành phần ứng dụng. Mỗi một thành phần lại có một vòng đời riêng và đảm nhiệm một tác vụ hệ thống, và được gọi đến với nhiều kịch bản. Ví dụ, một ứng dụng khi không còn được sử dụng mà vẫn chiếm quá nhiều tài nguyên, hệ thống sẽ tự động đóng tiến trình mà không đi qua các hàm xử lý như vòng đời thông thường, các ứng dụng phân tích phần mềm độc hại sẽ gặp rất nhiều khó khăn khi xây dựng kịch bản hoạt động.
- **Sự kiện:** Sự kiện trên hệ điều hành Android có thể chia làm hai phần là sự kiện hệ thống (Cảnh báo pin, GPS ...) và sự kiện người dùng (thao tác với UI). Các sự kiện này có thể xảy ra liên tục, chồng chéo nhau và có thể rơi vào bất kỳ thời điểm nào khi thiết bị hoạt động, khiến việc xây dựng và duy trì mô hình phân tích trở nên bất khả thi.
- **Giao tiếp liên thành phần:** Android có cơ chế giao tiếp giữa các thành phần, thậm chí còn diễn ra giữa hai ứng dụng khác nhau. Ứng dụng phân tích muốn quản lý các giao tiếp này yêu cầu một phương pháp phỏng đoán tiên tiến, nơi mà các liên kết được kích hoạt bằng các phương thức cụ thể với các tham số đa dạng. Mặt khác, các phương thức này còn được quản lý bởi vòng đời ứng dụng, hệ thống này chịu trách nhiệm trực tiếp trong quá trình khởi chạy.
- **Thư viện:** Thông thường, một ứng dụng Android sẽ sử dụng một hoặc nhiều thư viện để phục vụ người dùng, như thư viện quảng cáo hoặc mua bán. Những thư viện này có rất nhiều dòng mã, khiến kích thước của ứng dụng bị tăng lên đáng kể. Đối với phân tích tĩnh, việc đi sâu vào mã nguồn ứng dụng với các bộ thư viện này là không cần thiết và tiêu tốn rất nhiều thời gian, thậm chí còn nhiều hơn so với phân tích mã nguồn gốc. Mặt khác, phân tích các thư viện này tiềm ẩn nguy cơ thu thập các mẫu dương tính do cần yêu cầu các tài nguyên mà bị nhầm lẫn không cần thiết cho ứng dụng.

1.4.2. Phân tích động

Phân tích động bao gồm phân tích hành vi của ứng dụng. Nó kiểm tra sự tương tác của phần mềm độc hại với các tài nguyên di động và các dịch vụ như vị trí, mạng, gói, các hoạt động hệ điều hành. Đặc trưng của phân tích động bao gồm lời gọi hệ thống, lưu lượng mạng, luồng mạng, địa chỉ mạng. Phân tích động giám sát hành vi của hệ thống. Một số framework có sẵn để thực hiện phân tích động có thể kể đến như Ananas, TaintDroid, DroidScope, CopperDroid, Crowdroid. Các kỹ thuật phân tích động phổ biến được trình bày trong bảng sau:

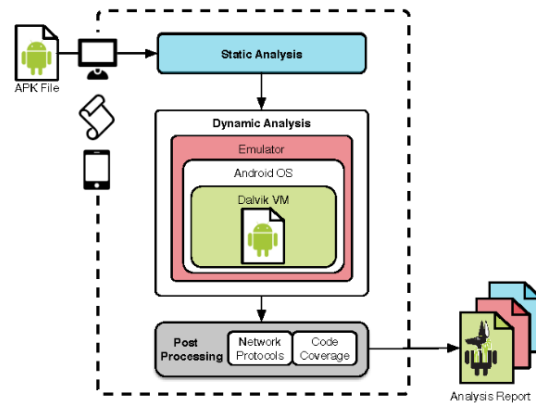
STT	Kỹ thuật phân tích	Cách thức hoạt động	Ưu điểm	Nhược điểm
1	Kỹ thuật phân tích dựa trên các hoạt động bất thường	Dựa trên việc quan sát hành vi thiết bị bằng cách theo dõi các tham số khác nhau và trạng thái của các thành phần thiết bị	Theo dõi các tham số khác nhau từ đó có cái nhìn rõ ràng về hệ thống	Số tham số tham gia càng nhiều thì các phép toán được yêu cầu càng nhiều
2	Kỹ thuật phân tích theo vết dữ liệu	Theo dõi nhiều nguồn dữ liệu nhạy cảm và phát hiện rò rỉ dữ liệu trong các ứng dụng di động	Theo dõi các dữ liệu nhạy cảm một cách hiệu quả	Không thực hiện theo dõi luồng kiểm soát
3	Kỹ thuật phân tích dựa trên mô phỏng	Phân tích ứng dụng dựa trên kỹ thuật giám sát thời gian chạy của máy ảo cấp hệ thống	Giám sát toàn bộ hệ thống bằng cách ở ngoài môi trường thực thi	Không phát hiện được các phần mềm độc hại mới

Bảng 1.2 Các kỹ thuật phân tích động

1.4.3. Phân tích lai

Phân tích lai là sự kết hợp của cả phân tích tĩnh và phân tích động. Quá trình bắt đầu với phân tích tĩnh. Phân tích tĩnh kiểm tra mã, quyền truy cập và các thành phần của ứng dụng. Sau đó, phân tích động thực hiện phân tích hành vi

tổng thể của ứng dụng. Chỉ có một số framework dùng cho phân tích lại như Mobile Sandbox, Andrubis. Hệ thống AASandbox (Android Application Sandbox) đưa ra phân tích hai bước cho các ứng dụng Android. Một ứng dụng di động gửi đến AASandbox, tại đó nó thực hiện phân tích tĩnh và động trong chế độ ẩn (offline). Phân tích tĩnh vô hiệu hóa mã nhị phân ứng dụng cài đặt và sử dụng từng đoạn mã để tìm các mẫu nghi vấn. Phân tích động thực thi mã nhị phân trên bộ mô phỏng Android và ghi lại các lời gọi hệ thống.



Hình 1.4 Kiến trúc của công cụ ANDRUBIS

Bên cạnh hai phương pháp phổ biến trên cũng có một số kỹ thuật khác cũng rất phổ biến như phân tích quyền ứng dụng, giám sát tuổi thọ pin hay gần đây nhất, với sự bùng nổ của công nghệ học máy và trí tuệ nhân tạo, việc phân tích đã được xây dựng tự động giúp rút ngắn tối đa thời gian cũng như chi phí phân tích và phát hiện phần mềm độc hại trên di động.

Phát hiện phần mềm độc hại là một yếu tố quyết định trong bảo mật hệ điều hành Android. Với sự phát triển nhanh chóng của các phương pháp học máy và trí tuệ nhân tạo, hướng nghiên cứu về việc phát triển các hệ thống tự động phát hiện phần mềm độc hại bằng cách sử dụng các kỹ thuật khai phá dữ liệu và học máy đang thu hút sự quan tâm của các nhà nghiên cứu trong lĩnh vực an ninh mạng. Với đề tài nghiên cứu này, chúng em sử dụng phương pháp phân tích tĩnh trong đó sử dụng thông tin về các lời gọi API và thông tin về mối quan hệ giữa các lời gọi API (các API cùng package, cùng block và cùng kiểu invoke), để xây dựng hệ thống phát hiện phần mềm độc hại dựa trên mạng thông tin không đồng nhất (HIN).

CHƯƠNG 2. TỔNG QUAN VỀ MẠNG THÔNG TIN KHÔNG ĐỒNG NHẤT

2.1. Giới thiệu về mạng thông tin không đồng nhất

Ngày nay chúng ta đang sống trong một xã hội kết nối, hầu hết các đối tượng đều được liên kết với nhau để tạo thành một mạng kết nối khổng lồ, có thể gọi chung là mạng thông tin (information network). Chẳng hạn như mạng xã hội, mạng world wide web, mạng giao thông giữa các thành phố, mạng sinh học, ... Những mạng thông tin phổ biến này tạo thành một thành phần quan trọng của cơ sở hạ tầng thông tin hiện đại.

Hầu hết các nghiên cứu hiện nay trong lĩnh vực phân tích mạng thông tin (information network analysis) đều giả thiết các mạng này là *mạng thông tin đồng nhất (homogeneous information network)*, nghĩa là xem các đối tượng (đỉnh) là cùng kiểu và các liên kết giữa các đối tượng trong mạng cũng cùng một kiểu. Ví dụ như mạng liên kết các tác giả trong lĩnh vực nghiên cứu chỉ chứa các đối tượng tác giả cùng mỗi liên kết đồng tác giả. Các mạng thông tin đồng nhất này được đơn giản hóa từ mạng thông tin trên thực tế bằng cách bỏ qua tính không đồng nhất giữa các đối tượng và liên kết, hoặc chỉ xét một loại liên kết giữa một loại đối tượng. Tuy nhiên, trong thực tế, các mạng thông tin đều chứa các thành phần khác nhau, các tương tác giữa chúng cũng khác nhau và có thể mô hình hóa như một *mạng thông tin không đồng nhất (heterogeneous information network - HIN)*, trong đó các đỉnh có thể là các kiểu đối tượng khác nhau và các cạnh có thể là các liên kết khác nhau.

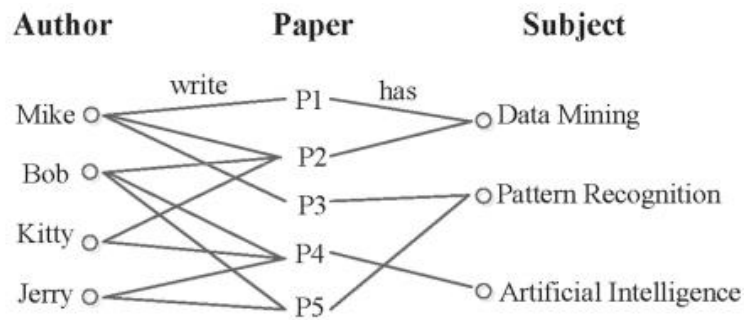
Chẳng hạn như mạng thông tin thư mục (Bibliographic information network) [3] DBLP, là một mạng không đồng nhất điển hình, bao gồm ba loại thực thể: bài báo (P), hội nghị (V), tác giả (A) và thuật ngữ (T). Đối với mỗi bài báo $p \in P$, nó sẽ có các mối liên kết dẫn tới một nhóm các tác giả, các hội nghị, một tập hợp các thuật ngữ. Ngoài ra, nó cũng có thể chứa thông tin trích dẫn cho một số bài báo, nghĩa là các liên kết tới một tập các bài báo được trích dẫn và liên kết từ một tập tới các bài báo trích dẫn đó. Lược đồ mạng cho một mạng thông tin thư mục và một thể hiện của nó được hiển thị như hình dưới.

tin bao gồm các đối tượng và mối liên kết giữa các đối tượng đó. Chúng ta có thể định nghĩa như sau:

Định nghĩa 1.1 Mạng thông tin [2]: Mạng thông tin là một đồ thị có hướng $G = (V, E)$ với mỗi kiểu thực thể là một ánh xạ $\varphi: V \rightarrow A$ và mỗi kiểu liên kết là một ánh xạ $\psi: E \rightarrow R$. Trong đó, mỗi đỉnh $v \in V$, thuộc về một kiểu đối tượng cụ thể của tập A : $\varphi(v) \in A$ và một liên kết $e \in E$ thuộc về một kiểu dữ liệu cụ thể của tập R : $\psi(e) \in R$.

Định nghĩa 1.2 Mạng thông tin được gọi là không đồng nhất nếu số lượng phân tử trong $A > 1$ hoặc số lượng phân tử trong $R > 1$

Dưới đây là một mạng thông tin không đồng nhất gồm ba loại đỉnh và hai loại liên kết. Ba loại đỉnh gồm Author (tác giả), Paper (bài báo) và Subject (chủ đề). Hai loại liên kết: liên kết *viết* giữa tác giả - bài báo và liên kết *thuộc về* giữa bài báo và chủ đề.



Hình 2.3 Ví dụ một mạng thông tin không đồng nhất [3]

2.2.2. Lược đồ

Để có thể hiểu rõ hơn các kiểu đối tượng và các kiểu liên kết trong mạng không đồng nhất, người ta đưa ra khái niệm về lược đồ mạng. Lược đồ mạng dùng để mô tả cấu trúc của mạng.

Định nghĩa 1.3 Lược đồ mạng [2]: Lược đồ mạng được biểu diễn bằng $T_G = (A, R)$ là một đồ thị có hướng với mỗi đỉnh là một kiểu thực thể trong A và mỗi cạnh là một kiểu liên kết trong R .

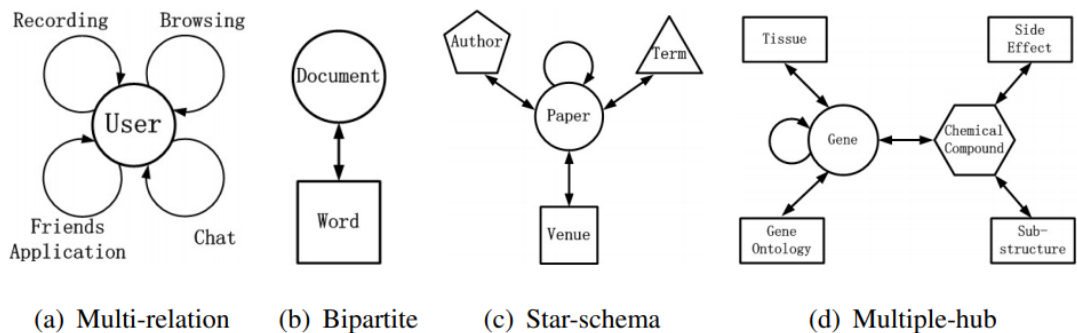
Mỗi kiểu liên kết R giữa đối tượng S và đối tượng T , nghĩa là $S \rightarrow^R T$, trong đó S được gọi là tập nguồn, T được gọi là tập đích, có thể được ký hiệu là $R_o S$ hoặc $R_o T$ tương ứng. Quan hệ nghịch đảo R^{-1} được định nghĩa $T \rightarrow^{R^{-1}} S$.

2.2.3. Phân loại mạng thông tin không đồng nhất

Ta có thể phân loại mạng thông tin không đồng nhất dựa trên thù hình mạng (Topology). Dựa vào tiêu chí này, ta phân ra làm những loại sau:

- Mạng đa quan hệ với một kiểu thực thể (multiple relation): Mạng có một kiểu nút nhưng có nhiều mối quan hệ giữa chúng. Mối quan hệ giữa người và người có dạng như vậy, giữa con người với nhau có thể tồn tại các cung bậc cảm xúc, từ yêu mến đến sợ hãi,...
- Mạng lưỡng phân (Bipartite network): gồm hai kiểu thực thể khác nhau và các liên kết giữa chúng. Mạng hai phía được sử dụng để mô phỏng tương tác giữa hai kiểu đối tượng. Một ví dụ điển hình có thể kể đến là quan hệ giữa các kiểu đối tượng, như user – item hay document – word.
- Mạng hình sao (Star - scheme network): là loại mạng phổ biến nhất, gồm một thực thể là thành phần trung tâm và những thực thể khác nằm xung quanh. Ví dụ về mạng thông tin thư mục ở trên thể hiện rõ nhất các khía cạnh của mạng hình sao.
- Mạng đa trung tâm (Multi - hub network): Các mạng hình sao có thể kết hợp với nhau, tạo thành mạng đa trung tâm. Mạng này chứa những cấu trúc phức tạp hơn, liên quan nhiều đến những đối tượng trung tâm. Mạng tương tác thành phần nhân tế bào của Tsuyuzaki và Nikaido chính là một điển hình cho loại mạng này.

Dưới đây là các hình ảnh minh họa của các loại mạng trên:



Hình 2.4 Phân loại các mô hình mạng thông tin không đồng nhất

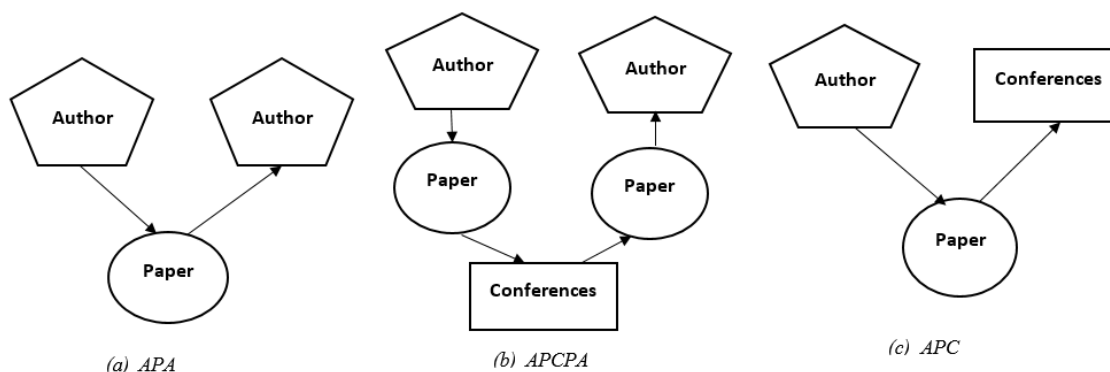
2.3. Siêu đường đi

Khác với mạng đồng nhất, hai đối tượng trong mạng thông tin không đồng nhất có thể kết nối với nhau thông qua các đường đi mang nhiều ngữ nghĩa khác nhau. Các đường này được gọi là siêu đường đi (Meta path) và được định nghĩa như sau.

Định nghĩa 1.4 Siêu đường đi [3]: Siêu đường đi (Meta-path) P là một đường đi trên đồ thị của lược đồ mạng $T_G = (A, R)$ và được ký hiệu dưới dạng $A_1 \rightarrow^{R_1} A_2 \rightarrow^{R_2} \dots \rightarrow^{R_l} A_{l+1}$, xác định mối quan hệ $R = R_1 \circ R_2 \circ \dots \circ R_l$ giữa các

đối tượng A_1, A_2, \dots, A_{l+1} Trong đó, \circ là toán tử biểu thị thành phần quan hệ và l là độ dài của đường đi P .

Trên mạng thông tin không đồng nhất DBLP, ta có thể có các siêu đường đi như sau:



Hình 2.5 Minh họa các siêu đường đi trong mạng DBLP

Các tác giả có thể liên kết với nhau thông qua các siêu đường đi sau “Tác giả - Bài báo – Tác giả” (APA), “Tác giả - Bài báo – Hội nghị - Bài báo – Tác giả” (APCPA), “Tác giả - Bài báo – Hội nghị” (ACP). Cụ thể, bảng sau cho thấy các thể hiện cũng như ngữ nghĩa của các siêu đường đi. Siêu đường đi APA cho thấy mối quan hệ giữa các tác giả cùng cộng tác trong một bài báo, trong khi đường đi APCPA lại biểu diễn cho quan hệ giữa các tác giả công bố bài báo trong cùng một hội nghị.

STT	Siêu đường đi	Ý nghĩa
1	Tác giả - Bài báo – Tác giả (APA)	Biểu diễn mối quan hệ đồng tác giả trong một bài báo.
2	Tác giả - Bài báo – Hội nghị - Bài báo -Tác giả (APCPA)	Các tác giả công bố bài báo trong cùng một hội nghị.
3	Tác giả - Bài báo – Hội nghị (APV)	Các tác giả công bố bài báo trong cùng một hội nghị.

Bảng 2.1 Mô tả ý nghĩa của các siêu đường đi

Thông qua quá trình ta phân tích ví dụ, ta có thể rút ra một số nhận xét về mạng không đồng nhất và siêu đường đi.

- Thứ nhất, siêu đường đi khác nhau sẽ mang ngữ nghĩa khác nhau. Sự phong phú về mặt ngữ nghĩa của siêu đường đi là một đặc tính quan trọng trong mạng thông tin không đồng nhất.

- Thứ hai, chúng ta thấy rằng, giữa A và B có nhiều mối quan hệ, nhưng sẽ có các quan hệ có trọng số lớn hơn những quan hệ khác. Ở ví dụ trên, nếu dựa theo siêu đường đi APA, các tác giả trong một bài báo (đồng tác giả) sẽ được đánh giá là “gần” nhau hơn. Tuy nhiên, nếu theo siêu đường đi APCPA thì các tác giả công bố bài báo trong cùng một hội nghị sẽ được đánh giá là “gần” nhau hơn. Độ tương đồng giữa hai tác giả sẽ được đánh giá khác nhau dựa trên các siêu đường đi khác nhau. Điều này là vì mạng không đồng nhất mang lại rất nhiều sự tương tác giữa các loại đối tượng. Do đó, khả năng dự đoán mối quan hệ giữa chúng cũng không hề nhất quán giữa các siêu đường đi.
- Thứ ba, và cũng là điểm quan trọng nhất, chúng ta sẽ cần có một tiêu chuẩn nhất định để đánh giá trọng số của chúng. Để làm vậy, chúng ta cần có các độ đo tương đồng, được trình bày ở dưới đây.

2.4. Độ đo tương đồng

Độ đo tương đồng được sử dụng để đánh giá sự tương đồng giữa các đối tượng. Hiện tại, có hai hướng tiếp cận khác nhau trên lĩnh vực này, là hướng đặc trưng và hướng liên kết.

Hướng đặc trưng hầu hết đều dựa vào các giá trị đặc trưng của các thành phần trên một siêu đường đi, chẳng hạn như độ tương đồng cosine (Cosine similarity score), hệ số Jaccard (Jaccard score) hay khoảng cách Euclid (Euclidean distance). Mặt khác, hướng tiếp cận liên kết lại lợi dụng các cấu trúc trên lược đồ, và đã tạo ra rất nhiều độ đo, như SimRank, đánh giá sự giống nhau của hai đối tượng bởi sự tương đồng giữa các kề cận của chúng; PathSim, được sử dụng để tìm các đối tượng ngang hàng trong mạng, tạo ra kết quả tốt hơn so với việc bước ngẫu nhiên cũng dựa trên phương pháp đo tương đồng; HeteSim để tính điểm phù hợp giữa các đối tượng thuộc các dạng khác nhau.[Laplacian][7].

Vậy nhưng các độ đo hướng liên kết này đều bộc lộ những yếu điểm: SimRank bị ảnh hưởng mạnh bởi nhiễu, và trong một số trường hợp, SimRank có thể bị lệch trọng tâm (Topic Drift). PathSim không những chỉ sử dụng được trên các đường đi đối xứng, mà còn chỉ có thể đo độ tương đồng giữa các đối tượng cùng loại, HeteSim khắc phục được những nhược điểm trên, nhưng vô cùng phức tạp, và hiệu suất thấp, không thể áp dụng cho các hệ thống với bộ dữ liệu lớn.

Để khắc phục những thiếu sót của HeteSim trong tính toán và nhu cầu bộ nhớ cao, Meng cùng các cộng sự đã đề xuất độ đo AverageSim để đánh giá độ đo tương tự thông qua bước đi ngẫu nhiên theo siêu đường đi và đường nghịch đảo tương ứng (P và P^{-1}) [3]. Độ đo AvgSim là độ đo đối xứng và có tính tổng quát để đánh giá mức độ liên quan của hai đối tượng cùng kiểu hay khác kiểu. Giá trị AvgSim của hai đối tượng là trung bình của xác suất có thể đến được theo đường đi đã cho và đường ngược lại.

2.5. Ứng dụng thực tế của mạng thông tin không đồng nhất trong bài toán phát hiện phần mềm độc hại Android

Mạng thông tin không đồng nhất đang là hướng đi mới trong phát hiện phần mềm độc hại Android, có thể thấy rõ qua các nghiên cứu trong vòng ba năm trở lại đây của các nhóm nghiên cứu trên thế giới.

Shifu Hou, Yanfang Ye và đồng nghiệp kết hợp sử dụng các lời gọi API và mối quan hệ giữa chúng để tìm ra những mối quan hệ ẩn giấu bên trong mạng, kết hợp với trích xuất đặc trưng và mô hình học máy MKL. [2]

Một năm sau, Yanfang Ye và đồng nghiệp cũng sử dụng lời gọi API, thế nhưng họ lại phân tích chúng với những mối liên hệ khác trong hệ sinh thái Android để có thể thu thập thêm thông tin về ứng dụng đó. Họ sử dụng mạng thông tin không đồng nhất với 5 loại thực thể (app, API, IMEI, chữ ký và affiliation) và cũng sử dụng khái niệm meta-path. Họ giới thiệu phương pháp trích xuất đỉnh nhúng trong ứng dụng HinLearning, và sử dụng mạng neuron học sâu. [4]

Năm 2019, Shifu Hou, Yanfang Ye và đồng nghiệp lại tiếp tục nghiên cứu và phát triển tiếp trên nền AIDroid, với mục đích áp dụng phương thức học biểu diễn (Representation learning). Các ứng dụng trong bộ huấn luyện được nhúng vào lược đồ mạng thông tin không đồng nhất, để từ đó họ có thể trích xuất thông tin từ các nút ngoài bộ huấn luyện mà không cần phải chạy lại hay chỉnh sửa mạng thông tin không đồng nhất. Họ sau đó thiết kế mô hình học sâu, lấy dữ liệu từ mạng để dự đoán phần mềm độc hại ở thời gian thực.[5]

Năm 2020, Hou và Ye lại có một hướng đi mới: sử dụng phương pháp học biểu diễn phân tích đặc trưng (Disentangled Representation Learning) để phân tích các phần mềm độc hại. Lần này, ngoài các đặc trưng của ứng dụng, Hou và Ye còn tập trung vào các yếu tố xã hội giữa ứng dụng, người lập trình và mẫu điện thoại để làm thông tin cho mạng. Họ đề ra phương pháp tích hợp các miền thông tin được tạo ra trước đó dưới các khung nhìn đặc trưng, như nội dung ứng

dụng, người sở hữu ứng dụng và quá trình cài đặt để tạo ra một bộ phân tách đặc trưng để trích xuất những thông tin, những hệ số biến đổi ngầm định trong mạng. Đây là lần đầu tiên phương pháp học biểu diễn phân tích đặc trưng được sử dụng trong dự đoán phần mềm độc hại. [6]

Ta thấy rằng, nhóm tác giả trên trung thành với hướng tiếp cận biểu diễn thông tin trên các mạng không đồng nhất, thực hiện trích xuất các thông tin trên mạng đó và sử dụng các mô hình học máy để dò tìm phần mềm độc hại, thế nhưng họ liên tục phát triển mô hình của họ theo các hướng khác nhau.

Điều này có thể thấy rõ trước hết qua số lượng kiểu đối tượng và liên kết họ sử dụng: đầu tiên là chỉ với ứng dụng và API (HINDroid), họ phát triển thành bộ (ứng dụng, IMEI, API, chữ ký và affiliation), và gần đây nhất là tích hợp các đối tượng xã hội để làm phong phú thông tin trong mô hình của họ.

Chúng ta cũng có thể thấy sự phát triển của họ qua các phương pháp trích xuất thông tin. Từ trích xuất bằng meta-path dựa trên MKL, họ tiếp tục chuyển sang phương pháp trích xuất đỉnh nhúng trong AIDroid, rồi mới đây nhất là sử dụng phương thức học biểu diễn và nhúng đồ thị. Bên cạnh đó, các nhóm cũng sử dụng nhiều mô hình học máy khác nhau: từ các mô hình học máy cơ bản, sang các mô hình học sâu.

Là một công cụ tích hợp và biểu diễn dữ liệu lớn hiệu quả, việc nghiên cứu về mạng thông tin không đồng nhất và các thuật toán trích xuất dữ liệu trên mạng này để có thể thu thập được nhiều thông tin ngữ nghĩa còn tiềm ẩn sau cấu trúc mạng hiện đang là một hướng nghiên cứu nổi trội và còn rất nhiều thách thức ở phía trước. Đặc biệt là đối với bài toán phát hiện mã độc nói chung và mã độc Android nói riêng.

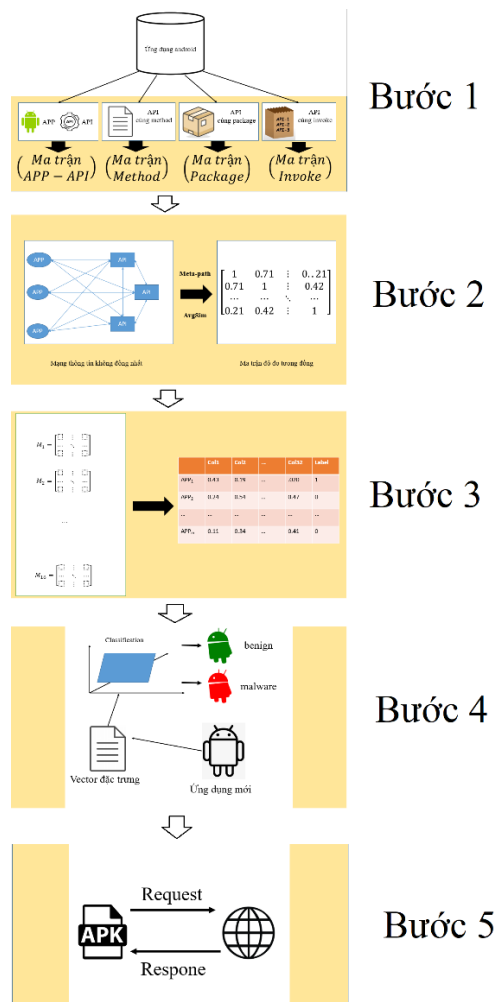
Cũng giống như hướng nghiên cứu của nhóm tác giả Shifu Hou và Yanfang Ye, chúng em dựa trên phương pháp phân tích lời gọi API và các mối liên hệ giữa chúng để xây dựng mạng thông tin không đồng nhất. Điểm khác biệt trong hướng nghiên cứu của nhóm là chúng em sử dụng độ đo AvgSim để đo độ tương đồng giữa hai ứng dụng Android bất kỳ, trên cơ sở các thông tin thu được, chúng em đề xuất công thức trích chọn vector đặc trưng để biểu diễn các ứng dụng Android thành các vector rồi đưa vào huấn luyện trong các mô hình học máy. Sau khi thực nghiệm trên các bộ dữ liệu được thu thập từ các nguồn đáng tin cậy và đánh giá kết quả, nhóm lựa chọn ba mô hình tốt nhất để xây dựng hệ thống phát hiện mã độc Android.

CHƯƠNG 3. ỨNG DỤNG MẠNG KHÔNG ĐỒNG NHẤT CHO BÀI TOÁN PHÁT HIỆN PHẦN MỀM ĐỘC HẠI ANDROID

3.1. Mô hình đề xuất

Quá trình xây dựng hệ thống phát hiện phần mềm độc hại Android được thực hiện qua các bước như sau:

- Bước 1: Xây dựng cơ sở dữ liệu cho bài toán.
- Bước 2: Xây dựng mạng thông tin không đồng nhất để biểu diễn dữ liệu mã độc.
- Bước 3: Đề xuất công thức trích chọn đặc trưng từ mạng thông tin không đồng nhất.
- Bước 4: Huấn luyện học máy, đánh giá và lựa chọn mô hình học hiệu quả nhất.
- Bước 5: Xây dựng hệ thống phát hiện phần mềm độc hại dựa trên các mô hình học máy đã lựa chọn.



Hình 3.1 Minh họa mô hình đề xuất

3.2. Xây dựng cơ sở dữ liệu

3.2.1. Thu thập dữ liệu

Hiện tại có rất nhiều nguồn dữ liệu dành cho các dự án về mã độc android được công bố trên khắp thế giới như Drebin, CIC Dataset, Android Malware Dataset (AMD), vv... Trong nghiên cứu này, chúng em thu thập các mẫu mã độc và mã lành từ ba nguồn dữ liệu đã được kiểm chứng là Drebin [8], Virusshare_apk_2012 [9], CIC Dataset [10]. Tổng cộng có 8000 ứng dụng độc hại android và 4696 ứng dụng lành tính.

Bộ Drebin chứa bao gồm 5560 ứng dụng độc hại Android được thu thập trong khoảng từ 2010 đến 2012. Trong đó chúng em sử dụng 2000 ứng dụng độc hại android trong bộ dữ liệu này.



Hình 3.2 Bộ dữ liệu Drebin

Bộ CIC Dataset chứa bao gồm 17,314 ứng dụng Android, trong đó có chứa khoảng 6000 ứng dụng android lành tính và 11,314 ứng dụng độc hại android. Chúng em sử dụng 4000 ứng dụng độc hại android và 4696 ứng dụng android lành tính ở trong bộ dữ liệu này.

CICMalDroid 2020

We are providing a new Android Malware dataset, namely CICMalDroid 2020, that has the following four properties:

1. **Big.** It has more than 17,341 Android samples.
2. **Recent.** It includes recent and sophisticated Android samples until 2018.
3. **Diverse.** It has samples spanning between five distinct categories: Adware, Banking malware, SMS malware, Riskware, and Benign
4. **Comprehensive.** It includes the most complete captured static and dynamic features compared with publicly available datasets.

Hình 3.3 Bộ dữ liệu CIC Dataset

Bộ virusshare_apk_2012 là một trong các bộ dữ liệu ở trên trang virusshare, bao gồm khoảng 5000 phần mềm độc hại. Chúng em sử dụng 2000 ứng dụng độc hại android ở bộ này.

VirusShare.com - Because Sharing is Caring

[Home](#) • [Hashes](#) • [Torrents](#) • [Research](#) • [About](#)

VirusShare is proud to have played a part in assisting the tireless efforts of the global research community and is thankful to the researchers who have contributed to the project. Below is an incomplete but growing list of publications that have cited VirusShare as a data source for their research.

• 2020 •

Abbasi, Muhammad Shabbir; Al-Sahaf, Harith; Welch, Ian; (2020). **Particle Swarm Optimization: A Wrapper-Based Feature Selection Method for Ransomware Detection and Classification.** *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, 181-196. Springer.

Ahmed, Yahye Abukar; Koçer, Barış; Al-rimy, Bander Ali Saleh; (2020). **Automated Analysis Approach for the Detection of High Survivable Ransomware.** *KSII Transactions on Internet and Information Systems (TIIS)*, 14(5), 2236-2257. 한국인터넷정보학회.

Ahmed, Yahye Abukar; Koçer, Barış; Huda, Shamsul; Al-rimy, Bander Ali Saleh; Hassan, Mohammad Mehedi; (2020). **A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection.** *Journal of Network and Computer Applications*, 102753. Elsevier.

Alaeiyan, Mohammadhadi; Dehghantanha, Ali; Dargahi, Tooska; Conti, Mauro; Parsa, Saeed; (2020). **A multilabel fuzzy relevance clustering system for malware attack attribution in the edge layer of cyber-physical networks.** *ACM Transactions on Cyber-Physical Systems*, 4(3), 1-22. ACM New York, NY, USA.

Hình 3.4 Bộ dữ liệu VirusShare

Vậy tổng cộng chúng em lấy và sử dụng 8000 ứng dụng độc hại android từ 3 bộ Drebin, VirusShare và CIC Dataset, 4696 ứng dụng lành tính từ bộ CICDataset.

3.2.2. Dịch ngược và trích xuất lời gọi API

File dạng .apk là file nén của một phần mềm android, chúng chứa các thông tin như là: file AndroidManifest.xml, file classes.dex, resource.arsc, các folder như là META-INF, res.

Name	Size	Packed	Type	Modified	CRC32
..			Local Disk		
META-INF			File folder		
res			File folder		
AndroidManifest.xml	6,276	1,650	XML Document	26/12/2013 12:30 AM	59FE59F1
classes.dex	69,896	26,451	DEX File	26/12/2013 12:30 AM	DC93EB34
resources.arsc	83,088	83,088	ARSC File	26/12/2013 12:30 AM	3C51F6A6

Hình 3.5 Cấu trúc một file apk

Chúng ta không thể xem trực tiếp nội dung file apk mà cần một phần mềm hỗ trợ cho việc này chính là dự án mở Apktool, được phát triển bởi iBotPeaches [11]. Phần mềm Apktool hỗ trợ dịch các file apk thành một hệ thống file smali. File smali có thể được hiểu như là một dạng mã nguồn hợp ngữ của ứng dụng android. Khi file apk ở dạng mã smali thì có thể dễ dàng tiến hành đọc thông tin mã nguồn của nó.

```
I: Using Apktool 2.4.1 on 0e9e953a0e160af4b2e41b5cdfa939cb50d6bb39896d1ff45702b6be4940e6ee
.d13ad86445cd0ce30f9cb0c4e285bb48
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Admin\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Hình 3.6 Quá trình dịch ngược file apk

Name	Date modified	Type	Size
Actor\$1.smali	17/3/2021 7:26 PM	SMALI File	7 KB
Actor.smali	17/3/2021 7:26 PM	SMALI File	171 KB
C2DMReceiver.smali	17/3/2021 7:26 PM	SMALI File	6 KB
Checker.smali	17/3/2021 7:26 PM	SMALI File	10 KB
DeviceRegistrar\$1.smali	17/3/2021 7:26 PM	SMALI File	6 KB
DeviceRegistrar\$2.smali	17/3/2021 7:26 PM	SMALI File	5 KB
DeviceRegistrar.smali	17/3/2021 7:26 PM	SMALI File	7 KB
Main\$1.smali	17/3/2021 7:26 PM	SMALI File	2 KB
Main\$2.smali	17/3/2021 7:26 PM	SMALI File	2 KB
Main\$3.smali	17/3/2021 7:26 PM	SMALI File	2 KB
Main\$4.smali	17/3/2021 7:26 PM	SMALI File	3 KB
Main\$AsyncLoader.smali	17/3/2021 7:26 PM	SMALI File	10 KB
Main.smali	17/3/2021 7:26 PM	SMALI File	34 KB
Manifest\$permission.smali	17/3/2021 7:26 PM	SMALI File	1 KB
Manifest.smali	17/3/2021 7:26 PM	SMALI File	1 KB
Msg.smali	17/3/2021 7:26 PM	SMALI File	4 KB

Hình 3.7 Các file smali thu được sau khi dịch ngược

Sau khi dịch ngược các file apk, chúng em đi trích xuất các lời gọi API từ các nội dung đã được dịch ngược. API là viết tắt của Application Programing Interface. API được định nghĩa là một giao thức, một hàm hoặc một công cụ hỗ trợ cho phép hai ứng dụng có thể giao tiếp được với nhau, cụ thể ở đây là giao tiếp giữa phần mềm android và hệ điều hành android. Các lời gọi API cho phép phần mềm truy cập vào tài các nguyên hệ thống như là bộ nhớ, camera hay là microphone.

Cấu trúc của một lời gọi API trong file smali có dạng bắt đầu bằng từ khóa invoke, tiếp theo đó là thông tin về danh sách tham số, tên lớp, tên lời gọi API, kiểu của danh sách tham số truyền vào và cuối cùng là kiểu trả về của API.

Ví dụ ở hình 3.8 có 2 lời gọi API sẽ được gọi khi ứng dụng thực thi đến đoạn mã này. Hai lời gọi API này có cùng một kiểu invoke là virtual. Đối với lời gọi thứ nhất, lời gọi này có một tham số là v0, tiếp theo “Landroid/content/Context” chính là tên lớp định nghĩa lời gọi API này, “getReources” là tên của lời gọi API, cuối cùng API này có kiểu trả về là một đối tượng thuộc lớp “Landroid/content/res/Resources”. Tương tự, lời gọi API thứ hai có hai tham số là v0 và v1, có tên lớp là “Landroid/content/res/Resources”, tên lời gọi API là “openRawResource”, kiểu trả về là “Ljava/io/InputStream”.


```

.line 173
:goto_0
:try_start_1|
iget-object v0, p0, Lcom/software/application/Actor;->mContext:Landroid/content/Context;

invoke-virtual {v0}, Landroid/content/Context;->getResources()Landroid/content/res/Resources;

move-result-object v0

const/high16 v1, 0x7f050000

invoke-virtual {v0, v1}, Landroid/content/res/Resources;->openRawResource(I)Ljava/io/InputStream;

move-result-object v0

```

Hình 3.8 Các lời gọi API trong file smali

Có rất nhiều cách tiếp cận để có thể trích xuất lời gọi API, chẳng hạn như có thể chỉ lọc những hướng dẫn gọi một số chức năng tiêu chuẩn của Android. Tuy nhiên, vì danh sách các lời gọi này có thể khá dài và quan trọng là nó có thể thay đổi so với các phiên bản Android khác nhau nên chúng em chỉ tập trung vào các lời gọi thuộc về không gian tên Android và Java (tức là các lớp bắt đầu bằng Landroid/LJava). Việc trích xuất toàn bộ lời gọi API trên toàn bộ dữ liệu là không khả thi và không có ý nghĩa trong quá trình huấn luyện học máy. Bởi vì, việc trích chọn quá nhiều thông tin để huấn luyện sẽ làm tăng thời gian xử lý, làm sai lệch các kết quả. Hơn nữa, có nhiều ứng dụng lành tính sử dụng các API mà các mã độc không hề sử dụng, lúc đó các API này không mang nhiều ý nghĩa cho việc phân loại. Một số nghiên cứu lại lựa chọn phương án trích xuất và xếp hạng các lời gọi API trên toàn bộ dữ liệu và sau đó lựa chọn theo thứ tự xếp hạng. Phương án này cũng có hạn chế đó là, có thể các phần mềm độc hại sẽ chỉ sử dụng một vài API với tần suất rất thấp, trong khi đó, các API xử lý bình thường như định dạng các chuỗi được sử dụng khá thường xuyên trong mọi ứng dụng nên sẽ có khả năng việc lựa chọn các API theo cách này sẽ bỏ sót các API có trong phần mềm độc hại dẫn tới bỏ sót các phần mềm độc hại khi tiến hành phân loại. Do vậy, giải pháp tốt nhất hiện tại đó chính là lọc các API mà các phần mềm độc hại thường hay sử dụng có trong các mẫu, nghĩa là các API sẽ được trích xuất theo tần suất xuất hiện trong các phần mềm độc hại nhiều nhất. Chúng em giới hạn số lượng API cần trích xuất vào khoảng 205.

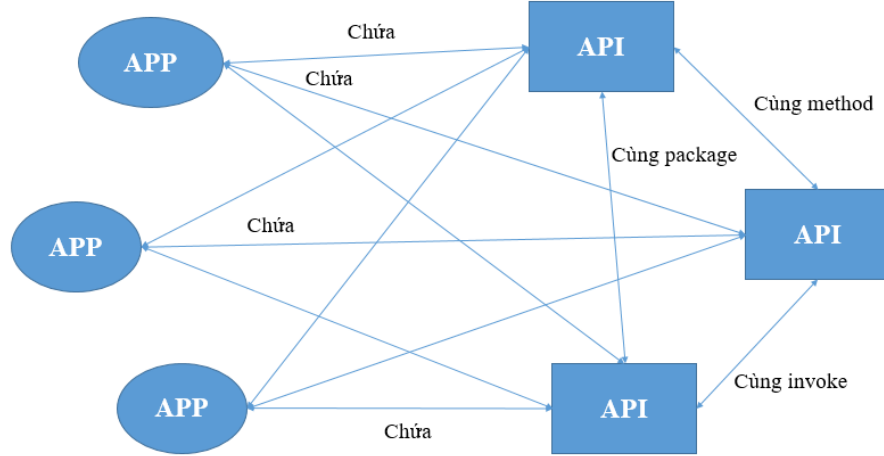
```
[
    "Landroid/content/ContentResolver;->insert",
    "Landroid/content/ContentResolver;->delete",
    "Landroid/content/ContentResolver;->update",
    "Landroid/content/ContentResolver;->query",
    "Landroid/content/Context;->startService",
    "Landroid/content/Context;->deleteFile",
    "Landroid/content/Context;->getDir",
    "Landroid/content/Context;->getFilesDir",
    "Landroid/content/Context;->getApplicationInfo",
    "Landroid/content/Context;->getCacheDir",
    "Landroid/content/Context;->getDatabasePath",
    "Landroid/content/Context;->openOrCreateDatabase",
    "Landroid/content/Context;->checkCallingPermission",
    "Landroid/content/Context;->openFileInput",
    "Landroid/content/Context;->openFileOutput",
    "Landroid/content/Context;->fileList",
    "Landroid/content/Context;->deleteDatabase",
    "Landroid/content/Context;->databaseList",
    "Landroid/content/Context;->stopService",
    "Landroid/content/Context;->getFileStreamPath",
```

Hình 3.9 Danh sách các lời gọi API phần mềm độc hại Android hay dùng

Dựa vào danh sách bao gồm 205 các lời gọi API mà các phần mềm độc hại hay sử dụng nhất, chúng em tiến hành quét trên toàn bộ các file smali của từng ứng dụng. Nếu như ứng dụng (*bắt đầu từ đây trở đi, chúng em sẽ gọi ứng dụng là APP*) nào có sử dụng đến các lời gọi API trong danh sách lời gọi API đã được lọc ra, các thông tin quan trọng như kiểu gọi invoke của API, method nào gọi API đó, và tên package của API sẽ được trích xuất. Những thông tin được trích xuất này là cơ sở để tạo nên bốn ma trận đại diện cho bốn mối quan hệ ở trong mạng thông tin không đồng nhất, đó là:

- Quan hệ giữa APP-API: được biểu diễn bằng ma trận APP-API_{mn}. trong đó m là số lượng app, n là số lượng các API. $a_{ij}=1$ khi App_i có sử dụng lời gọi API_j, ngược lại $a_{ij}=0$.
- Quan hệ giữa hai API cùng chung lời gọi invoke: được biểu diễn bằng ma trận INVOKE_{nn}, trong đó : $a_{ij}=1$ nếu API_i và API_j có cùng chung phương thức gọi invoke, ngược lại $a_{ij}=0$. Các phương thức invoke bao gồm:
 - Invoke – static: gọi một phương thức tĩnh với các biến số
 - Invoke – virtual: gọi một phương thức ảo với các biến số
 - Invoke – direct: gọi một phương thức với các biến số mà không cần phân giải phương thức ảo
 - Invoke – super: gọi một phương thức ảo của lớp cha ngay liền kề

- Invoke - interface : gọi một phương thức thuộc giao diện.
- Quan hệ giữa hai API cùng chung method: được biểu diễn bằng ma trận $METHOD_{nn}$, trong đó $a_{ij} = 1$ nếu API_i và API_j có cùng chung một method.
- Quan hệ giữa hai API cùng chung package: được biểu diễn bằng ma trận $PACKAGE_{nn}$, trong đó $a_{ij} = 1$ nếu API_i và API_j có cùng lời gọi đến một package.



Hình 3.10 Mạng thông tin không đồng nhất

3.2.3. Xây dựng ma trận độ đo tương đồng dựa trên mạng thông tin không đồng nhất.

Từ bốn ma trận thể hiện các mối quan hệ trong mạng thông tin không đồng nhất, chúng em sử dụng công thức tính độ đo tương đồng AvgSim theo các siêu đường đi để đo sự tương tự giữa một App_i đến App_j bất kỳ. Một siêu đường đi trên mạng thông tin không đồng nhất từ một đỉnh s đến một đỉnh t thể hiện mối quan hệ giữa s và t .

- Công thức AvgSim dùng cho đỉnh bất kỳ:

$$AvgSim(s, t|P) = \frac{1}{2} (RW(s, t|P) + RW(t, s|P^{-1}))$$

Trong đó : $RW(s, t|R_1 \circ R_2 \circ \dots \circ R_l) = \frac{1}{|o(s|R_1)|} \sum_{i=1}^{|o(s|R_1)|} (O_i(s|R_1), t|R_2 \circ \dots \circ R_l)$

- Công thức AvgSim dùng cho ma trận các mối quan hệ

$$AvgSim = \frac{1}{2} [R_{A_1 A_2} R_{A_2 A_3} \dots R_{A_l A_{l+1}} + C_{A_1 A_2} C_{A_2 A_3} \dots C_{A_l A_{l+1}}]$$

Trong đó : $R_{A_i A_j}$ là ma trận chuẩn hóa theo hàng, $C_{A_i A_j}$ là ma trận chuẩn hóa theo cột của ma trận $M_{A_i A_j}$, ma trận $M_{A_i A_j}$ là ma trận thể hiện mối quan hệ giữa đỉnh A_i và đỉnh A_j trong đồ thị.

Khi thực hiện xử lý dữ liệu, việc tính toán số lượng siêu đường đi từ một App đến một App khác trong mạng thông tin không đồng nhất khá mất thời gian, vì đây là bài toán duyệt có điều kiện trên đồ thị có số đỉnh lớn. Ngoài ra, một số nghiên cứu cho thấy, trong mạng thông tin không đồng nhất thì các siêu đường đi có độ dài lớn thường không đem lại nhiều thông tin để phân biệt mối quan hệ giữa các đỉnh trong mạng. Vì vậy trong nghiên cứu này, chúng em lựa chọn các đường đi đối xứng, độ dài tối đa của một siêu đường đi là 7 và số lượng siêu đường đi là 16.

STT	Siêu đường	Mô Tả
P ₁	AA ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₂	AMA ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng method}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₃	APA ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng package}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₄	AIA ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng invoke}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₅	AMPM ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng method}}$ API $\xrightarrow{\text{cùng package}}$ API $\xrightarrow{\text{cùng method}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₆	APMP ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng package}}$ API $\xrightarrow{\text{cùng method}}$ API $\xrightarrow{\text{cùng package}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₇	AMIM ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng method}}$ API $\xrightarrow{\text{cùng invoke}}$ API $\xrightarrow{\text{cùng method}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₈	AIMI ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng invoke}}$ API $\xrightarrow{\text{cùng method}}$ API $\xrightarrow{\text{cùng invoke}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₉	APIP ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng package}}$ API $\xrightarrow{\text{cùng invoke}}$ API $\xrightarrow{\text{cùng package}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP
P ₁₀	APII ^T A ^T	APP $\xrightarrow{\text{chứa}}$ API $\xrightarrow{\text{cùng invoke}}$ API $\xrightarrow{\text{cùng package}}$ API $\xrightarrow{\text{cùng invoke}^{-1}}$ API $\xrightarrow{\text{chứa}^{-1}}$ APP

P_{11}	$AMPIP^T M^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng method}} API \xrightarrow{\text{cùng package}} API \xrightarrow{\text{cùng invoke}} API$ $\xrightarrow{\text{cùng package}^{-1}} API \xrightarrow{\text{cùng method}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$
P_{12}	$APMIM^T P^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng package}} API \xrightarrow{\text{cùng method}} API \xrightarrow{\text{cùng invoke}} API$ $\xrightarrow{\text{cùng method}^{-1}} API \xrightarrow{\text{cùng package}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$
P_{13}	$AMIPI^T M^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng method}} API \xrightarrow{\text{cùng invoke}} API \xrightarrow{\text{cùng package}} API$ $\xrightarrow{\text{cùng invoke}^{-1}} API \xrightarrow{\text{cùng method}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$
P_{14}	$AIMPM^T I^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng invoke}} API \xrightarrow{\text{cùng method}} API \xrightarrow{\text{cùng package}} API$ $\xrightarrow{\text{cùng method}^{-1}} API \xrightarrow{\text{cùng invoke}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$
P_{15}	$AIPMP^T I^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng invoke}} API \xrightarrow{\text{cùng package}} API \xrightarrow{\text{cùng method}} API$ $\xrightarrow{\text{cùng package}^{-1}} API \xrightarrow{\text{cùng invoke}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$
P_{16}	$APIMI^T P^T A^T$	$APP \xrightarrow{\text{chứa}} API \xrightarrow{\text{cùng package}} API \xrightarrow{\text{cùng invoke}} API \xrightarrow{\text{cùng method}} API$ $\xrightarrow{\text{cùng invoke}^{-1}} API \xrightarrow{\text{cùng package}^{-1}} API \xrightarrow{\text{chứa}^{-1}} APP$

Bảng 3.1 Mười sáu siêu đường đi được sử dụng trong mô hình

Ứng với 16 siêu đường đi, ta áp dụng công thức AvgSim cho ma trận các mối quan hệ, kết quả thu được 16 ma trận độ đo tương đồng M_k trong đó có các phần tử a_{ij} thể hiện sự tương đồng giữa APP_i và APP_j , như vậy hàng thứ i của ma trận M_k sẽ thể hiện độ tương đồng của ứng APP_i so với tất cả các APP còn lại trong tập dữ liệu

3.2.4. Đề xuất công thức trích xuất vector đặc trưng

Sau khi tạo ra được các ma trận độ đo tương đồng giữa các APP, chúng em tiến hành trích xuất các vector đặc trưng để đưa vào mô hình học máy. Có thể nhận thấy được, hàng i của từng ma trận độ đo tương đồng thể hiện sự tương đồng của APP_i so với tất cả các APP còn lại ở trong bộ dữ liệu. Vì thế chúng em chia hàng i của từng ma trận độ đo tương đồng ra làm 2 phần, phần thứ nhất là độ đo tương đồng của APP_i so với tất cả các APP độc hại ở trong bộ dữ liệu, phần thứ 2 là độ đo tương đồng của APP_i so với tất cả các APP lành tính. Mục đích của việc chia hàng ra thành hai phần là để so sánh APP_i với các ứng dụng độc hại và các ứng dụng lành tính. Sau đó, chúng em cộng tổng phần thứ nhất lại tạo thành một đặt trưng thể hiện chỉ số khả năng APP_i là ứng dụng độc hại, tổng

phần thứ hai sẽ tạo thành một đặc trưng thể hiện chỉ số khả năng APP_i là ứng dụng lành tính. Ứng với mỗi siêu đường đi thì chúng em sẽ tính ra được hai đặc trưng của APP_i vậy với mười sáu siêu đường đi đã được đưa ra, chúng em xây dựng được vector đặc trưng gồm ba mươi hai phần tử cho mỗi APP . Và trên bộ dữ liệu gồm m APP thì sẽ xây dựng được một ma trận vector đặc trưng là $V_{m,32}$

- Công thức trích xuất đặc trưng:

$$\begin{cases} V_{i,j} = \sum_{l=0}^k M_n[i, l]; \text{ với } j = 2n - 1 \\ V_{i,j} = \sum_{l=k}^m M_n[i, l]; \text{ với } j = 2n \end{cases}$$

Trong đó:

- m là tổng số lượng ứng dụng được sử dụng
- k là tổng số lượng ứng dụng độc hại android được sử dụng
- $n = \overline{1,16}$
- $i = \overline{1,m}$
- **Chú ý:** ma trận độ đo tương đồng được sắp xếp theo thứ tự các ứng dụng độc hại android nằm trước các ứng dụng android lành tính. Mặt khác vì công thức này cho ra kết quả vector đặc trưng có số phần tử là 32 nên chúng em gọi đây là công thức v32.

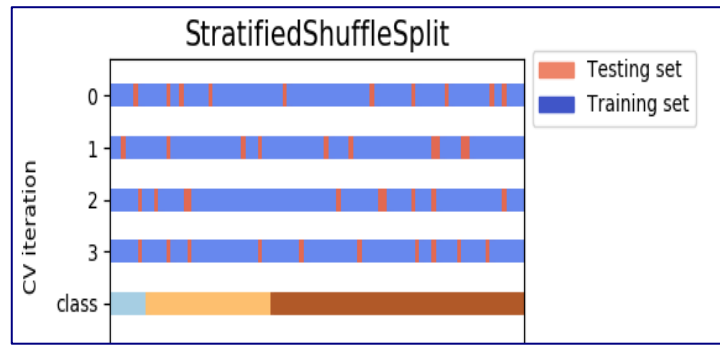
Như vậy mỗi App_i trong thực nghiệm của chúng em sẽ được biểu diễn bởi vector đặc trưng $v = (v_1, v_2, \dots, v_{32})$. Các vector đã thấy ở trên sẽ được đưa vào mô hình học máy. Ở đây, chúng em sẽ sử dụng các mô hình học máy như mô hình Decision Tree, Random Forest, Support Vector Machines và thuật toán học sâu Neural Network để thực hiện huấn luyện và đánh giá kết quả.

3.3. Huấn luyện học máy và đánh giá mô hình

3.3.1. Phương pháp đánh giá

Để đánh giá hiệu quả của các mô hình phân lớp, chúng em sử dụng kỹ thuật Stratified Shuffle Split để phân chia dữ liệu cho các thực nghiệm.

Kỹ thuật Stratified Shuffle Split (SSS) thực hiện việc xáo trộn bộ dữ liệu và sau đó chia thành n phần gọi là n_splits . Sau đó, SSS chọn một phần để sử dụng làm bộ kiểm tra, $n-1$ phần còn lại sẽ được chia tiếp để có được $n-1$ bộ kiểm tra ngẫu nhiên. Đặc điểm của kỹ thuật này là luôn xáo trộn dữ liệu trước mỗi lần chia tách, việc này đảm bảo cho các bộ kiểm tra giao nhau khác rỗng.



Hình 3.11 Stratified Shuffle Split

Với mỗi lần huấn luyện và kiểm thử một mô hình phân lớp, chúng em tính toán các độ đo sau đây:

- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
- $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$
- $\text{F1} = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$

Trong đó, TP, TN, FP, FN lần lượt là số lượng ứng dụng độc hại được phân lớp đúng; số lượng ứng dụng lành tính được phân lớp đúng; số lượng ứng dụng lành tính bị phân lớp sai; số lượng ứng dụng độc hại bị phân lớp sai.

Trong các độ đo trên, có hai độ đo được quan tâm hơn, đó là Recall và F1. Điểm Recall phụ thuộc vào giá trị FN. Trong khi đó, Precision lại chú trọng giá trị FP. Trong bài toán phát hiện phần mềm độc hại, điểm FN có sức nặng hơn FP vì nếu một ứng dụng độc hại android bị đoán nhầm thành ứng dụng lành tính sẽ nguy hiểm hơn mà một ứng dụng lành tính bị đoán nhầm thành một ứng dụng độc hại.

3.3.2. Thực nghiệm và đánh giá

Chúng em thực hiện 2 thực nghiệm dưới những điều kiện sau:

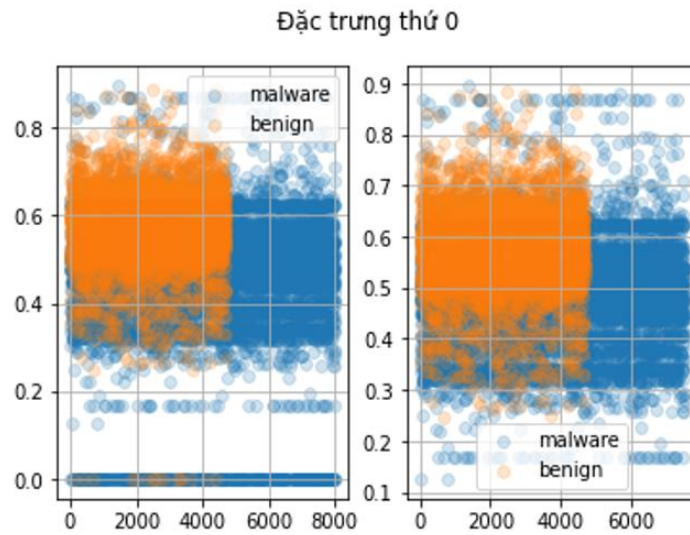
- Bộ huấn luyện - kiểm định (train - validation): 8000 độc, 4696 lành, sau khi khử nhiễu còn 7517 độc, 4676 lành
- Bộ kiểm tra (test): 2000 độc, 1000 lành
- Bộ dữ liệu được chia làm 10 phần, 9 phần dùng để huấn luyện và 1 phần dùng để kiểm định, dựa trên kỹ thuật SSS.

Các biến số bao gồm:

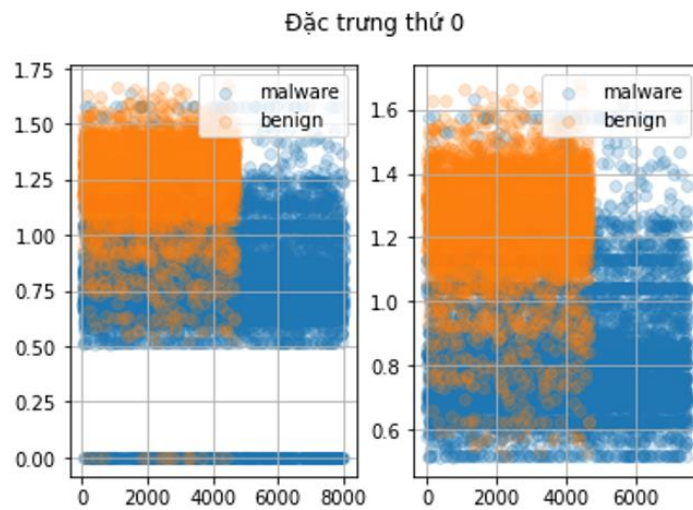
- Các mô hình học máy gồm Random Forest, Decision Tree, SVM, Mô hình Neural Network.

Chúng em sử dụng công thức trích xuất đặc trưng ở trong bài báo: “A feature representation method base on heterogeneros information network for

android malware detection” để so sánh với công thức của chúng em. Do công thức trích xuất vector đặc trưng của công thức này có 16 phần tử nên chúng em gọi công thức này là công thức v16.



Hình 3.12 Trước và sau khi khử nhiễu dữ liệu, v16



Hình 3.13 Trước và sau khi khử nhiễu dữ liệu, v32

Kết quả thử nghiệm được trình bày dưới các bảng sau:

Decision Tree						
Lần thử	Tham số	Chỉ số	Val v16	Val v32	Test v16	Test v32
1	Default	Acc	0.9459	0.9803	0.7733	0.9547

		F1	0.9561	0.9840	0.8291	0.9654
		Recall	0.9548	0.9801	0.8250	0.9475
		Precision	0.9573	0.9879	0.8333	0.9839
2	Default class_weight={0:0.5, 1:0.7}	Acc	0.9475	0.9820	0.4353	0.9567
		F1	0.9570	0.9853	0.4500	0.9669
		Recall	0.9468	0.9814	0.3465	0.9485
		Precision	0.9674	0.9893	0.6417	0.9860
3	Default class_weight={0:0.5, 1:0.9}	Acc	0.9418	0.9844	0.7153	0.9500
		F1	0.9521	0.9873	0.7862	0.9614
		Recall	0.9388	0.9840	0.7850	0.9335
		Precision	0.9658	0.9906	0.7874	0.9910
4	Default class_weight={0:0.5, 1:1.2}	Acc	0.9410	0.9836	0.7263	0.9460
		F1	0.9518	0.9866	0.7961	0.9581
		Recall	0.9455	0.9827	0.8015	0.9260
		Precision	0.9582	0.9906	0.7908	0.9925

Bảng 3.2 Kết quả của quá trình học máy sử dụng mô hình Decision Tree

Random Forest						
Lần thử	Tham số	Chỉ số	Val v16	Val v32	Test v16	Test v32
1	n_estimators=45 max_depth=7	Acc	0.9336	0.9795	0.8803	0.9640
		F1	0.9462	0.9834	0.9047	0.9728
		Recall	0.9468	0.9854	0.8525	0.9660
		Precision	0.9456	0.9815	0.9638	0.9797

2	n_estimators=45 max_depth=7 class_weight={0:0.5, 1:0.7}	Acc	0.9369	0.9795	0.8780	0.9647
		F1	0.9490	0.9835	0.9061	0.9733
		Recall	0.9521	0.9880	0.8830	0.9645
		Precision	0.9458	0.9789	0.9305	0.9822
3	n_estimators=45 max_depth=7 class_weight={0:0.5, 1:0.9}	Acc	0.9369	0.9795	0.9017	0.9593
		F1	0.9494	0.9835	0.9247	0.9693
		Recall	0.9601	0.9907	0.9060	0.9615
		Precision	0.9389	0.9764	0.9442	0.9771
4	n_estimators=45 max_depth=7 class_weight={0:0.5, 1:1.2}	Acc	0.9328	0.9738	0.8910	0.9623
		F1	0.9465	0.9789	0.9169	0.9716
		Recall	0.9654	0.9880	0.9020	0.9660
		Precision	0.9284	0.9700	0.9323	0.9772

Bảng 3.3 Kết quả của quá trình học máy sử dụng mô hình Random Forest

Support Vector Machine						
Lần thứ	Tham số	Chỉ số	Val v16	Val v32	Test v16	Test v32
1	Default	Acc	0.9131	0.9664	0.8927	0.9583
		F1	0.9300	0.9729	0.9173	0.9684
		Recall	0.9362	0.9787	0.8925	0.9565
		Precision	0.9239	0.9671	0.9434	0.9805
2	Default class_weight={0:0.5, 1:0.7}	Acc	0.9107	0.9615	0.8970	0.9623
		F1	0.9290	0.9690	0.9217	0.9717
		Recall	0.9481	0.9774	0.9095	0.9700

		Precision	0.9106	0.9608	0.9343	0.9734
3	Default class_weight={0:0.5, 1:0.9}	Acc	0.9057	0.9664	0.8890	0.9623
		F1	0.9257	0.9731	0.9172	0.9717
		Recall	0.9521	0.9880	0.9225	0.9700
		Precision	0.9006	0.9587	0.9120	0.9734
4	Default class_weight={0:0.5, 1:1.2}	Acc	0.9090	0.9615	0.8807	0.9613
		F1	0.9291	0.9694	0.9127	0.9710
		Recall	0.9668	0.9907	0.9360	0.9720
		Precision	0.8942	0.9490	0.8906	0.9701

Bảng 3.4 Kết quả của quá trình học máy sử dụng mô hình Support Vector Machine

Neural Network						
Lần thứ	Tham số	Chỉ số	Val v16	Val v32	Test v16	Test v32
1	1:5 layers, activation function: sigmoid, loss: binary_crossentropy, Input shape: (16,0) or (32,0)	Acc	0.9109	0.9545	0.8983	0.9603
		F1			0.9245	0.9705
		Recall			0.9340	0.9785
		Precision			0.9152	0.9626
2	2:6 layers, activation function: sigmoid, loss: binary_crossentropy, Input shape: (16,0) or (32,0)	Acc	0.9079	0.9536	0.8843	0.9620
		F1			0.9079	0.9713
		Recall			0.8550	0.9660
		Precision			0.9677	0.9767
3	3:7 layers,	Acc	0.9074	0.9553	0.8860	0.9647

activation function: sigmoid, loss: binary_crossentropy, Input shape: (16,0) or (32,0)	F1			0.9122	0.9736
	Recall			0.8885	0.9770
	Precision			0.9372	0.9702

Bảng 3.5 Kết quả của Neural Network

Vậy tổng hợp lại ta có bảng so sánh các mô hình có tham số cho kết quả tốt nhất:

Tổng hợp mô hình						
Lần thứ	Mô hình	Chỉ số	Val v16	Val v32	Test v16	Test v32
1	Decision Tree: Default, Class_weight={0:0.5, 1:0.7}	Acc	0.9475	0.9820	0.4353	0.9567
		F1	0.9570	0.9853	0.4500	0.9669
		Recall	0.9468	0.9814	0.3465	0.9485
		Precision	0.9674	0.9893	0.6417	0.9860
2	Random Forest: n_estimators=45, max_depth=7	Acc	0.9336	0.9795	0.8803	0.9640
		F1	0.9462	0.9834	0.9047	0.9728
		Recall	0.9468	0.9854	0.8525	0.9660
		Precision	0.9456	0.9815	0.9638	0.9797
3	Support Vector Machine: Default, Class_weight={0:0.5, 1:1,2}	Acc	0.9090	0.9615	0.8807	0.9613
		F1	0.9291	0.9694	0.9127	0.9710
		Recall	0.9668	0.9907	0.9360	0.9720
		Precision	0.8942	0.9490	0.8906	0.9701
4	Neural Network: activation function: sigmoid,	Acc	0.9074	0.9553	0.8860	0.9647
		F1			0.9122	0.9736

loss: binary_crossentropy, Input shape: (16,0) or (32,0)	Recall			0.8885	0.9770
	Precision			0.9372	0.9702

Bảng 3.6 Bảng so sánh các mô hình

Qua so sánh, chúng em lựa chọn ba mô hình học máy có độ chính xác cao nhất là Decision tree, Random forest và Support vector machine để đoán nhận.

3.4. Xây dựng hệ thống

3.4.1. Tổng quan về hệ thống

Từ những kết quả đã đạt được, chúng em đi xây dựng hệ thống phát hiện phần mềm độc hại thông minh dựa trên mạng thông tin không đồng nhất. Trong đó sử dụng ba mô hình để đoán nhận đó là Decision Tree, Random forest và SVM.

Những người dùng hệ thống là những người có mong muốn kiểm tra ứng dụng android mà mình tải về máy có phải là phần mềm độc hại hay không.

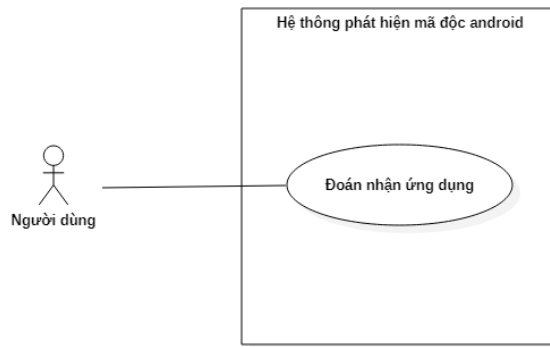
3.4.2. Các yêu cầu chức năng và phi chức năng của hệ thống

Chúng em đưa ra các yêu cầu thiết kế như sau:

- Yêu cầu chức năng:
 - Đoán nhận ứng dụng mà người dùng gửi lên, xem nó có phải là ứng dụng độc hại hay không.
- Yêu cầu phi chức năng:
 - Giao diện hệ thống đơn giản dễ hiểu, dễ sử dụng.
 - Hệ thống cần phải phản hồi lại kết quả sau trung bình 30 giây.
 - Hệ thống cần phải hoạt động tốt trên môi trường web của các thiết bị khác nhau.

3.4.3. Phân tích hệ thống

Từ những yêu cầu đã phân tích, chúng em đi xây dựng mô hình ca sử dụng:



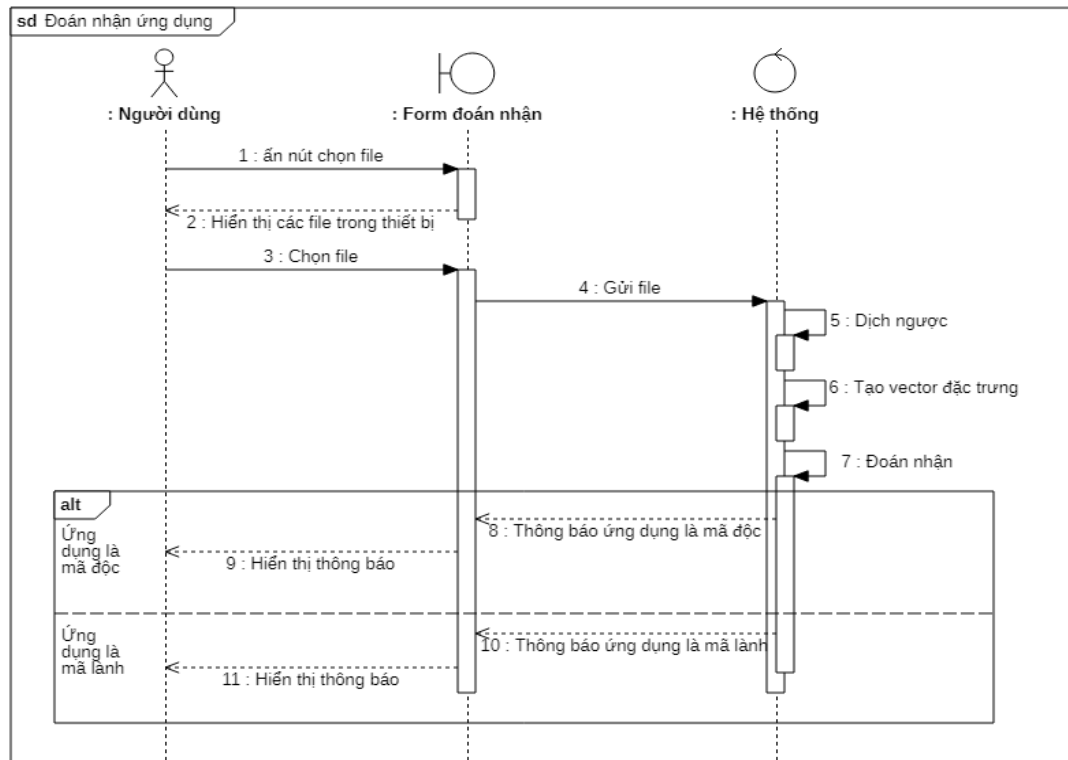
Hình 3.14 Mô hình ca sử dụng của hệ thống

Đặc tả ca sử dụng:

Ca sử dụng	Đoán nhận ứng dụng
Tác nhân	Người dùng
Mô tả ngắn gọn	Hệ thống đoán nhận ứng dụng được đưa vào
Điều kiện trước	
Các luồng cơ bản	<ol style="list-style-type: none"> 1. Người dùng ấn vào nút chọn file ở trên form đoán nhận ứng dụng 2. Người dùng chọn file muốn đoán nhận 3. Hệ thống nhận file 4. Hệ thống dịch ngược file cần đoán nhận 5. Hệ thống trích xuất các vector đặc trưng của app mới 6. Hệ thống đoán nhận, nếu kết quả là phần mềm độc hại, hệ thống trả về thông báo ứng dụng là phần mềm độc hại cho người dùng
Các luồng phụ	6.1 Nếu kết quả là mã lành, hệ thống trả về thông báo ứng dụng là mã lành cho người dùng
Điều kiện sau	
Các yêu cầu riêng	

Hình 3.15 Đặc tả ca sử dụng Đoán nhận ứng dụng

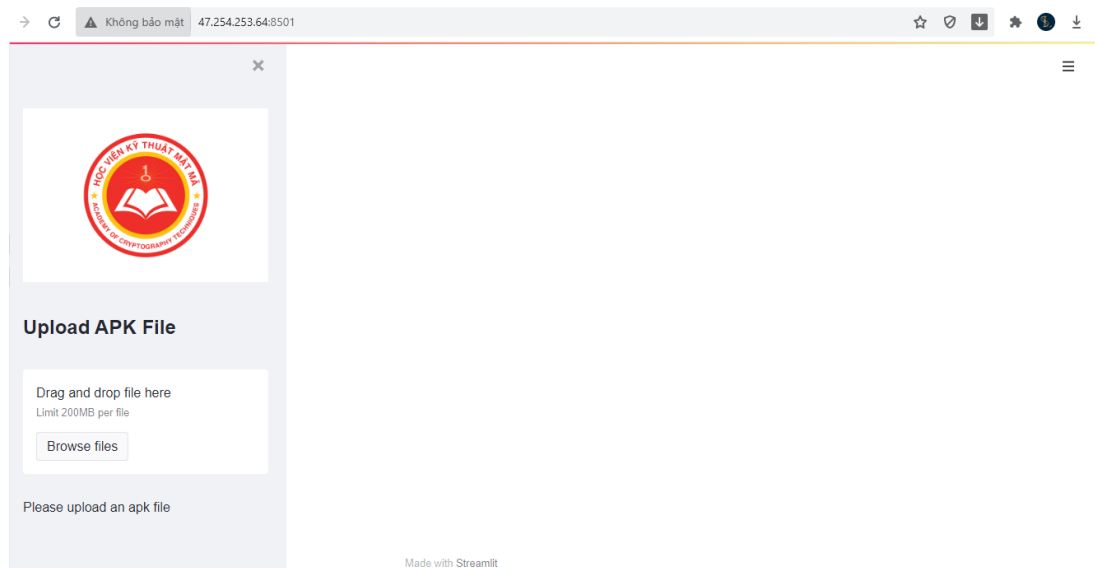
Phân tích ca sử dụng Đoán nhận ứng dụng



Hình 3.16 Biểu đồ tuần tự ca sử dụng Đoán nhận ứng dụng

3.4.4. Thiết kế


Thiết kế giao diện:



Hình 3.17 Giao diện hệ thống phát hiện phần mềm độc hại android

1e2495649e72a872883f49681a0bc1258097c70d34accfd1f9d05565653a1661.c48b80ff41c6286679b13f9e8c2d18df....	25/4/2016 9:07 PM	APK File	3,999 KB
4aeccf56981a32461ed3cad5e197a3eedb97a8dfb916affc67ce4b9e75b67d98.apk	25/11/2013 3:10 AM	APK File	5,290 KB
7d2b0b2a521478ef357add691ff39162.apk	17/7/2018 1:15 AM	APK File	8,707 KB
36e2688ae7cf80b37c7885861ce4feaa65174e5b32f5e027141ef963469c56af.a1def01e9364eb1b1aueb852a7c15991....	11/11/2013 1:31 AM	APK File	2,135 KB
116b4483b19011a42d9562a0e4a117aee96e85806a26dedd468dae3847658a3e.apk	10/11/2013 7:08 PM	APK File	4,291 KB
354af6efe7fb8008e3f71843852db0dc21d6cd6de3dfdf640fbbf05127857a77.7941dde43990038d52f2ec4bb6802a87...	25/4/2016 11:48 PM	APK File	1,818 KB
60317f725b5d42682586165d99bd3f9d79a3773709e82d1eea8912cd13aa6a82.952dda3ff4243977a7f3565793c80ee...	26/4/2016 4:05 AM	APK File	1,043 KB
80679eed58457733de9644bcea359f769f5234e42460c51236087dec6742276.c0bb7613d4e80c7ca088d89d2cd6a8...	25/4/2016 1:26 PM	APK File	1,069 KB
26689763234d194687a684cc2ec2be2dc31d35b6e1add7d7691e3080fe3d5699.eff12a0a012efd5bf54223036c634fa...	26/4/2016 1:07 AM	APK File	3,336 KB
ada1e237899d662513ff089aeb581679a0980fea2d2cbb65bcb92808616c8d2b.apk	10/11/2013 6:46 PM	APK File	4,124 KB
b2b3732a3afa0a72e1471f5f6bead12f.apk	17/7/2018 1:15 AM	APK File	9,693 KB
f52e55e008a035c317fe4a63388bfe30a49f5168031755de83174c90a5b0973a.apk	9/11/2017 1:33 AM	APK File	6,756 KB
VirusShare_b83df8e33875745aaf2459dcc9f302b2.apk	26/5/2019 10:36 PM	APK File	1,096 KB

Hình 3.18 Quá trình chọn file



Upload APK File

Drag and drop file here
Limit 200MB per file

Browse files

116b4483b19011a42d95...
4.2MB

Completely saved and decompiled

Time save and decompile: 9.192346811294556

	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8	col 9
0	0.6808	0.5300	0.7364	0.6768	0.8920	0.5535	0.6808	0.6528	0.7222

Prediction: 7.66% is BENIGN, 92.34% is MALWARE

Time create feature vector and predict: 1.2730674743652344

Made with Streamlit

Hình 3.19 Kết quả đoán nhận

Do hệ thống còn phụ thuộc nhiều vào phần mềm bên thứ 3 đó là trình dịch ngược APKTool, cho nên thời gian đoán nhận của một ứng dụng được đưa vào vẫn rất lâu, trung bình từ là 25 giây cho một ứng dụng nặng 15-20MB, trong đó khoảng 20 giây là thời gian dịch ngược của APKTool, 5 giây là thời gian phân tích và đưa ra kết quả cho người dùng.

CHƯƠNG 4. KẾT LUẬN

4.1. Các vấn đề đã làm được

- Hiểu được cấu trúc của .apk và sử dụng công cụ để hỗ trợ giải nén và dịch ngược các file apk thành hệ thống file smali
- Trích xuất thông tin từ file smali và xây dựng nên các ma trận kề, biểu thị các mối quan hệ giữa App – API, API – API.
- Xây dựng mô hình mạng thông tin không đồng nhất và tính độ đo tương đồng giữa các App dựa trên công thức AvgSim.
- Xây dựng công thức trích xuất đặc trưng để biểu diễn các ứng dụng thành các vector đặc trưng.
- Huấn luyện học máy và đánh giá các mô hình. Trên cơ sở đó lựa chọn các mô hình hiệu quả để xây dựng hệ thống phát hiện phần mềm độc hại Android.

4.2. Các vấn đề còn tồn đọng của hệ thống

Thời gian đoán nhận một ứng dụng còn mất nhiều thời gian.

4.3. Hướng nghiên cứu tiếp theo

- Bổ sung thêm các đối tượng cùng các mối liên kết mới vào mạng thông tin không đồng nhất như quyền truy cập, nhà sản xuất điện thoại, chữ ký, vv...
- Cải tiến các mô hình học máy để nâng cao hiệu suất dự đoán cho lớp bài toán phát hiện phần mềm độc hại Android.
- Cải tiến và đề xuất kỹ thuật trích xuất đặc trưng trên mạng thông tin không đồng nhất.

THAM KHẢO

- [1]. M. Nilay, P. Nitin, “*Review of Behavior Malware Analysis for Android*” *International Journal of Engineering and Innovative Technology*, vol 2, pp.230 – 235, 2013.
- [2]. Shifu Hou, Yanfang Ye, Yangqiu Song, Melih Abdulhayoglu, “*HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network*”.
- [3]. Ding Xiao, X. Meng, “*AVGSIM: Relevance measurement on massive data in heterogeneous networks*”.
- [4]. Shifu Hou, Yanfang Ye et al., “*AiDroid: When heterogeneous information network marries deep neural network for real-time Android malware detection*”.
- [5]. Shifu Hou, Yanfang Ye et al., “*Disentangled Representation Learning in Heterogeneous Information Network for Large-scale Android Malware Detection in the COVID-19 Era and Beyond*”.
- [6]. Shifu Hou, Yanfang Ye et al., “*Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection*”.
- [7]. Rongye Liu, Ning Yang, Xiangqian Ding, Lintao Ma, “*An Unsupervised Feature Selection Algorithm: Laplacian Score Combined with Distance-Based Entropy Measure*”.
- [8]. Drebin Android Dataset - <https://www.sec.tu-bs.de/~danarp/drebin/>
- [9]. VirusShare Android Dataset - <https://virusshare.com/about>
- [10]. Android Malware Dataset (CIC) - <https://www.unb.ca/cic/datasets/andmal2017.html>
- [11]. iBotPeaches APKTool - <https://ibotpeaches.github.io/Apktool/>